Journal of Computational Design and Engineering 6 (2019) 243–259

Contents lists available at ScienceDirect

Journal of Computational Design and Engineering

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

IWOA: An improved whale optimization algorithm for optimization problems

Seyed Mostafa Bozorgi, Samaneh Yazdani *

Department of Computer Engineering, North Tehran Branch, Islamic Azad University, Tehran, Iran

ARTICLE INFO

Article history:
Received 1 August 2018
Received in revised form 5 February 2019
Accepted 11 February 2019
Available online 15 February 2019

Keywords:
Whale optimization algorithm
Swarm intelligence
Meta-heuristic algorithm
Optimization
Differential evolution

ABSTRACT

The whale optimization algorithm (WOA) is a new bio-inspired meta-heuristic algorithm which is presented based on the social hunting behavior of humpback whales. WOA suffers premature convergence that causes it to trap in local optima. In order to overcome this limitation of WOA, in this paper WOA is hybridized with differential evolution (DE) which has good exploration ability for function optimization problems. The proposed method is named Improved WOA (IWOA). The proposed method, combines exploitation of WOA with exploration of DE and therefore provides a promising candidate solution. In addition, IWOA$^+$ is presented in this paper which is an extended form of IWOA. IWOA$^+$ utilizes re-initialization and adaptive parameter which controls the whole search process to obtain better solutions. IWOA and IWOA$^+$ are validated on a set of 25 benchmark functions, and they are compared with PSO, DE, BBO, DE/BBO, PSO/GSA, SCA, MFO and WOA. Furthermore, the effects of dimensionality and population size on the performance of our proposed algorithms are studied. The results demonstrate that IWOA and IWOA$^+$ outperform the other algorithms in terms of quality of the final solution and convergence rate.

1. Introduction

Nature-inspired algorithms are very popular for solving different problems in various fields such as engineering (Hadavandi, Mostafayi, & Soltani, 2018; Lucas, Nasiri-Gheidari, & Tootoonchian, 2010), bioinformatics (Das, Abraham, & Konar, 2008), economy (Hafezi, Shahrabi, & Hadavandi, 2015) and medicine (Lin, Mimori, & Chen, 2012), for they don't require prior knowledge about the problems (Ghasemi, Ghavidel, Rahmani, Roosta, & Falah, 2014). Swarm intelligence (SI) is one of the main categories of nature-inspired algorithms which is developed by simulating the social behavior of some simple organisms. Some of the most popular swarm intelligence algorithms are Particle Swarm Optimization (PSO) (Clerc & Kennedy, 2002; Eberhart & Shi, 2004), Krill-Herd (KH) (Gandomi & Alavi, 2012), Moth-flame Optimization (Mirjalili, 2015), Cuckoo Search (CS) (Yang & Deb, 2009), Biogeography-Based Optimization (BBO) (Simon & Member, 2008) and Grey Wolf Optimization (GWO) (Mirjalili, Mirjalili, & Lewis, 2014). The Whale Optimization Algorithm (WOA) (Mirjalili & Lewis, 2016) is a new swarm intelligence algorithm developed by Mirjalili and Lewis and it is inspired from social behavior of humpback whales.

WOA searches the global optimum through encircling prey, searching for prey and attacking the prey. The performance of WOA was tested on 29 optimization benchmark functions and some 6 structural design problems. Results indicates the effectiveness of WOA in comparison with some state-of-the-art nature-inspired algorithms (Mirjalili & Lewis, 2016).

Several studies of WOA are presented which can be divided into two categories, i.e., (1) improving the WOA's performance and (2) Appling the WOA to solve some optimization problems. Some of WOA's improvements are (Kaur & Arora, 2017; Ling, Zhou, & Luo, 2017; Sun, Wang, Chen, & Liu, 2018). Kaur and Arora proposed chaotic WOA (CWOA) (Kaur & Arora, 2017) in which the chaos theory is used to tuning the main parameters of WOA to enhance the convergence speed of it. The experimental results on 20 optimization benchmark functions demonstrated that CWOA can improve the performance of WOA. In Ling et al. (2017) Lévy flight trajectory-based WOA (LWOA) is introduced. The LWOA employed Lévy flight trajectory to increase the diversity of population. The results indicated that LWOA outperformed the original WOA. Sun et al. (2018) proposed modified WOA (MWOA) for solving large-scale global optimization problems. In MWOA the Lévy flight strategy was employed to improve WOA's exploration ability. To

