Expert Systems With Applications 112 (2018) 156–172

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Improved grasshopper optimization algorithm using opposition-based learning

Ahmed A. Ewees[a,b,*], Mohamed Abd Elaziz[c], Essam H. Houssein[d]

[a] University of Bisha, Saudi Arabia
[b] Department of Computer, Damietta University, Egypt
[c] Department of Mathematics, Faculty of Science, Zagazig University, Egypt
[d] Faculty of Computers and Information, Minia University, Egypt

ARTICLE INFO

Article history:
Received 9 November 2017
Revised 9 June 2018
Accepted 10 June 2018
Available online 15 June 2018

Keywords:
Grasshopper optimization algorithm
Opposition-based learning
Benchmark functions
Engineering problems optimization

ABSTRACT

This paper proposes an improved version of the grasshopper optimization algorithm (GOA) based on the opposition-based learning (OBL) strategy called OBLGOA for solving benchmark optimization functions and engineering problems. The proposed OBLGOA algorithm consists of two stages: the first stage generates an initial population and its opposite using the OBL strategy; and the second stage uses the OBL as an additional phase to update the GOA population in each iteration. However, the OBL is applied to only half of the solutions to reduce the time complexity. To investigate the performance of the proposed OBLGOA, six sets of experiment series are performed, and they include twenty-three benchmark functions and four engineering problems. The experiments revealed that the results of the proposed algorithm were superior to those of ten well-known algorithms in this domain. Eventually, the obtained results proved that the OBLGOA algorithm can provide competitive results for optimization engineering problems compared with state-of-the-art algorithms.

1. Introduction

Optimization problems have received increased attention in recent years because of their ubiquitous nature, with such problems observed in fields ranging from engineering applications and computer science to finance and decision-making. In nearly all applications in engineering and industry, optimization is required to minimize the cost and energy consumption or to maximize the profits, output, performance, and efficiency (Koziel & Yang, 2011). However, to optimize real problems, various difficulties, such as multiple-objectives (El Aziz, Ewees, Hassanien, Mudhsh & Xiong, 2018b; Marler & Arora, 2004), constraints (Coello, 2002), uncertainties (Beyer & Sendhoff, 2007), local solutions (Knowles, Watson & Corne, 2001), parameter estimation (Oliva, Ewees, Aziz, Hassanien & Peréz-Cisneros, 2017), and deceptive global solutions (Deb & Goldberg, 1993), must be addressed. In general, mathematical optimization is the study of such planning and design problems using mathematical tools.

Optimization problems can be roughly divided into three categories: 1) constrained optimization, 2) unconstrained optimization, and 3) constrained engineering optimization. In this context, several methods, such as conventional optimization methods, have been proposed to solve these problems; however, these methods are mostly depreciated because of their drawbacks, such as local optima stagnation (Kelley, 1999; Vogl, Mangis, Rigler, Zink & Alkon, 1988). The main alternatives for designers are stochastic optimization techniques, which consider problems as black boxes and approximate the optimal designs (El Aziz, Ewees & Hassanien, 2016). These approaches initialize the optimization process with a set of random candidate solutions for a given problem and improve them over a pre–defined number of steps.

Bio-inspired optimization techniques have become very popular because of their applicability to a wide range of optimization problems (Ahirwal, Kumar & Singh, 2013). Bio-inspired optimization methods are basically motivated by natural phenomena, such as biological processes (e.g., reproduction, mutation, and interaction) or social behaviour (e.g., a flock of birds, school of fish, and intelligence of a swarm of bees). These methods have been exploited in various fields, such as image retrieval (El Aziz, Ewees & Hassanien, 2018a), signal processing (Ahirwal, Kumar & Singh, 2014), image segmentation (El Aziz, Ewees & Hassanien, 2017), antenna optimization (Bayraktar, Komurcu, Bossard & Werner, 2013), and feature selection (Ewees, El Aziz & Hassanien, 2017). Among the bio-inspired algorithms, a special class of algorithms has been

* Corresponding author.
E-mail addresses: eewees@du.edu.eg (A.A. Ewees), essam.halim@mu.edu.eg (E.H. Houssein).

https://doi.org/10.1016/j.eswa.2018.06.023
0957-4174/© 2018 Elsevier Ltd. All rights reserved.

developed based on swarm intelligence. Swarm intelligence (SI) is focused on the collective, emerging behaviour of multiple, interacting agents that follow simple rules (Fister, Yang, Fister, Brest & Fister, 2013). Each agent can be considered unintelligent, whereas the whole system of multiple agents can show self-organization behaviour and thus can behave like a type of collective intelligence. Examples include ant colony foraging, bird flocking, animal herding, bacterial growth, honey bee behaviour, and fish schooling (additional details can be found in the literature (Hassanien & Emary, 2016). Therefore, many algorithms have been developed by drawing inspiration from SI systems in nature.

In general, optimization and swarm-based intelligence, in particular, are becoming a basic foundation for many modern applications in a variety of disciplines. Because of its simplicity and flexibility, SI has been applied to engineering applications (Mavrovouniotis, Li & Yang, 2017). Various SI algorithms, such as ant colony optimization (Maniezzo, 1992), particle swarm optimization (Eberhart & Kennedy, 1995), artificial bee colony (Karaboga, 2005), bacterial foraging optimization (Passino, 2002), firefly algorithm (Yang, 2010a), artificial fish swarm optimization (Li, Shao & Qian, 2002), have been proposed over the past decade. Originally, SI algorithms were designed for stationary optimization problems.

Recently, many SI optimization techniques have been proposed to solve constrained optimization, unconstrained optimization, and engineering design problems (Yadav, Deep, Kim & Nagar, 2016), due to their simplicity of coding and consistency in performance. These techniques use swarm behaviour to solve optimization problems, and they use the concept of group intelligence along with individual intelligence (Kannan, Slochanal, Subbaraj & Padhy, 2004). Initially, these techniques were applied only to solve unconstrained optimization problems, such as particle swarm optimization (Chen, Zhang, Chung, Zhong, Wu & Shi, 2010), the chaotic krill herd algorithm (Wang, Guo, Gandomi, Hao & Wang, 2014), the chaotic bat algorithm (Gandomi & Yang, 2014), the firefly algorithm with chaos (Gandomi, Yang, Talatahari & Alavi, 2013a), and the binary bat algorithm (Mirjalili, Mirjalili & Yang, 2014). However, SI techniques were subsequently developed to solve constrained optimization problems; for example, Coello (2002) published a comprehensive survey on constraint-handling approaches for a large number of optimization algorithms, and Mezura-Montes and Coello (2011) presented the future scope and trends of constraint-handling mechanisms. In the same manner, a chaotic grey wolf optimization has been proposed (Kohli & Arora, 2017) as has a dragonfly algorithm (Mirjalili, 2016).

In the same context, a new SI optimization algorithm called the grasshopper optimization algorithm (GOA) has been applied in structural optimization (Saremi, Mirjalili & Lewis, 2017). GOA mimics the behaviour of grasshopper swarms in nature for solving optimization problems. However, the GOA presents several shortcomings that must be resolved, including 1) the premature convergence problem and 2) the slow movements and small steps of the grasshoppers. According to the aforementioned considerations and the results presented in Xu, Erdbrink and Krzhizhanovskaya (2015) which are the motivation of this study, the opposition-based learning (OBL) strategy is used to address the premature convergence and slow movement issues.

The OBL strategy presented in Tizhoosh (2005a), aims to boost the efficiency of metaheuristic optimization algorithms. OBL is relevant to the candidate solutions generated by a stochastic iteration scheme and their opposite solutions found in the opposite regions of the search space, which are closer to the global optimum than a random solution. The OBL strategy has been hybridized with many bio-inspired optimization methods and provides shorter expected distances to the global optimum then randomly sampled solution pairs (Ibrahim, Oliva, Ewees & Lu, 2017; Xu et al., 2015). For example, in Li, Chen, Zhang and Li (2016) the OBL was used to improve the cuckoo optimization algorithm called CH-EOBCCS; in this algorithm, the population was updated using the OBL strategy with a chaotic version of the Cuckoo Search (CS) algorithm. Fuqing et al. (Zhao, Zhang, Wang & Zhang, 2015a) proposed an improvement to the shuffled complex evolution algorithm based on OBL called SCE-OBL, and it was used to solve the permutation flow shop scheduling problem, in which the OBL was used to generate the initial population. Zhao et al. proposed the OBL artificial bee colony algorithm (OBL-ABC) to improve the exploration of the ABC algorithm (Zhao, Lv & Sun, 2015b). In this study, only the opposite solutions for the employed bee and onlooker bee were calculated. Particle swarm optimization (PSO) was improved by the OBL strategy as presented in Shang, Sun, Li, Liu, Zheng and Zhang (2015), in which the OBL was used in the updating stage to enhance the experiences of the particles and the common knowledge of the swarm. Gao et al. introduced the modified harmony search (HS) (Gao, Wang, Ovaska & Zenger, 2012) based on the OBL strategy to improve the mutation problem, and this algorithm was used to solve 24 global optimization benchmark problems and an optimal wind generator design. Radha et al. (Thangaraj, Pant, Chelliah & Abraham, 2012) presented an improvement for the differential evolution algorithm via the chaotic and OBL approaches, where the aim of the OBL was to generate the initial population while the chaotic approach was used to update the mutation parameter. Gong (2016) proposed a modified version of the fireworks algorithm, in which the OBL was used to generate the initial population and force the population to jump into new solutions. In Ahandani and Alavi-Rad (2015), four versions of the shuffled frog learning (SFL) algorithm were proposed, the OBL was used in the initialization stage, and the difference among the four versions in generating the jumping stage was evaluated. In the optimization problems, the strategy of simultaneously examining a candidate and its opposite solution is implemented to accelerate the convergence rate toward a globally optimal solution.

In this study, the modified GOA algorithm consists of two stages: in the first stage, the GOA generates a random population and then calculates the fitness function for each solution, whereas in the second stage, the solutions are updated using the classical GOA and then the OBL strategy is applied to half the population (to reduce the time complexity). Subsequently, the population is updated according to the best solutions from the union of current solutions and their opposites. The modified GOA algorithm has been tested through a set of experimental series conducted to evaluate the performance of the proposed algorithm using 23 benchmark problems (Suganthan, Hansen, Liang, Deb, Chen, Auger et al., 2005), and 4 optimization engineering problems, namely, the welded beam design problem, tension/compression spring design problem, three-bar truss design problem, and pressure vessel design problem. The experimental results compared with those of other similar algorithms revealed that the OBLGOA obtains superior results in terms of performance and efficacy.

The remainder of this paper is organized as follows: the basic concept of the GOA and the OBL strategy are introduced in Section 2. The improved GOA based on OBL (OBLGOA) is presented in detail in Section 3. Then, the OBLGOA is validated by a number of experiments, and the results and discussion are given in Section 4. Finally, we discuss relevant issues and conclude the paper in the last section.

2. Materials and methods

In this section, the optimization problem, GOA, and OBL are explained in the following subsections.

2.1. Problem definition

The mathematical definition of the optimization problem is introduced in the following equations, which can be used to write most optimization problems in the generic form as follows (Yang, 2010b):

$$\min f(x)\, x = \{x_1, \dots x_k\} \in S \tag{1}$$

$$subjectto: g_j(x) \geq 0, \;\; (j = 1, 2, \dots, J) \tag{2}$$

$$h_K(x) = 0, \;\; k = 1, 2, \dots, K \tag{3}$$

$$L_i \leq x_i \leq U_i, \; i = 1, 2, \dots, n \tag{4}$$

where S represents the solution space; $g_j(x)$ and $h_k(x)$ represent the inequality and equality constraints, respectively; J is the number of inequality constraints; K is the number of equality constraints; and $[L_i, U_i]$ represents the boundaries of the ith variable.

If no constraints are imposed on the problem with $J = 0$ and $K = 0$, then the optimization is said to be an unconstrained problem, whereas when $J > 0$ or $K > 0$, , the problem is said to be a constrained problem.

2.2. Grasshopper optimization algorithm

The authors in Saremi et al. (2017) proposed the GOA swarm algorithm, which emulates the behaviour of grasshopper insects. These insects are a type of pest that affects crop production and agriculture, and their life cycle consists of three phases: egg, nymph and adulthood (Rogers, Matheson, Despland, Dodgson, Burrows & Simpson, 2003). In the nymph phase, the main characteristics of grasshopper movement include jumping and moving in rolling cylinders (with small steps and slow movements), and they eat vegetation found in their path. In the adulthood phase, grasshoppers migrate a long distance in a swarm (with long-range and abrupt movements).

These behaviours are mathematically formulated by considering the position of the grasshopper (x_i) as follows (Rogers et al., 2003; Saremi et al., 2017):

$$x_i = S_i + G_i + A_i, \qquad i = 1, 2, \dots, N \tag{5}$$

where S_i is social interaction of the ith grasshopper and it is defined as:

$$S_i = \sum_{j=1, i \neq j}^{N} s(d_{ij})\widehat{d_{ij}}, \qquad d_{ij} = |x_i - x_j| \tag{6}$$

where d_{ij} represents the distance between the ith and jth Grasshoppers; while s represents the strength of social forces function that defined as:

$$s(y) = fe^{\frac{-y}{l}} - e^{-y} \tag{7}$$

where f and l represent the intensity of attraction and the attractive length scale. In Eq. (5), the G_i and A_i represent the gravity force and wind advection for the ith grasshopper, respectively and are defined as:

$$G_i = -g\widehat{e}_g, \qquad A_i = u\widehat{e}_w \tag{8}$$

where g and u represent the gravitational constant and a constant drift; while e_g and e_w represent the unity vector towards the center of the earth and the direction of the wind, respectively.

