ICJK
Journal of Mathematics and Interdisciplinary Applications
http://dx.doi.org/10.62762/JMIA.2026.111913

RESEARCH ARTICLE

Check for updates

Dynamic Inertia Weight Whale Optimization Algorithm for Numerical and Engineering Optimization Problems

Wen Long [1,*] **and Tiantian Hu** [2]

[1] School of Mathematics and Statistics, Guizhou University of Finance and Economics, Guiyang 550025, China
[2] School of Information, Guizhou University of Finance and Economics, Guiyang 550025, China

Abstract

Whale optimization algorithm (WOA) is a relatively new population-based metaheuristic optimization method, which has the advantage of fewer control parameters, strong global optimization ability and easy to implement. However, when being used for high-dimensional problems, WOA may be trapped in the local optimum. In this study, we propose an effective whale optimization algorithm called EWOA. Inspired by particle swarm optimization (PSO), a modified position-updated equation by introducing dynamic inertia weight parameter to guide the search of new candidate individuals is presented. In addition, in order to make full use of and balance the exploration and the exploitation of WOA, a nonlinear distance control parameter strategy is proposed. The effectiveness of EWOA is tested based on 26 traditional high-dimensional benchmark problems (D = 30, 100, and 500) and four real-world engineering applications. The experimental and statistical test results demonstrate that EWOA converges to the global optima faster and provides more accurate results than the classical WOA, variants WOA, and other considered population-based metaheuristic algorithms in most cases, especially in solving an optimization problem that has high dimensionality.

Keywords: whale optimization algorithm, inertia weight, global optimization, high-dimensional complex problem, engineering application.

1 Introduction

During the last two decades, many metaheuristic optimization algorithms have been developed to solve many numerical optimization and real-world application problems which are highly nonlinear and complex constraints [1]. The most prominent and recent metaheuristic optimization algorithms that have been proposed are differential evolu- tion (DE) [2], particle swarm optimization (PSO) [3], Fourier transform optimizer (FTO) [4], secretary bird optimization algorithm (SBOA) [5], language education optimization (LEO) [6], queuing search algorithm (QSA) [7], prism refraction search (PRS) [8], fire hawk optimizer (FHO) [9], phototropic growth algorithm (PGA) [10], sailfish optimization (SFO) [11], trees social relations (TSR) [12], divine religions algorithm (DRA) [13], crested porcupine optimizer (CPO) [14], Bezier

Submitted: 27 April 2026
Accepted: 10 June 2026
Published: 21 June 2026

Vol. 2, **No.** 2, 2026.
10.62762/JMIA.2026.111913

***Corresponding author:**
Wen Long
lw227@mail.gufe.edu.cn

Citation
Long, W., & Hu, T. (2026). Dynamic Inertia Weight Whale Optimization Algorithm for Numerical and Engineering Optimization Problems. *Journal of Mathematics and Interdisciplinary Applications*, 2(2), 125–142.

curve-based optimization (BCO) [15], artificial algae algorithm (AAA) [16], simplified swarm optimization (SSO) [17], whale optimization algorithm (WOA) [18], grey wolf optimizer (GWO) [19], sine cosine algorithm (SCA) [20], wave optics optimizer (WOO) [21], among other.

In this paper, we concentrate on whale optimization algorithm (WOA), developed by Mirjalili and Lewis [18] in 2016 based on mimicking the bubble-net hunting mechanism of humpback whales in nature. Experimental comparisons showed that the performance of WOA is competitive to that of other metaheuristic optimization algorithms (i.e., PSO, GSA, and GWO) with an advantage of employing fewer control parameters and ease of implementation [18, 22]. Therefore, WOA has been widely applied to solve many real-world optimization design problems, such as parameter estimation of photovoltaic cells [23, 24], stock trend forecasting [25], flexible job shop scheduling [26], feature selection [27], parameters estima-tion in permanent magnet synchronous motor [28], enginee-ring optimization [29], IoV path planning [30], optimal placement of TCSC and SVC [31], renewable energy prediction [32], mobile robot scheduling [33], closed-loop supply chain network design [34], system identification [35], resource allocation [36], and so on.

Although the classical WOA algorithm has shown strong competitiveness compared with some representative metaheuristic optimization algorithms (i.e., PSO, GSA, and GWO), however, as the growth of the search space dimension, it still has some drawbacks such as slow convergence, and easily trapped into local optimum [18, 37]. Therefore, accelerating convergence speed and avoiding the local optima have become two most important and appealing goals in WOA research. A number of variant WOA algorithms have, hence, been developed to achieve these two goals. Ling et al. [37] presented an improved version of WOA (denoted as LWOA) based on levy flight trajectory to accelerate convergence and get rid of local optimum. Luo et al. [38]proposed an enhanced version of WOA by introducing a chaotic initialization, a chaotic local search and Gaussian mutation strategies. Hu et al. [39] introduced inertia weights and developed an improved WOA for solving global optimization problems and forecasting air quality index. In order to speed up the convergence and enhance the ability of jumping out of local optimum, Sun et al. [40] proposed a modified WOA for large-scale global optimization problems. Chen et al. [41] presented a balanced WOA with Lévy flight and chaotic local search for solving constrained engineering design problems. Wang et al. [42] developed an opposition-based multi-objective whale optimization algorithm with global grid ranking to accelerate convergence and improve solution diversity.

It is well-known that both global exploration and local exploitation are necessary for metaheuristic optimization algorithm. However, the two aspects contradict each other. In order to reach good performances on problem optimizations, they should be well coordinated [43]. While the position-updated equation of WOA, which is applied to produce new candidate individuals based on the information of previous global best individual, and is good at local search, which results in the above two insufficiencies. In order to improve the performance of WOA, one active research trend is to investigate its position-updated equation. Therefore, in this paper, an improved version of WOA based on a modified position-updated equation and a nonlinearly distance control parameter strategy is proposed and is called EWOA, whose purpose is to accelerate convergence and improve precision of standard WOA. The EWOA algorithm is tested on 26 traditional high-dimensional complex benchmark problems with dimensions ranging from 30 to 500 and four real-world engineering applications. The statistical results demonstrate that EWOA is effective and feasible, and, most notably, has superior convergence abilities in high-dimensional space.

The remainder of this study is organized as follows. In Section 2, the classical WOA algorithm is summarized. The improved version of WOA (EWOA) is proposed based on two modified characteristics in Section 3. Section 4 tests, compares and analyzes the performance of the proposed EWOA with other algorithms based on a set of benchmark functions and real engineering problems. Finally, the conclusion of this paper is summarized in Section 5.

2 Whale Optimization Algorithm

The whale optimization algorithm (WOA) is a novel and efficient metaheuristic optimization technique developed by Mirjalili and Lewis in 2016 [18]. WOA is inspired by the bubble-net hunting mechanism of humpback whales in nature. The classical WOA composed of three consecutive stages which are the encircling prey, the bubble-net attacking, and the search for prey. In the encircling prey stage, the

Table 1. 26 Widely used benchmark test functions.

Function	Search range
$f_1(x)=\sum_{i=1}^n x_i^2$	$[-100,100]$
$f_2(x)=\sum_{i=1}^n ix_i^2$	$[-10,10]$
$f_3(x)=\sum_{i=1}^n \lvert x_i\rvert+\prod_{i=1}^n \lvert x_i\rvert$	$[-10,10]$
$f_4(x)=\sum_{i=1}^n\left(\sum_{j=1}^i x_j\right)^2$	$[-100,100]$
$f_5(x)=\max_i\{\lvert x_i\rvert,1\le i\le n\}$	$[-100,100]$
$f_6(x)=\sum_{i=1}^n\left[100(x_{i+1}-x_i^2)^2+(x_i-1)^2\right]$	$[-30,30]$
$f_7(x)=\sum_{i=1}^n(\lvert x_i+0.5\rvert)^2$	$[-100,100]$
$f_8(x)=\sum_{i=1}^n ix_i^4$	$[-1.28,1.28]$
$f_9(x)=\sum_{i=1}^n ix_i^4+\text{random}[0,1)$	$[-1.28,1.28]$
$f_{10}(x)=\sum_{i=1}^n \lvert x_i\rvert^{(i+1)}$	$[-1,1]$
$f_{11}(x)=\exp\left(0.5\cdot\sum_{i=1}^n x_i\right)-1$	$[-1,1]$
$f_{12}(x)=\sum_{i=1}^n\left(10^6\right)^{(i-1)/(n-1)}x_i^2$	$[-100,100]$
$f_{13}(x)=\sum_{i=1}^n\left[x_i^2-10\cos(2\pi x_i)+10\right]$	$[-5.12,5.12]$
$f_{14}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^n x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^n\cos(2\pi x_i)\right)+20+e$	$[-32,32]$
$f_{15}(x)=\frac{1}{4000}\sum_{i=1}^n x_i^2-\prod_{i=1}^n\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	$[-600,600]$
$f_{16}(x)=\sum_{i=1}^n(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]+\sin^2(3\pi x_i)+\lvert x_n-1\rvert[1+\sin^2(3\pi x_i)]$	$[-10,10]$
$f_{17}(x)=\sum_{i=1}^n \lvert x_i\cdot\sin(x_i)+0.1\cdot x_i\rvert$	$[-10,10]$
$f_{18}(x)=0.1n-\left(0.1\sum_{i=1}^n\cos(5\pi x_i)-\sum_{i=1}^n x_i^2\right)$	$[-1,1]$
$f_{19}(x)=\sum_{i=1}^n x_i^2+\left(\sum_{i=1}^n 0.5x_i\right)^2+\left(\sum_{i=1}^n 0.5ix_i\right)^4$	$[-5,10]$
$f_{20}(x)=\sum_{i=2}^n\left(0.5+\frac{\sin^2(\sqrt{100x_{i+1}^2+x_i^2})-0.5}{1+0.001(x_{i+1}^2-2x_{i+1}x_i+x_i^2)^2}\right)$	$[-100,100]$
$f_{21}(x)=0.1(\sin^2(3\pi x_i)+\sum_{i=1}^n(x_i-1)^2(1+\sin^2(3\pi x_{i+1}))+(x_n-1)^2(1+\sin^2(2\pi x_n)))$	$[-5,5]$
$f_{22}(x)=(-1)^{n+1}\prod_{i=1}^n\cos(x_i)\cdot\exp\left(-\sum_{i=1}^n(x_i-\pi)^2\right)$	$[-100,100]$
$f_{23}(x)=1-\cos\left(2\pi\sqrt{\sum_{i=1}^n x_i^2}\right)+0.1\sqrt{\sum_{i=1}^n x_i^2}$	$[-100,100]$
$f_{24}(x)=0.5+\frac{\sin^2\left(\sqrt{\sum_{i=1}^n x_i^2}\right)-0.5}{\left(1+0.001\left(\sum_{i=1}^n x_i^2\right)\right)^2}$	$[-100,100]$
$f_{25}(x)=\sum_{i=1}^n\left(0.2x_i^2+0.1x_i^2\cdot\sin 2x_i\right)$	$[-10,10]$
$f_{26}(x)=\sum_{i=2}^n\left(x_{i-1}^2+x_i^2\right)^{0.25}\left[\sin^2\left(50(x_{i-1}^2+x_i^2)^{0.1}\right)^2+1\right]$	$[-10,10]$

mathematical model of the encircling mechanism is formulated as follows:

$$\bar{X}(t+1)=\bar{X}^*(t)-\bar{A}\cdot\left[\bar{C}\cdot\bar{X}^*(t)-\bar{X}(t)\right]\qquad(1)$$

where $\bar{X}$ is the position vector of the whales, t is the current iteration, $\bar{X}^*$ is the position vector of the global optima, $\lvert\cdot\rvert$ is the absolute value, $\cdot$ is the element-by-element multiplication, $\bar{A}$ and $\bar{C}$ are coefficient vectors and are formulated

$$\bar{A}=2\bar{a}\cdot\bar{r}-\bar{a},\quad \bar{C}=2\cdot\bar{r}\qquad(2)$$

where $\bar{r}$ is the random vector in $[0,1]$, $\bar{a}$ is called distance control parameter. It should be noted that the distance control parameter $\bar{a}$ decreases linearly from 2 to 0 during the process of iterations, namely

$$\bar{a}(t) = 2 - 2 \times \frac{t}{t_{\max}} \tag{3}$$

where $t_{\max}$ is the maximum number of iterations.