Peer review under responsibility of Society for Computational Design and Engineering.
* Corresponding author.
E-mail addresses: s.m.bozorgi@iau-tnb.ac.ir (S. Mostafa Bozorgi), s_yazdani@iau-tnb.ac.ir (S. Yazdani).

https://doi.org/10.1016/j.jcde.2019.02.002
2288-4300/

enhance exploitation ability of WOA, MWOA applied quadratic interpolation method. In addition, MWOA utilized nonlinear control strategy to control the whole search process and balance the exploration and the exploitation ability of WOA. Experimental results illustrated the superiority of MWOA to the other comparative algorithms in terms of accuracy and convergence rate. In addition, WOA has been applied to many real-world applications. In Bentouati, Chaib, and Chettih (2016), a new approach has been proposed by combining WOA and pattern search (PS) (Findler, Lo, & Lo, 1987) algorithms for power system design. Simulation results indicate the effectiveness of the proposed algorithm for solving different problems. In Mafarja and Mirjalili (2017), simulated annealing (SA) was embedded in WOA to present four feature selection methods. SA was applied as a local search to improve the exploitation ability of WOA. These four proposed methods were compared with three other wrapper methods and the obtained results indicated the effectiveness of the proposed methods. Aljarah, Faris, and Mirjalili (2018) applied WOA to Neural Networks training. The proposed method was evaluated and compared with seven other algorithms for 20 test functions. The results showed that the proposed method can find better solutions. In Aziz, Ewees, and Ella (2017), WOA and Moth-Flame Optimization (MFO) algorithms were applied for multi-level thresholding image segmentation. The proposed algorithms were compared with five other algorithms. The experimental results showed that MFO obtained better results than the other six comparative algorithms in term of quality of solutions. In ben oualid Medani, Sayah, and Bekrar (2018) WOA was utilized to solve the optimal reactive power dispatch problem. The obtained results illustrated that the proposed method obtained better results than two comparative algorithms. Yu, Wang, Li, Su, and Wu (2017) proposed levy flight WOA (LWOA) and used it to optimize the parameters of an active disturbance rejection control (ADRC) scheme for automatic carrier landing system (ACLS). Results demonstrated that the proposed method obtained better results than conventional ACLS. Wu et al. (2018) improved WOA and utilized it for proposing a novel path planning framework for a solar-powered UAV (SUAV) in urban environment. Simulation showed the effectiveness of the proposed method.

One of the main drawbacks of WOA is that it is not good at exploring the search space. To address this limitation, in this paper, WOA is hybridized with DE which has a good exploration ability and present an algorithm named Improved WOA (IWOA). Furthermore, we improve the performance of IWOA and present IWOA$^+$ in which a new control parameter is introduced. In addition, re-initialization is integrated into IWOA$^+$ to increase diversity of population. Experiments on 25 well-known optimization benchmark functions indicate the effectiveness of our proposed algorithms.

The reminder of this paper is organized as follows. The WOA is briefly introduced in Section 2. Section 3 describes the DE. Section 4 presents the proposed algorithms in detail. The proposed algorithms are evaluated by 25 optimization benchmark functions in Section 5. Section 6 presents main findings of this study.

2. Whale optimization algorithm

WOA is a new population-based algorithm which is presented in 2016 (Mirjalili & Lewis, 2016). This algorithm simulates the social behavior of humpback whales. Similar to other population-based algorithms, WOA uses a set of random candidate solutions (population) and uses three rules to update and improve the position of candidate solutions in each step which are encircling prey, spiral updating position and search for prey. They are described as follows:

- *Encircling prey: if ($p < 0.5$ and $|A| < 1$)*

The position of the candidate solution, $\vec{X}(t+1)$, is updated according to Eqs. (1) and (2):

$$\vec{D} = |\vec{C}.\vec{X^*} - \vec{X}(t)| \tag{1}$$

$$\vec{X}(t+1) = |\vec{X^*}(t) - \vec{A}.\vec{D}| \tag{2}$$

where $p \in [0.1]$ and $\overrightarrow{X^*}$ is the best candidate solution in current generation. $\vec{A}$ and $\vec{C}$ are calculated as Eqs. (3) and (4):

$$\vec{A} = |2\vec{a}.\vec{r} - \vec{a}| \tag{3}$$

$$\vec{C} = 2.\vec{r} \tag{4}$$

where $\vec{a}$ is linearly decreased from 2 to 0 and $\vec{r}$ is a randomly vector in [0,1].