However, Eq. (5) cannot be applied directly to find the solution of the optimization problem, so, the authors in Saremi et al. (2017) reformulated it as follows:

$$x_i = c\left(\sum_{j=1, i \neq j}^{N} c\frac{u - l}{2} s(|x_j - x_i|)\frac{x_j - x_i}{d_{ij}}\right) + \widehat{T_d} \tag{9}$$

where u and l represent the upper bound and lower bound of the search space, respectively; T_d represents the value of the best solution found thus far; and s is defined as in Eq. (7). However, in Eq. (9) gravity is not considered, and the wind direction is always considered to be toward $\widehat{T_d}$. Here c is a decreasing coefficient to shrink the comfort zone, repulsion zone, and attraction zone as follows:

$$c = c_{\max} - t\,\frac{c_{\max} - c_{\min}}{t_{\max}} \tag{10}$$

where $c_{\max}$ and $c_{\min}$ represent the maximum value (equal to 1) and minimum value (equal to 0.00001) of c, respectively; t refers to the current iteration, and $t_{\max}$ is the maximum number of iterations.

Finally, the pseudo code of the GOA is given in Algorithm 1,

Algorithm 1 Grasshopper optimization algorithm (GOA) (Saremi et al., 2017).

```
1:  Initialize the value of the parameters such as population size ($N$), $c_{\max}, c_{\min}$,
    and maximum number of iteration ($t_{\max}$)
2:  Generate a random population ($X$)
3:  Set the current iteration $t = 1$
4:  while ($t < t_{\max}$) do
5:      Compute the fitness function $f$
6:      Select the best solution $\widehat{T_d}$
7:      Update the value of $c$ using Eq.(10)
8:      for $i = 1 : N$ do
9:          Normalize the distance between the solutions in $X$ in the interval [1,4].
10:         Update $x_i \in X$ using Eq.(9)
11:     end for
12:     $t = t + 1$
13: end while
14: Return $\widehat{T_d}$.
```

where the GOA starts by generating a random population X of size N. The next step is to calculate the fitness function for each solution $x_i, i = 1, \dots, N$, and the best solution, $\widehat{T_d}$, is then selected according to the best fitness function. Thereafter, for each x_i, the following two steps are performed: 1) normalize the distance between the solutions in X to [1,4]; and 2) update the current solution xi using Eq. (4). The next step is to update the current iteration and repeat the previous steps until the stopping conditions are achieved.

2.3. Opposition-based learning

The OBL strategy is used to define the opposite solution to the current solution, and it then uses the value of the fitness function (f) to determine whether the opposite is better than the current solution. The basic definition of the OBL was proposed in Tizhoosh (2005b), by assuming the opposite value $\bar{x}$ for the real value $x \in [u, l]$, which can be calculated as follows:

$$\bar{x} = u + l - x \tag{11}$$

This definition can be generalized to n–dimensions by using the following equation:

$$\bar{x}_i = u_i + l_i - x_i, \qquad i = 1, 2, \dots, N \tag{12}$$

where $\bar{x} \in R^n$ is the opposite vector from the real vector $x \in R^n$. In addition, through the optimization process, the two solutions (x and $\bar{x}$) are compared, and the best of these solutions is stored, while the other is removed by comparing the fitness function. For example, if $f(x) \leq f(\bar{x})$ (for minimization), then x is saved; otherwise, $\bar{x}$ is stored.

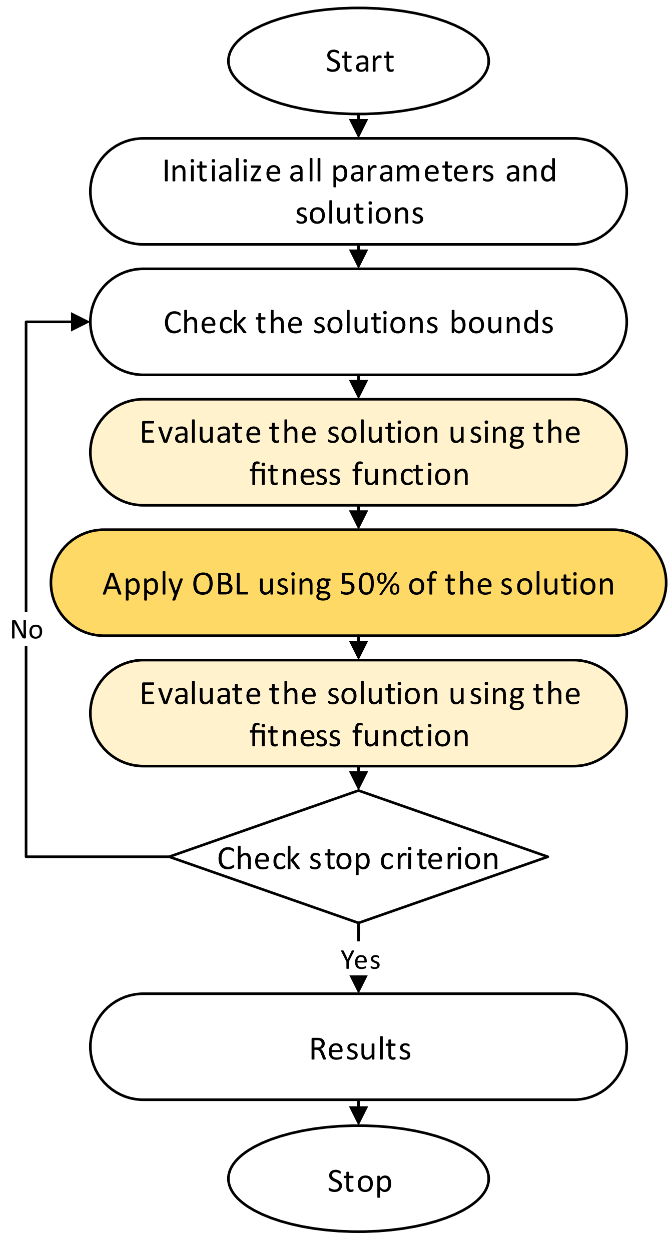
Start
Initialize all parameters and solutions
Check the solutions bounds
Evaluate the solution using the fitness function
Apply OBL using 50% of the solution
Evaluate the solution using the fitness function
Check stop criterion
No
Yes
Results
Stop

Fig. 1. Structure of the proposed algorithm.

3. Proposed method

In this section, the structure of the proposed algorithm is explained to improve the performance of the GOA algorithm. In this context, the GOA is modified by combining its original structure with the OBL strategy to introduce the ability to deeply explore the search domain and rapidly reach the optimal value. The proposed algorithm is called OBLGOA, and its main steps are shown in Fig. 1.

The OBL strategy is applied after finishing the GOA's exploration phase to maintain 50% of the domain space that has been calculated by the GOA. This step allows the original domain space to reach the optimal value quickly and repair the out-of-range values. In general, the proposed OBLGOA consists of two stages: 1) the initial stage and 2) the updating stage. The description of these stages is introduced in the following subsections.

Initial stage:

In this stage, the OBLGOA algorithm begins by determining the initial parameter values of the GOA; then, the GOA randomly creates a population of size N in dimension Dim. The OBL is used to calculate the opposite solution for each solution in the population, and the fitness function is calculated for each solution in X and its $\bar{X}$. The final step in the initial stage is to select the best N solutions from the union of the two populations (X and $\bar{X}$).

Updating stage:

In this stage, each solution is updated according to Eq. (9) and evaluated using the fitness function. Subsequently, the best solution and fitness value are saved. At the same time, the OBL technique receives the updated solutions from the GOA and selects a number of them (i.e., 50%) by calculating the opposite population for this part. In this study, the solutions in this part are the solutions that correspond to the best 50% according to the fitness functions. Afterward, the result is reevaluated by the fitness function; if the fitness value is better than the current value, the OBLGOA updates the population with this value in the next iteration. This mechanism is repeated until the stop condition is met (here, OBLGOA uses the maximum number of iterations as a stop criterion).

4. Experiments

To evaluate the performance of the proposed method, six experiments were performed. In the first experiment, the influence of selecting a different ratio from the OBL population to calculate the opposite solutions is studied. The second and third experiments show the influence of the maximum number of iterations and the difference in the population size, respectively. The fourth experiment compares the traditional GOA with the OBLGOA using twenty–three various benchmark functions. In the fifth experiment, the OBLGOA is compared with ten metaheuristics algorithms using twenty-three benchmark functions. Finally, the sixth experiment evaluates the performance of the OBLGOA in solving engineering problems. We also applied 30 independent runs to calculate the statistical results of the algorithms.

4.1. Performance measures

- Average value:
  The average value is the mean of the best values acquired by an algorithm for different runs, which can be calculated as follows:

$$\mu_F = \frac{1}{N_r}\sum_{i=1}^{N_r} F_i^* \tag{13}$$

  where N_r is the number of different runs and F_i^* is a best value.
- Best value:
  the best value is the best value that reached by an algorithm in different runs.

$$Best = \min_{1\le i\le N_r} F_i^* \tag{14}$$

- Worst value:
  The Worst value is the worst value that reached by an algorithm in different runs.

$$Worst = \max_{1\le i\le N_r} F_i^* \tag{15}$$

- Standard deviation:
  The standard deviation is calculated to test if an algorithm can reach the same best value in different runs and test the repeatability of the results of an algorithm

$$STD_F = \sqrt{\frac{1}{N_r-1}\sum_{i=1}^{N_r}(F_i-\mu_F)^2} \tag{16}$$

Table 1
A Description of the benchmark optimization functions.

ID	Equation	Lower	Upper	Dimension	Type	$f_{\min}$
F1	$f(x)=\sum_{i=1}^{n}x_i^2$	−100	100	10	Unimodal	0
F2	$f(x)=\sum_{i=1}^{n}\lvert x_i\rvert+\Pi_{i=1}^{n}\lvert x_i\rvert$	−10	10	10	Unimodal	0
F3	$f(x)=\sum_{i=1}^{n}(\sum_{j-1}^{i}x_i)^2$	−100	100	10	Unimodal	0
F4	$f(x)=max_i\{\lvert x_i\rvert,1\leqslant i\leqslant n\}$	−100	100	10	Unimodal	0
F5	$f(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	−30	30	10	Unimodal	0
F6	$f(x)=\sum_{i=1}^{n}([x_i+0.5])^2$	−100	100	10	Unimodal	0
F7	$f(x)=\sum_{i=1}^{n}ix_i^4+random[0,1]$	−1.28	1.28	10	Unimodal	0
F8	$f(x)=\sum_{i=1}^{n}-x_i sin(\sqrt{\lvert x_i\rvert})$	−500	500	10	Multimodal	418.9829x5
F9	$f(x)=\sum_{i=1}^{n}[x_i^2-10cos(2\pi x_i)+10]$	−5.12	5.12	10	Multimodal	0
F10	$f(x)=-20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2})-exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))+20+e$	−32	32	10	Multimodal	0
F11	$f(x)=\frac{1}{4000}\sum_{i=1}^{n}x_i^2-\Pi_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	−600	600	10	Multimodal	0
F12	$f(x)=\frac{\pi}{n}\{10sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10sin^2(\pi y_{i+1})]+(y_n-1)^2\}+\sum_{i=1}^{n}u(x_i,10,100,4)$	−50	50	10	Multimodal	0
	$u(x_i,a,k,m)=\begin{cases}k(x_i-a)^m, & x_i>a\\ 0, & -a\leqslant x_i\leqslant a\\ k(-x_i-a)^m, & x_i<-a\end{cases}$					
F13	$f(x)=0.1\{sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+sin^2(3\pi x_i+1)]+(x_n-1)^2[1+sin^2(2\pi x_n)]\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	−50	50	10	Multimodal	0
F14	$f(x)=(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{i=1}^{2}(x_i-a_{ij})^6})^-1$	−65.536	65.536	2	Multimodal	1
F15	$f(x)=(\sum_{i=1}^{11}[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}]^2$	−5	5	4	Multimodal	0.00030
F16	$f(x)=(4x_1^2-2.1x_1^4+\frac{1}{3}x_1^6+x_1x_2-x_2^2+4x_2^4$	−5	5	2	Multimodal	-1.0316
F17	$f(x)=(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6)^2+10(1-\frac{1}{8\pi})cosx_1+10$	−5	5	2	Multimodal	0.398
F18	$f(x)=[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)]$	−2	2	2	Multimodal	3
	$\times[30+(2x_1-3x_2)\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)]$					
F19	$f(x)=-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{3}a_{ij}(x_j-p_{ij})^2)$	1	3	3	Multimodal	-3.86
F20	$f(x)=-\sum_{i=1}^{4}c_i\exp(-\sum_{j=1}^{6}a_{ij}(x_j-p_{ij})^2)$	0	1	6	Multimodal	-3.32
F21	$f(x)=-\sum_{i=1}^{5}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	0	10	4	Multimodal	-10.1532
F22	$f(x)=-\sum_{i=1}^{7}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	0	10	4	Multimodal	-10.4028
F23	$f(x)=-\sum_{i=1}^{10}[(X-a_i)(X-a_i)^T+c_i]^{-1}$	0	10	4	Multimodal	-10.5363

Table 2
The parameter settings.

Algorithm	Value
GOA	$c_{\max} = 1$
	$c_{\min} = 0.00004$
OBL	OBL ratio = 50% of population

- Success rate:
  It checks the reliability of an algorithm by calculating the ratio of the number of times that an algorithm reached the required value-to-reach (*VTR*) and it is calculated as:

$$SR = \left(\frac{Number\ of\ times\ reached\ VTR}{N_r} \right) \times 100 \tag{17}$$

- Time average:
  The time average is used to calculate the time complexity of an algorithm over multiple runs. In this paper, this value is measured in seconds.

