Applied Intelligence
https://doi.org/10.1007/s10489-018-1362-4

A hybrid whale optimization algorithm based on modified differential evolution for global optimization problems

Jun Luo[1] · **Baoyu Shi**[1]

Abstract

Whale optimization algorithm(WOA) is a biological-inspired optimization algorithm with the advantage of global optimization ability, less control parameters and easy implementation. It has been proven to be effective for solving global optimization problems. However, WOA can easily get stuck in the local optimum and may lose the population diversity, suffering from premature convergence. In this work, a hybrid whale optimization algorithm called MDE-WOA was proposed. Firstly, in order to enhance local optimum avoidance ability, a modified differential evolution operator with strong exploration capability is embedded in WOA with the aid of a lifespan mechanism. Additionally, an asynchronous model is employed to accelerate WOA's convergence and improve its accuracy. The proposed MDE-WOA is tested with 13 numerical benchmark functions and 3 structural engineering optimization problems. The results show that MDE-WOA has better performance than others in terms of accuracy and robustness on a majority of cases.

Keywords Whale optimization algorithm · Differential evolution · Global optimization · Benchmark functions

1 Introduction

Global optimization problems(GOPs) are ubiquitous in engineering fields. Without loss of generality, GOP can be defined as follows.

Find $\mathbf{X}$ which optimizes $f(\mathbf{X})$
subject to:

$$\begin{aligned} &g_i(\mathbf{X}) \le 0, \quad i = 1, ..., n \\ &h_j(\mathbf{X}) = 0, \quad j = 1, ..., p \\ &lb_i \le x_i \le ub_i, \quad i = 1, ..., d \end{aligned} \tag{1}$$

where n is the number of inequality constraints and p is the number of equality constraints. $\mathbf{X}$ is the vector of solutions and d indicates the dimensions of variables. lb_i and ub_i denotes the range of variables.

Traditional methods to solve GOPs can be classified into two groups: gradient-based and direct methods. For gradient-based search methods, the first or/and second-order derivatives of the objective function and constraints are used to guide the search, whereas direct methods only use objective function and constraint value. The direct search methods usually require many function evaluations for convergence because derivative information is not used. The gradient-based methods can converge to an optimal or near-optimal solution quickly, but they are not efficient in non-differentiable problems. Additionally, the gradient search may rely on the position of an initial point if there are various local optimums in the problem. Furthermore, the search result may become unstable when the objective function and constraints have sharp peaks. The drawbacks of traditional numerical methods have forced researchers to focus on meta-heuristic algorithms. As the complexity and scale of real engineering design optimization problems has increased, meta-heuristic algorithms have been utilized as primary techniques for solving the GOPs [1, 2]. Most meta-heuristic algorithms are inspired from natural phenomena, and some of the widely used meta-heuristic algorithms include particle swarm optimization (PSO) inspired by the social behavior of bird flocking [3], ant colony optimization (ACO) mimicking the ants' foraging behavior [4], artificial bee colony (ABC) simulating the intelligent foraging behavior of honey bees [5], bat Algorithm (BA) based on the echolocation behavior of bats [6]. Researches have shown that meta-heuristic algorithms have great potential. Although the above algorithms have many advantages, the no free lunch (NFL) theorem has proven that none of these algorithms can solve all kinds of optimization problems [7]. The good performance of an optimizer for a

✉ Jun Luo
luojun@cqu.edu.cn

[1] Chongqing University, Chongqing 400030, China

Published online: 19 December 2018

set of optimization problems does not guarantee its success in solving a different set of problems. This motivates researchers to develop novel meta-heuristic algorithms for optimization. Some of the recently proposed are fruit fly optimization algorithm (FOA) [8], grey wolf optimization (GWO) [9], salp swarm algorithm (SSA) [10, 11], grasshopper optimization algorithm (GOA) [12], crow search algorithm (CSA) [13], sine cosine Algorithm (SCA) [14], bird swarm algorithm (BSA) [15], chicken swarm optimization (CSO) [16] and so on.

Based on the bubble-net hunting behavior of humpback whales, a novel algorithm named Whale Optimization Algorithm(WOA) was proposed in 2016 [17]. Compared with other well-known meta-heuristic methods, WOA has proved to be more competitive.WOA has been successfully applied to various engineering optimization problems, such as parameter estimation of photovoltaic cells [18], optimal sitting of capacitors in radial distribution network [19], wind speed forecasting [20], spam profile detection [21], multilevel threshold image segmentation [22], size optimization of skeletal structures [23], forecasting on energy-related carbon dioxide emissions [24] and so on. WOA has already attracted many researchers' attention for the scarcity of its control parameters and ease of implementation [25, 26]. However, through exploring optimization mechanism, it's not difficult to find that search agents get only close to the best solution over the latter half process of iteration, which makes WOA easily trap into local optima, resulting in losing population diversity and poor at balancing exploitation and exploration. In order to solve these defects and achieve good performance on dealing with GOPs, some strategies should be improved.

Differential Evolution (DE) was proposed by Storn and Price in 1997 [27]. Although it has strong global search capability due to its mutation scheme, it converges slowly and is weak in exploitation. Hybrid meta-heuristics is one of the most efficient methods for enhancing the performance of each technique, and it has been testified to be more efficient and converge faster than methods that use solely individual components. In the past few years, DE was widely promoted by combing with other meta-heuristic algorithms. Liang et al. [28] invented an improved artificial bee colony algorithm with adaptive differential operators, adaptively adjusting the two parameters (scale factor F, crossover rate C_r) of differential operators through Gaussian distribution. Zhu et al. [29] designed a hybrid grey wolf optimization with differential evolution operators called HGWO, which integrated DE into GWO to update the previous best position of grey wolves and tested scheduling for 3D stacked SoC. Gao et al. [30] put forward a hybrid algorithm called DGABC, and DE was combined with gbest-guided ABC by an evaluation scheme, which selected the search method based on the prior experience. DGABC attempted to take advantage of the previous search experience to accelerate the convergence speed. Sayah et al. [31] hybridized particle swarm optimization with differential evolution algorithm to handle with nonconvex economic dispatch problems. The main idea is to combine the PSO equation into the DE algorithm as an additional mutation operator. This strategy could improve the exploitation ability of DE and avoid the diversity loss of partial swarm. Literature [32] proposed a H-CRO-DE algorithm which combined Chemical Reaction Optimization and Differential Evolution to solve global optimization problems. The differential evolution random strategy was employed as a global search operator to increase the diversity of population and thus improve the exploration ability. The experimental results of these works demonstrated the performance of algorithms could be significantly enhanced by combing with DE. Therefore, fusing DE into WOA is a good approach to overcome the drawbacks of WOA and DE.

As discussed above, the standard DE has some advantages but it does not use the elite information about the search space and has no mechanism to memorize the previous process, so it easily results in a waste of computing power and needs a better balance between exploration and exploitation when it is hybridized with WOA. Swagatam et al. [33] described a variant of the DE /target-to-best/ strategy called DEGL, which linearly combined global and local mutation operators through a new parameter with an objective of trading off their effects. In this paper, a hybrid algorithm named MDE-WOA is proposed, combining modified DEGL algorithm and WOA. Specifically, the proposed algorithm adopts an aging leader mechanism to select the search method, namely, the selection of search equation depends on the lifespan of global best solution. Additionally, to improve the convergence and accuracy, an asynchronous model is employed to update the global best solution. Finally, a series of benchmark functions are employed to test the effectiveness of proposed algorithm.

The aim of this research work is to handle the problems of weak diversity and premature convergence in the WOA and the main contributions of our study can be summarized as follows:

- A novel hybridization approach based on WOA and modified DE is proposed.
- A modified DE algorithm is incorporated to WOA through a lifespan mechanism. Additionally, an asynchronous model is employed to improve the accuracy.
- The basic WOA, DE, three DE variants and five other well-known metaheuristic algorithms, namely PSO, SCA, GWO, chaotic salp swarm algorithm (CSSA) [34], adaptive fireworks algorithm (AFWA) [35] are

chosen as methods comparative analysis by thirteen unconstrained benchmark functions.
- Six evaluation criteria including mean, standard deviation, p-values of the Wilcoxon rank sum, convergence curves, success rate and runtime are used.
- The proposed MDE-WOA algorithm has been utilized to solve three constrained engineering optimization problems, such as (a) the optimal design problem of pressure vessel, (b) Tension/Compression spring design and (c) welded beam design.

The rest of this paper is organized as follows. The basic whale optimization algorithm and its limitations are briefly introduced in Section 2. Section 3 presents a detailed description of the proposed algorithm. In Section 4, a set of benchmarks and engineering design problems are employed to validate the performance of the algorithm through comparing the results with WOA and other well-known algorithms. Finally, conclusions are presented in Section 5.

2 Whale optimization algorithm and its limitations

The WOA is a new meta-heuristic optimization algorithm inspired from the bubble-net hunting behavior of humpback whales. There are three operators including searching for prey randomly, encircling prey and bubble-net foraging. The bubble-net attacking is done by creating bubbles along a constricted circle and spiral shape path simultaneously. Humpback whale follows the bubbles to swim up toward the surface and encircle prey. Mathematical model is given as follows.

2.1 Whale optimization algorithm

2.1.1 Searching for prey

Humpback whales search for prey randomly in terms of the position of each other. The mathematical model can be described as follows:

$$\begin{aligned} \mathbf{D} &= |\mathbf{C} \cdot \mathbf{X}_{rand} - \mathbf{X}_t| \\ \mathbf{X}_{t+1} &= \mathbf{X}_{rand} - \mathbf{A} \cdot \mathbf{D} \end{aligned} \tag{2}$$

where $\mathbf{X}$ is the position of an individual, and $\mathbf{X}_{rand}$ is randomly selected from the current generation and represents a position vector, t indicates the current iteration, the symbol || is the absolute value, $\mathbf{A}$ and $\mathbf{C}$ are coefficient vectors. It is worth mentioning that the value of $\mathbf{A}$ is greater than 1 or less than -1 in this phase. $|\mathbf{A}| \geq 1$ forces search agent to move far away from a reference whale. Therefore, WOA emphasizes exploration in various unknown regions in this phase.

The coefficient vectors $\mathbf{A}$ and $\mathbf{C}$ are calculated as in (3):

$$\begin{aligned} A &= 2\mathbf{a} \cdot \mathbf{r} - \mathbf{a} \\ \mathbf{C} &= 2 \cdot \mathbf{r} \\ \mathbf{a} &= 2 - \frac{2t}{t_{\max}} \end{aligned} \tag{3}$$

where $\mathbf{r}$ is a random vector in the range from 0 to 1, $t_{\max}$ is the max number of iterations, and $\mathbf{a}$ is linearly decreased from 2 to 0 over the course of iterations.