- *Search for prey: if ($p < 0.5$ and $|A| < 1$)*

Searching for prey is very similar to encircling prey, but in contrast to it, in searching for prey instead of using $\overrightarrow{X^*}$, a randomly candidate solution, $\overrightarrow{X_{rand}}$, is selected. Eqs. (5) and (6) describe the process.

$$\vec{D} = |\vec{C}.\overrightarrow{X_{rand}} - \vec{X}(t)| \tag{5}$$

$$\vec{X}(t+1) = |\overrightarrow{X_{rand}}(t) - \vec{A}.\vec{D}| \tag{6}$$

Searching for prey is used during the exploration phase and enables WOA to perform global search.

- *Spiral updating position: if $p \geq 0.5$*

Encircling prey and spiral updating position methods are utilized during the exploitation phase of the WOA. Spiral updating position changes the position of individuals as Eq. (7):

$$\vec{X}(t+1) = \vec{D'}.e^{bl}.\cos(2\pi l) + \vec{X^*}(t) \tag{7}$$

where $\vec{D} = \left|\overrightarrow{X^*}(t) - \vec{X}(t)\right|$ which is the distance between ith candidate solution and the best solution in the current generation. b is a constant and $l \in [-1.1]$. Algorithm 1 shows the pseudo-code of the WOA algorithm.

Algorithm 1 (*WOA algorithm*).

```
1.  Generate the initial population X_i (i = 1, 2, ..., NP)
2.  Evaluate the fitness for each candidate solutions in X_i
3.  X* = the best candidate solutions
4.  while The halting criterion is not satisfied do
5.    for i = 1 to NP do
6.      Update a, A, C, l and P;
7.      for j = 1 to n do
8.        if P < 0.5 then
9.          if (|A| < 1)
10.           D = | C. X* − X_i |
11.           X_i(j) = X*(j) − A.D
12.         else if (|A| ≥ 1)
13.           Select a random individual X_rand
14.           D = | C. X_rand − X_i |
```

```
15.            X_i(j) = X_rand(j) − A.D
16.          end if
17.        else if P ≥ 0.5 then
18.          D′ = |X* − X_i |
19.          X_i(j) = D′. e^bl. cos(2πl) + X*
20.        end if
21.      end for
22.      Evaluate the fitness for X_i
23.    end for
24.  end while
```

3. Differential evolution

Differential evolution is a simple and an easy-to-use population-based search algorithm which is presented by Storn and Price in 1995 (Brest, Zumer, & Maucec, 2006). It generates new individuals by performing mutation and crossover operators. The parent is replaced by its offspring if it is fitter than its corresponding parent. Algorithm 2 shows the pseudo-code of the DE.

Algorithm 2 (*The DE algorithm*).

```
1.  Generate the initial population X
2.  Evaluate the fitness for each individual in X
3.  while The halting criterion is not satisfied do
4.    for i = 1 to NP do
5.      Select uniform randomly r1 ≠ r2 ≠ r3 ≠ i
6.      j_rand = rndint(1,n)
7.      for j = 1 to n do
8.        if rndreal_j [0, 1) > CR or j == j_rand then
9.          U_i(j) = X* (j) + F × (X_r2 (j) − X_r3 (j))
10.       else
11.         U_i(j) = X_i(j)
12.       end if
13.     end for
14.   end for
15.   Evaluate the offspring U_i
16.   if U_i is better than X_i then
17.     Update individual i, X_i = U_i
18.     if U_i is better than X* then
19.       Update best individual , X* = U_i
20.     end if
21.   end if
22. end while
```

In Algorithm 2 X is the population and n is dimension of the problem. *rndint* generates a random integer number in the range of [1,n]. CR is a crossover probability and $rndreal_j[0,1)$ generates a random real number between 0 and 1. $X_i(j)$ is the jth variable of ith individual in the population, and U_i is the offspring. Algorithm 2 shows DE/rand/1/bin in which a binomial crossover is used. For more details, see Engelbrecht (2007). The exploration ability of DE's mutation is very good. So, DE can find the region of global optimum appropriately, but DE is not good at exploitation (Gong, Cai, & Ling, 2011).