In addition, convergence curves are illustrated to show the behavior of each algorithm in order to reach the optimal value. As well as, Wilcoxon rank sum test is used as a statistical measure to reveal the significant difference between OBLGOA and GOA.

4.2. Functions description

A description of the twenty-three functions is given in Table 1, and these functions correspond to two types: multimodal and unimodal (Saremi et al., 2017). The unimodal functions have only one extreme point in their domain and are used to assess the convergence speed of the algorithms, whereas the multimodal functions have more than one extreme point and are used to assess the performance of the algorithm to skip the local point.

4.3. Parameter settings

The following parameter settings for OBLGOA and GOA are used in all experiments as shown in Table 2 except for experiment series 1 (the OBL ratio parameter is tested using 25%, 50%, and 75% of the population to evaluate the effects of these ratios on updating the solutions).

4.4. Experiment series 1: Influence of the OBL ratio

In this experiment, the effect of the OBL solution's ratio is studied by applying fifteen functions (F1-F15), in which the maximum number of iterations (Maxiter) is fixed at 150 iterations, the VTR = 10E-5, and the population size is set to 50. The ratio of the OBL solution is tested over 25, 50, and 75. The results of this experiment are listed in Table 3.

Table 3, shows that the results for OBL–50 were superior to all results for OBL–25, whereas OBL–75 outperformed OBL–50 in only 5 functions in terms of the fitness value. In addition, OBL-50 had the best average computational time at 65.53 s, followed by OBL–75 at 70.34 s and OBL–25 at 72.39 s. Moreover, the SRs of OBL–50 and OBL–75 are equal and outperformed OBL–25 in most of the experiments. From the above results, we can conclude that the OBL ratio has a significant effect when it is greater than or equal to 50, and this effect is decreased slightly if this ratio is decreased. Therefore, we used OBL-50 as the default value in the following experiments.

4.5. Experiment series 2: influence of the maximum number of iterations

To study the effect of the Maxiter on the performance of the proposed OBLGOA, the same parameter settings as in the experiment series 1 were used. However, the Maxiter was tested on a different set of four numbers (100, 200, 300 and 500), the population size was fixed at 50 and 10 dimensions, and the VTR was set to 10E−5. The comparison results are provided in Table 4. which shows that the performance of the proposed OBLGOA is better than the performance of the GOA for all performance measures (time, fitness function, and SR), although the GOA is slightly faster than the OBLGOA at 200–500 iterations. Additionally, the SR for both algorithms is increased when exceeding the number of iterations, but in all experiments, the SR for the OBLGOA is higher than the SR for the GOA by 52%, 49%, 40%, and 31%. Therefore, we can conclude that the Maxiter has a large influence on the performance of both algorithms. However, the change in SR from 300 iterations to 500 iterations is 4.29% and 11.79% for the OBLGOA and GOA, respectively, which indicates that the GOA requires more iterations than the OBLGOA.

4.6. Experiment series 3: influence of the population size

The effect of the size of the population is studied in this experiment using only fifteen functions (F1-F15), in which the Maxiter is fixed at 200 iterations and the VTR = 10E−5. However, the population size is set to 30, 50, 100 and 150. The comparison results under these settings are provided in Table 5, which shows that (as expected) when the population size is increased, the performance of both algorithms increases. Additionally, when the time is increased, the OBLGOA performance is better than that of the GOA at the four different sizes, although for a population size of 50, the GOA is faster. Moreover, the SR of the OBLGOA is higher than the SR of the GOA by 53%, 49%, 49% and 46% for populations of 30, 50, 100, and 150, respectively. This finding indicates that the GOA must increase the size of the population to enhance its SR since the difference between the SR for both algorithms decreases when the population size is increased. Additionally, the SR of the OBLGOA is enhanced very little for populations great than 50 (only 2% with pop = 100) and does not enhance for populations greater than 100.

4.7. Experiment series 4: comparison between OBLGOA and GOA

In this experiment, the performances of the OBLGOA and GOA algorithms are evaluated to determine the optimal solution for the functions illustrated in Table 1. The comparison results are given in Table 6 the size of the population is 50, the dimension is 10, and the Maxiter is set to 100. The results presented in this table show that in terms of the average fitness function values, the OBLGOA provides better results than the GOA for most functions, and the two algorithms provide nearly equivalent results for functions F16 and F20. However, the worst OBLGOA result is better than the best GOA result for certain functions, such as F1-F4, F7, F9-F11, and F13. Moreover, the stability of the proposed OBLGOA is higher than the stability of the original GOA, as illustrated by the values of the STD measure for both algorithms except for function F16. In addition, the OBLGOA generates an SR value that is 19% higher than that of the GOA (0.5855 versus 0.3913); however, both algorithms fail to provide solutions for functions F5, F6, F12 and F13.

In addition, Fig. 2 shows the convergence curve for the OBLGOA and GOA and indicates that the behaviour of the algorithms is changed according to the given functions at the iterations. For example, the convergence rate of the OBLGOA is better than that of the GOA for most functions except for F16-F20, and the convergence is the same as that of the original GOA is very slightly better than that of the OBLGOA.

To provide more evidence to prove the superiority of the proposed OBLGOA, a nonparametric Wilcoxon rank sum test is used, which determines if a significant difference occurs between the proposed algorithm and the GOA, with the significance level set

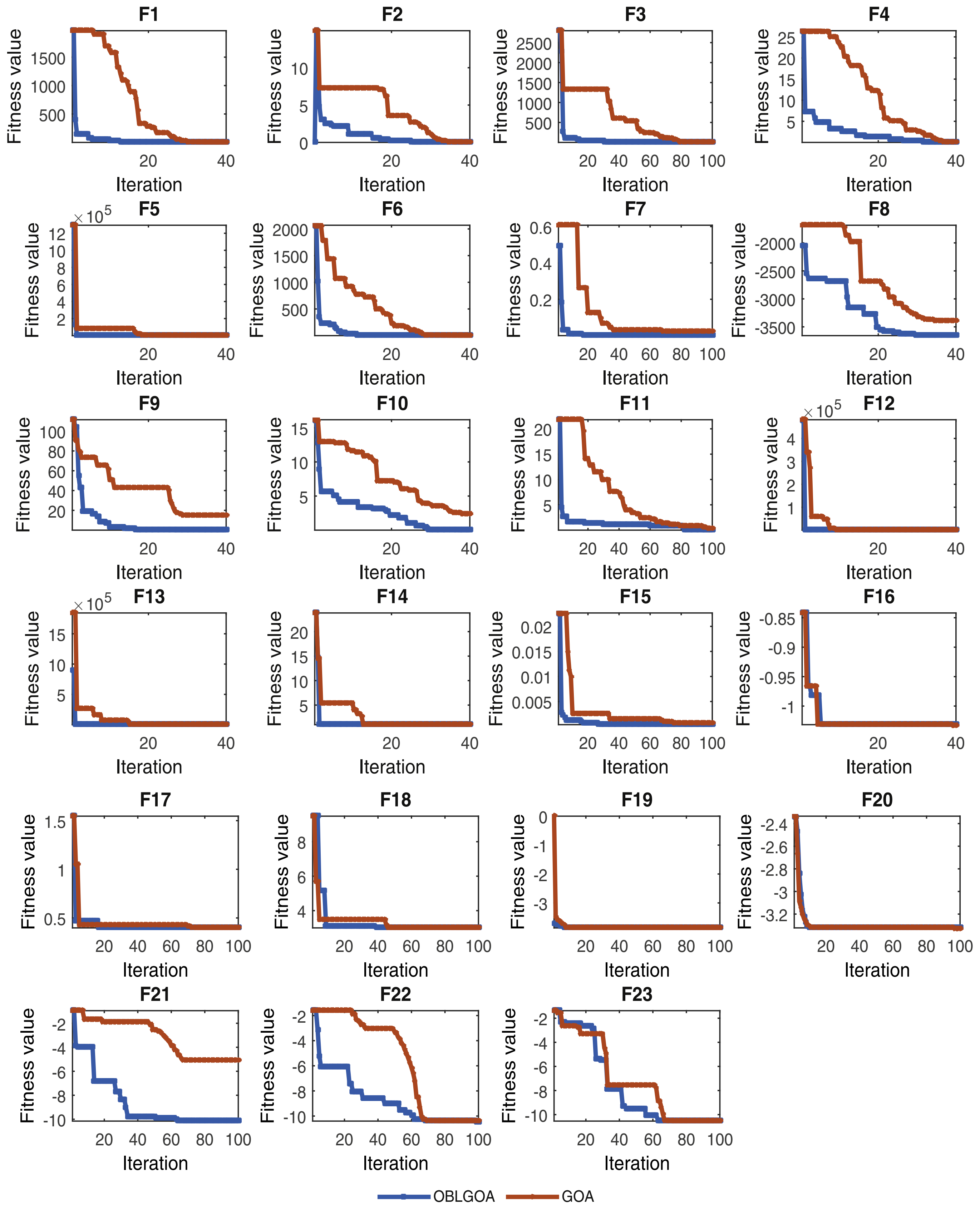

Fig. 2. Convergence curves for both algorithms overall functions.

Table 3
Effective of the variance of the OBL solution's ratio.

	OBL-25			OBL-50			OBL-75		
Func.	Time	Fitness	SR	Time	Fitness	SR	Time	Fitness	SR
F1	79.19	4.713E-04	100	66.079	6.001E-06	100	90.880	1.713E-05	100
F2	86.73	1.277E-02	71	65.983	8.716E-04	100	89.260	8.806E-04	100
F3	85.89	9.380E-04	100	47.738	6.2125E-04	100	53.861	6.0737E-04	100
F4	75.36	5.925E-03	71	87.853	1.160E-03	100	73.652	1.163E-03	100
F5	73.38	8.785E+00	0	76.038	8.774E+00	0	81.186	8.831E+00	0
F6	78.01	1.576E-01	0	74.532	6.618E-02	0	75.301	6.521E-02	0
F7	77.83	3.447E-04	100	75.563	2.828E-04	100	80.742	3.131E-04	100
F8	77.94	−2.622E+03	100	75.238	−2.615E+03	100	81.349	−2.987E+03	100
F9	86.54	7.586E-01	29	73.578	2.868E-06	100	79.543	2.800E-06	100
F10	85.432	1.396E-03	100	76.070	3.212E-04	100	77.565	1.316E-04	100
F11	80.33	4.005E-01	0	83.823	1.038E-04	100	71.975	9.222E-05	100
F12	76.38	3.153E-02	43	85.649	1.910E-02	50	74.116	1.922E-02	50
F13	73.99	2.4337E-02	0	46.918	1.7458E-02	0	76.870	1.7005E-02	0
F14	17.95	7.301E-01	0	18.086	8.066E-01	0	18.698	8.066E-01	0
F15	30.80	4.074E-03	57	29.879	1.316E-03	100	30.049	1.271E-03	100

Table 4
The fitness and VTR values of OBLGOA and GOA at Maxiter (100, 200, 300, 500).

F. No	Maxiter = 100						Maxiter = 200					
	OBLGOA			GOA			OBLGOA			GOA		
	CPU time	Fitness	SR	CPU time	Fitness	SR	CPU time	Fitness	SR	CPU time	Fitness	SR
F1	48.89	5.42E-05	100	50.34	1.97E-01	0	111.42	3.54E-09	100	108.23	1.51E-02	20
F2	47.46	1.73E-04	100	51.82	6.25E-02	20	102.75	1.98E-05	100	103.59	5.91E-02	60
F3	58.45	1.35E-03	100	59.59	2.91E+01	0	90.34	5.65E-04	100	88.59	9.88E-01	0
F4	48.79	3.50E-03	80	54.40	2.81E-01	0	89.29	6.94E-05	100	88.96	1.89E-01	0
F5	50.43	8.90E+00	0	55.90	1.36E+02	0	89.29	8.12E+00	0	88.85	8.22E+00	0
F6	47.01	7.65E-02	0	50.65	1.21E-01	0	89.35	1.44E-02	40	88.14	1.45E-02	40
F7	46.74	5.57E-04	100	50.92	1.11E-02	17	89.20	1.68E-04	100	88.45	1.67E-02	20
F8	46.58	−2.54E+03	100	45.88	−2.70E+03	100	89.74	−2.75E+03	100	89.02	−2.84E+03	100
F9	48.02	1.85E-05	100	46.00	3.30E+01	0	90.18	8.49E-09	100	88.34	2.49E+01	0
F10	52.82	1.56E-03	100	51.32	1.61E+00	0	100.57	2.09E-05	100	98.82	4.18E-02	60
F11	48.05	1.30E-04	80	49.53	3.95E-01	0	95.79	4.79E-05	80	93.44	2.87E-01	0
F12	52.72	2.76E-02	0	50.67	1.84E+00	0	91.62	1.09E-02	60	89.12	1.34E+00	0
F13	48.93	2.13E-02	0	46.54	3.53E-02	0	89.44	9.30E-03	20	88.56	2.25E-02	0
F14	11.35	1.39E+00	0	10.26	1.59E+00	0	22.49	9.98E-01	0	20.52	9.98E-01	0
F15	18.70	1.81E-03	100	18.56	9.23E-03	40	40.73	1.23E-03	100	37.68	8.70E-03	60
Avg.	45.00		64	46.16		11.8	85.48		73.3	84.02		24
F. No	Maxiter = 300						Maxiter = 500					
	OBLGOA			GOA			OBLGOA			GOA		
	CPU time	Fitness	SR	CPU time	Fitness	SR	CPU time	Fitness	SR	CPU time	Fitness	SR
F1	134.19	1.52E-09	100	136.02	1.37E-03	100	276.15	1.46E-09	100	263.47	2.29E-06	100
F2	142.56	5.88E-06	100	141.36	6.02E-03	60	244.33	2.72E-06	100	247.04	1.75E-03	100
F3	135.38	4.46E-04	100	133.72	1.21E-01	0	236.52	9.92E-05	100	231.39	4.66E-02	25
F4	134.04	4.29E-05	100	133.62	6.51E-02	20	225.81	7.73E-06	100	229.79	1.64E-02	60
F5	134.91	8.57E+00	0	133.51	5.75E+02	0	224.39	8.04E+00	0	221.16	3.93E+01	0
F6	135.02	8.20E-04	100	133.28	1.19E-03	100	250.15	9.33331E-06	100	239.76	1.61E-05	100
F7	133.43	1.78E-04	100	132.26	1.00E-02	60	258.94	1.11E-04	100	242.07	3.65E-03	100
F8	134.52	−2.79E+03	100	132.72	−2.80E+03	100	240.89	−2.69E+03	100	234.76	−2.77E+03	100
F9	134.22	3.07E-09	100	132.75	2.71E+01	0	225.29	3.82E-11	100	222.35	1.75E+01	0
F10	133.65	2.05E-05	100	132.62	1.13E-03	60	223.07	5.52E-06	100	221.86	1.04E-03	60
F11	145.98	4.42E-05	80	151.27	3.25E-01	0	224.51	6.99E-09	100	222.59	2.12E-01	0
F12	164.17	5.64E-03	100	161.21	3.04E-01	20	225.24	2.14E-04	100	222.46	6.77E-01	20
F13	169.34	1.97E-03	60	169.91	3.15E-03	60	224.33	2.05E-04	80	221.99	2.51E-04	80
F14	44.72	9.98E-01	0	41.92	9.98E-01	0	55.73	9.98E-01	0	50.61	9.98E-01	0
F15	57.44	1.14E-03	100	65.96	8.62E-03	60	92.62	1.12E-03	100	90.10	8.63E-03	60
Avg.	128.90		82.7	128.81		43	215.20		85.3	210.76		54