Whales have a special hunting behavior of spiral bubble-net feeding as depicted in Figure 1.

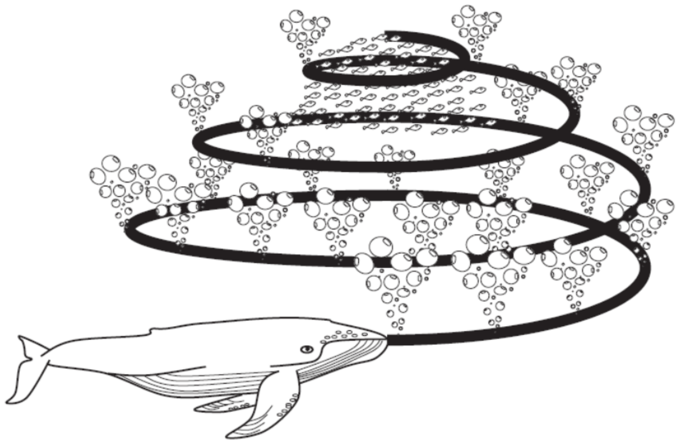


Figure 1. Bubble-net hunting behavior of humpback whales.

In the bubble-net attacking stage, this behavior is described by two mechanisms of shrinking encircling and spiral updating position. The mathematical model of spiral updating position mechanism is formulated as:

$$\bar{X}(t+1) = \bar{X}^*(t) + \left|\bar{X}^*(t) - \bar{X}(t)\right| \cdot e^{bl} \cdot \cos(2\pi l) \tag{4}$$

where b is a constant number, and l is a random number in $[-1, 1]$.

It should be noted that whales swim around the prey within a shrinking circle and move with a logarithmic spiral motion simultaneously. For simplicity, we assume that there is a probability of 50% to select between either the shrinking encircling mechanism or the logarithmic spiral motion during optimization, namely

$$\bar{X}(t+1) = \begin{cases} \bar{X}^*(t) - \bar{A} \cdot \left|\bar{C} \cdot \bar{X}^*(t) - \bar{X}(t)\right|, & \text{if } p < 0.5 \\ \bar{X}^*(t) + \left|\bar{X}^*(t) - \bar{X}(t)\right| \\ \quad \cdot e^{bl} \cdot \cos(2\pi l), & \text{if } p \geq 0.5 \end{cases} \tag{5}$$

where p is a random number in $[0, 1]$.

In the search for prey stage, whales search randomly according to the position of each other. The mathematical model is formulated by

$$\bar{X}(t+1) = \bar{X}_{\text{rand}} - \bar{A} \cdot \left|\bar{C} \cdot \bar{X}_{\text{rand}} - \bar{X}\right| \tag{6}$$

where $\bar{X}_{\text{rand}}$ is a random candidate solution selected from population.

3 Proposed EWOA Algorithm

The traditional WOA suffers from some shortcomings such as trapping in local optima, slow convergence, and time-consuming [18]. In this section, the proposed EWOA is introduced which improve the performance of traditional WOA. The No-Free-Lunch (NFL) theorem [44] mentioned that an optimization algorithm could not be used to solve all the optimization problems. This is the main motivation to develop the EWOA. The proposed algorithm does not affect the configuration of canonical WOA and improves the WOA through two modified strategies. The two strategies are explained in details as follows.

3.1 Modified Position Updating Equation

According to the position-updated equations (1) or (4), the right side of them is made up of two parts. The first part of (1) or (4) represents the previous global best position, and the second part of (1) or (4) is the one contributing to the previous global best position. The new candidate individuals are generated by equations (1) or (4) based on the information of previous global best position, which is good at local search. Therefore, one active research trend of WOA algorithm is to investigate its position-updated equation for improving the performance of WOA. Luo et al. [38] proposed a novel adaptive WOA (AWOA) based on modified position-updated equation for solving global optimization problems. However, AWOA still has some drawbacks such as easily trapped into a local optima, slow convergence and low precision.

Particle swarm optimization (PSO) is a popular metaheuristic algorithm developed by Kennedy and Eberhart in 1995 [45], which is inspired by collective behavior of bird flocking or fish schooling. In 1998, In 1998, Shi and Eberhart [46] proposed a modified version of PSO by introducing the concept of inertia weight for the original version of PSO to coordinate the ability of global search and local search in the evolution process. In addition, they pointed out that a larger value of inertia weight is beneficial to global search while a smaller value of inertia weight is conducive to local search [46]. Therefore, by changing the values of the inertia weight dynamically, the characteristics of global search and local search of PSO algorithm are dynamically adjusted.

For population-based metaheuristic optimization algorithms, appropriate coordinate of global search and local search is vital to find the global best solution effectively. Inspired by PSO, this work proposes a modified position-updated equation by introducing the parameter of inertia weight (denoted as w) for the original position-updated equations (1) and (4) to balance the ability of global search and local search effectively. The proposed modified position-updated equations are given as follows:

$$\bar{X}(t+1) = w(t) \cdot \bar{X}^*(t) - \bar{A} \cdot \left[\bar{C} \cdot \bar{X}^*(t) - \bar{X}(t)\right] \quad (7)$$

$$\bar{X}(t+1) = w(t) \cdot \bar{X}^*(t) + \left|\bar{X}^*(t) - \bar{X}(t)\right| \cdot e^{b \cdot \cos(2\pi l)} \quad (8)$$

where w is the inertia weight.

Generally, for metaheuristic optimization algorithms, maintaining a high diversity of population is very important in the early phase of the search [46]. The main purpose of this phase is to encourage the candidate individuals to traverse through the entire search space as possible. Therefore, in the equations (7) and (8), the inertia weight should be set to a relatively smaller value so as to achieve this purpose. On the other hand, accelerate convergence of population is also crucial in the latter phase of the search. The main objective of this phase is to find the global optimum effectively. Consequently, the inertia weight of the equations (7) and (8) should be set to a relatively larger value [43]. Based on the above considerations, in this study, we present a linearly increasing inertia weight throughout the course of iterations, namely

$$w(t) = w_{\min} + (w_{\max} - w_{\min}) \times \frac{t}{t_{\max}} \quad (9)$$

where $w_{\max}$ and $w_{\min}$ are represent the maximum and minimum values of the inertia weight, respectively. From equations (7), (8), and (9), the global search and local search of WOA are dynamically adjusted by changing the values of the inertia weight dynamically. Compared with reference [39], the position-updated equations of EWOA (namely, Eqs. (7), (8), and (6)) different from IWOA. In addition, unlike reference [39], the inertia weight of EWOA increases non-linearly during the process of iterations.

3.2 Nonlinear Distance Control Parameter

In the classical WOA algorithm, there are four main parameters such as $\bar{a}$, $\bar{A}$, $\bar{C}$, and b, and b. It is noted that parameter $\bar{a}$ is crucial for balancing between global exploration and local exploitation. When $|\bar{a}| > 1$, candidate individuals in the population tend to diverge from the current optimal individual to enhance the ability of global exploration; and when $|\bar{a}| < 1$, the candidate solutions in the population converge towards the current optimal solution to improve the ability of local exploitation. This goal is achieved by adjusting the distance control parameter $\bar{a}$. However, from Eq. (3), the distance control parameter $\bar{a}$ decreased linearly from 2 to 0 throughout the course of iterations to transition between global exploration and local exploitation.

Although the linear distance control parameter strategy shows faster convergence and higher accuracy in the early phase for most cases by empirical investigations with some well-known problems [18]. However, the actual search process of WOA is nonlinear and much more complicated, so it cannot be truly reflected by the linearly decreasing adaptive parameter $\bar{a}$. Therefore, this study proposes a new nonlinear distance control parameter strategy as follows:

$$\bar{a}(t) = \bar{a}_{\text{initial}} + (\bar{a}_{\text{initial}} - \bar{a}_{\text{final}}) \times \exp\left(-\lambda \times \frac{t^2}{t_{\max}^2}\right) \quad (10)$$

where $\bar{a}_{\text{initial}}$ and $\bar{a}_{\text{final}}$ are represent the initial and final values of the distance control parameter $\bar{a}$, respectively; λ is the nonlinear adjust factor.

Different from Eq. (3), the Eq. (10) is a nonlinear distance control parameter ($\bar{a}$) strategy which increases from 0 to 2 during the process of iterations. The reasons for proposing this nonlinear increasing strategy are as follows. In general, at the early phase of WOA, the population has a higher diversity. The higher population diversity means that the ability to explore the global space is stronger. The purpose of this phase is to accelerate convergence. Therefore, the value of $\bar{a}$ should be set to a smaller value and the increasing trend is faster. On the contrary, all of the whales are attracted to the current global optima at the latter phase, and they maybe converge prematurely without enough exploration of the search space. This will lead the whales trap into a local optimum. The purpose of this phase is to avoid falling into local optima, and encourage further exploration. Thus, the value of $\bar{a}$ should be set to a larger value and the increasing trend is slower. In other words, the main reasons for making this modification is that the smaller values of $\bar{a}$ at the beginning facilitate "local exploitation" and larger values of $\bar{a}$ afterward facilitate "global exploration".

We conducted several experiments on WOA with a

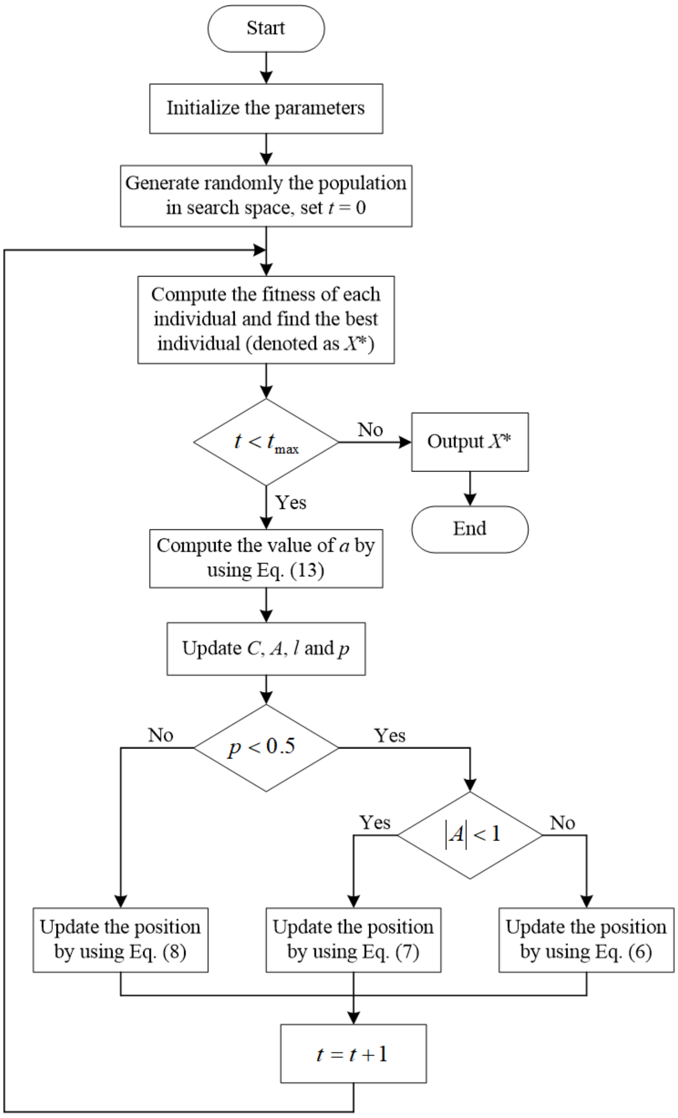


Figure 2. The flow chart of EWOA algorithm.

nonlinear modulation index λ in the interval $[0.1, 1.0]$ on average exhibited better performance on average, but a larger $\lambda(> 1.0)$ resulted in more failure in convergence. Especially, the performance of EWOA with $\lambda = 0.1$ is the best. Therefore, the value of λ is set to 0.1.