2.1.2 Encircling prey

The best candidate solution obtained so far is considered as the target prey and the other search agents aim at updating their positions towards the best solution. To describe the encircling behavior, the following equations are put forward:

$$\begin{aligned} \mathbf{D} &= \left|\mathbf{C} \cdot \mathbf{X}_{g\_best} - \mathbf{X}_t\right| \\ \mathbf{X}_{t+1} &= \mathbf{X}_{g\_best} - \mathbf{A} \cdot \mathbf{D} \end{aligned} \tag{4}$$

where $\mathbf{X}_{g\_best}$ is the best solution acquired so far, and should be updated after each iteration if there is a better solution. When $|\mathbf{A}|$ is less than 1, the search agent will update its position around the best solution and encircle the prey in this exploitation phase. To clearly see the effects of (4), a two-dimensional position vector of whales and some potential neighbors are demonstrated in Fig. 1. As can be seen from this picture, each individual can reach different places around the prey by adjusting the coefficient vectors $\mathbf{A}$ and $\mathbf{C}$. Additionally, random vector $\mathbf{r}$ can assist each search agent in arriving in any situations among the presented points in Fig. 1.

2.1.3 Bubble-net attacking

In order to simulate the bubble-net foraging behavior of humpback whales, two approaches are proposed:

1. Shrinking encircling mechanism: This technique is achieved by decreasing the value of $\mathbf{a}$ linearly and $\mathbf{A}$ is a random vector in the range [-$\mathbf{a}$, $\mathbf{a}$] according to (3).
2. Spiral updating position: A spiral equation is used to mimic the helix-shaped shift of humpback whales as in (5).

$$\mathbf{X}_{t+1} = \mathbf{D}' \cdot e^{bl} \cdot \cos(2\pi l) + \mathbf{X}_{g\_best} \tag{5}$$

where $\mathbf{D}' = \left|\mathbf{X}_{g\_best} - \mathbf{X}_t\right|$ shows the distance between the ith whale and prey, l is a random value in [-1,1] and b is a constant for defining the spiral shape.

In WOA, there is 50% probability that search agents choose either the encircling mechanism or the spiral path

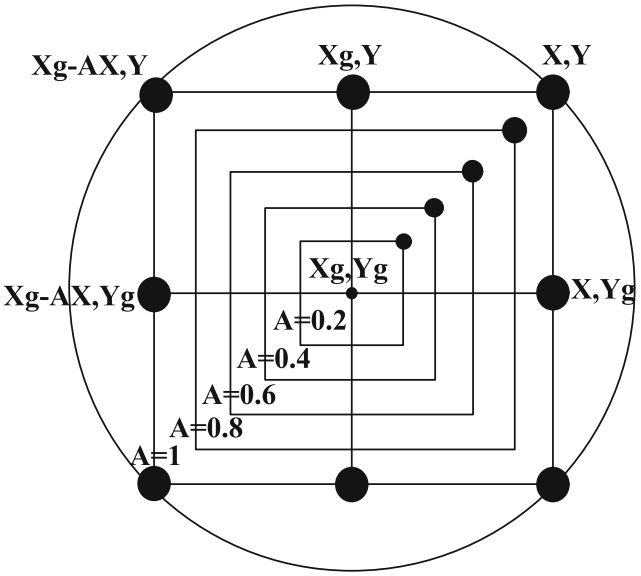

Fig. 1 shrinking encircling mechanism

through a random variable p. The general steps of the WOA can be summarized in the pseudo code shown in Fig. 2.

2.2 Limitations of WOA

In WOA, we can summarize that (4) forces the whales to reposition themselves around the best solution and (5) makes them move in a spiral-shaped path towards the prey rapidly. These two equations are like local search which provides high exploitation. Equation (2) requires whales to diverge from the prey and move randomly around each other during the initial steps of the iteration that is similar to global search. Since these phases are done alternatively, the WOA includes exploration and exploitation ability.

As we all know the random vector $\mathbf{A}$ allows the WOA to balance exploration and exploitation, some iterations are applied to exploration ($|\mathbf{A}| \geq 1$) and the rest are devoted to exploitation ($|\mathbf{A}| < 1$). However, in some cases, search agents of WOA are tended to stagnate in local optima after exploitative motions. According to (3), search agents only communicate with the elite vector of population during the latter half of the iteration. Then the population diversity will drop rapidly and the likelihood of getting into local optimum will increase. For instance, $|\mathbf{A}|$ is always less than 1 in the last 500 iterations when the max iteration number is set to 1000, then all of the search agents can only update their positions by (4) and (5). In general, the WOA is likely to fall into stagnation, which is a common disadvantage of local search operation. Once the population falls into the local peak, WOA has no effective mechanism to jump out the attraction of the local optimum.

Therefore, we propose a hybrid WOA with the aim of mitigating the stagnation problem.

3 Hybrid whale optimization algorithm based on modified differential evolution operators

3.1 Motivation

As previously mentioned, WOA can be considered as a global optimizer because it has exploration and exploitation ability, but it is attracted by elite vector in the latter process, then the whole whale population is easy to fall into prematurity. One approach to overcome this shortcoming is to use a hybrid model where at least two algorithms are combined to enhance the performance of each optimizer. According to Talbi [36], combining a meta-heuristic with a complementary algorithm is a common manner of hybridization model which can be divided into two groups: low-level and high-level. In the low-level hybridization, a given operator in a meta-heuristic is replaced with another search strategy. In contrast, self-contained meta-heuristics are executed in a sequence in high-level hybrid model.

In this paper, the low-level teamwork hybrid model (LTH) is employed to alleviate the stagnation problem in WOA. Specifically, modified differential evolution (MDE) operators are used as components in the WOA on the basis of a lifespan mechanism. WOA and MDE share a single population. In addition, the basic WOA utilizes synchronization model to update the $\mathbf{X}_{g\_best}$. In most cases, asynchronous model converges faster than the synchronization model. Therefore, MDE-WOA updates the optimal solution by asynchronous mechanism.

3.2 Lifespan mechanism

As Goldsmith pointed out [37], an appropriate lifespan is an important and significant characteristic for evolution because it is good for increasing the diversity of the population. In MDE-WOA, a lifespan mechanism is used to determine when to embed modified differential evolution operators into WOA. It is assumed that the initial lifespan of the leader is S and its current age is s. The age control for the leader is adaptively adjusted on the basis of the leader's power, t indicates the current iteration. The updating formula of s can be expressed as:

$$s = \begin{cases} s+1 & \text{if } (\delta_{\mathbf{X}_{g\_best}} = 0) \\ s & \text{if } (\delta_{\mathbf{X}_{g\_best}} < 0) \end{cases} \tag{6}$$

where $\delta_{\mathbf{X}_{g\_best}} = f(\mathbf{X}_{g\_best,t}) - f(\mathbf{X}_{g\_best,t-1})$. We assumed that the best solution obtained so far is the minimum of object function. When the current population finds a better solution than the prey in parent population, the age of leader s remains unchanged. On the contrary, it will increase by one. If s equals to S (*the fitness of* $\boldsymbol{X}_{g\_best}$ *has not been*

Fig. 2 The procedure of WOA

Algorithm: WOA

Initialize the whales population (i = 1, 2, ..., NP)
Calculate the fitness of each search agent, $\vec{X}_{g\_best}$*=the best search agent*
while (t < maximum iterations)
  for each search agent
    Update ***a****,* ***A****,* ***C****, l, and p*
    if1 (p<0.5)
      if2 *(|**A**| < 1)*
        Update the position of the current search agent by the ***Eq. (4)***
      else if2 *(|**A**|≥1)*
        Select a random search agent
        Update the position of the current search agent by the ***Eq. (2)***
      end if2
    else if1 (p≥0.5)
      Update the position of the current search by the ***Eq. (5)***
    end if1
    Check if any search agent goes beyond the search space and amend it
    Calculate the fitness of each search agent, update $\vec{X}_{g\_best}$ *if there is a better solution*
  end for
  t=t+1
end while
return $\vec{X}_{g\_best}$

updated for S times), (4) will be replaced by the improved differential evolution search strategy.

3.3 Differential evolution with global and local neighborhood-based operators

DE is one of the most popular evolutionary algorithms to solve GOPs due to its simplicity and remarkable performance [38]. The classic DE employs mutation, crossover and selection operators to find the optimum at each iteration. Each individual considered as a parent generates an offspring by three steps: firstly, a donor vector is created by mutation, then the trial vector is produced by crossover between the donor and parent vector, finally greedy selection between parent and the trial vector is performed where the better one will be chosen as offspring. The performance of DE mainly depends on two factors, one is the trial vector generation strategy and the other is control parameters (population size NP, scaling factor F, crossover control parameter C_r) [39].

Although DE is successfully applied in solving a number of practical problems, it also has some drawbacks. Therefore, a lot of works have been done to improve the performance of DE [40]. The improvement strategies can be divided into three main categories: adaption and self-adaption variants, which adjust the control parameters adaptively or self-adaptively. Examples of this class are JADE [41], SADE [42], LSHADE [43] and so on. In JADE, a new mutation strategy DE /current-to-pbest/ with external optional archive is implemented and control parameters are updated in an adaptive manner. The second group is memetic DE algorithms, which combines global search scheme with local search methods. Algorithm proposed in [44] employs three local methods to coordinate with classical DE. The third group is topological neighborhood-based DE algorithms, which introduces global and local neighborhood -based mutation operators. One of the topological methods is DEGL which modifies DE/target-to-best/ mutation operator by integrating local neighborhood-based and global mutation models. DEGL could balance the exploration and exploitation abilities without imposing serious additional burdens. This paper embeds a modified DE (MDE) operators into WOA which uses the concept of neighborhood mutation operator in DEGL. The details of MDE will be described as follows.

3.3.1 Mutation

MDE made up of the local model and global mutation model utilizes the concept of the neighborhood of each individual. The local model is mutated using the best solution obtained so far in a small neighborhood of current search agent, devoting to explore unknown regions of feasible search space. The global model takes the globally best vector $\mathbf{X}_{g\_best,t}$ of the entire population into account for mutation operation, which contributes to converge to the near-optimal solution as quickly as possible. Finally, we combine the local model and global model through a weight factor in order to acquire the final donor vector. The balance between exploration and exploitation is realized with the weight factor.