to 5%. The *pvalues* given in Table 7 show that a significant difference occurs between the two algorithms for most functions except for nine (F13–F15, F17–F19 and F21–F23).

4.8. Experiment series 5: comparison between OBLGOA and the other algorithms

In this experiment, the performance of the proposed OBLGOA is compared with the original GOA and nine other well-known algorithms: Multi-Verse Optimizer (MVO), Ant Lion Optimizer (ALO), Sine-Cosine Algorithm (SCA), Particle Swarm Optimization (PSO), Genetic Algorithm (GA), Bat Algorithm (BAT), Dragonfly Algorithm (DA), Mothflame Optimization (MFO) (Mirjalili, 2015), and Differential Evolution (DE). Table 8 lists the parameter settings of all algorithms used in this experiment.

Table 9 and Fig. 3 show the performance of the proposed algorithm against the compared algorithms. Table 9 shows that the OBLGOA achieved better performance over all other algorithms in terms of the fitness function value. For example, in terms of the average values for the unimodal functions (F1–F7), the proposed algorithm has the best value, and the same findings are observed

Table 5
The fitness and VTR values of OBLGOA and GOA at pop (30, 50, 100, 150).

F. No	Population = 30						Population = 50					
	OBLGOA			GOA			OBLGOA			GOA		
	CPU time	Fitness	SR	CPU time	Fitness	SR	CPU time	Fitness	SR	CPU time	Fitness	SR
F1	40.08	1.70E-08	100	42.81	1.72E-02	20	111.42	3.54E-09	100	108.23	1.51E-02	20
F2	35.64	2.52E-04	100	37.94	1.31E-02	50	102.75	1.98E-05	100	103.59	5.91E-02	60
F3	38.18	7.15E-04	100	40.50	5.65E+00	0	90.34	5.65E-04	100	88.59	9.88E-01	0
F4	38.49	2.93E-04	100	44.12	4.72E-01	0	89.29	6.94E-05	100	88.96	1.89E-01	0
F5	39.04	8.61E+00	0	44.11	8.79E+02	0	89.29	8.12E+00	0	88.85	8.22E+00	0
F6	35.77	3.24E-02	0	36.75	3.42E-02	0	89.35	1.44E-02	40	88.14	1.45E-02	40
F7	32.53	2.19E-04	100	35.15	2.47E-02	0	89.20	1.68E-04	100	88.45	1.67E-02	20
F8	33.53	−2.63E+03	100	32.02	−2.73E+03	100	89.74	−2.75E+03	100	89.02	−2.84E+03	100
F9	34.66	3.68E-07	67	32.30	2.89E+01	0	90.18	8.49E-09	100	88.34	2.49E+01	0
F10	32.65	3.14E-05	100	31.92	1.33E+00	0	100.57	2.09E-05	100	98.82	4.18E-02	60
F11	34.57	6.89E-02	67	32.36	2.49E-01	0	95.79	4.79E-05	80	93.44	2.87E-01	0
F12	32.95	1.62E-02	50	32.03	2.26E+00	0	91.62	1.09E-02	60	89.12	1.34E+00	0
F13	32.75	1.80E-02	20	34.82	4.25E-02	0	89.44	9.30E-03	20	88.56	2.25E-02	0
F14	9.87	2.97E+00	0	8.33	2.98E+00	0	22.49	9.98E-01	0	20.52	9.98E-01	0
F15	13.64	1.30E-03	100	13.37	5.25E-03	33	40.73	1.23E-03	100	37.68	8.70E-03	60
Avg.	32.29		67	33.24		13.56	85.48		73.33	84.02		24.00
F. No	Population = 100						Population = 150					
	OBLGOA			GOA			OBLGOA			GOA		
	CPU time	Fitness	SR	CPU time	Fitness	SR	CPU time	Fitness	SR	CPU time	Fitness	SR
F1	358.11	7.94E-09	100	358.00	1.56E-02	20	803.59	3.98E-09	100	872.53	1.56E-02	20
F2	384.10	1.16E-05	100	420.21	1.55E-02	67	801.45	1.04E-05	100	869.13	8.27E-03	66.7
F3	400.13	3.74E-04	100	421.73	1.06E+00	0	801.70	1.38E-06	100	867.00	1.86E-01	0
F4	421.96	5.21E-05	100	408.77	2.45E-01	0	825.65	5.16E-05	100	922.61	1.23E-01	0
F5	573.33	8.52E+00	0	553.72	3.45E+01	0	793.96	8.53E+00	0	868.25	1.27E+03	0
F6	414.32	1.60E-02	33	435.59	1.76E-02	33	795.43	1.96E-02	33	871.01	1.80E-02	33.3
F7	459.14	2.33E-04	100	555.86	8.76E-03	67	812.20	1.41E-04	100	915.54	7.32E-03	66.7
F8	391.68	−2.78E+03	100	377.55	−3.18E+03	100	948.41	−2.79E+03	100	930.15	−3.33E+03	100
F9	355.74	8.87E-09	100	354.73	2.16E+01	0	885.06	8.59E-09	100	888.54	3.25E+01	0
F10	355.01	5.95E-05	100	354.54	1.00E+00	0	794.07	2.05E-05	100	797.47	2.27E-01	66.7
F11	354.44	1.54E-05	100	355.62	3.93E-01	0	830.44	1.42E-05	100	841.58	2.06E-01	0
F12	354.23	8.85E-03	67	354.84	2.66E-01	33	797.37	2.38E-03	100	813.42	2.12E-01	33.3
F13	355.52	7.38E-03	33	355.66	1.62E-02	20	1014.43	8.38E-03	33.3	904.23	1.43E-02	20.0
F14	80.22	9.98E-01	0	77.39	9.98E-01	0	171.65	9.98E-01	0	165.87	9.98E-01	0
F15	143.41	1.25E-03	100	143.33	1.38E-02	67	319.39	1.29E-03	100	318.52	1.46E-02	66.7
Avg.	360.09		76	368.50		27.11	759.65		77.78	789.72		31.56

Table 6
The comparison of optimization results obtained over all benchmark functions; VTR=10E-5.

No.	OBLGOA					GOA				
	Avg.	Best	Worst	STD	SR	Avg.	Best	Worst	STD	SR
F1	5.42E-05	3.78E-07	1.95E-04	7.26E-05	100	1.97E-01	7.32E-02	3.68E-01	1.03E-01	0
F2	1.73E-04	9.55E-05	2.24E-04	4.42E-05	100	6.25E-02	8.10E-03	1.96E-01	6.79E-02	20
F3	1.35E-03	1.05E-04	2.79E-03	1.05E-03	100	2.91E+01	4.06E+00	8.32E+01	2.87E+01	0
F4	3.50E-03	1.30E-04	1.64E-02	6.45E-03	80	2.81E-01	1.49E-01	5.18E-01	1.25E-01	0
F5	8.90E+00	8.87E+00	8.95E+00	2.84E-02	0	1.36E+02	1.00E+01	5.63E+02	2.15E+02	0
F6	7.65E-02	2.15E-02	2.10E-01	6.87E-02	0	1.21E-01	2.55E-02	3.34E-01	1.09E-01	0
F7	5.57E-04	2.78E-04	7.20E-04	1.78E-04	100	1.11E-02	6.31E-03	2.03E-02	4.88E-03	40
F8	−2.54E+03	−2.77E+03	−2.10E+03	3.11E+02	100	−2.70E+03	−2.86E+03	−2.53E+03	1.35E+02	100
F9	1.85E-05	7.72E-06	3.32E-05	8.61E-06	100	3.30E+01	2.19E+01	4.97E+01	9.19E+00	0
F10	1.56E-03	3.61E-04	2.13E-03	6.99E-04	100	1.61E+00	5.99E-01	2.64E+00	8.48E-01	0
F11	1.30E-04	1.98E-06	3.35E-04	1.37E-04	80	3.95E-01	2.07E-01	5.86E-01	1.47E-01	0
F12	2.76E-02	1.12E-02	6.17E-02	1.78E-02	0	1.84E+00	9.08E-02	5.85E+00	2.15E+00	0
F13	2.13E-02	1.52E-02	2.67E-02	5.09E-03	0	3.53E-02	1.88E-02	5.06E-02	1.26E-02	0
F14	1.39E+00	9.98E-01	2.98E+00	7.94E-01	100	1.59E+00	9.98E-01	2.98E+00	7.94E-01	100
F15	1.81E-03	1.26E-03	2.15E-03	3.64E-04	100	5.37E-03	1.47E-03	2.04E-02	7.52E-03	80
F16	−1.03	−1.03	−1.03	2.38E-4	100	−1.03	−1.03	−1.03	2.88E-10	100
F17	3.98E-1	3.98E-1	3.98E-1	4.27E-10	100	3.98E-1	3.98E-1	3.98E-1	4.98E-10	100
F18	3.00	3.00	3.00	4.08E-9	100	3.00	3.00	3.00	2.55E-9	100
F19	−3.86	−3.8608	−3.8627	7.50E-4	100	−3.85	−3.8616	−3.6841	3.20E-2	100
F20	−3.28	−3.32	−3.16	6.99E-2	100	−3.27	−3.32	−3.12	7.00E-2	100
F21	−1.02E+1	−1.02E+1	−1.02E+1	2.00E-6	100	−7.15	−1.02E+1	−2.63	3.50	100
F22	−1.04E+1	−1.04E+1	−1.04E+1	4.90E-5	100	−7.93	−2.75	−1.04E+1	3.35	100
F23	−1.05E+1	−1.05E+1	−1.05E+1	1.91E-5	100	−7.24	−1.05E+1	−2.42	3.58	100

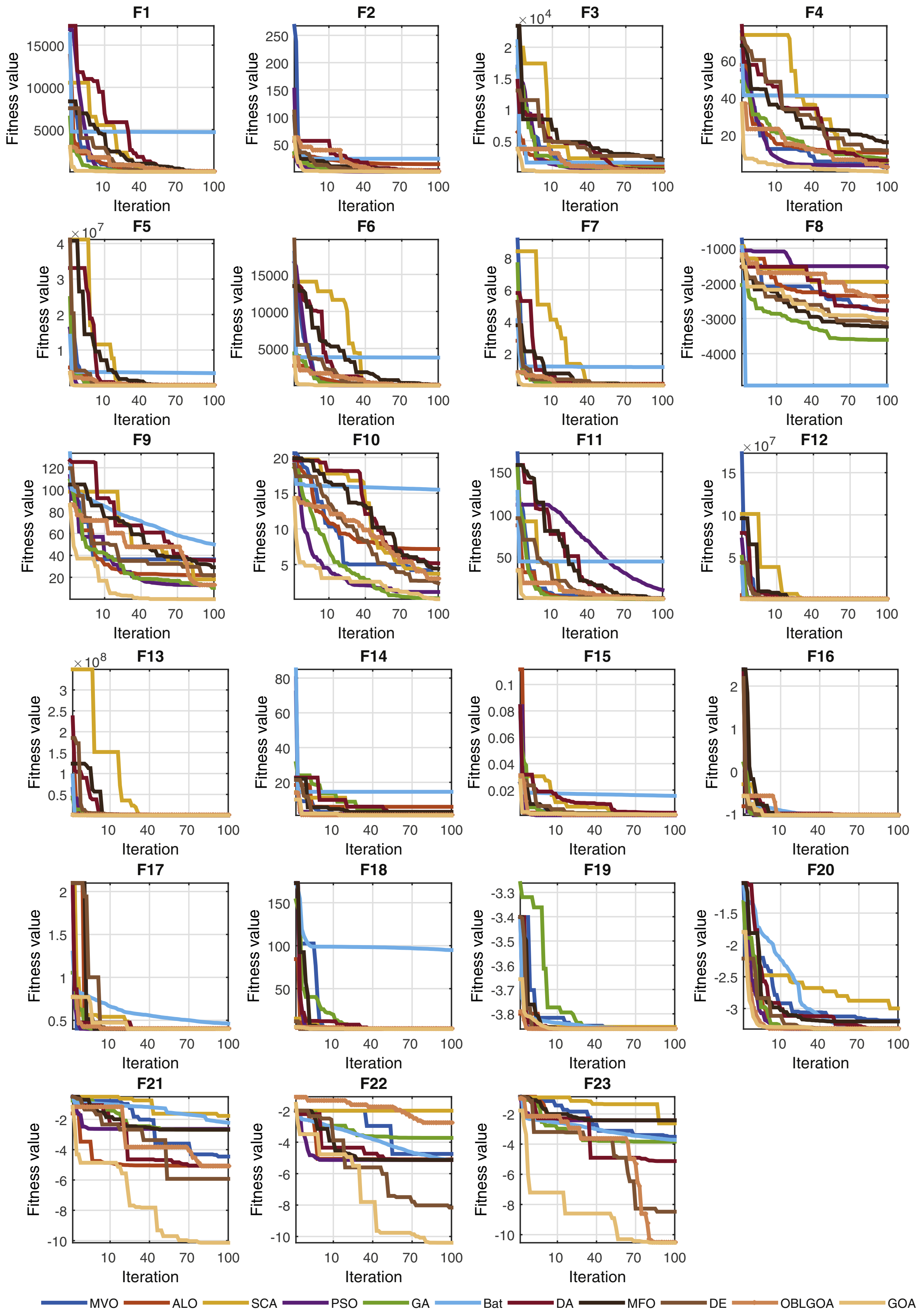

Fig. 3. Convergence curves for all algorithms.