With the descriptions above, the flow chart of EWOA algorithm is shown in Figure 2.

To verify the performance of our algorithm, EWOA is compared with the classical WOA, the WOA with nonlinear distance parameter strategy (abbreviated as NWOA), and the WOA with dynamic inertia weight strategy (abbreviated as DWOA). For making a fair comparison, we take the same parameter settings for all algorithms. The control parameters' setting is given in Table 2.

Table 2. Parameters settings.

Parameters' description	Parameters' settings
Population size	$N = 30$
Maximum number of iterations	$t_{\max} = 500$
Number of fitness evaluations	NFEs $= 15{,}000$
Initial and final values of a	$a_{\text{initial}} = 2.0,\ a_{\text{final}} = 0.0$
Maximum and minimum of w	$w_{\max} = 1,\ w_{\min} = 0$
Nonlinear modulation index	$\lambda = 0.1$

All source codes are written in MATLAB 7.11.0 (win64) and executed in an Apple iMac computer with 2.8 GHz Intel Core i5, 8 GB 1867 MHz DDR3 under MAC OS operation system. The mean and standard deviation results of 26 benchmark tests functions with $D = 30$ after 30 trials are shown in Table 3. In Table 3, Wilcoxon's rank sum test at a 0.05 significance level is performed between EWOA and other three WOA variants. "+", "–", and "≈" denote that the performance of EWOA is better than, worse than, and similar to that of the corresponding approach, respectively. To attract attention, the best results are marked in boldface.

From Table 3, EWOA is able to find the global optima value 0 on nineteen functions (i.e., f_1–f_2, f_4, f_7–f_8, f_{10}–f_{13}, f_{15}–f_{16}, f_{18}–f_{22}, f_{24}–f_{26}) with fixed number of FEs 15000. For the functions f_3, f_5, f_{17}, and f_{23}, the mean values obtained by EWOA are 1e-217, 1e-198, 1e-210, and 1e-173, respectively, which show their solutions are highly near global optimum. Compared with WOA, NWOA finds better results on fifteen functions (i.e., f_1–f_4, f_8–f_{10}, f_{12}, f_{16}–f_{17}, f_{20}–f_{21}, f_{23}–f_{26}) and similar results on eleven functions (i.e., f_5, f_7, f_{11}, f_{13}–f_{15}, f_{18}–f_{19}, f_{22}). With respect to WOA, DWOA can provide better results on nineteen test functions (i.e., f_1–f_5, f_8–f_{10}, f_{12}, f_{14}, f_{16}–f_{17}, f_{19}–f_{21}, f_{23}–f_{26}) and similar results on seven functions (i.e., f_6, f_7, f_{11}, f_{13}, f_{15}, f_{18}, f_{22}). Compared with WOA, EWOA finds better and similar results for nineteen and six functions, respectively. However, for function f_6, the better result obtained by WOA. Compared to the NWOA algorithm, EWOA is able to obtain better and similar results on eighteen (i.e., f_1–f_5, f_8–f_9, f_{12}, f_{14}, f_{16}–f_{17}, f_{19}–f_{21}, f_{23}–f_{26}) and eight (i.e., f_6–f_7, f_{10}–f_{11}, f_{13}, f_{15}, f_{18}, f_{22}) test functions, respectively. With respect to DWOA, EWOA gets better and similar results on seven (i.e., f_3, f_5, f_6, f_{17}, f_{19}–f_{20}, f_{23}) and eighteen (i.e., f_1–f_2, f_4, f_7–f_8, f_{10}–f_{16}, f_{18}, f_{21}–f_{22}, f_{24}–f_{26}) test functions, respectively. However, the better result is obtained by DWOA on test function f_6.

Figure 3 shows the convergence curves of WOA, NWOA, DWOA, and EWOA on some representative

Table 3. Comparisons of EWOA with WOA, NWOA, and DWOA on 26 benchmark functions with 30 dimensions.

Functions	WOA (Mean ± St.dev)	NWOA (Mean ± St.dev)	DWOA (Mean ± St.dev)	EWOA (Mean ± St.dev)
f_1	1.60E-73 ± 7.91E-74	+6.96E-120 ± 1.80E-119	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_2	7.34E-81 ± 1.51E-80	+2.74E-124 ± 4.30E-124	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_3	2.16E-52 ± 4.65E-52	+5.95E-73 ± 1.73E-72	+1.09E-193 ± 0.00E+00	+4.26E-217 ± 0.00E+00
f_4	5.29E+04 ± 5.37E+03	+2.25E+04 ± 4.09E+03	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_5	3.68E+01 ± 2.66E+01	+3.68E+01 ± 3.90E+01	+1.38E-184 ± 0.00E+00	+4.49E-198 ± 0.00E+00
f_6	2.80E+01 ± 6.00E-01	−2.87E+01 ± 9.89E-02	≈ 2.80E+01 ± 1.75E-01	−2.87E+01 ± 6.96E-02
f_7	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ −0.00E+00 ± 0.00E+00
f_8	8.23E-119 ± 1.05E-118	+5.99E-224 ± 0.00E+00	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_9	1.92E-03 ± 1.91E-03	+8.48E-04 ± 9.79E-04	+1.10E-04 ± 4.82E-05	+4.98E-05 ± 3.69E-05
f_{10}	1.51E-118 ± 2.84E-118	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ 0E+00 ± 0.00E+00
f_{11}	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ −0.00E+00 ± 0.00E+00
f_{12}	7.86E-72 ± 8.24E-72	+6.18E-120 ± 6.55E-120	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{13}	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ −0.00E+00 ± 0.00E+00
f_{14}	5.86E-15 ± 1.94E-15	+5.15E-15 ± 2.97E-15	+8.88E-16 ± 0.00E+00	≈ 8.88E-16 ± 0.00E+00
f_{15}	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ −0.00E+00 ± 0.00E+00
f_{16}	7.24E-81 ± 6.57E-81	+5.17E-124 ± 6.00E-124	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{17}	2.75E-52 ± 3.22E-52	+2.71E-73 ± 3.03E-73	+3.06E-194 ± 0.00E+00	+3.88E-210 ± 0.00E+00
f_{18}	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ −0.00E+00 ± 0.00E+00
f_{19}	4.76E+02 ± 8.72E+01	+4.72E+02 ± 6.75E+01	+1.14E-322 ± 0.00E+00	+0.00E+00 ± 0.00E+00
f_{20}	1.97E-03 ± 4.36E-03	+2.09E-01 ± 4.64E-01	+2.80E-03 ± 6.26E-03	+0.00E+00 ± 0.00E+00
f_{21}	4.20E-81 ± 4.66E-81	+1.57E-122 ± 2.52E-122	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{22}	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ −0.00E+00 ± 0.00E+00
f_{23}	1.40E-01 ± 8.94E-02	+7.99E-02 ± 8.36E-02	+2.00E-02 ± 3.66E-02	+1.64E-173 ± 0.00E+00
f_{24}	1.52E-02 ± 1.23E-02	+1.13E-02 ± 1.52E-02	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{25}	1.12E-79 ± 1.05E-79	+5.12E-122 ± 6.17E-122	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{26}	4.23E-30 ± 4.47E-30	+9.47E-37 ± 1.97E-36	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
$+/\approx/-$	19/6/1	18/8/0	7/18/1	

benchmark test functions with $D = 30$. As can be seen from Figure 3, EWOA can obtain higher precision and faster convergence speed than the basic WOA and the EWOA variant. In addition, DWOA is also able to get faster convergence and obtain higher precision in most test functions. This is because the dynamic inertia weight strategy is beneficial for balancing convergence speed and convergence precision.

In order to further verify the scalability of the proposed algorithm, EWOA is applied to test higher-dimensional functions (i.e., $D = 100, 500$). Tables 4 and 5 provide the mean results (Mean) and the standard deviation results (Std.dev) for 30 independent runs on the all test functions in Table 1. In Tables 4 and 5, Wilcoxon's rank sum test at a 0.05 significance level is performed between EWOA and other three WOA variants. "+", "-", and "≈" denote that the performance of EWOA is better than, worse than, and similar to that of the corresponding approach, respectively. To attract attention, the best results are marked in boldface. It is noted that these results are obtained with same parameter settings for above experiments and that it does not require any increase in population size or number of function evaluations.

From Table 4, when growing the dimension of problem, the success of EWOA continues to reach best results except for function f_{20}. Compared with WOA, EWOA is able to find better results on 20 test functions with $D = 100$ (i.e., f_1–f_5, f_8–f_{12}, f_{14}, f_{16}–f_{17}, f_{19}–f_{21}, f_{23}–f_{26}). For the testing functions, two algorithms obtain similar results. EWOA significantly outperforms NWOA on nineteen functions with $D =$

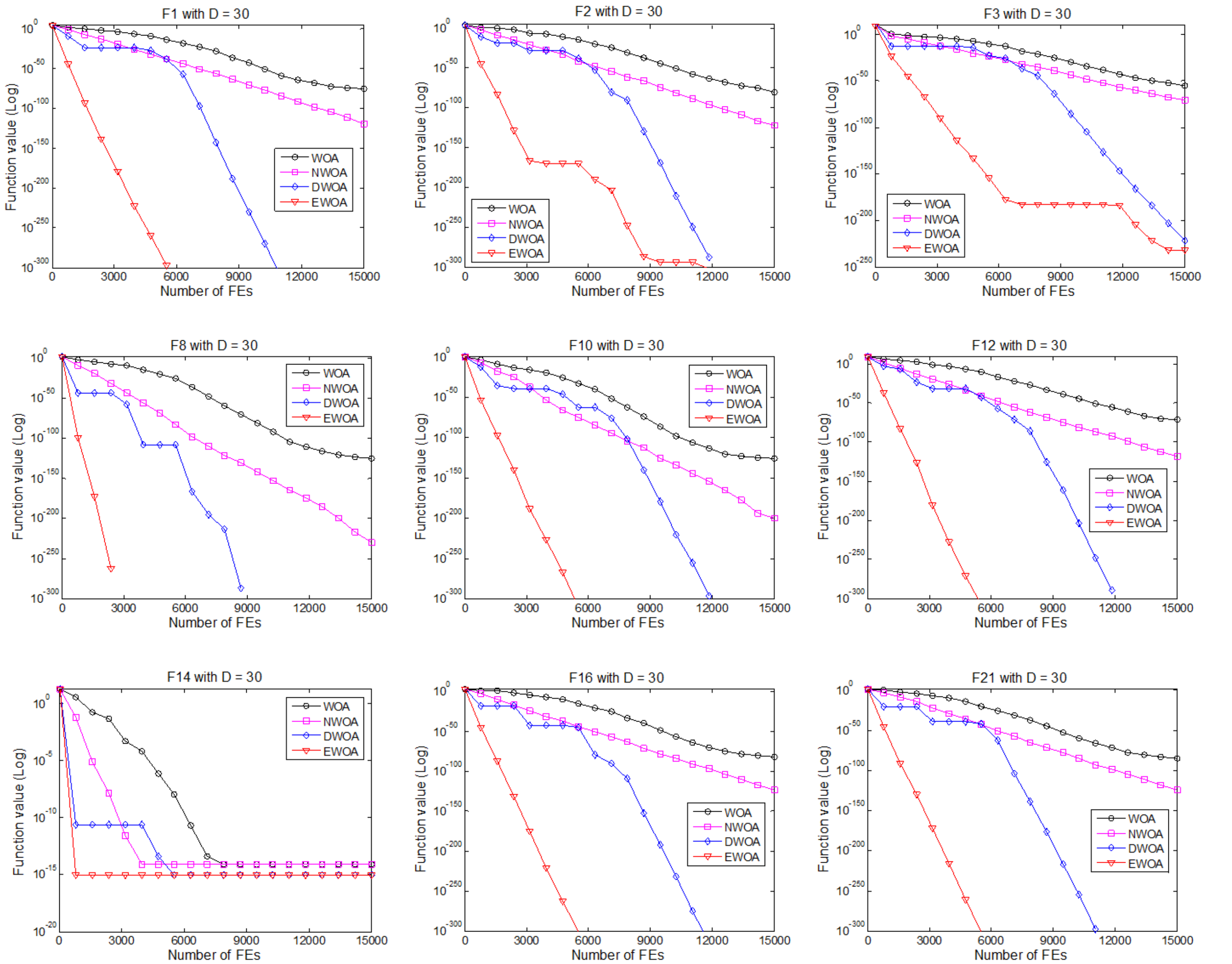


Figure 3. Convergence graphs of EWOA and other three algorithms on some representative test functions with D = 30.