For each vector $\mathbf{X}_{i,t}$, the neighborhood of radius k (*where* k *is a nonzero integer in the range*$[1, (NP-1)/2]$) is defined. If k equals to 1, the neighborhood shown in Fig. 3 consists of vectors $\mathbf{X}_{i-1,t}$, $\mathbf{X}_{i,t}$, $\mathbf{X}_{i+1,t}$. The local donor vector is created by the best solution in the neighborhood of $\mathbf{X}_{i,t}$ and two other vectors randomly chosen from the same neighborhood. The model can be expressed as in (7).

$$\mathbf{L}_{i,t} = \mathbf{X}_{i,t} + \alpha_l \cdot (\mathbf{X}_{l\_best,t} - \mathbf{X}_{i,t}) + \beta_l \cdot (\mathbf{X}_{p,t} - \mathbf{X}_{q,t}) \quad (7)$$

where $\mathbf{X}_{l\_best,t}$ is the best solution in the neighborhood of $\mathbf{X}_{i,t}$, and $p, q \in [i-k, i+k](p \neq q \neq i)$, α_l, β_l are random disturbance factors based on the fixed scaling factor F, $\alpha_l = \beta_l = \lambda \times rand(NP, D) + F$, the disturbance could cut down the probability of falling into local optimum. Similarly, the global donor vector is described as in (8).

$$\mathbf{G}_{i,t} = \mathbf{X}_{g\_best,t} + F \cdot (\mathbf{X}_{g\_best,t} - \mathbf{X}_{i,t}) + F \cdot (\mathbf{X}_{r_1,t} - \mathbf{X}_{r_2,t}) \quad (8)$$

where $\mathbf{X}_{g\_best,t}$ is the optimum vector acquired so far in the i^{th} iteration, and r_1, r_2 are random numbers within the global range. It is worth mentioning that the first item in (8) utilizes the globally best vector instead of $\mathbf{X}_{i,t}$ in DEGL, and it could improve the convergence ability. The final donor vector is calculated as follows.

$$\mathbf{V}_{i,t} = \omega \cdot \mathbf{G}_{i,t} + (1-\omega) \cdot \mathbf{L}_{i,t} \quad (9)$$

where the weight factor $\omega \in [0, 1]$, small value of ω makes the MDE vulnerable to local neighborhood component, and brings about better exploration. In contrast, large value of ω favors the global solution and improves exploitation. In order to reduce control parameters and maintain a balance, the ω is set to middle point 0.5.

3.3.2 Crossover

After generating the donor vector, crossover operation is employed to further increase the population diversity. Exponential and binomial distributions are the most common crossover strategies. Here binomial crossover is

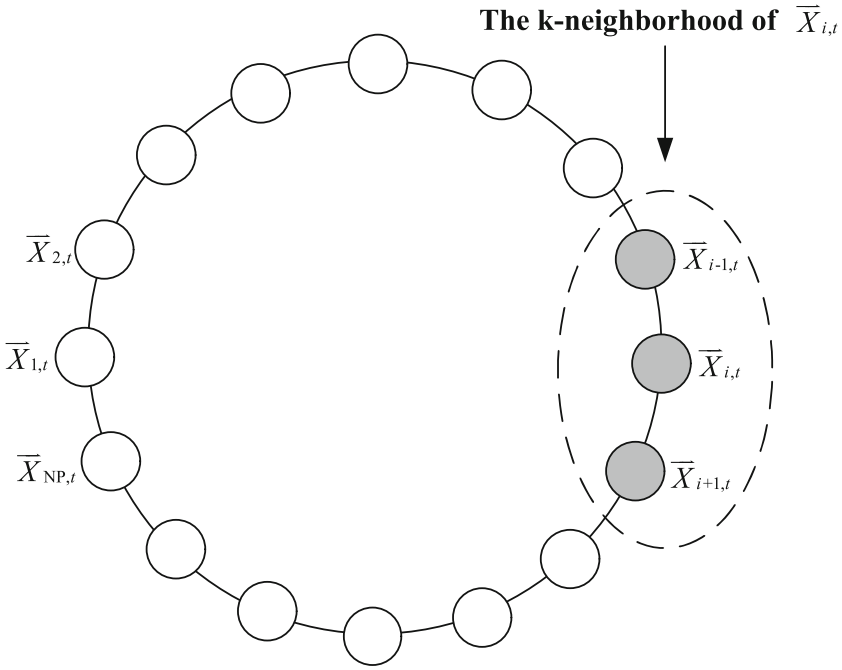

Fig. 3 The ring topology of k-neighborhood, where k = 1

Algorithm: MDE-WOA

Initialize the whales population (i = 1, 2, ..., NP),scaling factor F, crossover rate Cr ,lifespan S
Calculate the fitness of each search agent, $\vec{X}_{g\_best}$*=the best search agent, s=0*
while (t < maximum iterations)
  for each search agent
    Update ***a****,* ***A****,* ***C****, l, and p*
    if1 (p<0.5)
      if2 *(|**A**| < 1)& (s<S)*
        Update the position of the current search agent by the ***Eq. (4)***
      else if2 *(|**A**|≥1) & (s<S)*
        Select a random search agent
        Update the position of the current search agent by the ***Eq. (2)***
      else if2*(s=S)*
        Local and global neighborhood-based mutations
          Generate a donor vector by the ***Eq. (9)***
        Crossover
          Generate a trial vector by the ***Eq. (10)***
        Greedy selection
          Evaluate the trial vector and update the position of the current search by the ***Eq. (11)***
      end if2
    else if1 (p≥0.5)
      Update the position of the current search by the ***Eq. (5)***
    end if1
    Check if any search agent goes beyond the search space and amend it
    Calculate the fitness of each search agent, update $\vec{X}_{g\_best}$ *if there is a better solution*
  end for
  t=t+1, update s by the ***Eq. (6)***
end while
return $\vec{X}_{g\_best}$

Fig. 4 Framework of MDE-WOA

adopted. The donor vector $\mathbf{V}_{i,t}$ exchanges its components with $\mathbf{X}_{i,t}$ through crossover operator to form the trial vector.

$$\mathbf{U}_{i,j} = \begin{cases} \mathbf{V}_{i,j} & rand \leqslant C_r \ or \ j = j_{rand} \\ \mathbf{X}_{i,j} & other \end{cases} \tag{10}$$

where $j_{rand} \in [1, 2, ...D]$ is randomly chosen from index, which can guarantee that the trial vector gets at least one component from $\mathbf{V}_{i,t}$. C_r controls the crossover probability.

3.3.3 Selection

The greedy selection mechanism (the better between target and trial vector is selected) employed by DE could be expressed as:

$$\mathbf{X}_{i,t+1} = \begin{cases} \mathbf{U}_{i,t} & f(\mathbf{U}_{i,t}) \leqslant f(\mathbf{X}_{i,t}) \\ \mathbf{X}_{i,t} & f(\mathbf{U}_{i,t}) > f(\mathbf{X}_{i,t}) \end{cases} \tag{11}$$

Table 1 Description of unimodal benchmark functions

Function	Dim	Range	Equation	fmin
Sphere	30	[−100,100]	$f_1(x) = \sum_{i=1}^{n} x_i^2$	0
Schwefel 2.22	30	[−10,10]	$f_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	0
Schwefel 1.2	30	[−100,100]	$f_3(x) = \sum_{i=1}^{n} \left(\sum_{j=1}^{i} x_j \right)^2$	0
Schwefel 2.21	30	[−100,100]	$f_4(x) = \max_i \{ \lvert x_i \rvert , 1 \le i \le n \}$	0
Rosenbrock	30	[−30,30]	$f_5(x) = \sum_{i=1}^{n-1} \left[100 \left(x_{i+1} - x_i^2 \right) + (x_i - 1)^2 \right]$	0
Step	30	[−100,100]	$f_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	0
Quartic	30	[−1.28,1.28]	$f_7(x) = \sum_{i=1}^{n} i x_i^4 , + random\,[0, 1)$	0

3.4 The flowchart of MDE-WOA

For clarity, the pseudo code of MDE-WOA is given in Fig. 4.

4 Results and discussion

To validate the performance of MDE-WOA from different perspectives, several experiments by solving 13 classical benchmark functions and 3 structural constraint engineering problems [17] are conducted. These benchmark functions are listed in Tables 1 and 2, where range denotes the boundary of search space, Dim indicates the dimension of variables and fmin is the optimum of function. These functions can be divided into two types: unimodal test functions (Table 1), and high-dimension multimodal functions (Table 2). Unimodal test functions have only one global optimum, aiming at validating the exploitation ability and convergence of algorithm. The high-dimension multimodal cost functions have various local optima. An algorithm should avoid all the local optima to approach and approximate the global optimum. Therefore, local optima avoidance and exploration of algorithm can be tested by multi-modal cost functions. These two types of cases with different difficulty levels are chosen to efficiently benchmark the performance of the algorithms from different perspectives.

The experimental environment was a 3.80 GHz AMD Athlon(tm)X4 760K CPU with RAM of 4GB in the Windows7 system. Each of the experiments was repeated 30 times independently on the Matlab 7.1.

4.1 Analysis of parameters

The parameters of MDE-WOA are determined through preliminary numerical experiments or based on common settings. We empirically choose $C_r = 0.9$, $F = 0.8$, $k = 3$ (*neighborhood size equal to 10% of the population provides reasonably accurate results*) according to the literature [33].The population size $NP = 60$, maximum number iteration is 1000, the λ that controls the disturbance range is 0.0001.

The lifespan S of leader has great impact on the population diversity. If the age of leader increases from 0 to S, the MDE-WOA needs to replace the strategy of encircling prey by MDE operators. The higher S is, the faster diversity reduction is. Introducing MDE into WOA too early may lead to the decrease of exploitation ability of WOA, so we need to test a suitable S. In this experiment, four typical benchmark functions were chosen to test the MDE-WOA algorithm with S=70,80,90,100,110,120 and 130, respectively. For MDE-WOA, f_1 and f_2 are unimodal so the global minimum are not difficult to find. The valley of f_5 is easy to find, but realizing the global convergence is difficult. It is also difficult to find the global optima for WOA on f_{12}, because the whole whale population is easy to fall into prematurity when WOA is solving multi-modal function. All these functions were tested with dimensions 30, and all experiments was repeated 30 times independently. The reported results are the means and standard deviations of the statistical data.

Table 3 shows the statistical optimization results. It can be seen that MDE-WOA with S=70-100 underperforms WOA for f_1 and f_2, because small S can not make full use of exploitation and quick convergence abilities of WOA. MDE-WOA with S=70-120 outperforms WOA for f_5 and f_{12}, because small S contributes to the increase of diversity. The results of MDE-WOA with $S = 110$ are identical to that of $S = 120$. Both of them obtain the superior results for all the functions (f_1, f_2, f_5, f_{12}). It can be observed that for f_5 and f_{12}, the mean value begins to increase (which means the solution gets worse) when S equals to 130. Therefore, we can conclude that S in the interval [110,120] is suitable for MDE-WOA to balance exploration and exploitation. Finally, we set S as 120 to conduct the below experiments.