Table 7
The results of Wilcoxon rank sum test.

Func. no	*p*-value	Func. no	*p*-value	Func. no	*p*-value	Func. no	*p*-value
F1	0.000	F7	0.000	F13	0.897	F19	0.48
F2	0.000	F8	0.000	F14	0.888	F20	0.018
F3	0.000	F9	0.000	F15	0.419	F21	0.327
F4	0.000	F10	0.000	F16	0.000	F22	0.286
F5	0.000	F11	0.000	F17	0.953	F23	0.096
F6	0.000	F12	0.000	F18	0.519		

Table 8
The parameter settings for the algorithms.

Algorithm	Parameters values
MVO	$WEP_{\min} = 0.2,\ WEP_{\max} = 1$
ALO	$Dim, lower\ bound, and\ Upper\ bound = according\ to\ each\ problem$
SCA	$a = 2$
PSO	$wMax = 0.9, wMin = 0.2, C1 = 2, C2 = 2$
GA	$pc = 0.8, gamma = 0.2,\ pm = 0.3,\ mu = 0.02, beta = 8$
BAT	$Loudness = 0.5, Pulse\ rate = 0.5, Frequency\ minimum = 0, Frequency\ \max = 2$
DA	$Delta\ \max = 0.1$
MFO	$t \in [-1, 1], b = 1$
DE	$pCR = 0.2,\ \beta_{\min} = 0.2,\ \beta_{\max} = 0.8$

for functions F9–F12 and F21–F23, which belong to the multimodal category. However, for functions F8 and F15, the average of the proposed algorithm does not reach the optimal value, whereas most of the other algorithms reach the optimum. Additionally, for function F14, the GA has the best average among the algorithms, followed by the DE algorithm, and the proposed algorithm obtains the third-best results. Moreover, most of the compared algorithms present the same performance for function F20, although the DE presents the best results.

In terms of the standard deviation, the proposed algorithm achieved a better performance for most of the functions except for the following: F8 and F13–F20.

Fig. 3 shows the convergence curves for all functions based on the compared algorithms. This figure shows that the proposed OBLGOA obtained the best convergence along all fitness functions except for F8 and F20. Moreover, the OBL strategy has a larger effect on the convergence of the proposed algorithm, especially compared with the GOA, and this trend is obvious in functions F9, F21, and F22. In addition, the convergence of the OBLGOA for the first ten iterations is better than that of the other algorithms, which indicates the high influence of the OBL on the initial population and the updated population.

These results indicate that the proposed OBLGOA can efficiently identify the optimal global values of 23 benchmark functions at a fast convergence rate and the use of OBL to update the solutions effectively helps the GOA converge to the optimal values.

4.9. Experiment series 6: engineering problems

In this section, we test the proposed algorithm using four engineering optimization problems: the welded beam design problem, tension/compression spring design problem, three-bar truss design problem, and pressure vessel design problem. The following subsections show the results of the OBLGOA compared with the results of the state-of-art methods.

4.9.1. Welded beam design problem

Estimating the optimal value for the four parameters that minimize the cost of the fabrication is the aim of the welded beam design problem as shown in Fig. 4. The parameters are called the thickness of the bar (b), length of an attached part of the bar (l), height of the bar (t) and thickness of the weld (h). The mathematical definition of the welded beam design problem is given by the following equation:

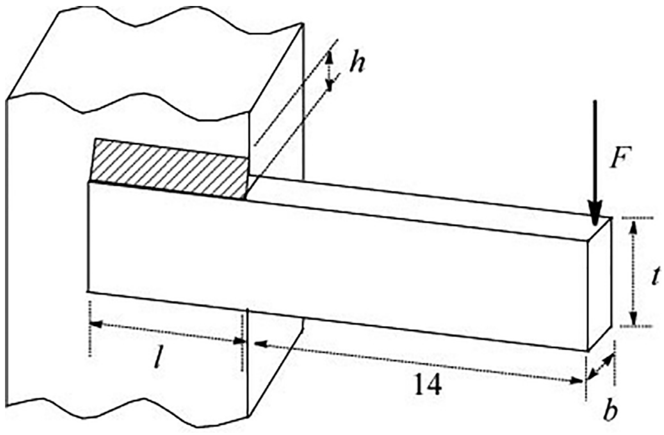

Fig. 4. Parameters of the welded beam design problem.

$$\begin{aligned}
&\textit{Consider} && \vec{x} = [x_1\ x_2\ x_3\ x_4] = [h\ l\ t\ b],\\
&\textit{Minimize} && f(\vec{x}) = 1.10471x_1^2x_2 + 0.04811x_3x_4(14.0 + x_2),\\
&\textit{Subject to} && g_1(\vec{x}) = \tau(\vec{x}) - \tau_{max} \leqslant 0,\\
& && g_2(\vec{x}) = \sigma(\vec{x}) - \sigma_{max} \leqslant 0,\\
& && g_3(\vec{x}) = \delta(\vec{x}) - \delta_{max} \leqslant 0,\\
& && g_4(\vec{x}) = x_1 - x_4 \leqslant 0,\\
& && g_5(\vec{x}) = P - P_c(\vec{x}) \leqslant 0,\\
& && g_6(\vec{x}) = 0.125 - x_1 \leqslant 0,\\
& && g_7(\vec{x}) = 1.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2)\\
& && \quad - 5.0 \leqslant 0 \qquad (18)\\
&\textit{Variables range} && 0.1 \leqslant x_1 \leqslant 2,\\
& && 0.1 \leqslant x_2 \leqslant 10,\\
& && 0.1 \leqslant x_3 \leqslant 10,\\
& && 0.1 \leqslant x_4 \leqslant 2\\
&\textit{where} && \tau(\vec{x}) = \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2},\\
& && \tau' = \frac{p}{\sqrt{2x_1x_2}},\quad \tau'' = \frac{MR}{J},
\end{aligned}$$

Table 9
The comparison results of all algorithms over 23 functions.

Func.		OBLGOA	GOA	MVO	ALO	SCA	PSO	GA	BAT	DA	MFO	DE
F1	Average	3.00E-06	1.28E-01	3.95E-01	3.83E+01	7.16E+00	2.19E+00	1.90E-01	4.64E+03	2.66E+01	8.05E+00	4.00E-01
	Min	1.31E-08	2.68E-02	1.54E-01	1.79E-03	4.02E-02	8.39E-08	9.12E-03	9.39E+02	2.92E-02	1.36E+00	9.91E-02
	Max	5.46E-05	2.54E-01	8.16E-01	3.34E+02	4.58E+01	3.03E+01	9.31E-01	1.11E+04	1.97E+02	2.58E+01	1.04E+00
	STD	9.84E-06	6.79E-02	1.68E-01	8.01E+01	1.23E+01	5.50E+00	2.17E-01	2.27E+03	3.75E+01	6.02E+00	2.15E-01
F2	Average	9.38E-05	2.32E-01	1.99E-01	1.30E+01	1.03E-01	4.26E-03	1.15E-02	2.43E+01	2.65E+00	4.21E-01	8.42E-02
	Min	2.45E-05	8.75E-03	7.77E-02	1.34E+00	1.35E-03	8.64E-06	3.31E-04	1.07E+01	2.08E-01	1.39E-01	5.07E-02
	Max	2.62E-04	7.51E-01	4.85E-01	4.24E+01	2.89E-01	6.55E-02	4.73E-02	4.03E+01	6.48E+00	9.10E-01	1.62E-01
	STD	6.43E-05	2.04E-01	8.79E-02	1.01E+01	8.06E-02	1.16E-02	1.26E-02	8.08E+00	1.47E+00	2.04E-01	2.27E-02
F3	Average	2.97E-04	5.87E+00	2.86E+00	9.03E+02	2.46E+02	1.03E+02	5.57E+02	7.94E+03	3.21E+02	1.42E+03	2.08E+03
	Min	3.10E-07	4.37E-01	5.55E-01	4.32E+01	5.60E+00	8.85E-02	8.33E+01	1.46E+03	3.00E+01	1.12E+02	9.14E+02
	Max	1.60E-03	2.07E+01	1.71E+01	2.26E+03	3.05E+03	7.29E+02	1.76E+03	2.06E+04	1.16E+03	1.01E+04	3.47E+03
	STD	4.03E-04	5.09E+00	2.84E+00	6.06E+02	5.42E+02	1.52E+02	4.58E+02	4.60E+03	2.77E+02	2.13E+03	6.10E+02
F4	Average	3.36E-04	3.12E-01	5.08E-01	1.05E+01	4.34E+00	3.52E+00	4.95E+00	4.23E+01	6.79E+00	1.12E+01	8.39E+00
	Min	4.30E-05	7.89E-02	2.40E-01	2.32E-01	3.56E-01	6.14E-02	1.13E+00	2.47E+01	1.01E+00	4.51E+00	5.09E+00
	Max	7.23E-04	9.14E-01	1.29E+00	2.65E+01	1.74E+01	9.71E+00	2.04E+01	6.18E+01	1.94E+01	3.96E+01	1.34E+01
	STD	1.74E-04	2.01E-01	2.28E-01	5.98E+00	3.73E+00	2.35E+00	3.78E+00	1.19E+01	4.84E+00	7.00E+00	2.05E+00
F5	Average	8.80E+00	1.57E+02	3.40E+02	2.95E+03	4.01E+02	3.40E+01	1.22E+02	4.37E+06	4.79E+03	6.68E+02	1.65E+02
	Min	8.47E+00	8.26E+00	8.82E+00	2.94E-01	1.42E+01	8.14E-04	8.49E+00	4.10E+05	1.45E+01	4.82E+01	5.74E+01
	Max	8.93E+00	1.80E+03	2.84E+03	2.85E+04	3.24E+03	2.16E+02	1.04E+03	1.13E+07	6.66E+04	3.43E+03	2.88E+02
	STD	1.47E-01	3.48E+02	7.89E+02	5.57E+03	7.68E+02	5.75E+01	2.37E+02	2.98E+06	1.19E+04	8.27E+02	6.19E+01
F6	Average	5.22E-02	1.20E-01	3.88E-01	1.20E+01	3.20E+00	2.83E+00	4.50E-01	4.95E+03	2.08E+01	6.37E+00	3.16E-01
	Min	7.51E-03	2.44E-02	6.07E-02	1.76E-04	8.76E-01	2.01E-09	6.51E-05	8.05E+02	1.04E-01	7.33E-01	6.73E-02
	Max	1.29E-01	2.98E-01	1.01E+00	1.60E+02	1.81E+01	4.77E+01	3.05E+00	1.13E+04	8.59E+01	2.01E+01	6.68E-01
	STD	3.31E-02	5.83E-02	2.09E-01	3.04E+01	3.22E+00	8.10E+00	7.12E-01	2.79E+03	2.58E+01	4.52E+00	1.49E-01
F7	Average	2.15E-04	2.38E-02	1.25E-02	1.12E-01	2.30E-02	2.68E-02	2.15E-02	1.13E+00	6.33E-02	4.41E-02	3.33E-02
	Min	5.35E-05	5.66E-03	2.82E-03	2.81E-03	1.90E-03	4.34E-03	3.41E-03	1.07E-01	5.93E-03	8.78E-03	9.66E-03
	Max	5.13E-04	1.22E-01	2.94E-02	5.47E-01	7.16E-02	6.81E-02	5.23E-02	2.69E+00	2.12E-01	1.37E-01	5.77E-02
	STD	1.18E-04	2.20E-02	6.32E-03	1.01E-01	1.77E-02	1.41E-02	1.29E-02	7.35E-01	5.08E-02	2.68E-02	1.13E-02
F8	Average	−3.02E+03	−2.83E+03	−2.82E+03	−2.40E+03	−1.96E+03	−1.62E+03	−3.63E+03	−2.12E+61	−2.79E+03	−3.22E+03	−3.55E+03
	Min	−3.64E+03	−3.52E+03	−3.72E+03	−4.16E+03	−2.47E+03	−2.42E+03	−4.07E+03	−7.29E+62	−3.61E+03	−3.71E+03	−3.92E+03
	Max	−2.43E+03	−2.20E+03	−2.15E+03	−1.81E+03	−1.60E+03	−1.28E+03	−3.24E+03	−1.29E+52	−2.17E+03	−2.40E+03	−3.19E+03
	STD	3.46E+02	3.22E+02	3.52E+02	5.07E+02	1.94E+02	2.24E+02	2.01E+02	1.21E+62	3.81E+02	3.11E+02	1.88E+02
F9	Average	2.59E-08	2.24E+01	2.70E+01	2.23E+01	1.81E+01	1.25E+01	6.26E+00	4.41E+01	3.54E+01	2.67E+01	1.60E+01
	Min	5.74E-09	3.98E+00	6.13E+00	5.02E+00	1.24E+00	4.17E+00	2.51E+00	1.78E+01	7.86E+00	6.09E+00	9.70E+00
	Max	1.17E-07	6.17E+01	5.49E+01	6.37E+01	6.43E+01	2.99E+01	1.17E+01	6.87E+01	7.50E+01	7.08E+01	2.48E+01
	STD	2.40E-08	1.27E+01	1.43E+01	1.14E+01	1.46E+01	5.42E+00	2.46E+00	1.52E+01	1.42E+01	1.35E+01	3.29E+00
F10	Average	2.02E-04	1.41E+00	1.01E+00	7.15E+00	3.86E+00	9.49E-01	2.59E-01	1.53E+01	5.08E+00	2.49E+00	6.93E-01
	Min	4.45E-05	1.44E-01	2.29E-01	3.93E-01	1.55E-01	2.76E-05	1.03E-02	1.15E+01	1.53E+00	6.15E-01	2.82E-01
	Max	4.89E-04	2.49E+00	2.81E+00	1.37E+01	2.00E+01	6.21E+00	1.67E+00	1.85E+01	1.17E+01	3.94E+00	1.81E+00
	STD	1.38E-04	8.87E-01	6.44E-01	3.28E+00	6.08E+00	1.23E+00	3.95E-01	1.49E+00	2.13E+00	7.20E-01	3.66E-01
F11	Average	9.25E-05	3.17E-01	6.86E-01	4.09E-01	7.33E-01	3.60E+00	2.61E-01	4.57E+01	9.06E-01	9.44E-01	6.66E-01
	Min	9.05E-06	1.45E-01	4.05E-01	8.51E-02	1.94E-02	1.30E+00	5.45E-02	1.84E+01	1.68E-01	5.79E-01	3.80E-01
	Max	3.80E-04	6.44E-01	9.91E-01	2.06E+00	1.24E+00	6.00E+00	9.68E-01	8.24E+01	1.64E+00	1.15E+00	9.34E-01
	STD	8.99E-05	1.44E-01	1.65E-01	3.85E-01	2.49E-01	1.42E+00	1.70E-01	1.83E+01	4.59E-01	1.53E-01	1.23E-01
F12	Average	2.11E-02	1.27E+00	4.95E-01	8.00E+00	1.50E+00	2.25E-01	3.28E-02	6.60E+06	4.50E+00	2.43E+00	3.63E-02