100 (i.e., f_1–f_5, f_8–f_{12}, f_{16}–f_{17}, f_{19}–f_{21}, f_{23}–f_{26}). Both EWOA and NWOA can consistently provide similar results on seven functions (i.e., f_6–f_7, f_{13}–f_{15}, f_{18}, and f_{22}). With respect to DWOA, EWOA provides better and similar results on seven and nineteen test functions, respectively.

As given in Table 5, for functions with $D = 500$, EWOA has shown very good scalability to the search dimension, i.e., the performance did not deteriorate seriously as the dimension increased. EWOA significantly outperforms WOA on 21 test functions with $D = 500$. With respect to NWOA, EWOA obtains better and similar results on 19 and seven test functions, respectively. Compared with DWOA, EWOA provides better and similar results on seven and eighteen functions with $D = 500$, respectively.

For further illustration, the convergence curves of WOA, NWOA, DWOA and EWOA on some typical functions with D = 100 and D = 500 are shown in Figures 4 and 5. As seen from Figures 4 and 5, EWOA has better higher precision and faster convergence speed than WOA and NWOA in most functions. Additionally, with respect to DWOA, EWOA provides faster convergence speed on all the test functions.

For a better evaluation of the metaheuristic algorithms, not only the solution quality and the computational times should be investigated [20]. Under the same program run environment, Table 6 provides the mean CPU execution time results (in seconds) obtained by the WOA, NWOA, DWOA, and the EWOA on 30, 100, and 500 dimensions benchmark functions respectively in Table 6.

From Table 6, for 26 test functions with different

Table 4. Comparisons of EWOA with WOA, NWOA, and DWOA on 26 benchmark functions with 100 dimensions.

Functions	WOA (Mean ± St.dev)	NWOA (Mean ± St.dev)	DWOA (Mean ± St.dev)	EWOA (Mean ± St.dev)
f_1	5.80E-70 ± 1.81E-69	1.02E-115 ± 1.27E-115	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_2	7.00E-75 ± 7.08E-75	9.59E-120 ± 1.52E-119	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_3	4.73E-49 ± 2.71E-49	3.31E-69 ± 4.16E-69	4.05E-187 ± 0.00E+00	≈ 1.26E-206 ± 0.00E+00
f_4	1.18E+06 ± 5.45E+04	8.59E+05 ± 5.52E+04	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_5	5.06E+01 ± 3.92E+01	5.19E+01 ± 2.23E+00	1.64E-178 ± 0.00E+00	≈ 1.65E-192 ± 0.00E+00
f_6	9.83E+01 ± 1.41E-01	≈ 9.82E+01 ± 5.17E-02	9.80E+01 ± 1.81E-01	≈ 9.82E+01 ± 4.96E-02
f_7	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ 1.00E+00 ± 0.00E+00
f_8	1.97E-115 ± 1.78E-115	8.71E-216 ± 0.00E+00	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_9	4.15E-03 ± 3.93E-03	2.44E-03 ± 4.22E-03	4.73E-05 ± 2.33E-05	≈ 7.70E-05 ± 8.12E-05
f_{10}	3.82E-112 ± 1.17E-112	1.05E-215 ± 0.00E+00	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{11}	4.44E-17 ± 9.93E-17	4.44E-17 ± 9.93E-17	≈ 0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{12}	3.38E-68 ± 5.36E-68	5.31E-116 ± 5.51E-116	≈ 0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{13}	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ 1.00E+00 ± 0.00E+00
f_{14}	4.44E-15 ± 2.51E-15	3.73E-15 ± 2.97E-15	≈ 8.88E-16 ± 0.00E+00	≈ 8.88E-16 ± 0.00E+00
f_{15}	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ 1.00E+00 ± 0.00E+00
f_{16}	3.26E-77 ± 5.58E-77	1.38E-119 ± 2.31E-119	≈ 0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{17}	1.02E-48 ± 8.58E-49	4.04E-68 ± 9.91E-69	≈ 3.42E-188 ± 0.00E+00	≈ 5.87E-204 ± 0.00E+00
f_{18}	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ 1.00E+00 ± 0.00E+00
f_{19}	1.84E+03 ± 2.71E+02	1.61E+03 ± 2.00E+02	≈ 2.16E-230 ± 0.00E+00	≈ 1.75E-233 ± 0.00E+00
f_{20}	9.29E-01 ± 1.40E+00	7.35E-02 ± 1.64E-01	≈ 4.46E-03 ± 3.85E-03	≈ 3.53E-06 ± 2.05E-06
f_{21}	1.78E-75 ± 1.50E-75	6.56E-117 ± 7.06E-117	≈ 0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{22}	0.00E+00 ± 0.00E+00	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ 1.00E+00 ± 0.00E+00
f_{23}	1.60E-01 ± 5.48E-02	1.40E-01 ± 1.34E-01	≈ 2.00E-02 ± 4.47E-02	≈ 5.36E-155 ± 0.00E+00
f_{24}	2.07E-02 ± 1.51E-02	1.95E-02 ± 3.31E-02	≈ 1.94E-03 ± 4.35E-03	≈ 0.00E+00 ± 0.00E+00
f_{25}	4.95E-74 ± 6.39E-74	1.90E-116 ± 2.36E-116	≈ 0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{26}	1.08E-28 ± 1.64E-28	2.34E-36 ± 3.99E-36	≈ 0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
+/ ≈ /−	20/0/6	19/7/0	7/19/0	

dimensions, the mean CPU execution times of four algori-thms are very close to each other. In other words, there is no significant difference among four algorithms on 26 functions in terms of the mean CPU execution times.

3.3 Compared with Other Algorithms

In this section, EWOA is also compared with six well-known population-based algorithms: nonlinear inertia weight particle swarm optimization (NIW-PSO) [47], multi-verse optimizer [48] (MVO), teaching-learning-based optimization (TLBO) [49], grey wolf optimization (GWO) [19], elite opposition-based differential evolution (EODE) [50], and improved WOA (IWOA) [27]. 26 test functions provided in Table 1 were used to investigate the performance of EWOA and other six population-based algorithms. The parameter settings for all of the algorithms are set as follows, i.e., the number of fitness evaluations (NFEs) is 15,000. Table 7 shows the results obtained by EWOA and other six population-based algorithms for 26 functions with D = 30, where "Mean" indicates the mean best fitness value and "St.dev" represents the corresponding standard deviation value. The best results are highlighted in bold.

From Table 7, with respect to NIW-PSO, EWOA can find better results for all of the test functions including f_{22}. Compared with MVO, EWOA obtains better and similar results for twenty-four and two (f_7 and f_{22}) test functions, respectively. Compared to the TLBO algorithm, EWOA is able to provide better and similar results for twenty-two and three (f_7, f_{15}, and f_{22}) test

Table 5. Comparisons of EWOA with WOA, NWOA, and DWOA on 26 benchmark functions with 500 dimensions.

Functions	WOA (Mean ± St.dev)	NWOA (Mean ± St.dev)	DWOA (Mean ± St.dev)	EWOA (Mean ± St.dev)
f_1	1.50E-68 ± 1.92E-68	+1.27E-114 ± 1.33E-114	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_2	1.53E-68 ± 9.21E-69	+2.19E-114 ± 2.42E-114	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_3	2.43E-46 ± 3.26E-46	+1.05E-65 ± 1.48E-65	+7.16E-186 ± 0.00E+00	+7.40E-202 ± 0.00E+00
f_4	3.12E+07 ± 1.33E+07	+3.46E+07 ± 9.35E+06	+1.07E-295 ± 0.00E+00	+1.44E-299 ± 0.00E+00
f_5	8.70E+01 ± 1.26E+01	+8.30E+01 ± 9.44E+00	+6.28E-176 ± 0.00E+00	+1.06E-185 ± 0.00E+00
f_6	4.96E+02 ± 6.31E-01	≈ 4.95E+02 ± 1.91E-01	≈ 4.95E+02 ± 1.14E-01	≈ 4.95E+02 ± 2.26E-01
f_7	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ 1.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_8	5.26E-107 ± 9.68E-108	+3.18E-210 ± 0.00E+00	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_9	6.17E-03 ± 9.28E-03	+2.56E-03 ± 1.81E-03	+8.06E-05 ± 8.12E-05	≈ 1.07E-04 ± 8.48E-05
f_{10}	6.39E-108 ± 8.15E-108	+2.69E-203 ± 0.00E+00	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{11}	1.33E-16 ± 1.22E-16	+4.44E-17 ± 9.93E-17	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{12}	4.41E-63 ± 2.45E-63	+1.27E-113 ± 1.75E-113	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{13}	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ 1.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{14}	5.15E-15 ± 2.97E-15	+4.44E-15 ± 3.55E-15	≈ 8.88E-16 ± 0.00E+00	≈ 8.88E-16 ± 0.00E+00
f_{15}	7.50E-01 ± 1.68E-01	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ 1.00E+00 ± 0.00E+00
f_{16}	2.29E-71 ± 2.43E-71	+1.72E-114 ± 1.61E-114	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{17}	1.41E-46 ± 1.11E-46	+1.44E-66 ± 1.92E-66	+2.75E-186 ± 0.00E+00	+1.73E-197 ± 0.00E+00
f_{18}	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ 1.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{19}	8.23E+03 ± 1.86E+03	+7.40E+03 ± 1.43E+03	+2.92E-62 ± 6.53E-62	−3.38E-21 ± 7.56E-21
f_{20}	2.29E+00 ± 4.31E+00	+8.07E-01 ± 1.76E+00	+1.26E-02 ± 2.82E-02	+1.30E-04 ± 2.89E-04
f_{21}	3.36E-72 ± 5.43E-72	+1.33E-114 ± 2.71E-114	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{22}	0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00	≈ 1.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{23}	2.44E-01 ± 2.29E-01	+2.00E-01 ± 1.96E-01	+9.98E-02 ± 6.55E-02	+4.20E-82 ± 1.16E-81
f_{24}	2.89E-02 ± 3.00E-02	+2.07E-02 ± 1.51E-02	+9.73E-03 ± 4.92E-03	+0.00E+00 ± 0.00E+00
f_{25}	2.31E-70 ± 2.27E-70	+1.09E-115 ± 1.40E-115	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
f_{26}	1.59E-28 ± 1.82E-28	+1.60E-35 ± 1.87E-35	+0.00E+00 ± 0.00E+00	≈ 0.00E+00 ± 0.00E+00
+/ ≈ /−	21/5/0	19/7/0	7/18/1	

functions, respectively. However, the better result of f_6 is obtained by TLBO. Compared with GWO, EWOA gets better and similar results on twenty-two and three (f_7, f_{18}, and f_{22}) functions, respectively. However, GWO provides better result on function f_6. EWOA significantly outperforms EODE on 23 test functions. For f_8, f_{10}, and f_{22}, EWOA and EODE find similar results. With respect to the IWOA, EWOA can get better and similar results on eight and 16 test functions, respectively. However, IWOA obtains better values on two functions (i.e., f_3 and f_7). Based on the above comparisons, the total performance of EWOA is best.