Table 2 Multimodal benchmark functions

Function	Dim	Range	Equation	fmin
Schwefel 2.26	30	$[-500,500]$	$f_8(x) = \sum_{i=1}^{n} -x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	-418.9829×30
Rastrigin	30	$[-5.12,5.12]$	$f_9(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	0
Ackley	30	$[-32,32]$	$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}\right) - \exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right) + 20 + e$	0
Griewank	30	$[-600,600]$	$f_{11}(x) = \frac{1}{400}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	0
Penalized1.1	30	$[-50,50]$	$f_{12}(x) = \frac{\pi}{n}\left\{10\sin(\pi y_1) + \sum_{i=1}^{n-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right] + (y_n-1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$ $y_i = 1 + \frac{x_i+1}{4}$, $u(x_i, a, k, m) = \begin{cases} k(x_i-a)^m & x_i > a \\ 0 & a < x_i < a \\ k(-x_i-a)^m & x_i < -a \end{cases}$	0
Penalized1.2	30	$[-50,50]$	$f_{13}(x) = 0.1\left\{\sin^2(3\pi x_1) + \sum_{i=1}^{n}(x_i-1)^2\left[1+\sin^2(3\pi x_i+1)\right] + (x_n-1)^2\left[1+\sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	0

4.2 Accuracy and stability test

To validate the accuracy and stability of proposed algorithm, the MDE-WOA is compared with basic WOA, basic DE [27] and three DE variants including DEGL [33], JADE [41] and CoDE [45], and several classical or powerful meta-heuristic algorithms like PSO [3], SCA [14], GWO [9], CSSA [34], and AFWA [35]. These nature-inspired algorithms have been investigated by benchmark problems, and the results demonstrated that they have good performance in search efficiency. This experiment was run 30 times independently for 13 benchmark functions. To make the comparison fair, we set the same population size (60) and iteration number (1000) for all algorithms. Additionally, the population for all problems tested was initialized using the same random seeds. The parameters of DE are configured with: $F = 0.5$; C_r=0.9. The PSO parameter configuration is carried out according to the literature [46]: Vmax=6; wMax=0.9; wMin=0.2; c1=2; c2=2. The common parameters of DEGL and MDE-WOA are configured with: $F = 0.8$; $C_r = 0.9$; $\omega = 0.5$; $k = 3$. For other parameters, we follow these settings in the original papers. The statistical results are reported in Tables 4 and 5. Note that the best and worst values, the average value of the minimums (ave) and their standard deviations (std) are provided. Moreover, statistical test, considering each run's results and proving that the results are statistically significant, should also be conducted due to the stochastic nature of the algorithms. Therefore, the Wilcoxon rank-sum test is carried out to verify if two sets of solutions have significant difference or not. Note that the best outcomes among all algorithms are shown in boldface.

According to Table 4, the MDE-WOA is quite competitive compared with others. It almost outperforms all others for the best and worst results on $f_1 - f_{13}$ except f_3, f_4, f_{10}. For f_3 and f_4, it is GWO that obtains the best results while MDE-WOA follows it. WOA is the most efficient optimizer for f_{10} while MDE-WOA achieves near accuracy. In particular, the results of MDE-WOA for f_6, f_9, f_{11} obtain the theoretical optimal solution zero. The results for $f_1 - f_{13}$ reported in Table 4 obviously illustrate that the improved algorithm performs better than others in terms of solving accuracy. From Table 5, it can be obviously concluded that the MDE-WOA's average f_{min} and standard deviation on nearly all functions are less than the other nine algorithms. Specifically, MDE-WOA provides the best or near results on eleven out of thirteen test functions, and it ranks second in two other test functions including f_3 and f_4. Table 6 shows p-values obtained from Wilcoxon's rank sum test for the results of MDE-WOA against other meta-heuristics at 5% significance level. In this work, MDE-WOA is statistically significant if and only if it results in a p-value less

Table 3 Statistical results of the functions by WOA algorithm and MDE-WOA with S =70-130 respectively

F	Statistics	WOA	S =70	S =80	S =90	S =100	S =110	S =120	S =130
f_1	Ave	2.78E-179	1.78E-55	4.98E-60	5.88E-60	3.08E-61	1.25E-286	**1.42E-289**	2.46E-289
	Std	0	9.70E-55	1.66E-59	2.92E-59	1.48E-60	5.33E-300	0	0
f_2	Ave	2.58E-112	5.08E-27	5.60E-29	2.91E-29	1.64E-35	8.27E-158	**3.59E-162**	5.16E-161
	Std	4.69E-112	1.90E-26	1.50E-28	1.27E-28	4.21E-34	4.17E-159	1.95E-161	1.58E-160
f_5	Ave	2.58E+01	**6.77E-18**	9.49E-18	1.61E-17	9.10E-18	5.04E-17	1.63E-16	2.86E-05
	Std	2.77E-01	2.44E-17	2.36E-17	4.61E-17	2.36E-17	1.95E-16	4.51E-16	1.89E-06
f_{12}	Ave	2.36E-04	**1.42E-33**	8.75E-33	1.59E-32	3.64E-32	1.57E-32	1.57E-32	1.39E-15
	Std	2.71E-05	2.88E-50	6.63E-42	1.71E-35	6.62E-33	2.33E-34	1.84E-34	1.26E-16

The bolded entries in the above tables are used to highlight the best obtained results

than 0.05. As it can be observed, the p-values are more than 0.05 between MDE-WOA and basic WOA for f_7, f_9, f_{10} and f_{11}. Both of them have obtained the theoretical optimal solution zero for f_9 and f_{11} and obtained near results for f_7 and f_{10}. Except such cases, most of p-values are far less than 0.05, showing this superiority is statistically significant. From these three tables, it can be concluded that the accuracy and stability of the proposed algorithm are greatly improved.

4.3 Convergence behavior analysis

To confirm the convergence of the proposed algorithm, the convergence curves of MDE-WOA and other seven algorithms based on eight cases are provided in Fig. 5. Note that the average optimum solution obtained so far denotes the average of best fitness value in each iteration 30 times.

From the simulation results, we can see that MDA-WOA converged rapidly in the early stage of the iteration process due to the asynchronous model which helps agent search for the promising regions and converges to the optima more quickly than WOA. This behavior is evident in all test functions. It can be observed that WOA is easy to fall into stagnation while whales are swimming around the prey with a shrinking circle and along a spiral-path. This phenomenon is obvious in f_5, f_6, f_{12}, f_{13}. MDE-WOA has a strong ability to jump out from local minimum for f_5, f_6, f_{12}, f_{13} and thus can obtain a better solving accuracy than basic WOA, because we embedded MDE with strong search ability in later evaluations. Especially for f_{12} and f_{13}, the curve clearly indicates that MDE-WOA effectively escapes from the attraction of local best and can find a better solution in the middle and later evaluations. The comparison of convergence curves in some of the benchmark problems further shows that the superiority of proposed method.

4.4 Successful rate and convergence rate test

The successful rate is an important target to measure the performance of the algorithm which is defined as in Eq. (14).

$$SR = \frac{N_s}{N} \times 100\% \tag{12}$$

where SR is the successful rate, N is the total number of experiments, and N_s is the number of times of attaining the predefined accuracy. Moreover, average runtime is another important target, which marks the complexity of the algorithm. To fairly compare the speed and successful rate of different algorithms, we run each algorithm on $f_1 - f_7$ and $f_9 - f_{13}$ with 30 dimensions for 30 times and stop as soon as the best fitness value falls below the expected threshold or the current iteration exceeds the max iterations.

From Table 7, we can see that MDE-WOA has the best performance in aspect of SR and runtime. For sphere function (f_1) , nearly all of the eight algorithms can acquire 100% successful rate but the advantage of MDE-WOA is obvious on runtime. Although MDE-WOA is more complex than WOA due to the improved DE operators, MDE-WOA converges below the expected threshold faster than other seven algorithms. Not only does MDE-WOA acquire the highest successful rate for nearly all of 12 benchmark functions, but it also consumes the least amount of computational time, particularly for the complex multimodal functions. Although in some cases MDE-WOA consumes more computing time, it obtains higher successful rate, such as f_{11} compared with GWO.

4.5 Effectiveness of the two modifications

In this subsection, we cover experiments to validate the effectiveness of each modifications performed in two

Table 4 Comparison of results (**Best, Worst**) gained for the unimodal and multimodal benchmark functions