4. The proposed approach

In this section, two improvements for WOA are presented. In order to improve the exploration ability of WOA and address the WOA's premature convergence, in this study the mutation of DE is integrated into WOA. For better exploration and exploitation balancing, the IWOA is improved and IWQA$^+$ is introduced. We use a new parameter which is called *search mode* to change the exploration phase and exploitation of IWOA automatically. These two proposed algorithms are described as follows:

4.1. Improved WOA

The core of our proposed method, IWOA, is hybridized the WOA's operators with DE's mutation operator to improve WOA's exploration-exploitation tradeoff. The main operator of IWOA is a hybrid operator (see lines 10–28 of Algorithm 3) which combines DE's mutation and the components of WOA, namely, encircling prey, search for pray and spiral updating position. IWOA has two main parts: the exploration part (see lines 11–18 of Algorithm 3) and exploitation one (lines 19–27 of Algorithm 3). According to Algorithm 3, when $rand < \lambda$ the exploration part changes the individuals. λ is adjusted by Eq. (8).

$$\lambda = 1 - \frac{t}{t_{\max}} \tag{8}$$

where t is current generation and t_{max} shows the maximum number of generations. We apply λ to control the exploration and the exploitation ability of IWOA. According to Eq. (8) the value of λ is decreased over time from 1 to 0. So, the individuals are allowed to explore in the initial generation, while doing exploitation as time increases.

Exploration part contains DE's mutation and search for pray of the WOA. IWOA integrates DE's mutation because of its superiority in exploring the search space (Gong et al., 2011). The exploitation part of IWOA is similar to WOA. In contrast to WOA, IWOA is an elitism method. Namely, the new position for ith individual in the next generation is the fitter one between parent X_i and offspring U_i. It is important to note that, solutions should consider boundary constraints. If these constraints are violated, the repairing rule is applied according to Eq. (9).

$$X_i(j) = \begin{cases} \delta_j + rndreal(0,1) \times (\mu_j - \delta_j) & if \quad X_i(j) < \delta_j \\ \mu_j - rndreal(0,1) \times (\mu_j - \delta_j) & if \quad X_i(j) < \mu_j \end{cases} \tag{9}$$

μ_j and δ_j are upper bound and lower bound of jth dimension respectively. $X_i(j)$ is the jth dimension of ith solution. *rndreal*(0, 1) is a random number between 0 and 1.

Algorithm 3 (*IWOA algorithm*).

```
1.  Generate the initial population X_i (i = 1, 2, ..., NP)
2.  Evaluate the fitness for each individual in X_i
3.  X* = the best individual
4.  while The halting criterion is not satisfied do
5.    for each individual, map the fitness to the number of
    species
6.    for i = 1 to NP do
7.      Select uniform randomly r1 ≠ r2 ≠ r3 ≠ i
8.      Update a = 2-t.(2/t_Max), A = 2a.r.a , C = 2.r , l = rndreal
    (−1, 1) , p = rndreal(0, 1), λ = 1 − (t/t_Max);
9.      j_rand = rndint(1, n)
10.     for j = 1 to n do
11.       if p ≤ λ then
12.         if rndreal_j [0, 1) ≤ CR or j == j_rand then
13.           U_i(j) = X* (j) + F × (X_r2 (j) − X_r3 (j))
14.         else
15.           Select a random individual X_rand
16.           D = | C. X_rand −X_i |
17.           U_i(j) = X_rand(j) −A.D
```