3.4 EWOA for Engineering Design Problems

In this section, the efficacy of EWOA was evaluated using four constrained real-world engineering applications: a welded beam, a pressure vessel, a three-bar truss, and a tension/compression spring. These engineering applications were used to show the validity and effectiveness of the proposed method in the real-world problems. The detailed information of these applications could be found in their original papers. Since the engineering applications of this section have different constraints. We need to use a constraint-handling technique. Deb's feasibility-based rule [51] is one of the most common constraint-handling tech-niques. The advantage of this technique is simplicity and ease of implementation. However, this technique does not utilize the information of in-feasible solutions. For the sake of simplicity, we equip the EWOA with the Deb's feasibility-based rules in this section to handle

Table 6. Mean CPU execution time (in seconds) obtained by WOA, NWOA, DWOA, and EWOA on 26 test functions with 30D, 100D, and 500D.

Func	WOA			NWOA			DWOA			EWOA		
	$D=30$	$D=100$	$D=500$	$D=30$	$D=100$	$D=500$	$D=30$	$D=100$	$D=500$	$D=30$	$D=100$	$D=500$
f_1	1.038370	3.012480	14.66778	1.026556	2.878610	13.71042	1.032491	2.971574	14.34531	0.993139	2.845417	13.45688
f_2	1.071700	3.052560	14.71461	1.065136	2.916122	13.82436	1.064504	3.011702	14.45284	1.033194	2.871549	13.55964
f_3	1.059627	3.041608	14.70697	1.041499	2.900056	13.75940	1.044141	2.991836	14.40076	1.004750	2.867264	13.50959
f_4	1.715728	4.072717	29.06811	1.710539	3.976532	28.28050	1.731120	5.334957	28.82784	1.667934	5.213205	26.39784
f_5	1.079576	3.071014	14.65019	1.073819	2.903967	13.78883	1.076633	3.017077	14.47623	1.038609	2.884643	13.56921
f_6	1.148665	3.098953	14.84192	1.116761	2.973821	13.93086	1.117027	3.066769	14.51019	1.073114	2.929476	13.63469
f_7	1.043159	3.016907	14.73099	1.050150	2.895541	13.72988	1.064028	3.004039	14.48548	1.017031	2.843878	13.59719
f_8	1.156800	3.260351	15.65425	1.114514	3.122054	14.77796	1.130635	3.193373	15.39298	1.084445	3.065536	14.44867
f_9	1.134958	3.263768	15.65914	1.122139	3.106055	14.76279	1.149117	3.219957	15.50015	1.080886	3.079999	14.53041
f_{10}	1.132054	3.240423	15.36324	1.110051	3.073150	14.43642	1.123284	3.173115	15.11602	1.084627	3.034107	14.28630
f_{11}	1.041492	3.033796	14.63141	1.034343	2.877978	13.70887	1.051978	2.963892	14.34653	1.021500	2.836354	13.49793
f_{12}	1.142042	3.271977	15.63488	1.120814	3.117549	14.76651	1.144873	3.210436	15.46604	1.080996	3.084687	14.56201
f_{13}	1.089659	3.081776	14.75626	1.070065	2.928015	13.77987	1.092840	3.030924	14.47914	1.040055	2.896815	13.65602
f_{14}	1.128897	3.113128	14.78601	1.110585	2.969123	13.81281	1.130345	3.076043	14.56350	1.083997	2.932189	13.64773
f_{15}	1.100728	3.090928	14.78180	1.078725	2.945028	13.85080	1.103817	3.051214	14.55664	1.051627	2.909915	13.66388
f_{16}	1.119384	3.113406	14.79106	1.106915	2.967685	13.86635	1.113533	3.066363	14.60351	1.076312	2.911805	13.47781
f_{17}	1.067215	3.040821	14.67590	1.046176	2.900791	13.76864	1.065883	3.008708	14.40689	1.013964	2.872994	13.51672
f_{18}	1.090083	3.077235	14.71559	1.063903	2.934664	13.76806	1.093465	3.047069	14.50605	1.033209	2.887334	13.60201
f_{19}	1.125138	3.096443	14.75113	1.095944	2.955568	13.80967	1.104385	3.052168	14.52264	1.075981	2.918973	13.62490
f_{20}	1.195568	3.214548	15.01701	1.175390	3.065069	14.07260	1.187097	3.164337	14.78657	1.143919	3.014710	13.86535
f_{21}	1.121239	3.090465	14.75511	1.085034	2.939718	13.84259	1.099847	3.051028	14.54321	1.049252	2.911616	13.62072
f_{22}	1.175441	3.163483	14.99110	1.127322	3.040320	14.01318	1.143956	3.138462	14.67568	1.100603	2.998801	13.82264
f_{23}	1.115322	3.075925	14.74467	1.105250	2.959920	14.03826	1.111565	3.056898	14.55078	1.099735	2.931785	13.58749
f_{24}	1.101000	3.073900	14.74108	1.078764	2.947219	13.98021	1.093489	3.041965	14.48588	1.051020	2.905130	13.54963
f_{25}	1.074246	3.065386	14.73708	1.065096	2.918773	13.95371	1.071149	3.021457	14.52801	1.030965	2.873838	13.62819
f_{26}	1.253069	3.596493	16.67026	1.239766	3.407728	15.98762	1.242712	3.479621	16.49671	1.204986	3.364898	15.68911

constraints. EWOA parameters for these real-world engineering applic-ations were set as follows, i.e., the number of fitness evaluations (NFEs) was set 50,000.

Tables 8, 9, 10 and 11 compare the optimization results obtained by EWOA with those of other optimization methods reported in the literature on pressure vessel design, tension/ compression spring design, three-bar truss design, and welded beam design problems, respectively. Please note that the results of other methods are taken from their original papers. "NFEs" indicates the number of function evaluation. Note that "N/A" denotes the results are not available in Tables 8, 9, 10 and 11.

From Table 8, for the pressure vessel design problem, it is seen that in terms of the best index (i.e., f(best)), the results obtained by EWOA were the same as the results found by SCPSO, CS, and HPSO, but superior to those obtained by the remaining algorithms. In terms of the mean, worst, and st.dev indices, EWOA get more promising results than most algorithms. In addition, the number of FEs by EWOA (50,000) is moderate among the compared algorithms.

For the tension/compression spring design problem, the best index in Table 9 shows that the results obtained by EWOA was inferior to SCPSO, CSA, EEGWO, and HPSO. In terms of the mean, worst, and st.dev indices, the results obtained by EWOA were better than those obtained by the other algorithms except for GWO, EEGWO, and CSA. As far as the number of FEs is concerned, AATM had the minimum number of FEs (25,000), while CPSO had a considerable number of FEs (200,000). EWOA had the moderate number of FEs (50,000).

As it can be seen in Table 10, the best results for the three-bar truss design problem were found by EWOA, SC, AATM, DEDS, and HEAA. For the mean, worst, and st.dev results, EWOA is better than SC, SCA, GSA, and WOA and inferior to the remaining algorithms.

Table 7. Comparisons of EWOA and other six algorithms on 26 benchmark functions with 30 dimensions.

Function	Statistical result	NIW-PSO	MVO	TLBO	GWO	EODE	IWOA	EWOA
f_1	Mean	5.46E-09	7.42E-08	7.03E-120	1.18E-29	1.20E-178	**0.00E+00**	**0.00E+00**
	St.dev	2.81E-09	1.54E-07	1.00E-119	1.93E-29	0.00E+00	**0.00E+00**	**0.00E+00**
f_2	Mean	3.27E-10	1.02E-07	4.45E-121	2.92E-30	5.00E-180	**0.00E+00**	**0.00E+00**
	St.dev	2.69E-10	8.28E-08	4.50E-121	3.35E-30	0.00E+00	**0.00E+00**	**0.00E+00**
f_3	Mean	2.83E-04	2.70E-04	1.44E-60	4.49E-18	1.02E-94	**0.00E+00**	4.26E-217
	St.dev	3.00E-04	1.67E-04	6.76E-61	2.86E-18	2.28E-94	**0.00E+00**	0.00E+00
f_4	Mean	9.50E-11	3.23E-06	6.97E-119	1.17E-07	6.00E-177	1.03E+00	**0.00E+00**
	St.dev	1.06E-10	5.04E-06	8.00E-119	1.16E-07	0.00E+00	6.96E-01	**0.00E+00**
f_5	Mean	4.31E+00	1.39E-04	4.75E-51	2.14E-07	8.96E-73	6.37E-01	**4.49E-198**
	St.dev	6.41E-01	7.47E-05	2.79E-51	2.10E-07	2.00E-72	4.98E-01	**0.00E+00**
f_6	Mean	5.22E+01	2.89E+01	**2.64E+01**	2.76E+01	2.92E+01	2.88E+01	2.87E+01
	St.dev	3.51E+01	1.44E-02	**8.00E-01**	1.14E+00	1.63E-01	3.13E-01	6.96E-02
f_7	Mean	1.20E+00	**0.00E+00**	**0.00E+00**	**0.00E+00**	6.00E-01	**0.00E+00**	**0.00E+00**
	St.dev	8.37E-01	**0.00E+00**	**0.00E+00**	**0.00E+00**	5.48E-01	**0.00E+00**	**0.00E+00**
f_8	Mean	1.23E-15	8.69E-21	3.42E-238	5.68E-56	**0.00E+00**	**0.00E+00**	**0.00E+00**
	St.dev	1.48E-15	8.68E-21	0.00E+00	5.08E-56	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_9	Mean	2.49E-02	2.81E-04	8.75E-04	2.94E-03	2.23E-03	4.65E-02	**4.98E-05**
	St.dev	6.16E-03	3.93E-04	1.98E-04	1.03E-03	1.98E-03	1.90E-02	**3.69E-05**
f_{10}	Mean	5.51E-28	1.22E-19	2.86E-277	2.34E-103	**0.00E+00**	**0.00E+00**	**0.00E+00**
	St.dev	9.86E-28	1.74E-19	0.00E+00	5.20E-103	**0.00E+00**	**0.00E+00**	**0.00E+00**
f_{11}	Mean	3.06E-07	1.98E-02	4.59E-09	6.54E-07	4.00E-02	**0.00E+00**	**0.00E+00**
	St.dev	0.00E+00	5.09E-03	0.00E+00	7.18E-07	3.83E-02	**0.00E+00**	**0.00E+00**
f_{12}	Mean	3.32E-05	7.57E-03	1.55E-115	1.05E-25	2.20E-174	**0.00E+00**	**0.00E+00**
	St.dev	2.92E-05	1.28E-02	2.20E-115	9.25E-26	0.00E+00	**0.00E+00**	**0.00E+00**
f_{13}	Mean	6.19E+01	2.63E-07	1.89E+01	1.59E-13	1.24E+02	**0.00E+00**	**0.00E+00**
	St.dev	1.34E+01	2.92E-07	1.93E+01	1.68E-13	9.95E+01	**0.00E+00**	**0.00E+00**
f_{14}	Mean	1.14E-05	9.35E-05	2.22E-15	7.05E-14	3.55E-15	**8.88E-16**	**8.88E-16**
	St.dev	4.73E-06	5.06E-05	0.00E+00	6.45E-15	1.99E-15	**0.00E+00**	**0.00E+00**
f_{15}	Mean	2.92E-02	2.39E-06	**0.00E+00**	3.04E-03	2.95E-02	**0.00E+00**	**0.00E+00**
	St.dev	6.52E-02	3.89E-06	**0.00E+00**	6.80E-03	6.59E-02	**0.00E+00**	**0.00E+00**
f_{16}	Mean	1.70E+01	1.33E-09	1.36E-120	2.53E-30	1.72E+02	**0.00E+00**	**0.00E+00**
	St.dev	1.97E+01	1.29E-09	2.00E-120	2.96E-30	3.16E+02	**0.00E+00**	**0.00E+00**
f_{17}	Mean	4.48E-03	3.62E-05	1.14E-60	1.85E-04	1.34E+01	**0.00E+00**	3.88E-210
	St.dev	8.65E-03	2.19E-05	1.40E-60	2.93E-04	1.55E+01	**0.00E+00**	0.00E+00
f_{18}	Mean	7.09E-01	2.52E-10	4.23E-123	**0.00E+00**	2.40E-185	**0.00E+00**	**0.00E+00**
	St.dev	2.19E-01	3.36E-10	6.80E-123	**0.00E+00**	0.00E+00	**0.00E+00**	**0.00E+00**
f_{19}	Mean	3.61E+00	3.78E+08	2.87E-18	7.44E-08	6.95E+08	1.71E+01	**0.00E+00**
	St.dev	1.02E+00	3.69E+08	4.69E-18	1.59E-07	1.07E+09	8.34E+00	**0.00E+00**
f_{20}	Mean	1.12E+01	3.82E-05	6.04E+00	1.09E+01	1.02E+01	5.47E-02	**0.00E+00**
	St.dev	4.66E-01	3.61E-05	1.25E+00	9.54E-01	2.60E+00	5.06E-02	**0.00E+00**
f_{21}	Mean	4.39E-03	5.24E-10	1.55E-121	1.58E-32	1.40E-183	**0.00E+00**	**0.00E+00**
	St.dev	6.02E-03	8.09E-10	3.40E-121	2.41E-32	0.00E+00	**0.00E+00**	**0.00E+00**
f_{22}	Mean	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
	St.dev	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00	0.00E+00
f_{23}	Mean	6.20E-01	2.88E-05	9.99E-02	1.80E-01	2.00E-01	1.80E-01	**1.64E-173**
	St.dev	1.30E-01	2.39E-05	0.00E+00	4.47E-02	0.00E+00	1.30E-01	**0.00E+00**
f_{24}	Mean	1.47E-01	4.86E-07	9.72E-03	3.17E-02	7.23E-02	9.72E-03	**0.00E+00**
	St.dev	2.81E-02	5.69E-07	0.00E+00	1.23E-02	7.63E-02	0.00E+00	**0.00E+00**
f_{25}	Mean	1.99E-11	2.68E-12	5.23E-121	1.21E-31	1.20E-186	**0.00E+00**	**0.00E+00**
	St.dev	2.13E-11	8.57E-13	1.10E-120	2.05E-31	0.00E+00	**0.00E+00**	**0.00E+00**
f_{26}	Mean	1.28E+01	8.64E-02	6.25E-29	7.70E-08	2.13E-47	**0.00E+00**	**0.00E+00**
	St.dev	6.87E+00	9.27E-02	2.77E-29	3.49E-08	3.31E-47	**0.00E+00**	**0.00E+00**