F	Statistics	MDE-WOA	WOA	DE	PSO	DEGL	JADE	CoDE	SCA	GWO	CSSA	AFWA
f_1	Best	**1.73E-295**	9.53E-192	9.18E-14	1.51E-13	3.22E-58	3.12E-39	1.63E-11	1.65E-08	2.07E-76	7.14E-09	1.34E-13
	Worst	**7.03E-286**	1.17E-175	2.39E-13	2.97E-12	8.47E-55	5.76E-31	2.74E-10	9.10E-02	2.03E-73	1.76E-08	5.61E-11
f_2	Best	**7.70E-164**	1.36E-115	2.00E-07	5.61E-08	3.79E-27	1.64E-18	1.17E-06	9.83E-09	3.66E-44	7.10E-05	5.48E-08
	Worst	**2.45E-161**	1.28E-112	8.40E-07	2.51E-06	5.37E-24	1.19E-11	5.94E-06	1.26E-05	1.63E-42	3.08E+00	3.33E-01
f_3	Best	1.91E-20	2.62E+02	2.96E+01	2.78E+01	1.50E-10	9.24E-10	1.25E-03	7.39E+01	**1.06E-27**	1.31E+00	1.13E-04
	Worst	2.56E-08	1.55E+05	1.13E+02	8.83E+01	2.56E-08	2.29E-06	7.58E-02	6.14E+03	**2.41E-19**	9.87E+01	2.00E-03
f_4	Best	**4.33E-27**	1.04E+00	2.21E+00	3.68E-01	2.38E-02	8.15E-08	1.09E-02	1.78E-01	7.73E-20	7.38E-01	4.92E-06
	Worst	3.62E-08	8.44E+01	1.02E+01	5.27E-01	8.12E-02	3.60E-06	4.25E-02	4.24E+01	**9.53E-18**	1.22E+01	1.42E-04
f_5	Best	**2.47E-19**	2.62E+01	2.21E+01	1.99E+01	4.27E-10	1.65E-03	1.77E+01	2.75E+01	25.1669	2.25E+01	1.81E+01
	Worst	**1.66E-18**	2.68E+01	8.29E+01	1.08E+02	3.99E+01	1.05E+02	1.97E+01	1.59E+02	27.9291	4.48E+02	4.01E+02
f_6	Best	**0**	1.16E-03	3.58E-13	1.12E-15	0	0	1.91E-11	3.66E+00	5.10E-06	7.61E-09	5.65E-14
	Worst	1.23E-32	3.89E-02	3.58E-13	2.04E-12	4.93E-32	2.16E-32	5.50E-10	4.75E+00	7.53E-01	1.64E-08	1.41E-10
f_7	Best	**1.90E-06**	3.67E-05	3.31E-02	1.87E-02	6.22E-02	1.33E-03	3.87E-03	9.78E-04	9.70E-05	1.91E-02	6.80E-03
	Worst	1.47E-04	1.26E-03	7.24E-02	6.34E-02	1.31E-01	5.85E-03	1.82E-02	1.16E-01	9.57E-04	8.67E-02	3.49E-02
f_8	Best	**−12569.4866**	−12569.0355	−7985.3267	−6745.8908	−9824.0772	−12569.49	−12569.4866	−4743.8	−7449.8	−8754.3	−10576
	Worst	**−12569.4866**	−12350.2077	−5261.5667	−5797.8315	−8759.7381	−12332.61	−12569.4864	−3673.5	−3017.3	−6725.8	−7515.9
f_9	Best	**0**	0	1.57E+02	2.68E+01	2.78E+01	4.01E-05	1.95E+01	2.24E-07	0	2.69E+01	5.27E+01
	Worst	**0**	0	1.96E+02	5.57E+01	7.46E+01	5.13E-04	2.79E+01	6.23E+01	5.82E+00	9.45E+01	1.51E+02
f_{10}	Best	4.44E-15	4.44E-15	1.22E-07	1.37E-07	6.48E-14	4.44E-15	8.94E-07	1.41E-04	7.99E-15	2.54E-05	4.10E-09
	Worst	2.28E-13	7.99E-15	2.03E-07	1.11E-06	4.46E+01	7.99E-15	4.36E-06	2.02E+01	1.51E-14	3.62E+00	3.09E+01
f_{11}	Best	**0**	0	3.67E-13	5.20E-14	1.96E-02	0	6.25E-11	3.37E-05	0	3.42E-08	2.55E-15
	Worst	**0**	0	4.20E-12	1.96E-02	4.43E-02	1.23E-02	1.64E-06	8.62E-01	1.79E-02	2.95E-02	3.70E-02
f_{12}	Best	**1.60E-32**	2.12E-04	1.38E-14	6.60E-16	1.57E-32	1.57E-32	5.08E-13	3.99E-01	6.30E-03	3.94E-01	3.66E-16
	Worst	**1.57E-32**	3.39E-04	1.07E-12	1.20E-14	1.33E+01	6.55E-26	9.60E-12	8.38E+01	3.94E-02	1.03E+01	1.19E+00
f_{13}	Best	**1.35E-32**	5.60E-03	1.46E-13	2.22E-15	1.09E-02	1.35E-32	3.48E-12	1.84E+01	1.26E-05	4.80E-10	5.06E-16
	Worst	1.84E-32	1.00E-01	1.69E-09	1.76E-12	1.91E+01	1.78E-26	4.50E-11	7.70E+04	5.06E-01	5.48E-02	1.10E-02

The bolded entries in the above tables are used to highlight the best obtained results

Table 5 Comparison of results (**Average, Standard**) gained for $f_1 - f_{13}$

F	Statistics	MDE-WOA	WOA	DE	PSO	DEGL	JADE	CoDE	SCA	GWO	CSSA	AFWA
f_1	Mean	**1.41E-286**	2.63E-176	1.67E-13	9.98E-13	1.97E-55	2.26E-32	1.12E-10	6.90E-04	2.32E-74	1.26E-08	8.90E-12
	Std.Dev	**0**	0	5.27E-14	1.16E-12	3.67E-55	1.05E-31	7.72E-12	1.80E-02	3.94E-74	2.86E-09	1.50E-11
f_2	Mean	**7.31E-162**	2.61E-113	5.00E-07	9.03E-07	1.74E-24	4.10E-13	2.50E-06	1.82E-06	3.39E-43	8.45E-01	1.11E-02
	Std.Dev	**1.04E-161**	5.69E-113	2.31E-07	9.85E-07	2.13E-24	2.17E-12	9.95E-07	3.22E-06	3.80E-43	8.50E-01	6.09E-02
f_3	Mean	6.25E-09	1.07E+04	6.38E+00	4.51E+00	1.43E-08	3.17E-07	9.74E-03	1.88E+03	**9.31E-21**	1.91E+01	7.30E-04
	Std.Dev	1.09E-08	5.32E+03	3.12E+00	2.47E+00	3.17E-08	5.66E-07	1.34E-02	1.84E+03	**4.38E-20**	1.92E+01	5.10E-04
f_4	Mean	7.63E-09	2.03E+01	5.16E+01	4.31E-01	4.73E-02	8.94E-07	2.82E-02	1.29E+01	**2.28E-18**	4.43E+00	4.14E-05
	Std.Dev	1.60E-08	3.62E+01	3.24E+01	8.79 E-02	3.30E-02	1.04E-06	6.89E-03	1.15E+01	**2.30E-18**	2.60E+00	3.16E-05
f_5	Mean	**1.00E-18**	2.66E+01	3.54E+01	4.98E+01	2.39E+00	6.48E+00	1.88E+01	3.88E+01	2.65E+01	7.79E+01	8.74E+01
	Std.Dev	**6.82E-19**	2.86E-01	2.66E+01	3.82E+01	2.18E+00	1.88E+01	5.50E-01	2.68E+01	7.41E-01	9.87E+01	1.14E+02
f_6	Mean	**2.47E-33**	2.40E-03	1.97E-13	1.14E-12	1.48E-32	5.24E-33	1.14E-10	4.18E+00	3.45E-01	1.20E-08	1.46E-11
	Std.Dev	**5.51E-33**	1.20E-03	9.42E-14	8.43E-13	2.03E-32	6.43E-33	9.99E-11	3.17E-01	2.51E-01	2.60E-09	2.96E-11
f_7	Mean	**5.66E-05**	3.34E-04	5.33E-02	3.67E-02	9.05E-02	3.50E-03	1.08E-02	1.73E-02	3.87E-04	4.71E-02	1.49E-02
	Std.Dev	**9.42E-05**	5.22E-04	1.45E-02	1.64E-02	2.56E-02	1.12E-03	3.33E-03	2.20E-02	2.24E-04	1.72E-02	6.30E-03
f_8	Mean	**-1.26E+04**	−1.25E+04	−6.28E+03	−6.45E+03	−9.36E+03	−1.25E+04	−1.26E+04	−4106.4	−6.15E+03	−7.88E+03	−9.03E+03
	Std.Dev	**4.36E-12**	9.78E+01	1.05E+03	6.22E+02	4.04E+02	6.73E+01	3.61E-05	2.94E+02	8.04E+02	5.69E+02	7.67E+02
f_9	Mean	**0**	0	1.77E+02	3.72E+01	4.91E+01	1.56E-04	2.31E+01	1.14E+01	1.94E-01	5.5.6E+01	8.75E+01
	Std.Dev	**0**	0	1.38E+01	1.46E+01	1.78E+01	9.51E-05	2.28E+00	1.87E+01	1.06E+00	1.95E+01	2.61E+01
f_{10}	Mean	1.30E-14	**5.86E-15**	1.49E-07	4.55E-07	2.63E+00	6.69E-15	2.30E-06	1.29E+01	4.99E-14	1.67E+00	1.05E+00
	Std.Dev	9.97E-14	1.95E-15	3.27E-08	3.86E-07	1.78E+00	1.74E-15	8.21E-07	9.28E+00	2.74E-15	1.00E+00	9.64E-01
f_{11}	Mean	**0**	0	1.99E-12	8.40E-03	2.90E-02	6.57E-04	7.53E-08	2.08E-01	2.40E-03	6.40E-03	1.23E-02
	Std.Dev	**0**	0	1.57E-12	8.80E-03	9.30E-03	2.58E-03	3.05E-07	2.26E-01	5.20E-03	9.10E-03	1.01E-02
f_{12}	Mean	**1.58E-32**	2.88E-04	2.51E-13	3.31E-15	2.87E-01	2.18E-27	3.18E-12	1.11E+00	2.19E-02	3.36E+00	2.23E-01
	Std.Dev	**1.44E-34**	5.07E-05	4.60E-13	4.89E-15	5.87E-01	1.19E-26	2.34E-12	1.54E+00	8.60E-03	2.07E+00	3.32E-01
f_{13}	Mean	**1.45E-32**	2.75E-02	3.42E-10	3.62E-13	4.55E+00	1.04E-27	1.80E-11	3.09E+03	2.42E-01	1.09E-02	2.60E-03
	Std.Dev	**2.20E-33**	4.08E-02	7.55E-10	7.81E-13	8.28E+00	3.61E-27	1.04E-11	1.42E+04	1.40E-01	1.22E-02	4.70E-03

Table 6 Comparison of pvalues of the Wilcoxon ranksum test gained for $f_1 - f_{13}$ ($p \geq 0.05$ are underlined)

F	WOA	DE	PSO	DEGL	JADE	SCA	GWO	CSSA	AFWA
f_1	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
f_2	1.07E-07	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	1.07E-07	3.02E-11	3.02E-11
f_3	3.02E-11	3.02E-11	3.02E-11	2.61E-10	0.00073	3.02E-11	1.07E-07	3.02E-11	6.72E-10
f_4	3.02E-11	3.02E-11	3.02E-11	3.02E-11	7.77E-09	3.02E-11	5.57E-10	3.02E-11	5.49E-11
f_5	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11	3.02E-11
f_6	7.77E-12	7.77E-12	7.77E-12	3.83E-10	6.47E-02	7.77E-12	7.77E-12	7.77E-12	7.77E-12
f_7	0.435192	3.02E-11	3.02E-11	6.70E-11	7.09E-08	1.33E-10	0.346956	3.02E-11	4.50E-11
f_8	3.53E-08	2.88E-11	2.88E-11	2.88E-11	3.17E-07	2.88E-11	2.88E-11	2.88E-11	3.53E-11
f_9	0.570163	2.36E-12	2.36E-12	2.36E-12	2.36E-12	2.36E-12	0.610741	2.36E-12	2.36E-12
f_{10}	0.630839	1.45E-11	1.45E-11	1.45E-11	7.01E-08	1.45E-11	2.42E-10	1.45E-11	1.45E-11
f_{11}	0.160802	1.21E-12	1.21E-12	1.21E-12	8.15E-02	1.21E-12	0.041926	1.21E-12	1.21E-12
f_{12}	9.27E-12	9.27E-12	9.27E-12	9.27E-12	3.36E-06	9.27E-12	9.27E-12	9.27E-12	9.27E-12
f_{13}	5.18E-12	5.18E-12	5.16E-12	5.18E-12	7.46E-08	5.18E-12	5.18E-12	5.18E-12	5.18E-12

groups. For the first group, we design another two WOA variants named WOA1 and WOA2 where introduced basic DE and modified DE operators respectively into WOA. The other parts of WOA1 and WOA2 remain consistent with traditional WOA algorithm. Thus, a comparison between WOA, WOA1 and WOA2 could validate the effectiveness of the first modification. The second group of experiments tests the competitiveness of asynchronous model. We design WOA3 whose $\mathbf{X}_{g\_best}$ is updated if there is a better solution after each agent changed. The remaining parts of the algorithm remain identical to the WOA algorithm.