(continued on next page)

```
            end if
          else if p > λ then
            if rndreal_j [0, 1) ≤ 0.5 then
              D = | C. X* − X_i |
              U_i(j) = X*(j) − A.D
            else
              D'= |X* − X_i |
              U_i(j) = D'. e^bl . cos(2πl) + X*
            end if
          end if
        end for
        Evaluate the offspring U_i
        if U_i is better than X_i then
          Update individual i , X_i = U_i
          if U_i is better than X* then
            Update best individual , X* = U_i
          end if
        end if
      end for
      t = t + 1;
    end while
```

According to Algorithm 3, IWOA has a very simple structure. It is important to mention that the only difference between IWOA and WOA is that the mutation of DE is integrated into WOA. In addition, the selection part of IWOA is similar to the selection method of DE. As a result, the computational complexity of IWOA is similar to WOA which is $O(MaxGen * NP * O(fitness))$.[1] *MaxGen* is the maximum number of generations, and $O(fitness)$ is determined by the application.

4.2. Our extended approach: IWOA+

The success of population-based algorithms is related to the exploration- exploitation trade-off. IWOA like WOA has two main parts: exploration and exploitation. When and how long these two parts should be performed have a great impact on the performance of IWOA. In this section, the IWOA is developed and IWOA+ is presented. The major difference between IWOA and IWOA+ is that IWOA+ utilizes a new parameter which is called the *search mode*. As it is mentioned above, in IWOA, the value of λ is decreased over time to a small value and lets the exploration part change the solutions in the initial search steps, as it activates the exploitation part as time increases. However, in IWOA+ the active part (exploration or exploitation) of the algorithm is changed adaptively through the *search mode* parameter by tracking the behavior of solutions in the population. To promote exploration in the first initial steps of the search, the *search mode* is set to 1. If *search mode* is set to 1 then *Ps* is set to 0.9 (Algorithm 4). In this situation, the exploration part is more active than exploitation one.

If the quality of the best solution is not changed over the number of iterations, then for changing the active part in IWOA+ the value of the *search mode* is updated to 2. In this situation, the exploitation part is more active than exploration one. According to Algorithm 4, the *search mode* parameter not only determines the active part adaptively but also justifies the duration that the exploration or exploitation parts are active and can change the positions of the solutions (see lines 39–56 of Algorithm 4). In this algorithm, *count-fail* is used to track the behavior of the best solution in the population over time. *Thf* is a threshold and is used to change the *search mode*. In order to control mode, a counter called *count fail* and a predetermined threshold are used. One challenge is to determine the initial value of this threshold. The threshold should not be very large or not very small. Initial value of the threshold is very important. The smaller value of *Thf* may cause the faster changes in search mode. So, the optimization process can't perform appropriately. The larger values of *Thf* may cause IWOA+ may become trap in local optimum. In this paper, the value of *Thf* is set to 50 which is founded empirically. The value of *Thf* is smaller than t_{max}.

If the population can't improve after the above mentioned phases, 80% of population are discarded and replaced with a randomly-generated population. 20% of the population is selected randomly from the previous ones which is included of the best individual of the population. As a result, the history of the population is saved. It is important to mention that, after re-initialization the values of all the *search mode*, *count-fail* and *Thf* are reinitialized.

Algorithm 4 (*IWOA+ algorithm*).

```
1.  Generate the initial population X_i (i = 1, 2, ..., NP)
2.  Evaluate the fitness for each individual in X_i
3.  X* = the best individual in iteration t = 1
4.  Ps = 0.9; search mode = 1; // explore mode
5.  count-fail = 0; Thf = round[t_Max/50]; // threshold fail
6.  while The halting criterion is not satisfied do
7.    for each individual, map the fitness to the number of species
8.    for i = 1 to NP do
9.      Select uniform randomly r_1 ≠ r_2 ≠ r_3 ≠ i;
10.     Update a = 2 − t.(2/t_Max), A = 2a.r.a , C = 2.r , l = rndreal (−1, 1); j_rand = rndint(1, n); k_rand= rndreal(0, 1);
11.     for j = 1 to n do
12.       if (search mode ==1 and k_rand <=ps) or (search mode ==2 and k_rand > ps) then // explore mode
13.         if rndreal_j [0, 1) ≤ CR or j == j_rand then
14.           U_i(j) = X* (j) + F × (X_r2 (j) − X_r3 (j))
15.         else
16.           Select a random individual X_rand
17.           V = | C. X_rand − X_i |
18.           U_i(j) = X_rand(j) − A.V
19.         end if
20.       else if (search mode ==2 and k_rand <=ps) or (search mode ==1 and k_rand > ps) then // exploit mode
21.         if rndreal_j [0, 1) ≤ 0.5 then
22.           V = | C. X* − X_i |
23.           U_i(j) = X*(j) − A.V
24.         else
25.           V * = |X* − X_i |
26.           U_i(j) = V *. e^bl . cos(2πl) + X*
27.         end if
28.       end if
29.     end for
30.     Evaluate the offspring U_i
31.     if U_i is better than X_i then
32.       Update individual i , X_i = U_i
33.       if U_i is better than X* then
34.         Update best individual , X* = U_i
35.       end if
36.     end if
37.   end for
38.   t = t + 1;
39.   if best individual t ≤ best individual t-1 then
40.     count-fail = 0;
41.   else
```