(*continued on next page*)

Table 9 (*continued*)

Func.		OBLGOA	GOA	MVO	ALO	SCA	PSO	GA	BAT	DA	MFO	DE
	Min	1.82E-03	1.38E-02	8.87E-03	2.59E+00	1.56E-01	7.85E-10	1.89E-06	3.52E+01	5.90E-01	2.16E-02	5.89E-03
	Max	4.84E-02	5.79E+00	2.12E+00	2.72E+01	4.58E+00	2.80E+00	3.57E-01	5.97E+07	2.16E+01	1.18E+01	1.06E-01
	STD	1.31E-02	1.32E+00	6.10E-01	5.71E+00	1.28E+00	5.08E-01	9.11E-02	1.14E+07	4.06E+00	2.62E+00	2.31E-02
F13	Average	6.93E-02	3.54E-02	4.34E-02	2.82E+01	1.58E+00	1.36E-01	2.63E-02	1.70E+07	7.75E+00	2.68E+00	7.85E-02
	Min	1.60E-02	4.17E-03	1.14E-02	7.66E-04	4.46E-01	9.29E-09	7.72E-04	8.83E+04	1.94E-02	1.63E-01	2.20E-02
	Max	1.84E-01	1.14E-01	1.62E-01	5.37E+02	1.35E+01	2.89E+00	1.13E-01	9.02E+07	1.40E+02	1.24E+01	1.35E-01
	STD	3.97E-02	2.28E-02	2.45E-02	9.64E+01	2.24E+00	5.42E-01	2.60E-02	1.87E+07	2.31E+01	2.83E+00	2.76E-02
F14	Average	1.26E+00	1.36E+00	2.62E+00	6.42E+00	2.41E+00	2.82E+00	1.08E+00	2.28E+01	3.64E+00	3.25E+00	1.03E+00
	Min	9.98E-01	9.98E-01	9.98E-01	9.98E-01	9.98E-01	9.98E-01	9.98E-01	9.98E-01	9.98E-01	9.98E-01	9.98E-01
	Max	1.99E+00	2.98E+00	1.74E+01	1.55E+01	5.94E+00	1.55E+01	1.99E+00	4.15E+02	1.55E+01	1.17E+01	1.99E+00
	STD	4.40E-01	6.53E-01	3.58E+00	4.28E+00	1.09E+00	2.87E+00	2.43E-01	6.75E+01	3.03E+00	2.59E+00	1.66E-01
F15	Average	1.85E-03	3.71E-03	4.35E-03	4.50E-03	1.25E-03	8.91E-04	3.03E-03	1.19E-02	4.34E-03	1.14E-03	1.25E-03
	Min	3.74E-04	7.40E-04	5.11E-04	3.16E-04	4.04E-04	3.86E-04	7.14E-04	8.97E-04	5.96E-04	6.32E-04	7.09E-04
	Max	1.62E-02	2.30E-02	2.05E-02	2.49E-02	2.80E-03	2.89E-03	2.08E-02	1.12E-01	2.04E-02	2.24E-03	2.87E-03
	STD	2.73E-03	6.18E-03	6.99E-03	5.12E-03	5.10E-04	4.13E-04	4.44E-03	2.12E-02	6.01E-03	4.02E-04	4.44E-04
F16	Average	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.01E+00	-1.03E+00	-1.03E+00	-1.03E+00
	Min	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00
	Max	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-1.03E+00	-2.16E-01	-1.03E+00	-1.03E+00	-1.03E+00
	STD	2.38E-04	2.88E-10	1.08E-05	9.66E-13	4.66E-04	3.16E-16	4.56E-07	1.36E-01	9.08E-08	2.71E-16	3.36E-10
F17	Average	3.98E-01	3.98E-01	3.98E-01	3.98E-01	4.11E-01	3.98E-01	4.09E-01	4.85E-01	3.98E-01	3.98E-01	3.98E-01
	Min	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01	3.98E-01
	Max	3.98E-01	3.98E-01	3.98E-01	3.98E-01	4.84E-01	3.98E-01	6.09E-01	1.47E+00	3.98E-01	3.98E-01	3.98E-01
	STD	4.28E-10	4.98E-10	2.00E-05	2.24E-12	1.97E-02	0.00E+00	4.05E-02	2.12E-01	2.13E-07	3.15E-12	5.14E-07
F18	Average	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00
	Min	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00
	Max	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.02E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00	3.00E+00
	STD	4.08E-09	2.55E-09	6.47E-05	1.04E-11	3.12E-03	5.84E-13	8.67E-05	7.32E-04	8.32E-07	4.85E-15	4.66E-12
F19	Average	-3.86E+00	-3.86E+00	-3.86E+00	-3.85E+00	-3.85E+00	-3.86E+00	-3.86E+00	-3.79E+00	-3.86E+00	-3.86E+00	-3.86E+00
	Min	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00	-3.86E+00
	Max	-3.86E+00	-3.68E+00	-3.86E+00	-3.72E+00	-3.83E+00	-3.86E+00	-3.86E+00	-3.09E+00	-3.79E+00	-3.86E+00	-3.86E+00
	STD	7.50E-04	3.20E-02	7.52E-05	2.94E-02	6.98E-03	2.42E-15	2.28E-08	2.15E-01	1.19E-02	3.31E-14	1.86E-14
F20	Average	-3.28E+00	-3.27E+00	-3.26E+00	-3.25E+00	-2.82E+00	-3.27E+00	-3.29E+00	-3.17E+00	-3.24E+00	-3.23E+00	-3.32E+00
	Min	-3.32E+00	-3.32E+00	-3.32E+00	-3.32E+00	-3.12E+00	-3.32E+00	-3.32E+00	-3.32E+00	-3.32E+00	-3.32E+00	-3.32E+00
	Max	-3.16E+00	-3.12E+00	-3.12E+00	-3.02E+00	-1.91E+00	-3.20E+00	-3.20E+00	-1.70E+00	-3.07E+00	-3.16E+00	-3.30E+00
	STD	6.99E-02	7.00E-02	7.39E-02	9.06E-02	3.12E-01	5.88E-02	5.37E-02	3.51E-01	9.14E-02	5.35E-02	6.11E-03
F21	Average	-1.02E+01	-7.15E+00	-5.27E+00	-4.84E+00	-1.98E+00	-5.65E+00	-3.34E+00	-3.82E+00	-5.47E+00	-7.44E+00	-8.34E+00
	Min	-1.02E+01	-1.02E+01	-1.02E+01	-1.02E+01	-4.77E+00	-1.02E+01	-5.10E+00	-1.02E+01	-1.02E+01	-1.02E+01	-1.02E+01
	Max	-1.02E+01	-2.63E+00	-2.63E+00	-2.63E+00	-4.96E-01	-2.63E+00	-2.63E+00	-8.67E-01	-2.63E+00	-2.63E+00	-2.60E+00
	STD	2.00E-06	3.50E+00	2.86E+00	2.65E+00	1.48E+00	3.52E+00	9.00E-01	2.81E+00	2.55E+00	3.40E+00	2.27E+00
F22	Average	-1.03E+01	-8.15E+00	-6.62E+00	-5.25E+00	-2.84E+00	-5.25E+00	-4.36E+00	-3.80E+00	-6.15E+00	-6.52E+00	-8.61E+00
	Min	-5.13E+00	-2.75E+00	-1.04E+01	-1.04E+01	-4.95E+00	-1.04E+01	-9.23E+00	-1.04E+01	-1.04E+01	-1.04E+01	-1.04E+01
	Max	-1.04E+01	-1.04E+01	-1.84E+00	-1.84E+00	-5.22E-01	-1.84E+00	-2.75E+00	-6.70E-01	-1.84E+00	-1.84E+00	-3.49E+00
	STD	8.67E-01	3.26E+00	3.73E+00	3.18E+00	1.30E+00	3.30E+00	1.93E+00	2.60E+00	2.84E+00	3.46E+00	2.03E+00
F23	Average	-1.05E+01	-7.25E+00	-7.24E+00	-5.62E+00	-2.48E+00	-6.07E+00	-7.98E+00	-3.30E+00	-5.04E+00	-7.63E+00	-8.74E+00
	Min	-1.05E+01	-1.05E+01	-1.05E+01	-1.05E+01	-5.76E+00	-1.05E+01	-1.05E+01	-1.05E+01	-1.05E+01	-1.05E+01	-1.05E+01
	Max	-1.05E+01	-2.42E+00	-2.42E+00	-1.68E+00	-9.32E-01	-1.68E+00	-2.87E+00	-1.03E+00	-1.86E+00	-1.86E+00	-3.45E+00
	STD	1.91E-05	3.58E+00	3.75E+00	3.62E+00	1.15E+00	3.62E+00	3.01E+00	2.50E+00	2.52E+00	3.54E+00	1.76E+00

Table 10
The results of the welded beam design problem.