The parameter settings are same as those in Section 4.1. Figures 6 and 7 show the comparison results for the two experimental groups on several representative benchmarks, showing that each modification greatly improves upon the traditional WOA. From Fig. 6, we can see that the hybridization with DE/rand/1 can overcome the premature convergence problem of WOA, and WOA2 achieves the better accuracy for f_5, f_6, f_{12}, f_{13} compared with WOA1. This result also illustrates the balance between exploration and exploitation capabilities of MDE operators. For the asynchronous model, two unimodal functions f_1 and f_2 are chosen to test its effectiveness, and the curve demonstrates the competitiveness of this modification.

4.6 Constrained engineering problems

In this part, three structural optimization problems will be presented to investigate the accuracy and robustness of proposed approach. These constrained examples include a pressure vessel, a tension/compression spring, and a welded beam design problem. An effective method to handle constraints is to employ penalty functions. There are different types of penalty functions, such as static, dynamic, adaptive, death penalty, and so on [47]. The static penalty, assigning a big fitness value in terms of the degree of constrained violation (in the case of minimization problem), is a common method. The infeasible solutions will be automatically discarded by the algorithm. Since finding a good constraints handling method for the proposed approach is out of the scope of this work, we cope with constraints using a static penalty function represented as:

$$F(\mathbf{x}) = f(\mathbf{x}) + \sum_{i=1}^{n} k \times [\max(g_i(\mathbf{x}), 0)] \tag{13}$$

where $\mathbf{x}$ is the design variables and $f(\mathbf{x})$ is the objective function value, k is a big positive constant number, $g_i(\mathbf{x})$ is the i^{th} constraint for solution.

4.6.1 Pressure vessel design

Details of this problem can be found in [48, 49]. The goal of this problem is the minimization of the total cost including material, forming and welding. The design variables of this problem are: the thickness $T_s(x_1)$ of the shell, the thickness $T_h(x_2)$ of the head, radius of cylindrical shell $R(x_3)$, and the shell length $L(x_4)$. In this problem, x_1 and x_2 are in multiples of 0.0625 inches, x_3 and x_4 are continuous variables.

This case was solved by lots of techniques, such as improved partial swarm optimization (PSO) [48], modified harmony search (HS) [49], GA [50], real-coded GA [52], and so on. Note that 30 individuals and a maximum number of 500 iterations (15000 function evaluations) have been utilized to solve this problem. The other parameters of the proposed method are same as the previous experiments. Under theses conditions, experimental results of MDE-WOA, compared with other methods, are shown in Tables 8

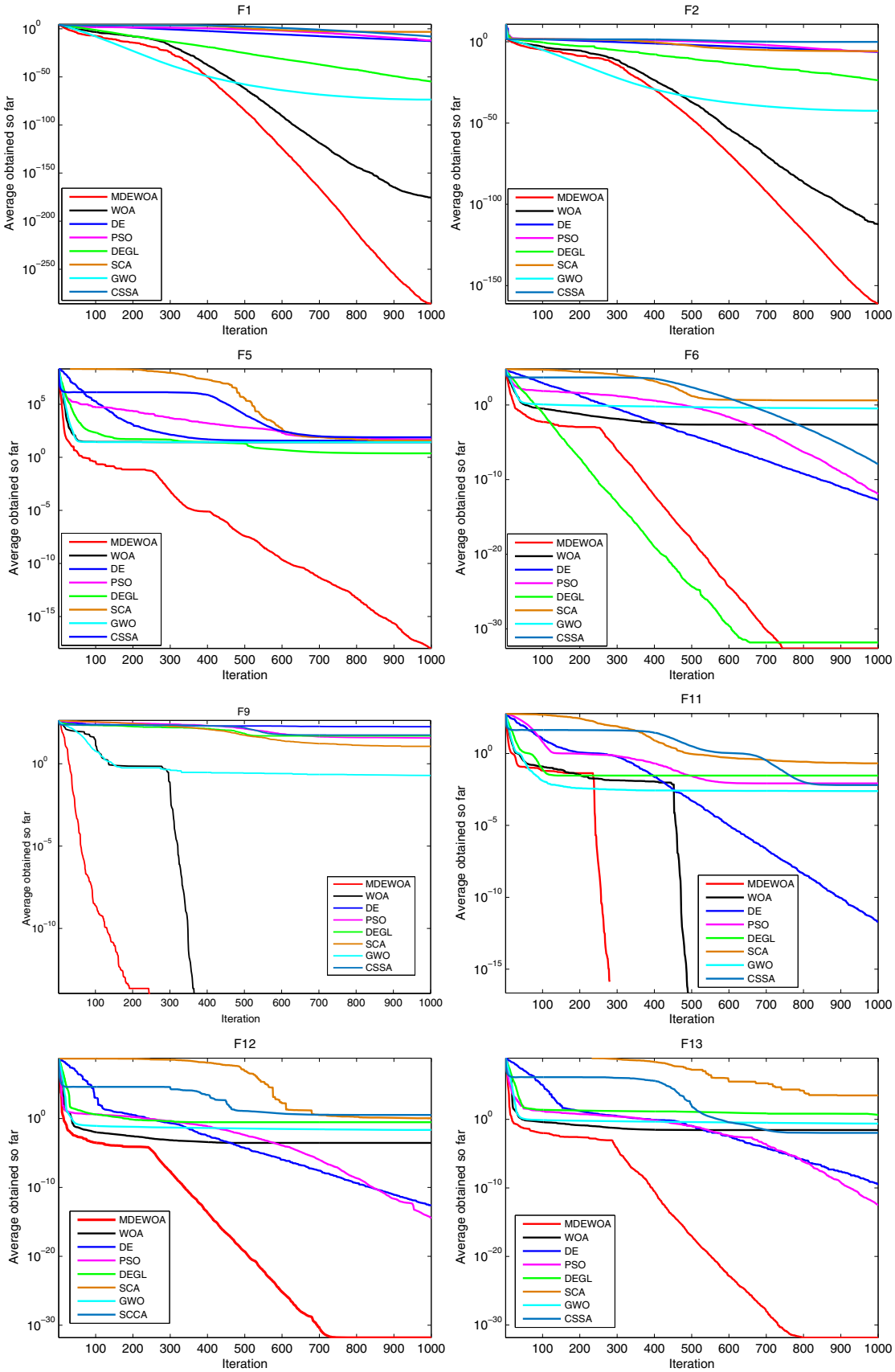

Fig. 5 Comparison of convergence curves of MDE-WOA and other seven algorithms obtained in some of the benchmark problems

Table 7 The successful rate and runtime

F	Threshold	Runtime/s and successful rate/%								
		MDE-WOA	WOA	DE	PSO	DEGL	SCA	GWO	CSSA	AFWA
f_1	1E-10	**0.1596/100**	0.2525/100	1.1491/100	0.4226/100	2.1444/100	0.6287/0	0.1690/100	0.9468/0	0.9873/100
f_2	1E-10	0.3330/100	0.3351/100	1.4640/0	0.5058/0	4.8936/100	0.7574/0	0.2345/100	0.9969/0	1.4851/0
f_3	1E-10	5.9921/70	3.2279/0	8.2346/0	2.6170/0	11.7282/100	3.4389/0	1.2274/100	3.7118/0	9.6677/0
f_4	1E-10	3.6341/70	1.0716/0	1.4567/0	0.5020/0	5.3122/0	0.9596/0	0.3374/100	1.2263/0	2.0926/0
f_5	1E-10	**3.1554/100**	1.2258/0	2.0554/0	0.7291/0	6.3053/0	1.0132/0	1.2160/0	1.2688/0	2.1868/0
f_6	1E-10	1.0781/100	1.0199/0	1.1743/100	**0.4344/100**	2.0938/100	1.0354/0	1.2390/0	1.2841/0	1.4978/100
f_7	1E-05	7.6118/26.7	**1.7601/20**	1.4640/0	0.7283/0	8.06985/0	1.0149/0	1.1983/0	1.8792/0	2.4081/0
f_9	1E-10	**0.1737/100**	0.3018/100	2.1312/0	0.7283/0	6.0387/0	0.8929/0	0.2930/96.7	1.1602/0	1.6887/0
f_{10}	1E-10	**0.3552/100**	0.3960/100	2.0369/0	0.7045/0	6.2558/20	1.2418/0	0.3668/100	1.4898/0	2.4094/0
f_{11}	1E-10	**0.2653/100**	0.4550/96.7	2.6120/86.7	0.9346/26.7	6.0756/25	1.3027/0	0.2277/96.7	1.6210/0	2.5450/33.3
f_{12}	1E-10	**2.1423/100**	3.2177/0	6.1032/96.7	2.2990/100	8.5511/68	2.4018/0	2.5570/0	2.1371/0	5.4395/53.3
f_{13}	1E-10	**2.5825/100**	3.1748/0	6.4892/100	2.4330/80	8.5876/40	2.3690/0	2.8893/0	2.6346/0	4.9015/80

The bolded entries in the above tables are used to highlight the best obtained results

and 9. For fair comparison, all the results of compared algorithms come from their corresponding references.