[1] We assume the maximum number of generations is considered as termination condition.

```
42.        count-fail = count fail + 1;
43.        if count fail > thf then
44.          if search mode = 1 then
45.            search mode = 2; count-fail =0; thf = thf × 2;
46.          else if search mode = 2 then
47.            search mode = 1; count-fail = 0; Thf = round
    [tMax/50];
48.            Re-initialization and Generate new population Xi (i
    = 1, 2... NP).
49.            // keep 20% of the current population include X*, X"
    and re-initialization 80% of the population.
50.            Evaluate the fitness for each individual in Xi
51.            if Xi is better than X* then
52.              Update best individual , X* = Xi
53.            end if
54.          end
55.        end if
56.      end if
57.    end while
```

5. Simulation results and discussion

To verify the performance of IWOA and IWOA$^+$ different experiments have been conducted over 25 benchmark functions which have been widely used in other studies (Cheng & Lien, 2012; Jamil & Blekinge, 2013; Yao, Liu, & Lin, 1999). These functions are classified into unimodal functions with a few numbers of local minima and multimodal functions with a lot of local minima (Table 1). Functions f01-f15 are high-dimensional and scalable problems. Functions f01-f05 are unimodal and functions f06-f13 are multimodal functions. Functions f14-f18 and f19-f25 are low dimensional functions for unimodal and multimodal functions respectively.

5.1. Experimental setup and performance criteria

Our proposed algorithms, IWOA and IWOA$^+$, have been compared with eight other nature-inspired algorithms: WOA (Mirjalili & Lewis, 2016), PSO (Eberhart & Shi, 2004), BBO (Simon & Member, 2008), SCA (Mirjalili, 2016), MFO (Mirjalili, 2015), PSO/GSA (Mirjalili & Hashim, 2010), DE (Brest et al., 2006) and DE/BBO (Gong et al., 2011). In order to compare these algorithms, some parameters have to be determined. Table 2 shows these parameters which are taken from (Gong et al., 2011; Mirjalili & Lewis, 2016). For all experiments, the parameters are set according to Table 2, unless we mention a change. To have a fair comparison, the experiment is repeated 50 times for each benchmark functions and five performance criteria are used. The definitions of them are as follows (Gong et al., 2011): (1) Average error: Error of solution X is defined as absolute of $(f(\bar{X}) - f(X^*))$ where $\bar{X}$ is the best solution in the population and X^* is the optimum. (2) Number of function calls (*NFCs*): Number of fitness function calls are recorded till *VTR* is obtained. Mean and standard deviation of *NFC* are also calculated. (3) Number of successful runs (*SR*): The number of runs for which the algorithm successfully reaches the *VTR* before maximum number of function calls (*Max-NFCs*) terminates the trial. (4) Convergence graphs: Show mean error of the algorithm in different simulations. (5) Acceleration rate (*AR*): This measure is defined as $AR = NFC_{Other}/NFC_{IWOA}$, and it is used

Table 2
The parameter settings of 10 comparative algorithms.