Algorithm	H	L	t	b	Optimal cost
OBLGOA	0.205769	3.471135	9.032728	0.2059072	1.7257
RO (Kaveh & A., 2012)	0.203687	3.528467	9.004233	0.207241	1.735344
HS (Lee & Geem, 2005)	0.2442	6.2231	8.2915	0.2443	2.3807
DAVID (Ragsdell & Phillips, 1976)	0.2434	6.2552	8.2915	0.2444	2.3841
SIMPLEX (Ragsdell & Phillips, 1976)	0.2792	5.6256	7.7512	0.2796	2.5307
CPSO (Krohling & dos Santos Coelho, 2006)	0.202369	3.544214	9.04821	0.205723	1.72802
MVO (Mirjalili et al., 2016)	0.205463	3.473193	9.044502	0.205695	1.72645
GA (Deb, 1991)	0.205986	3.471328	9.020224	0.20648	1.728226
GSA (Rashedi et al., 2009)	0.182129	3.856979	10	0.202376	1.87995
CSCA (Huang et al., 2007)	0.203137	3.542998	9.033498	0.206179	1.733461
WOA (Mirjalili & Lewis, 2016)	0.205396	3.484293	9.037426	0.206276	1.730499

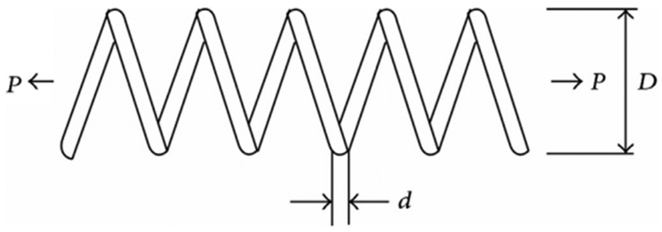

Fig. 5. parameters of the tension/compression spring problem.

$$M = P(L + \frac{x_2}{2}),$$
$$R = \sqrt{\frac{x_2^2}{4} + \left(\frac{x_1 + x_3}{2}\right)^2},$$
$$J = 2\left\{\sqrt{2x_1x_2}\left[\ \frac{x_2^2}{4} + (\frac{x_1+x_3}{2})^2\ \right]\right\},$$
$$\sigma(\vec{x}) = \frac{6PL}{x_4x_3^2}, \quad \delta(\vec{x}) = \frac{6PL^3}{Ex_3^2x_4}$$
$$P_c(\vec{x}) = \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}\left(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}\right),$$
$$P = 6000\ lb, L = 14\ in., \quad \delta_{max} = 0.25\ in.,$$
$$E = 30 \times 1^6\ psi, \quad G = 12 \times 10^6\ psi,$$
$$\tau_{max} = 13,600\ psi, \quad \sigma_{max} = 30,000\ psi \quad (18)$$

The proposed OBLGOA algorithms are used to determine the optimal values for the four parameters, and the results are compared with those of the following algorithms: Whale Optimization Algorithm (WOA) (Mirjalili & Lewis, 2016), Ray Optimization (RO) (Kaveh & A., 2012), Simplex method (SIMPLEX) (Ragsdell & Phillips, 1976) Coevolutionary Particle Swarm Optimization (CPSO) (Krohling & dos Santos Coelho, 2006), Gravitational Search Algorithm (GSA) (Rashedi, Nezamabadi-Pour & Saryazdi, 2009), GA (Deb, 1991), Co-evolutionary Differential Evolution (CSCA) (Huang, Wang & He, 2007), Harmony Search (HS) (Lee & Geem, 2005), MVO (Mirjalili, Mirjalili & Hatamlou, 2016), and Davidon-Fletcher-Powell (DAVID) (Ragsdell & Phillips, 1976). Table 10 shows the comparison results, which indicate that the proposed OBLGOA provides better results than the other algorithms.

4.9.2. Tension/compression spring design problem

The main goal of the tension/compression spring problem is to design a spring for a minimum weight through estimating the optimum values for three parameters namely, wire diameter (d), a number of active coils (N) and mean coil diameter (D). Fig. 5 shows the structure of this problem, as well as, the mathematical definition of it is given by the following equation:

$$\begin{aligned}
&\text{Consider} && \vec{x} = [x_1\ x_2\ x_3] = [d\ D\ N],\\
&\text{Minimize} && f(\vec{x}) = (x_3 + 2)x_2x_1^2,\\
&\text{Subject to} && g_1(\vec{x}) = 1 - \frac{x_2^3x_3}{71785x_1^4} \leqslant 0,\\
& && g_2(\vec{x}) = \frac{4x_2^2 - x_1x_2}{12566(x_2x_1^3 - x_1^4)} + \frac{1}{5108x_1^2} \leqslant 0,\\
& && g_3(\vec{x}) = 1 - \frac{140.45x_1}{x_2^2x_3} \leqslant 0,\\
& && g_4(\vec{x}) = \frac{x_1 + x_2}{1.5} - 1 \leqslant 0,\\
&\text{Variables range} && 0.05 \leqslant x_1 \leqslant 2\\
& && 0.25 \leqslant x_2 \leqslant 1.30\\
& && 2.00 \leqslant x_3 \leqslant 15
\end{aligned} \quad (19)$$

There are several algorithms have been used to solve this problem such as MVO (Mirjalili et al., 2016), WOA (Mirjalili & Lewis, 2016), CSCA (Huang et al., 2007), CPSO (Krohling & dos Santos Coelho, 2006), Evolution Strategy (ES) (Mezura-Montes & Coello, 2008), GA (Coello, 2000), GSA (Mirjalili & Lewis, 2016), Ray-Saini method (RAY & SAINI, 2001), and mathematical method by Belegundu and Arora (1985).

The comparison results of the proposed OBLGOA with other algorithms are given in Table 11, in which it can observe that the proposed OBLGOA algorithm is better than most of the algorithms, except WOA, ES, and CPSO

4.9.3. Three-bar truss design problem

The main objective of the three-bar truss design problem is to find the optimal values for two parameters (A_1 and A_2, although a third value, $A_3 = A_1$, is also observed) that can minimize the weight when designing a truss as illustrated in Fig. 6.

The mathematical model of the formulation of the three-bar truss design problem is given by the following equation:

$$\begin{aligned}
&\text{Consider} && \vec{x} = [x_1\ x_2] = [A_1\ A_2],\\
&\text{Minimize} && f(\vec{x}) = (2\sqrt{2x_1} + x_2) * l,\\
&\text{Subject to} && g_1(\vec{x}) = \frac{\sqrt{2x_1} + x_2}{\sqrt{2x_1^2} + 2x_1x_2}P - \sigma \leqslant 0,\\
& && g_2(\vec{x}) = \frac{x_2}{\sqrt{2x_1^2} + 2x_1x_2}P - \sigma \leqslant 0,\\
& && g_3(\vec{x}) = \frac{1}{\sqrt{2x_2} + x_1}P - \sigma \leqslant 0,\\
&\text{Variables range} && 0 \leqslant x_1, x_2 \leqslant 1,\\
&\textit{where} && l{=}100\ \text{cm}, P{=}2\ \text{KN/cm}^2, \sigma{=}2\ \text{KN/cm}^2
\end{aligned} \quad (20)$$

Table 11
The results of the tension/compression spring design problems.

Algorithm	d	D	N	Optimal cost
OBLGOA	0.0530178	0.38953229	9.6001616	0.01270136
Belegundu-Arora method (Belegundu & Arora, 1985)	0.0500	0.3177	14.026	0.012730
GA (Coello, 2000)	0.05148	0.351661	11.632201	0.01270478
WOA (Mirjalili & Lewis, 2016)	0.051207	0.345215	12.004032	0.0126763
CPSO (Krohling & dos Santos Coelho, 2006)	0.051728	0.357644	11.244543	0.0126747
ES (Mezura-Montes & Coello, 2008)	0.051643	0.35536	11.397926	0.012698
MVO (Mirjalili et al., 2016)	0.05251	0.37602	10.33513	0.012790
GSA(Mirjalili & Lewis, 2016)	0.050276	0.323680	13.525410	0.0127022
Ray-Saini method (RAY & SAINI, 2001)	0.321532	0.050417	13.979915	0.013060

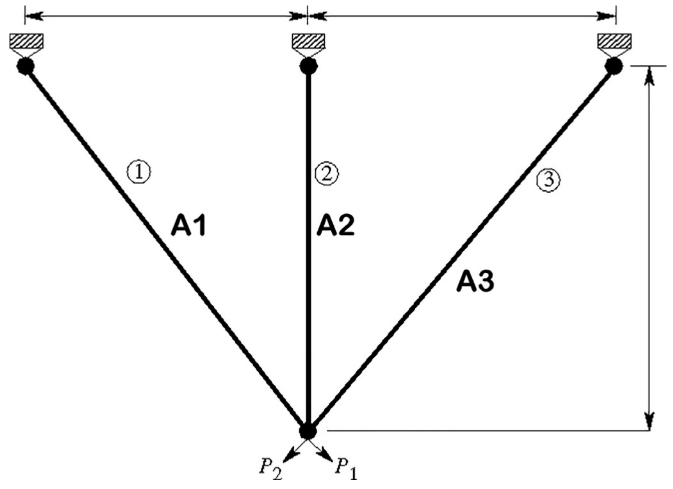

Fig. 6. Parameters of the three-bar truss design problem.

Table 12
The results of the three-bar truss design problems.

Algorithm	× 1	× 2	Optimal cost
OBLGOA	0.78866365	0.408280786	263.895844
GOA (Saremi et al., 2017)	0.788897556	0.40761957	263.8958815
PSO-DE (Liu et al., 2010)	0.7886751	0.4082482	263.8958434
DSS-MDE (Zhang et al., 2008)	0.78867513	0.40824828	263.8958434
MBA (Sadollah et al., 2013)	0.788565	0.4085597	263.895852
CS (Gandomi et al., 2013b)	0.78867	0.40902	263.97156

To evaluate the performance of the proposed OBLGOA to solve this problem, its results are compared against the results of the following algorithms: DSS-MDE (Zhang, Luo & Wang, 2008), CS (Gandomi, Yang & Alavi, 2013b), PSO-DE (Liu, Cai & Wang, 2010), GOA(Rogers et al., 2003), and MBA (Sadollah, Bahreininejad, Eskandar & Hamdi, 2013). Table 12 shows the estimated parameters and the optimal cost obtained by the proposed algorithm and the other algorithms. This table shows that the results of the OBLGOA are very close to those of all the other algorithms except for the CS algorithm. However, the optimal cost obtained by the proposed algorithm is regarded as an indication of the high performance of the OBLGOA in solving this problem.

4.9.4. Pressure vessel design problem

The minimization of the total cost of the cylindrical pressure vessel is the main objective of this problem, in which the total costs consist of the cost of welding, materials and forming as shown in Fig. 7. To solve this problem, the optimal values for four parameters must be determined: thickness of the head *Th*, the thickness *Ts*, the length of the cylindrical section of the vessel *L* and the inner radius *R*.

The performance of the proposed algorithm is compared with the performance of previously published algorithms, including the

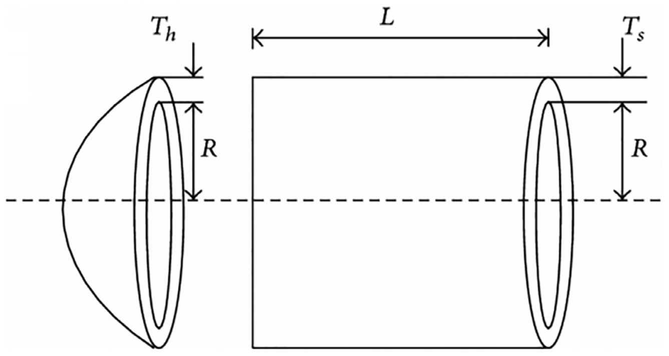

Fig. 7. Parameters of the pressure vessel design problem.

PSO-DE (Liu et al., 2010), HPSO (He & Wang, 2007), ACO (Kaveh & Talatahari, 2010), CDE (Huang et al., 2007), ES (Mezura-Montes & Coello, 2008), GA (Coello, 2000), and GSA (Mirjalili & Lewis, 2016), as shown in Table 13.

Table 13summarizes the comparison results between the proposed OBLGOA algorithm and the other algorithms. This table shows that the OBLGOA achieved the minimum value among all algorithms, and the PSO-DE and HPSO were better than the other algorithms and achieved the second-best results.

$$\begin{aligned} \text{Consider} \quad & \vec{x} = [x_1\ x_2\ x_3\ x_4] = [T_s\ T_h\ R\ L], \\ \text{Minimize} \quad & 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 \\ & + 3.1661x_1^2x_4 + 19.84x_1^2x_3, \\ \text{Subject to} \quad & g_1(\vec{x}) = -x_1 + 0.0193x_3 \leqslant 0, \\ & g_2(\vec{x}) = -x_2 + 0.00954x_3 \leqslant 0, \\ & g_3(\vec{x}) = -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1296000 \leqslant 0, \\ & g_4(\vec{x}) = x_4 - 240 \leqslant 0, \\ \text{Variables range} \quad & 0 \leqslant x_1 \leqslant 99, \\ & 0 \leqslant x_2 \leqslant 99, \\ & 10 \leqslant x_3 \leqslant 200, \\ & 10 \leqslant x_4 \leqslant 200 \end{aligned} \tag{21}$$

These results show that the proposed OBLGOA algorithm has a strong ability to solve real problems without a known search domain. The superiority of the OBLGOA is due to several facts: 1) the OBL strategy is used at the initial stage to generate a better initial population, and this approach will lead to an improved ability to explore and increase the convergence of the GOA algorithm; 2) the GOA algorithm can be easily exploited because of the larger attraction forces between grasshoppers; 3) the OBL can be used to update the current population, which improves the ability of the GOA to find the target value.

Table 13
The results of the pressure vessel design problem.

	Optimal values for variables				
Algorithm	*Th*	*Ts*	*R*	*L*	Optimal cost
OBLGOA	0.81622	0.40350	42.291138	174.811191	5966.67160
PSO-DE (Liu et al., 2010)	0.8125	0.4375	42.098446	176.6366	6059.71433
HPSO (He & Wang, 2007)	0.8125	0.4375	42.0984	176.6366	6059.7143
ACO (Kaveh & Talatahari, 2010)	0.8125	0.4375	42.098353	176.637751	6059.7258
CDE (Huang et al., 2007)	0.8125	0.4375	42.098411	176.63769	6059.734
ES (Mezura-Montes & Coello, 2008)	0.8125	0.4375	42.098087	176.640518	6059.7456
GA (Coello, 2000)	0.8125	0.4375	42.097398	176.65405	6059.94634
GSA (Mirjalili & Lewis, 2016)	1.125	0.625	55.9886598	84.4542025	8538.8359

5. Conclusions and future works

This paper has presented an improved version of the original GOA algorithm using the OBL strategy, and the new approach is called the OBLGOA. This strategy focuses on increasing the convergence rate of metaheuristic algorithms by calculating the opposite solution to the current solution. This modification improves the exploration ability of the algorithm. To evaluate the performance of the proposed OBLGOA, a set of experiment series was performed. In series 1, the OBLGOA was compared with the classical GOA algorithm to explain the influence of the OBL approach on the GOAs behaviour. In experiment series 2 and 3, the effectiveness of the iteration number and population size was tested to determine the optimal number of iterations and the population size that lead to a convergence to the optimal value by the OBLGOA that is faster than that of the classical approach. In experiment series 4, the influence of the selected OBL ratio was explored, and the performance of the OBLGOA was compared with that of other previously published algorithms. In this context, 23 benchmark test functions were used. The experimental results revealed that the proposed OBLGOA algorithm outperformed the compared algorithms in terms of the performance measures. In addition, the statistical results showed the ability of the proposed algorithm to solve a set of real engineering problems that included the welded beam design, tension/compression spring design, three-bar truss design, and pressure vessel design problems. According to the powerful results obtained for the proposed OBLGOA algorithm, in future work, we will attempt to introduce the OBLGOA to several fields, such as feature selection, image segmentation, and multi–objective optimization algorithms, and use it as a prediction method for renewable energy problems.