As seen from Table 8, the proposed method converged to the second best design. We can conclude that it is able to provide very competitive optimal solutions. Table 9 shows the average, standard deviation and number of fitness analysis. The average obtained solution was $f(x) = 6067.74$, which is better than any of the solutions produced by four other techniques. Moreover, it may be observed that the proposed method is better than others in terms of

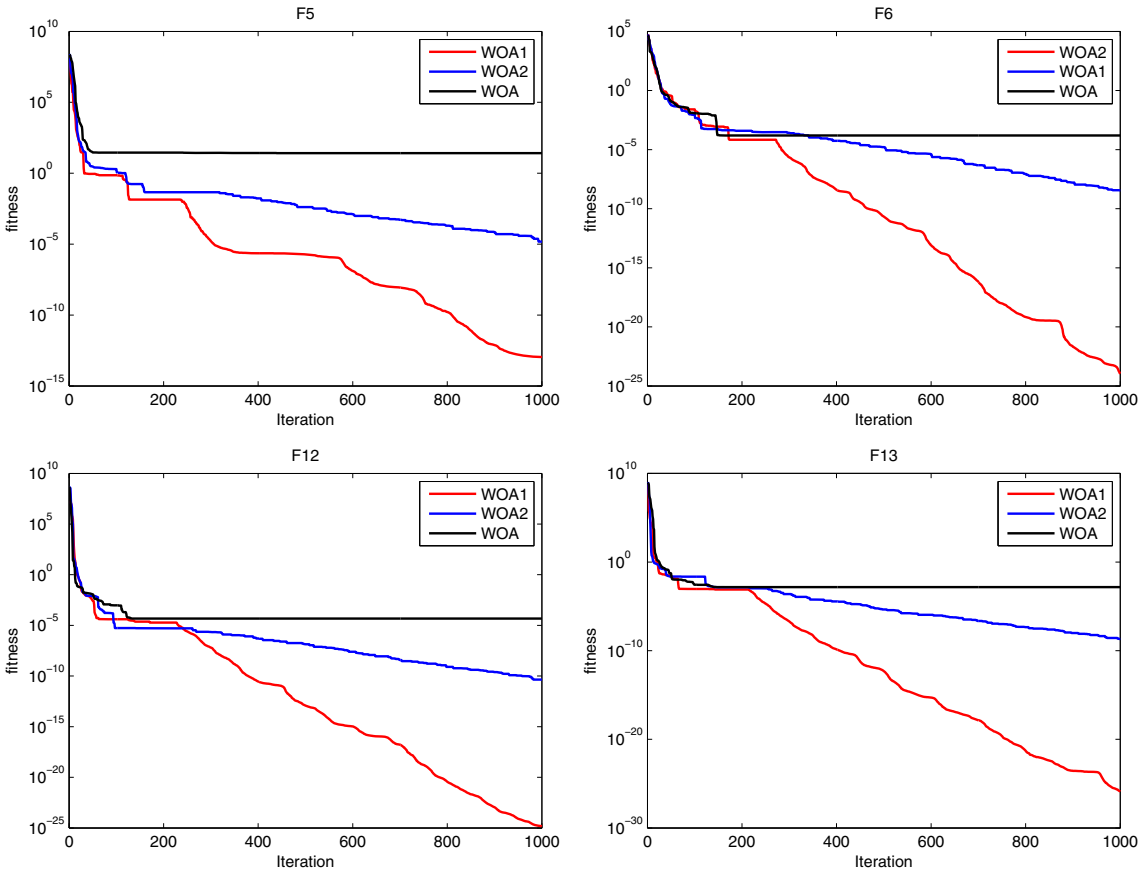

Fig. 6 Demonstration of the hybridizationś effectiveness

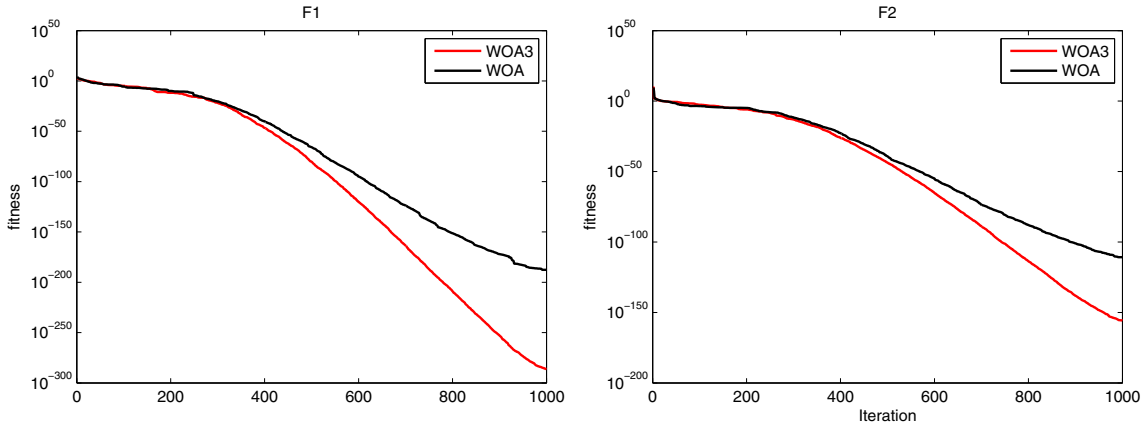

Fig. 7 Demonstration of the second modificationś effectiveness

Table 8 Optimal results for the pressure vessel problem

Design variables and objective function	MDE-WOA	WOA [17]	PSO [48]	HS [49]	GA [50]	GA [52]
$x_1(T_s)$	0.8125	0.8125	1.125	1.125	0.8125	Unavailable
$x_2(T_h)$	0.4375	0.4375	0.625	0.625	0.4375	Unavailable
$x_3(R)$	42.098446	42.0982699	58.2909	58.290138	40.3239	Unavailable
$x_4(L)$	176.636596	176.638998	43.6881	43.6927585	200	Unavailable
$f(\mathbf{x})$	6059.7143	6059.7410	7197.98	7197.73	6288.7445	**6059.71**

The bolded entries in the above tables are used to highlight the best obtained results

Table 9 Statical results of the pressure vessel problem

Algorithm	Ave	Std	Max. eval
MDE-WOA	**6067.74**	**15.0528**	15000
WOA	6068.05	65.6519	6300
AIS-GA [53]	6385.942	Unavailable	80000
ABC [54]	6245.31	205	30000
GA [52]	6075.16	16.4132	40000

The bolded entries in the above tables are used to highlight the best obtained results

Table 11 Statical results of the Tension/Compression spring problem

Algorithm	Ave	Std	Max. eval
MDE-WOA	**0.012665**	**4.89E-06**	15000
WOA	0.0126763	0.0003	4410
AIS-GA [53]	0.012974	Unavailable	36000
ABC [54]	**0.012665**	0.012813	30000
GA [52]	0.012705	2.74E-05	20000

The bolded entries in the above tables are used to highlight the best obtained results

Table 10 Optimal results for minimization of the weight of a spring

Design variables and objective function	MDE-WOA	WOA [17]	AIS-GA [53]	HS [49]	GA [50]	cricket algorithm [51]
$x_1(d)$	0.0517045	0.051207	0.051301897	0.0518606	0.051480	0.052036
$x_2(D)$	0.357089	0.345215	0.34747463	0.3608578	0.351661	0.365113
$x_3(N)$	11.2673	12.004032	11.852177	11.050339	11.632201	10.815329
$f(\mathbf{x})$	**0.012665**	0.0126763	0.012668	0.0126657	0.01270478	0.012670

The bolded entries in the above tables are used to highlight the best obtained results

Table 12 Optimal results for welded beam design problem

Design variables and objective function	MDE-WOA	WOA	PSO [48]	HS [49]	GA [50]	GA [52]
$x_1(h)$	0.20573	0.205396	0.2444	0.20573	0.2088	Unavailable
$x_2(l)$	3.47049	3.484293	6.2180	3.47049	3.4205	Unavailable
$x_3(t)$	9.03662	9.037426	8.2916	9.03662	8.9975	Unavailable
$x_4(b)$	0.20573	0.206276	0.2444	0.20573	0.2100	Unavailable
$f(\mathbf{x})$	**1.7248**	1.7305	2.3810	1.7248	1.7483	1.7248

The bolded entries in the above tables are used to highlight the best obtained results

robustness. Although number of analysis by MDE-WOA is more than basic WOA, the solution accuracy will not be improved by WOA with more function evaluations because of the stagnation problem.

4.6.2 Tension/compression spring design problem

The tension/compression spring is a real-world problem that has been usually employed as a benchmark for testing the performance of meta-heuristic algorithms. The objective of this design problem is the minimization of the weight subject to constraints on shear stress, deflection and surge frequency. This design problem includes three variables: wire diameter $d(x_1)$, mean coil diameter $D(x_2)$ and number of active coils $N(x_3)$, respectively. For details of this problem refer to [49, 50].

Table 10 lists the optimal solution of Tension /Compression spring problem obtained by MDE-WOA and compares them with the values presented by Bernardino et al. [53], Majid and Esmaile [49], Coello [50], Canayaz et al. [51]. It may be concluded that the result acquired from MDE-WOA is consistent with the previous best solution reported by Majid and Esmaile.

The statistic results for solving the vessel design problem are listed in Table 11. We have used 30 individuals with 500 iterations to find the solution. It can be seen that the similar results are obtained by the improved WOA and other algorithms, but the maximum evaluations number of MDE-WOA is smaller than that of the ABC. MDE-WOA again shows better performance on stability when compared with basic version of WOA.

4.6.3 Design of a welded beam

The aim for this problem is to minimize the fabrication cost subject to shear stress τ, bending stress σ, end deflection of the beam δ, buckling load P_c, and side constraints. There are four variables including thickness $h(x_1)$ of the weld, length $l(x_2)$ of the clamped joint, height $t(x_3)$ of the beam and width $b(x_4)$ of the beam. For details about these problems, the reader can refer to [48, 49].

This test problem was solved by many methods (such as improved PSO [48], modified HS [49], GA [50], real-coded GA [52], and so on). Optimization results are shown in Tables 12 and 13. As shown in Table 12, the optimal results obtained by MDE-WOA and real-coded tied for first place. According to Table 13, once more, the proposed method outperforms the WOA on accuracy and stability. This table also shows that the MDE-WOA outperforms ABC and GA in average and requires less number of analysis. We have used 30 search agents in MDE-WOA instead of 20 in WOA, so the number of function evaluations was more than basic algorithm.

Based on the experimental results, we can find that the basic WOA consumed the minimum number of evaluations on these problems, but the accuracy is difficult to be improved due to the aforementioned stagnation problem. The proposed MDE-WOA shows great performance on accuracy and robustness compared with basic WOA. These results are reasonable because the combined MDE operators increase the population diversity and relieve the stagnation problem of WOA.

As summary, the results reveal that the proposed MDE-WOA has capability in handling real engineering design problems and can obtain competitive solutions. Therefore, the proposed MDE-WOA is an attractive and effective approach for solving both constrained and unconstrained optimization problems.

Table 13 Statical results of the welded beam design problem

Algorithm	Ave	Std	Max. eval
MDE-WOA	**1.7248**	**1.20E-03**	15000
WOA	1.7320	2.26E-02	9900
AIS-GA [53]	2.38992	Unavailable	320000
ABC [54]	1.724852	3.10E-02	30000
GA [52]	1.739539	1.68E-02	40000

The bolded entries in the above tables are used to highlight the best obtained results

5 Conclusion

The original WOA is easy to fall into stagnation and lose population diversity when it carries out the operation of prey encircling. To address these issues, an improved WOA by hybridizing WOA with modified DE called MDE-WOA is proposed in this paper, which takes advantage of the strong search ability of differential evolution operators to make WOA jump out of the local optima and increase the population diversity. At the same time, the asynchronous model is proposed to improve the convergence speed and accuracy. The experimental results tested on 13 benchmark functions show that MDE-WOA with appropriate parameter outperforms the native WOA algorithm and other state-of-the-art meta-heuristic methods in terms of solving accuracy, stability, successful rate and runtime. Moreover, the MDE-WOA acquired optimal results or near-best solutions on 3 constrained engineering design problems.