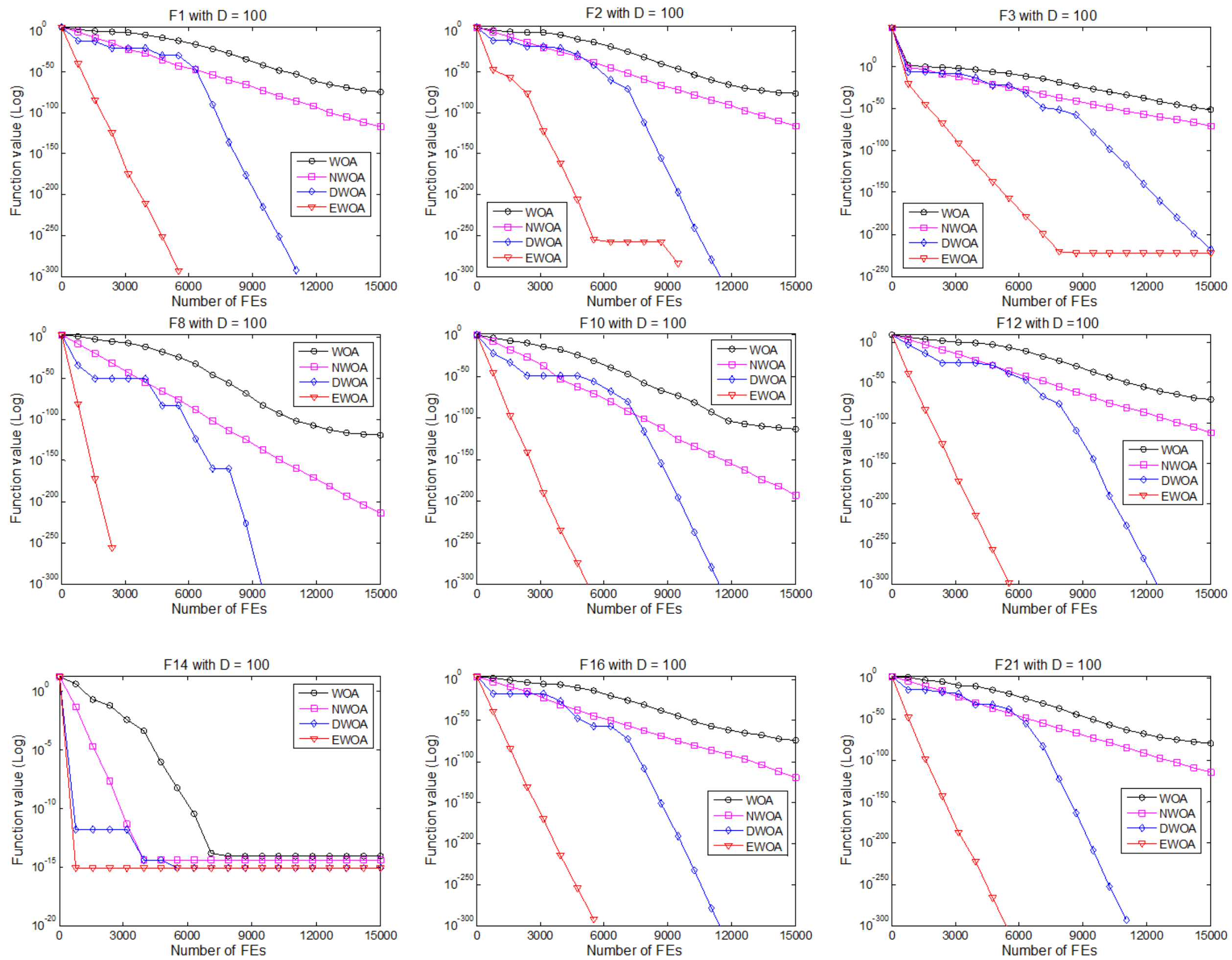


Figure 4. Convergence graphs of EWOA and other three algorithms on some representative test functions with D = 100.

Table 8. Comparison of EWOA and other ten algorithms for pressure vessel design problem.

Algorithm	$x_1(T_s)$	$x_2(T_h)$	$x_3(R)$	$x_4(L)$	f(best)	f(mean)	f(worst)	St.dev	NFEs
GA [52]	0.8125	0.4375	40.3239	200.00	6288.7445	6293.8432	6308.497	7.4133	900,000
CPSO [53]	0.8125	0.4375	42.0913	176.7465	6061.0777	6147.1332	6363.8041	86.45	240,000
CDE [54]	0.8125	0.4375	42.0984	176.6377	6059.734	6085.2303	6371.0455	43.013	204,800
SCPSO [55]	0.8125	0.4375	42.0984	176.6366	6059.7143	6358.157	7332.8415	372.71	70,650
CS [56]	0.8125	0.4375	42.0984	176.6366	6059.714	6447.736	6459.347	502.693	15,000
CSA [57]	0.8125	0.4375	42.0984	176.6366	6059.7144	6342.4991	7332.8416	384.9454	250,000
EEGWO [58]	13.0929	6.7922	42.0976	176.6495	6059.8704	6066.722	6091.0922	10.64121	50,000
HPSO [59]	0.8125	0.4375	42.0984	176.6366	6059.7143	6099.9323	6288.677	86.2	81,000
SPGA [60]	0.8125	0.4375	42.0974	176.654	6059.9463	6177.2533	6469.322	130.9297	80,000
WOA	0.8125	0.4375	42.0983	176.639	6059.741	6319.7386	6679.8199	244.7956	50,000
EWOA	0.8125	0.4375	42.0984	176.6366	6059.7143	6100.6204	6185.589	51.1511	50,000

Additionally, EWOA had a considerable number of FEs (50,000).

For the welded beam design problem, in terms of the best index in Table 11 shows that result found by HEAA was superior to those obtained by the remaining algorithms. In addition, the mean, worst, and st.dev results obtained by EWOA is better than all of the algorithms except for HEAA and EEGWO. For the number of FEs, EWOA had the moderate number of FEs (50,000).

4 Conclusion

In this paper, a novel variant of WOA, called EWOA, was developed based on two modifications. In

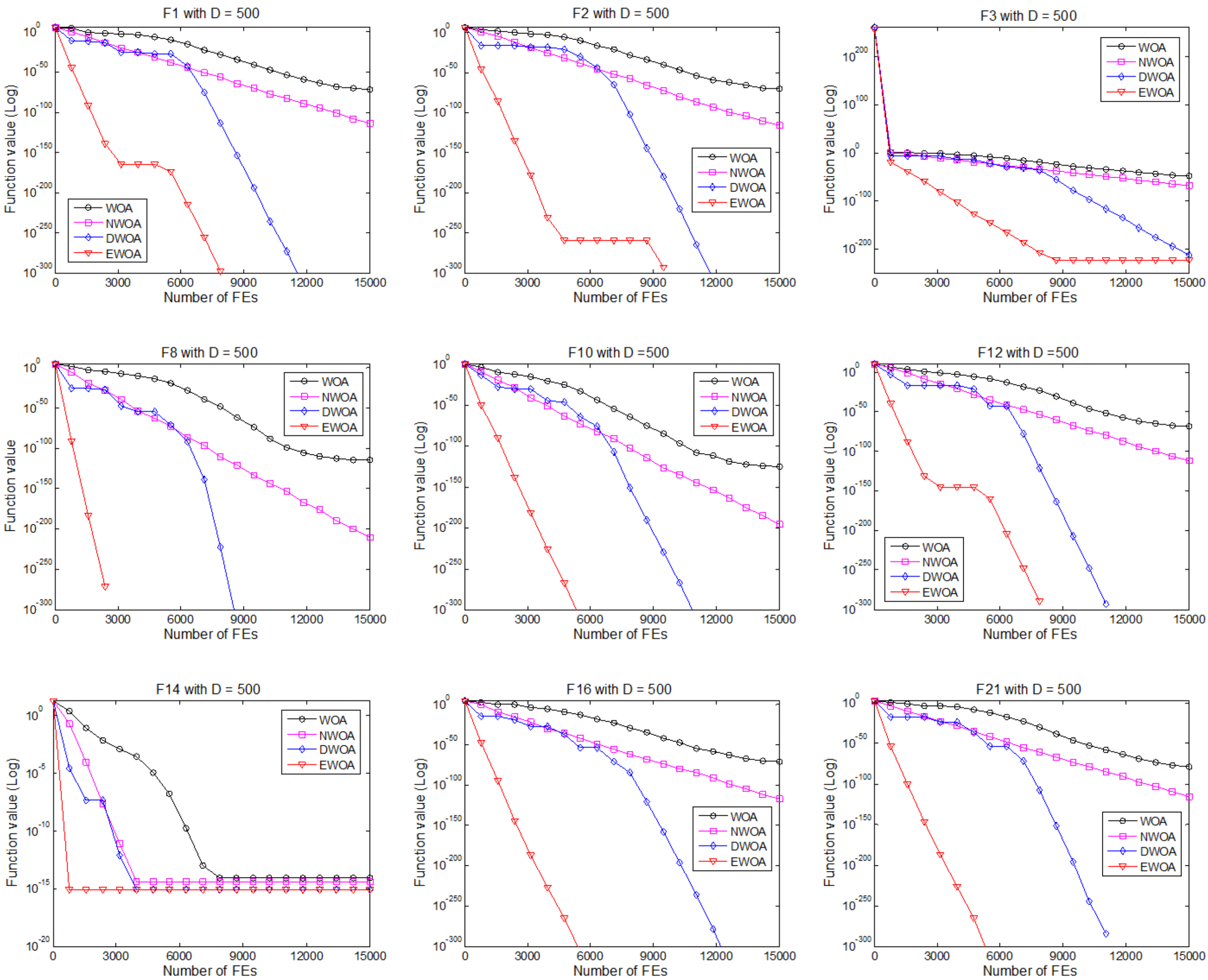


Figure 5. Convergence graphs of EWOA and other three algorithms on some representative test functions with D = 500.