References

Ahandani, M. A., & Alavi-Rad, H. (2015). Opposition-based learning in shuffled frog leaping: An application for parameter identification. *Information Sciences, 291*, 19–42.

Ahirwal, M. K., Kumar, A., & Singh, G. K. (2013). Eeg/erp adaptive noise canceller design with controlled search space (css) approach in cuckoo and other optimization algorithms. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB), 10*, 1491–1504.

Ahirwal, M. K., Kumar, A., & Singh, G. K. (2014). Adaptive filtering of eeg/erp through noise cancellers using an improved pso algorithm. *Swarm and Evolutionary Computation, 14*, 76–91.

Bayraktar, Z., Komurcu, M., Bossard, J. A., & Werner, D. H. (2013). The wind driven optimization technique and its application in electromagnetics. *IEEE Transactions on Antennas and Propagation, 61*, 2745–2757.

Belegundu, A. D., & Arora, J. S. (1985). A study of mathematical programming methods for structural optimization. part i: Theory. *International Journal for Numerical Methods in Engineering, 21*, 1583–1599.

Beyer, H. G., & Sendhoff, B. (2007). Robust optimization–a comprehensive survey. *Computer Methods in Applied Mechanics and Engineering, 196*, 3190–3218.

Chen, W. N., Zhang, J., Chung, H. S., Zhong, W. L., Wu, W. G., & Shi, Y. H. (2010). A novel set-based particle swarm optimization method for discrete optimization problems. *IEEE Transactions on Evolutionary Computation, 14*, 278–300.

Coello, C. A. C. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry, 41*, 113–127.

Coello, C. A. C. (2002). Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: A survey of the state of the art. *Computer Methods in Applied Mechanics and Engineering, 191*, 1245–1287.

Deb, K. (1991). Optimal design of a welded beam via genetic algorithms. *AIAA Journal, 29*, 2013–2015.

Deb, K., & Goldberg, D. E. (1993). Analyzing deception in trap functions. *Foundations of Genetic Algorithms, 2*, 93–108.

Eberhart, R., & Kennedy, J. (1995). A new optimizer using particle swarm theory. In *Micro machine and human science, 1995. MHS'95., Proceedings of the sixth international symposium on* (pp. 39–43). IEEE.

El Aziz, M. A., Ewees, A. A., & Hassanien, A. E. (2016). Hybrid swarms optimization based image segmentation. In *Hybrid soft computing for image segmentation* (pp. 1–21). Springer.

El Aziz, M. A., Ewees, A. A., & Hassanien, A. E. (2017). Whale optimization algorithm and moth-flame optimization for multilevel thresholding image segmentation. *Expert Systems with Applications, 83*, 242–256.

El Aziz, M. A., Ewees, A. A., & Hassanien, A. E. (2018a). Multi-objective whale optimization algorithm for content-based image retrieval. *Multimedia Tools and Applications*, 1–38.

El Aziz, M. A., Ewees, A. A., Hassanien, A. E., Mudhsh, M., & Xiong, S. (2018b). Multi-objective whale optimization algorithm for multilevel thresholding segmentation. In *Advances in Soft Computing and Machine Learning in Image Processing* (pp. 23–39). Springer.

Ewees, A. A., El Aziz, M. A., & Hassanien, A. E. (2017). Chaotic multi-verse optimizer-based feature selection. *Neural Computing and Applications*, 1–16.

Fister Jr, I., Yang, X. S., Fister, I., Brest, J., & Fister, D. (2013). A brief review of nature-inspired algorithms for optimization. *Elektrotehniski Vestnik, 80*, 116–122.

Gandomi, A., Yang, X. S., Talatahari, S., & Alavi, A. (2013a). Firefly algorithm with chaos. *Communications in Nonlinear Science and Numerical Simulation, 18*, 89–98.

Gandomi, A. H., & Yang, X. S. (2014). Chaotic bat algorithm. *Journal of Computational Science, 5*, 224–232.

Gandomi, A. H., Yang, X. S., & Alavi, A. H. (2013b). Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems. *Engineering with Computers, 29*, 17–35.

Gao, X., Wang, X., Ovaska, S., & Zenger, K. (2012). A hybrid optimization method of harmony search and opposition-based learning. *Engineering Optimization, 44*, 895–914.

Gong, C. (2016). Opposition-based adaptive fireworks algorithm. *Algorithms, 9*, 43.

Hassanien, A. E., & Emary, E. (2016). *Swarm intelligence: Principles, advances, and applications*. CRC Press.

He, Q., & Wang, L. (2007). A hybrid particle swarm optimization with a feasibility-based rule for constrained optimization. *Applied Mathematics and Computation, 186*, 1407–1422.

Huang, F. z., Wang, L., & He, Q. (2007). An effective co-evolutionary differential evolution for constrained optimization. *Applied Mathematics and computation, 186*, 340–356.

Ibrahim, R. A., Oliva, D., Ewees, A. A., & Lu, S. (2017). Feature selection based on improved runner-root algorithm using chaotic singer map and opposition-based learning. In *International conference on neural information processing* (pp. 156–166). Springer.

Kannan, S., Slochanal, S. M. R., Subbaraj, P., & Padhy, N. P. (2004). Application of particle swarm optimization technique and its variants to generation expansion planning problem. *Electric Power Systems Research, 70*, 203–210.

Karaboga, D. (2005). An idea based on honey bee swarm for numerical optimization. *Technical report technical report-tr06*. Erciyes university, engineering faculty, computer engineering department.

Kaveh, A., & Talatahari, S. (2010). An improved ant colony optimization for constrained engineering design problems. *Engineering Computations, 27*, 155–182.

Kaveh, K. M. A (2012). A new meta-heuristic method: Ray optimization. *Computers & Structures, 112–113*, 283–294.

Kelley, C. (1999). Detection and remediation of stagnation in the nelder–mead algorithm using a sufficient decrease condition. *SIAM Journal on Optimization, 10*, 43–55.

Knowles, J. D., Watson, R. A., & Corne, D. W. (2001). Reducing local optima in single-objective problems by multi-objectivization. In *International conference on evolutionary multi-criterion optimization* (pp. 269–283). Springer.

Kohli, M., & Arora, S. (2017). Chaotic grey wolf optimization algorithm for constrained optimization problems. *Journal of Computational Design and Engineering*.

Koziel, S., & Yang, X. S. (2011). *Computational optimization, methods and algorithms*. Springer. Volume 356.

Krohling, R. A., & dos Santos Coelho, L. (2006). Coevolutionary particle swarm optimization using gaussian distribution for solving constrained optimization problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 36*, 1407–1416.

Lee, K. S., & Geem, Z. W. (2005). A new meta-heuristic algorithm for continuous engineering optimization: Harmony search theory and practice. *Computer Methods in Applied Mechanics and Engineering, 194*, 3902–3933.

Li, J., Chen, T., Zhang, T., & Li, Y. X. (2016). A cuckoo optimization algorithm using elite opposition-based learning and chaotic disturbance. *Journal of Software Engineering, 10*, 16–28.

Li, X. l., Shao, Z. j., Qian, J. X., et al. (2002). An optimizing method based on autonomous animats: Fish-swarm algorithm. *System Engineering Theory and Practice, 22*, 32–38.

Liu, H., Cai, Z., & Wang, Y. (2010). Hybridizing particle swarm optimization with differential evolution for constrained numerical and engineering optimization. *Applied Soft Computing, 10*, 629–640.

Maniezzo, A. (1992). Distributed optimization by ant colonies. In *Toward a practice of autonomous systems: Proceedings of the first European conference on artificial life* (p. 134). Mit Press.

Marler, R. T., & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and Multidisciplinary Optimization, 26*, 369–395.

Mavrovouniotis, M., Li, C., & Yang, S. (2017). A survey of swarm intelligence for dynamic optimization: Algorithms and applications. *Swarm and Evolutionary Computation, 33*, 1–17.

Mezura-Montes, E., & Coello, C. A. C. (2008). An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. *International Journal of General Systems, 37*, 443–473.

Mezura-Montes, E., & Coello, C. A. C. (2011). Constraint-handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation, 1*, 173–194.

Mirjalili, S. (2015). Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm. *Knowledge-Based Systems, 89*, 228–249.

Mirjalili, S. (2016). Dragonfly algorithm: A new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems. *Neural Computing and Applications, 27*, 1053–1073.

Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51–67.

Mirjalili, S., Mirjalili, S. M., & Hatamlou, A. (2016). Multi-verse optimizer: A nature-inspired algorithm for global optimization. *Neural Computing and Applications, 27*, 495–513.

Mirjalili, S., Mirjalili, S. M., & Yang, X. S. (2014). Binary bat algorithm. *Neural Computing and Applications, 25*, 663–681.

Oliva, D., Ewees, A. A., Aziz, M. A. E., Hassanien, A. E., & Peréz-Cisneros, M. (2017). A chaotic improved artificial bee colony for parameter estimation of photovoltaic cells. *Energies*, 865. Volume 10.

Passino, K. M. (2002). Biomimicry of bacterial foraging for distributed optimization and control. *IEEE Control Systems, 22*, 52–67.

Ragsdell, K., & Phillips, D. (1976). Optimal design of a class of welded structures using geometric programming. *Journal of Engineering for Industry, 98*, 1021–1025.

Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009). Gsa: A gravitational search algorithm. *Information sciences, 179*, 2232–2248.

RAY, T., & SAINI, P. (2001). Engineering design optimization using a swarm with an intelligent information sharing among individuals. *Engineering Optimization, 33*, 735–748.

Rogers, S. M., Matheson, T., Despland, E., Dodgson, T., Burrows, M., & Simpson, S. J. (2003). Mechanosensory-induced behavioural gregarization in the desert locust schistocerca gregaria. *Journal of Experimental Biology, 206*, 3991–4002.

Sadollah, A., Bahreininejad, A., Eskandar, H., & Hamdi, M. (2013). Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems. *Applied Soft Computing, 13*, 2592–2612.

Saremi, S., Mirjalili, S., & Lewis, A. (2017). Grasshopper optimisation algorithm: Theory and application. *Advances in Engineering Software, 105*, 30–47.

Shang, J., Sun, Y., Li, S., Liu, J. X., Zheng, C. H., & Zhang, J. (2015). An improved opposition-based learning particle swarm optimization for the detection of snp-snp interactions. *BioMed Research International, 2015*.

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y. P., Auger, A., & Tiwari, S. (2005). Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization. *KanGAL Report, 2005005*, 2005.

Thangaraj, R., Pant, M., Chelliah, T. R., & Abraham, A. (2012). Opposition based chaotic differential evolution algorithm for solving global optimization problems. In *Nature and biologically inspired computing (NaBIC), 2012 fourth world congress on* (pp. 1–7). IEEE.

Tizhoosh, H. R. (2005a). Opposition-based learning: A new scheme for machine intelligence. In *Computational intelligence for modelling, control and automation, 2005 and international conference on intelligent agents, web technologies and internet commerce, international conference on* (pp. 695–701). IEEE. Volume. 1.

Tizhoosh, H. R. (2005b). Opposition-based learning: A new scheme for machine intelligence. In *Computational intelligence for modelling, control and automation, 2005 and international conference on intelligent agents, web technologies and internet commerce, international conference on* (pp. 695–701). IEEE. Volume 1.

Vogl, T. P., Mangis, J., Rigler, A., Zink, W., & Alkon, D. (1988). Accelerating the convergence of the back-propagation method. *Biological Cybernetics, 59*, 257–263.

Wang, G. G., Guo, L., Gandomi, A. H., Hao, G. S., & Wang, H. (2014). Chaotic krill herd algorithm. *Information Sciences, 274*, 17–34.

Xu, H., Erdbrink, C. D., & Krzhizhanovskaya, V. V. (2015). How to speed up optimization? Opposite-center learning and its application to differential evolution. *Procedia Computer Science, 51*, 805–814.

Yadav, A., Deep, K., Kim, J. H., & Nagar, A. K. (2016). Gravitational swarm optimizer for global optimization. *Swarm and Evolutionary Computation, 31*, 64–89.

Yang, X. S. (2010a). *Nature-inspired metaheuristic algorithms*. Luniver press.

Yang, X. S. (2010b). *Nature-inspired metaheuristic algorithms*. Luniver Press.

Zhang, M., Luo, W., & Wang, X. (2008). Differential evolution with dynamic stochastic selection for constrained optimization. *Information Sciences, 178*, 3043–3074.

Zhao, F., Zhang, J., Wang, J., & Zhang, C. (2015a). A shuffled complex evolution algorithm with opposition-based learning for a permutation flow shop scheduling problem. *International Journal of Computer Integrated Manufacturing, 28*, 1220–1235.

Zhao, J., Lv, L., & Sun, H. (2015b). Artificial bee colony using opposition-based learning. In *Genetic and evolutionary computing* (pp. 3–10). Springer.