The adaptively MDE-WOA which could adjust parameters including lifespan threshold and weight factor dynamically is currently under development. In addition, a multi-objective version of MDE-WOA will be explored in the future.

Acknowledgments The authors are grateful for the valuable comments and suggestions of editor and anonymous reviewers.

Compliance with Ethical Standards

Conflict of interests The authors declared that they have no conflicts of interest to this work.

References

1. Yang XS (2013) Metaheuristic Optimization: Nature-Inspired Algorithms and Applications. Springer, Berlin
2. Sotoudeh-Anvari A, Ashkan H (2018) A bibliography of metaheuristics-review from 2009 to 2015. International Journal Of Knowledge-based And Intelligent Engineering Systems 22(1): 83–95
3. Kennedy J, Eberhart R (2002) Particle swarm optimization. In: IEEE international conference on neural networks, 1995. Proceedings, vol 4, pp 1942–1948
4. Dorigo M, Birattari M, Stutzle T (2006) Ant colony optimization. IEEE Comput Intell Mag 1(4):28–39
5. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. J Glob Optim 39(3):459–471
6. Yang XS, Gandomi AH (2012) Bat algorithm: a novel approach for global engineering optimization. Eng Comput 29(5-6):464–483
7. Wolpert DH, Macready WG (1997) No free lunch theorems for optimization. IEEE Trans Evol Comput 1(1):67–82
8. Pan WT (2012) A new fruit fly optimization algorithm: Taking the financial distress model as an example. Knowl-Based Syst 26:69–74
9. Mirjalili S, Mirjalili SM, Lewis A (2014) Grey Wolf Optimizer. Adv Eng Softw 69:46–61
10. Mirjalili S, Gandomi AH, Mirjalili SZ et al (2017) Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. Adv Eng Softw 114:163–191
11. Faris H, Mafarja MM, Heidari AA et al (2018) An efficient binary salp swarm algorithm with crossover scheme for feature selection problems. Knowl-Based Syst 154:43–67
12. Mirjalili SZ, Mirjalili S, Saremi S, Faris H, Aljarah I (2018) Grasshopper optimization algorithm for multi-objective optimization problems. Appl Intell 48(4):805–820
13. Askarzadeh A (2016) A novel metaheuristic method for solving constrained engineering optimization problems: Crow search algorithm. Comput Struct 169:1–12
14. Mirjalili S (2016) SCA: A Sine Cosine Algorithm for solving optimization problems. Knowl-Based Syst 96:120–133
15. Meng XB, Gao XZ, Lu L, Liu Y, Zhang H (2016) A new bio-inspired optimisation algorithm: Bird swarm algorithm. J Exp Theor Artif Intell 28(4):673–687
16. Meng X, Liu Y, Gao X, Zhang H (2014) A new bio-inspired algorithm: Chicken swarm optimization. In: Tan Y, Shi Y, Coello CAC (eds) Advances In Swarm Intelligence, PT1, Lecture Notes in Computer Science, vol 8794, pp 86–94
17. Mirjalili S, Lewis A (2016) The Whale Optimization Algorithm. Adv Eng Softw 95:51–67
18. Oliv D, Abd El Aziz M, Hassanien AE (2017) Parameter estimation of photovoltaic cells using an improved chaotic whale optimization algorithm. Appl Energy 200:141–154
19. Prakash DB, Lakshminarayana C (2017) Optimal siting of capacitors in radial distribution network using Whale Optimization Algorithm. Alex Eng J 56(4):499–509
20. Wang J, Du P, Niu T, Yang W (2017) A novel hybrid system based on a new proposed algorithm-Multi-Objective Whale Optimization Algorithm for wind speed forecasting. Appl Energy 208:344–360
21. Al-Zoubi Ala'M, Faris H et al (2018) Evolving support vector machines using whale optimization algorithm for spam profiles detection on online social networks in different lingual contexts. Knowl-Based Syst 153:91–104
22. Abd El Aziz M, Ewees AA, Hassanien AE (2017) Whale Optimization Algorithm and Moth-Flame Optimization for multilevel thresholding image segmentation. Expert Syst Appl 83:242–256
23. Kaveh A, Ghazaan MI (2017) Enhanced whale optimization algorithm for sizing optimization of skeletal structures. Mechanics Based Design of Structures And Machines 45(3, SI):345–362
24. Zhao H, Guo S, Zhao H (2017) Energy-related CO2 emissions forecasting using an improved LSSVM model optimized by whale optimization algorithm. Energies 10(7):874–888
25. Ling Y, Zhou Y, Luo Q (2017) Levy flight trajectory-based whale optimization algorithm for global optimization. IEEE ACCESS 5:6168–6186
26. Kumar CHS, Rao RS (2016) A novel global MPP tracking of photovoltaic system based on whale optimization algorithm. Int J Renewable Energy Dev 5(3):225–232
27. Storn R, Price K (1997) Differential evolution - A simple and efficient heuristic for global optimization over continuous spaces. J Glob Optim 11(4):341–359
28. Liang Z, Hu K, Zhu Q, Zhu Z (2017) An enhanced artificial bee colony algorithm with adaptive differential operators. Appl Soft Comput 58:480–494
29. Zhu A, Xu C, Li Z, Wu J, Liu Z (2015) Hybridizing grey wolf optimization with differential evolution for global optimization and test scheduling for 3D stacked SoC. J Syst Eng Electron 26(2):317–328

30. Gao Wf, Huang Ll, Wang J, Liu Sy, Qin Cd (2016) Enhanced artificial bee colony algorithm through differential evolution. Appl Soft Comput 48:137–150
31. Sayah S, Hamouda A (2013) A hybrid differential evolution algorithm based on particle swarm optimization for nonconvex economic dispatch problems. Appl Soft Comput 13(4):1608–1619
32. Nouioua M, Li Z (2017) Using differential evolution strategies in chemical reaction optimization for global numerical optimization. Appl Intell 47(3):935–961
33. Das S, Abraham A, Chakraborty UK, Konar A (2009) Differential Evolution Using a Neighborhood-Based Mutation Operator. IEEE Trans Evol Comput 13(3):526–553
34. Sayed GI, Khoriba G, Haggag MH (2018) A novel chaotic salp swarm algorithm for global optimization and feature selection. Appl Intell 5:1–20
35. Li J, Zheng S, Tan Y (2014) Adaptive Fireworks Algorithm. In: 2014 IEEE congress on evolutionary computation (CEC), pp 3214–3221
36. Talbi E (2002) A taxonomy of hybrid metaheuristics. J Heuristics 8(5):541–564
37. Goldsmith T (2006) The evolution of aging. Nature Education Knowledge 156(10):927–931
38. Piotrowski AP (2013) Adaptive Memetic Differential Evolution with Global and Local neighborhood-based mutation operators. Inf Sci 241:164–194
39. Wang Y, Cai Z, Zhang Q (2011) Differential evolution with composite trial vector generation strategies and control parameters. IEEE Trans Evol Comput 15(1):55–66
40. Bhowmik P, Das S, Konar A, Das S, Nagar AK (2010) A new differential evolution with improved mutation strategy. In: 2010 IEEE congress on evolutionary computation (CEC), IEEE congress on evolutionary computation, vol 1210, pp 1–8
41. Zhang J, Sanderson AC (2009) JADE: Adaptive Differential Evolution With Optional External Archive. IEEE Trans Evol Comput 13(5):945–958
42. Qin A, Suganthan P (2005) Self-adaptive differential evolution algorithm for numerical optimization. In: 2005 IEEE ongress on Evolutionary Computation, vol 1-3, Proceedings IEEE Congress on Evolutionary Computation, pp 1785–1791
43. Tanabe R, Fukunaga AS (2014) Improving the search performance of SHADE using linear population size reduction. In: 2014 IEEE congress on evolutionary computation (CEC), pp 1658–1665
44. Tirronen V, Neri F, Karkkainen T, Majava K, Rossi T (2008) A Memetic Differential Evolution in filter design for defect detection in paper production. In: Giacobini M (ed) Proceedings of applications Of evolutionary computing, vol 16, pp 529–555
45. Wang Y, Cai Z, Zhang Q (2011) Differential Evolution with Composite Trial Vector Generation Strategies and Control Parameters. IEEE Trans Evol Comput 15(1):55–66
46. Mirjalili S, Lewis A (2013) S-shaped versus V-shaped transfer functions for binary Particle Swarm Optimization. Swarm Evolut Comput 9:1–14
47. Coello 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(11-12):1245–1287
48. Kiran MS (2017) Particle swarm optimization with a new update mechanism. Appl Soft Comput 60:670–678
49. Jaberipour M, Khorram E (2010) Two improved harmony search algorithms for solving engineering optimization problems. Commu Nonlinear Sci Num Simul 15(11):3316–3331
50. Coello C (2000) Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 41(2):113–127
51. Canayaz M, Karci A (2016) Cricket behaviour-based evolutionary computation technique in solving engineering optimization problems. Appl Intell 44(2):362–376
52. Pathan MV, Patsias S, Tagarielli VL (2018) A real-coded genetic algorithm for optimizing the damping response of composite laminates. Comput & Struct 198:51–60
53. Bernardino HS, Barbosa HJC, Lemonge ACC (2007) A hybrid genetic algorithm for constrained optimization problems in mechanical engineering. In: 2007 IEEE Congress on Evolutionary Computation, VOLS 1-10, Proceedings, pp 646–653
54. Akay B, Karaboga D (2012) Artificial bee colony algorithm for large-scale problems and engineering design optimization. J Int Manag 23(4):1001–1014

Jun Luo received his B.E in precision instruments and machinery and M.E degrees in radio communications technology from Chongqing University, Chongqing, China, in 1983 and 1993, respectively. Currently, he is a full professor and Ph. D supervisior at college of Optoelectronic Engineering, Chongqing University. His research interests include swarm intelligence algorithms, artificial intelligence and neural computing, measurement technology and instrumentation, and embedded systems.

Baoyu Shi received her bachelor's degree in Measurement and Control Technology and Instruments in 2016. She is currently pursuing her master degree in the Ministry of Education Key Laboratory of Optoelectronic Technology and systems, Chongqing University. Her research interests include global optimization evolutionary algorithms, and application in machine vision.