Table 9. Comparison of EWOA and other eleven algorithms for tension/compression spring design problem.

Algorithm	$x_1(d)$	$x_2(D)$	$x_3(P)$	f(best)	f(mean)	f(worst)	St.dev	NFEs
SSO [17]	12.004032	0.345215	0.051207	0.0126763	N/A	N/A	N/A	30,000
SC [61]	10.6484423	0.3681587	0.0521602	0.012669	0.012923	0.016717	5.92E-04	30,000
CPSO [53]	11.244543	0.357644	0.054728	0.0126747	0.01273	0.012924	5.20E-05	200,000
GWO [19]	12.04249	0.344541	0.051178	0.0126723	0.0126971	0.0127208	2.10E-05	30,000
SCPSO [55]	11.289687	0.356705	0.051688	0.0126652	0.0127576	0.0146117	2.70E-04	40,000
AATM [62]	11.119253	0.051813	0.35969	0.0126683	0.0127081	0.0128614	4.50E-05	25,000
CSA [57]	11.289012	0.356717	0.051689	0.0126652	0.012666	0.0126702	1.36E-06	50,000
EEGWO [58]	11.3113	0.35634	0.051673	0.012665	0.012685	0.01272	2.22E-05	50,000
HPSO [59]	11.265038	0.357126	0.051706	0.0126652	0.0127072	0.0127191	1.58E-05	81,000
MVO [48]	14.22623	0.315956	0.05	0.0128169	0.0144644	0.0178397	1.62E-03	30,000
WOA	11.1029	0.360405	0.0518539	0.012698	0.0129066	0.013191	2.27E-04	50,000
EWOA	11.3695	0.355461	0.0516393	0.0126673	0.0127021	0.012866	8.38E-05	50,000

Table 10. Comparison of EWOA and other eleven algorithms for three-bar truss design problem.

Algorithm	x_1	x_2	f(best)	f(mean)	f(worst)	St.dev	NFEs
SSO [17]	0.7886654	0.40827578	263.895843	N/A	N/A	N/A	30,000
SC [60]	0.788621	0.408401	263.8958	263.9033	263.9697	1.30E-02	17,610
AATM [61]	0.788682	0.408229	263.8958	263.8966	263.9004	1.10E-03	17,000
GWO [19]	0.788409	0.409403	263.8959	263.8966	263.898	4.37E-04	30,000
MVO [62]	0.788993	0.407351	263.8959	263.8961	263.8971	2.49E-04	30,000
DEDS [63]	0.788651	0.408316	263.8958	263.8958	263.8958	9.70E-07	15,000
SCA [22]	0.789086	0.407162	263.8984	263.9356	263.9951	2.88E-02	30,000
EEGWO [57]	0.78841	0.40899	263.896	263.8963	263.8966	2.19E-04	50,000
GSA [64]	0.777662	0.448853	264.8299	271.0348	279.7925	4.13E+00	30,000
HEAA [65]	0.78868	0.408234	263.8958	263.8959	263.8961	4.90E-05	15,000
WOA	0.78915	0.40691	263.897	263.9143	263.9298	1.32E-02	50,000
EWOA	0.78863	0.40834	263.8958	263.8979	263.9006	1.70E-03	50,000

Table 11. Comparison of EWOA and other eleven algorithms for welded beam design problem.

Algorithm	$x_1(h)$	$x_2(l)$	$x_3(t)$	$x_4(b)$	f(best)	f(mean)	f(worst)	St.dev	NFEs
FSA [66]	0.2443	6.2158	8.2939	0.2443	2.3811	2.4042	2.489	N/A	56,243
SC [61]	0.2444	6.238	8.2886	0.2446	2.3854	3.2551	6.3997	9.60E-01	33,095
AATM [62]	0.2441	6.2209	8.2982	0.2444	2.3823	2.387	2.3916	2.20E-03	30,000
HEAA [65]	0.2444	6.2175	8.2915	0.2444	2.3810	2.3810	2.3810	1.30E-05	30,000
GWO [19]	0.2444	6.2069	8.3036	0.2444	2.3822	2.3831	2.3846	9.37E-04	50,000
EEGWO [58]	0.2444	6.217	8.2928	0.2444	2.3813	2.3817	2.3824	4.18E-04	50,000
WOA	0.2447	6.2078	8.2892	0.2448	2.3839	2.3916	2.4031	8.08E-03	50,000
EWOA	0.2443	6.2144	8.2982	0.2443	2.3814	2.3823	2.3834	7.60E-04	50,000

EWOA, as inspired by particle swarm optimization, a modified position-updating equation was proposed by introducing a dynamic inertia weight to enhance the search ability. In addition, a modified distance control parameter strategy based on the exponential function was presented to achieve a trade-off between exploration and exploitation. The 26 widely used benchmark test functions and four well-known engineering applications were utilized to investigate the effectiveness and efficiency of EWOA. The experimental results demonstrate that the proposed EWOA algorithm could provide very competitive results compared to well-known and state-of-the-art population-based optimization algorithms in most cases.

Although EWOA is efficient, it has some shortcomings that should be addressed in future studies. One of the limitations in EWOA is that finding an optimal solution is not 100% guaranteed (such as f_5). In future work, we would investigate how to extend the EWOA to handle constrained single- and multi-objective optimization, and combined optimization. The application of EWOA is desirable to solve more complex real-world problems.

Data Availability Statement

Data will be made available on request.

Funding

This work was supported in part by the National Natural Science Foundation of China under Grant 12361106; in part by the Guizhou Provincial Science and Technology Plan Key Projects under Grant Qiankehe Jichu ZK[2023]003; in part by the Guizhou Provincial High Level Innovative Talent Training Plan Project under Grant Qiankehe Platform Talent GCC[2023]006.

Conflicts of Interest

Wen Long served as an Associate Editor of the *Journal of Mathematics and Interdisciplinary Applications* at the time of manuscript submission. To ensure the integrity of the peer-review process, Wen Long was not involved in the editorial handling, peer review, or decision-making process for this manuscript, which was handled independently by another editor. The remaining authors declare no conflicts of interest.

AI Use Statement

The authors declare that no generative AI was used in the preparation of this manuscript.

Ethical Approval and Consent to Participate

Not applicable.

References

[1] Long, W., Wang, Y., Long, Q., Yang, Y., & Xu, M. (2026). Phased-Enhancement Marine Predators Algorithm for Global Optimization and Medical Insurance Fraud Detection. *Journal of Bionic Engineering*, *23*(2), 1088-1111. [CrossRef]

[2] Skakovski, A., & Jędrzejowicz, P. (2019). An island-based differential evolution algorithm with the multi-size populations. *Expert Systems with Applications*, *126*, 308-320. [CrossRef]

[3] Xiang, Z., Ji, D., Zhang, H., Wu, H., & Li, Y. (2019). A simple PID-based strategy for particle swarm optimization algorithm. *Information Sciences*, *502*, 558-574. [CrossRef]

[4] Saad, M. R., Emam, M. M., Hosney, M. E., Samee, N. A., Alkanhel, R. I., & Houssein, E. H. (2026). Fourier transform optimizer: A novel physics-inspired metaheuristic algorithm for optimization problems. *Knowledge-Based Systems*, 115651. [CrossRef]

[5] Fu, Y., Liu, D., Chen, J., & He, L. (2024). Secretary bird optimization algorithm: a new metaheuristic for solving global optimization problems. *Artificial Intelligence Review*, *57*(5), 123. [CrossRef]

[6] Trojovský, P., Dehghani, M., Trojovská, E., & Milkova, E. (2023). Language Education Optimization: A New Human-Based Metaheuristic Algorithm for Solving Optimization Problems. *CMES-Computer Modeling in Engineering and Sciences*, *136*(2), 1527-1573. [CrossRef]

[7] Zhang, J., Xiao, M., Gao, L., & Pan, Q. (2018). Queuing search algorithm: A novel metaheuristic algorithm for solving engineering optimization problems. *Applied Mathematical Modelling*, *63*, 464-490. [CrossRef]

[8] Kundu, R., Chattopadhyay, S., Nag, S., Navarro, M. A., & Oliva, D. (2024). Prism refraction search: a novel physics-based metaheuristic algorithm. *The Journal of Supercomputing*, *80*(8), 10746-10795. [CrossRef]

[9] Azizi, M., Talatahari, S., & Gandomi, A. H. (2023). Fire Hawk Optimizer: A novel metaheuristic algorithm. *Artificial Intelligence Review*, *56*(1), 287-363. [CrossRef]

[10] Bohat, V. K., Hashim, F. A., Batra, H., & Abd Elaziz, M. (2025). Phototropic growth algorithm: A novel metaheuristic inspired from phototropic growth of plants. *Knowledge-Based Systems*, *322*, 113548. [CrossRef]

[11] Shadravan, S., Naji, H. R., & Bardsiri, V. K. (2019). The Sailfish Optimizer: A novel nature-inspired metaheuristic algorithm for solving constrained engineering optimization problems. *Engineering Applications of Artificial Intelligence*, *80*, 20-34. [CrossRef]

[12] Alimoradi, M., Azgomi, H., & Asghari, A. (2022). Trees social relations optimization algorithm: a new swarm-based metaheuristic technique to solve continuous and discrete optimization problems. *Mathematics and computers in simulation*, *194*, 629-664. [CrossRef]

[13] Mozhdehi, A. T., Khodadadi, N., Aboutalebi, M., El-Kenawy, E. S. M., Hussien, A. G., Zhao, W., ... & Mirjalili, S. (2025). Divine Religions Algorithm: A novel social-inspired metaheuristic algorithm for engineering and continuous optimization problems. *Cluster Computing*, *28*(4), 253. [CrossRef]

[14] Abdel-Basset, M., Mohamed, R., & Abouhawwash, M. (2024). Crested Porcupine Optimizer: A new nature-inspired metaheuristic. *Knowledge-Based Systems*, *284*, 111257. [CrossRef]

[15] Zhao, W., Xie, Y., Wang, L., Zhang, Z., Khodadadi, N., & Mirjalili, S. (2026). An effective Bezier curve-based optimization (BCO) for large-scale numerical problems and 3D unmanned aerial vehicle path planning with efficient multiple threats evasion. *Advanced Engineering Informatics*, *73*, 104524. [CrossRef]

[16] Uymaz, S. A., Tezel, G., & Yel, E. (2015). Artificial algae algorithm (AAA) for nonlinear global optimization. *Applied soft computing*, *31*, 153-171. [CrossRef]

[17] Mirjalili, S., Gandomi, A. H., Mirjalili, S. Z., Saremi, S., Faris, H., & Mirjalili, S. M. (2017). Salp Swarm Algorithm: A bio-inspired optimizer for engineering design problems. *Advances in engineering software*, *114*, 163-191. [CrossRef]

[18] Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in engineering software*, *95*, 51-67. [CrossRef]

[19] Mirjalili, S., Mirjalili, S. M., & Lewis, A. (2014). Grey wolf optimizer. *Advances in engineering software*, *69*, 46-61. [CrossRef]

[20] Derrac, J., García, S., Molina, D., & Herrera, F. (2011). A practical tutorial on the use of nonparametric statistical tests as a methodology for comparing evolutionary and swarm intelligence algorithms. *Swarm and Evolutionary Computation*, *1*(1), 3-18. [CrossRef]

[21] Peng, Y., Gu, S., Liang, Y., Ouyang, K., Li, Y., Wang, K., ... & Fan, C. (2025). Wave Optics Optimizer: A novel meta-heuristic algorithm for engineering optimization. *Communications in Nonlinear Science and Numerical Simulation*, 109337. [CrossRef]

[22] Gharehchopogh, F. S., & Gholizadeh, H. (2019). A comprehensive survey: Whale Optimization Algorithm and its applications. *Swarm and Evolutionary Computation*, *48*, 1-24. [CrossRef]

[23] Oliva, D., Abd El Aziz, M., & Hassanien, A. E. (2017). Parameter estimation of photovoltaic cells using an improved chaotic whale optimization algorithm. *Applied energy*, *200*, 141-154. [CrossRef]

[24] Abd Elaziz, M., & Oliva, D. (2018). Parameter estimation of solar cells diode models by an improved opposition-based whale optimization algorithm. *Energy conversion and management*, *171*, 1843-1859. [CrossRef]

[25] Li, Z. (2026). An improved chaotic whale optimization algorithm for MHA-MLP stock trend forecasting. *Alexandria Engineering Journal*, *134*, 69-80. [CrossRef]

[26] Manafi, E., Domenech, B., Tavakkoli-Moghaddam, R., & Ranaboldo, M. (2025). A self-learning whale optimization algorithm based on reinforcement learning for a dual-resource flexible job shop scheduling problem. *Applied Soft Computing*, *180*, 113436. [CrossRef]

[27] Tubishat, M., Abushariah, M. A., Idris, N., & Aljarah, I. (2019). Improved whale optimization algorithm for feature selection in Arabic sentiment analysis. *Applied Intelligence*, *49*(5), 1688-1707. [CrossRef]

[28] Yousri, D., Allam, D., & Eteiba, M. B. (2019). Chaotic whale optimizer variants for parameters estimation of the chaotic behavior in Permanent Magnet Synchronous Motor. *Applied Soft Computing*, *74*, 479-503. [CrossRef]

[29] Zhang, Y., Li, J., Ma, H., Su, Z., & Wu, J. (2026). MIWOA: A multi-strategy improved whale optimization algorithm with hybrid perturbation for engineering optimization. *Expert Systems with Applications*, *302*, 130245. [CrossRef]

[30] Yang, W., Shang, W., & Liu, Z. (2025). Dynamic Cooperative Whale Optimization Algorithm for Multivehicle IoV Path Planning. *IEEE Internet of Things Journal*, *12*(15), 29844-29859. [CrossRef]

[31] Raj, S., & Bhattacharyya, B. (2018). Optimal placement of TCSC and SVC for reactive power planning using Whale optimization algorithm. *Swarm and Evolutionary Computation*, *40*, 131-143. [CrossRef]

[32] Chifu, V. R., Cioara, T., Anitei, C. D., Pop, C. B., Anghel, I., & Toderean, L. (2025). A federated learning model with the whale optimization algorithm for renewable energy prediction. *Computers and Electrical Engineering*, *123*, 110259. [CrossRef]

[33] Petrović, M., Miljković, Z., & Jokić, A. (2019). A novel methodology for optimal single mobile robot scheduling using whale optimization algorithm. *Applied Soft Computing*, *81*, 105520. [CrossRef]

[34] Ghahremani-Nahr, J., Kian, R., & Sabet, E. (2019). A robust fuzzy mathematical programming model for the closed-loop supply chain network design and a whale optimization solution algorithm. *Expert systems with applications*, *116*, 454-471. [CrossRef]

[35] Zhang, J., Liu, W., Zhang, G., & Zhang, T. (2025). Quantum encoding whale optimization algorithm for global optimization and adaptive infinite impulse response system identification. *Artificial Intelligence Review*, *58*(5), 158. [CrossRef]

[36] Soltanpour, M., & Zhang, H. (2024). Near-global-optimal resource allocation for uplink SCMA systems based on whale optimization algorithm. *IEEE Transactions on Wireless Communications*, *24*(2), 1089-1103. [CrossRef]

[37] Ling, Y., Zhou, Y., & Luo, Q. (2017). Lévy flight trajectory-based whale optimization algorithm for global optimization. *IEEE Access*, *5*, 6168-6186. [CrossRef]

[38] Luo, J., Chen, H., Heidari, A. A., Xu, Y., Zhang, Q., & Li, C. (2019). Multi-strategy boosted mutative whale-inspired optimization approaches. *Applied Mathematical Modelling*, *73*, 109-123. [CrossRef]

[39] Hu, H., Bai, Y., & Xu, T. (2017). Improved whale optimization algorithm based on inertia weights and their applications. *International Journal of Circuits, Systems and Signal Processing*, *11*, 12-26. https://www.naun.org/main/NAUN/circuitssystemssignal/2017/a042005-aat.pdf

[40] Sun, Y., Wang, X., Chen, Y., & Liu, Z. (2018). A modified whale optimization algorithm for large-scale global optimization problems. *Expert Systems with Applications*, *114*, 563-577. [CrossRef]

[41] Chen, H., Xu, Y., Wang, M., & Zhao, X. (2019). A balanced whale optimization algorithm for constrained engineering design problems. *Applied Mathematical Modelling*, *71*, 45-59. [CrossRef]

[42] Wang, W. L., Li, W. K., Wang, Z., & Li, L. (2019). Opposition-based multi-objective whale optimization algorithm with global grid ranking. *Neurocomputing*, *341*, 41-59. [CrossRef]

[43] Liu, J., Mei, Y., & Li, X. (2015). An analysis of the inertia weight parameter for binary particle swarm optimization. *IEEE Transactions on Evolutionary Computation*, *20*(5), 666-681. [CrossRef]

[44] Wolpert, D. H., & Macready, W. G. (2002). No free lunch theorems for optimization. *IEEE transactions on evolutionary computation*, *1*(1), 67-82. [CrossRef]

[45] Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks* (Vol. 4, pp. 1942-1948). ieee. [CrossRef]

[46] Shi, Y., & Eberhart, R. (1998, May). A modified particle swarm optimizer. In 1998 IEEE International Conference on Evolutionary Computation Proceedings. *IEEE World Congress on Computational Intelligence* (*Cat. No. 98TH8360*) (pp. 69-73). IEEE. [CrossRef]

[47] Chatterjee, A., & Siarry, P. (2006). Nonlinear inertia weight variation for dynamic adaptation in particle swarm optimization. *Computers & operations research*, *33*(3), 859-871. [CrossRef]

[48] Mirjalili, S., Mirjalili, S. M., & Hatamlou, A. (2016). Multi-verse optimizer: a nature-inspired algorithm for global optimization. *Neural computing and applications*, *27*(2), 495-513. [CrossRef]

[49] Rao, R. V., Savsani, V. J., & Vakharia, D. (2012). Teaching–learning-based optimization: an optimization method for continuous non-linear large scale problems. *Information sciences*, *183*(1), 1-15. [CrossRef]

[50] Zhou, X., Wu, Z., & Wang, H. (2012, December). Elite opposition-based differential evolution for solving large-scale optimization problems and its implementation on GPU. In *2012 13th international conference on parallel and distributed computing, applications and technologies* (pp. 727-732). IEEE. [CrossRef]

[51] Deb, K. (2000). An efficient constraint handling method for genetic algorithms. *Computer methods in applied mechanics and engineering*, *186*(2-4), 311-338. [CrossRef]

[52] Coello, C. A. C. (2000). Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, *41*(2), 113-127. [CrossRef]

[53] He, Q., & Wang, L. (2007). An effective co-evolutionary particle swarm optimization for constrained engineering design problems. *Engineering applications of artificial intelligence*, *20*(1), 89-99. [CrossRef]

[54] Becerra, R. L., & Coello, C. A. C. (2006). Cultured differential evolution for constrained optimization. *Computer Methods in Applied Mechanics and Engineering*, *195*(33-36), 4303-4322. [CrossRef]

[55] Maruta, I., Kim, T. H., & Sugie, T. (2009). Fixed-structure H_∞ controller synthesis: A meta-heuristic approach using simple constrained particle swarm optimization. *Automatica*, *45*(2), 553-559. [CrossRef]

[56] Gandomi, A. H., Yang, X. S., & Alavi, A. H. (2013). Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. *Engineering with computers*, *29*(1), 17-35. [CrossRef]

[57] Askarzadeh, A. (2016). A novel metaheuristic method for solving constrained engineering optimization problems: crow search algorithm. *Computers & structures*, *169*, 1-12. [CrossRef]

[58] Long, W., Jiao, J., Liang, X., & Tang, M. (2018). An exploration-enhanced grey wolf optimizer to solve high-dimensional numerical optimization. *Engineering Applications of Artificial Intelligence*, *68*, 63-80. [CrossRef]

[59] 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. [CrossRef]

[60] Coello, C. A. C., & Montes, E. M. (2002). Constraint-handling in genetic algorithms through the use of dominance-based tournament selection. *Advanced Engineering Informatics*, *16*(3), 193-203. [CrossRef]

[61] Ray, T., & Liew, K. M. (2003). Society and civilization: an optimization algorithm based on the simulation of social behavior. *IEEE Transactions on Evolutionary Computation*, *7*(4), 386-396. [CrossRef]

[62] Wang, Y., Cai, Z., & Zhou, Y. (2009). Accelerating adaptive trade-off model using shrinking space technique for constrained evolutionary optimization. *International journal for numerical methods in engineering*, *77*(11), 1501-1534. [CrossRef]

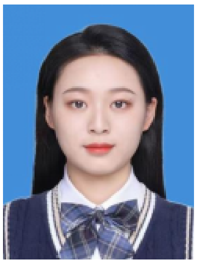
[63] Zhang, M., Luo, W., & Wang, X. (2008). Differential evolution with dynamic stochastic selection for constrained optimization. *Information Sciences*, *178*(15), 3043-3074. [CrossRef]

[64] Rashedi, E., Nezamabadi-Pour, H., & Saryazdi, S. (2009). GSA: a gravitational search algorithm. *Information sciences*, *179*(13), 2232-2248. [CrossRef]

[65] Wang, Y., Cai, Z., Zhou, Y., & Fan, Z. (2009). Constrained optimization based on hybrid evolutionary algorithm and adaptive constraint-handling technique. *Structural and Multidisciplinary Optimization*, *37*(4), 395-413. [CrossRef]

[66] Hedar, A. R., & Fukushima, M. (2006). Derivative-free filter simulated annealing method for constrained continuous global optimization. *Journal of Global optimization*, *35*(4), 521-549. [CrossRef]

Wen Long received the Ph.D. degree in control science and engineering from Central South University, Changsha 410083, China, in 2011. (Email: lw227@mail.gufe.edu.cn)



Tiantian Hu received the B.S. degree in E-commerce from Wuchang University of Science and Technology, Wuhan 430223, China, in 2023. (Email: 2421341034@qq.com)