Parameters	The amount
Population size (NP) (Mirjalili & Lewis, 2016)	100
Maximum number of Iteration (Mirjalili & Lewis, 2016)	500
Maximum number of function calls (*Max-NFCs*)	50,000
Value to reach (VTR) (Gong et al., 2011)	10^{-8}
Crossover probability for DE (Brest et al., 2006), DE/BBO (Gong et al., 2011), IWOA	0.9
Probability of mutation for BBO (Simon & Member, 2008)	0.05
Scaling factor for DE (Brest et al., 2006), DE/BBO (Gong et al., 2011), IWOA	Random between (0.2, 0.8)
DE mutation scheme for DE (Brest et al., 2006), DE/BBO (Gong et al., 2011), IWOA	DE/best/1/bin
Habitat modification probability for DE/BBO (Gong et al., 2011)	1
a in WOA (Mirjalili & Lewis, 2016), SCA (Mirjalili, 2016), IWOA	is decreased from 2 to 0
c_1 and c_2 for PSO (Eberhart & Shi, 2004)	2

Table 1
The brief description of 25 benchmark functions.

Name/Function			Dim.	Range	f min
Unimodal functions	F1	Sphere function	30	[−100, 100]	0
	F2	Sum of different powers function	30	[−1, 1]	0
	F3	Rotated hyper-ellipsoid function	30	[−100, 100]	0
	F4	Schwefel 2.21 function	30	[−100, 100]	0
	F5	Rosenbrock function	30	[−30, 30]	0
Multimodal functions	F6	Styblinski-tank function	30	[−5, 5]	−39.16599
	F7	Ackley function	30	[−32, 32]	0
	F8	Ackley N.4 function	30	[−35, 35]	−4.590101633799122
	F9	Griewank function	30	[−600, 600]	0
	F10	Generalized penalized function 1	30	[−50, 50]	0
	F11	Generalized penalized function 2	30	[−50, 50]	0
	F12	Shubert 3 function	30	[−10, 10]	−29.6733337
	F13	Shubert 4 function	30	[−10, 10]	−25.740858
Fixed-dimension unimodal functions	F14	Ackley N.2 function	2	[−32, 32]	−200
	F15	Drop-wave function	2	[−5.2, 5.2]	−1
	F16	Schaffer N.1 function	2	[−100, 100]	0
	F17	Schaffer N.2 function	2	[−100, 100]	0.00156685
	F18	Three-hump camel function	2	[−5, 5]	0
Fixed-dimension multimodal functions	F19	Michalewicz function	2	[0, π]	−1.8013
	F20	Kowalik's function	4	[−5, 5]	0.003075
	F21	Six-hump camel function	2	[−5, 5]	−1.0316285
	F22	Holder table function	2	[−10, 10]	−19.2085
	F23	Goldstein-price function	2	[−2, 2]	3
	F24	Hartmann 3-dimensional function	3	[1, 3]	−3.86
	F25	Levy N.13 function	2	[−10, 10]	0

to compare convergence speed of IWOA and other algorithms. If $AR > 1$, IWOA is faster than the compared algorithm.

5.2. General performance

Table 3 shows the mean and standard deviation of error of the ten compared algorithms on 25 benchmark functions. From Table 3:

- IWOA obtains the second rank for F1, F3, F5, F6 and F7 among other algorithms. The performance of IWOA for f12, F13, F15, and F20 is not good. In the remaining 14 test functions, IWOA reaches the best mean error. It is worth noting that for some of the test functions the obtained mean errors are very similar to IWOA$^+$.
- IWOA$^+$ obtains the best performance on all functions except for F1, F5, F8, and F24. For them, IWOA$^+$ obtains second rank among other comparative algorithms.
- We can see that IWOA is better than WOA on 20 test functions. In addition, IWOA$^+$ has performance improvement over WOA and IWOA on 23 and 18 test functions respectively. The better performance of IWOA and IWOA$^+$ are related to their capability to balance the exploration and exploitation.

Table 3 shows the ability of IWOA$^+$ on multimodal functions. The outstanding performance of IWOA$^+$ is due to its ability to change the active part of the algorithm. Re-initialization and changing the active operators (exploration part or exploitation part) help the IWOA$^+$ to obtain a better performance than WOA

Table 3
Comparison of errors of 10 comparative algorithms on 25 test functions over 50 independent runs.

Fun.	PSO		BBO		SCA		MFO		PSO/GSA	
	mean	Std.	mean	Std.	mean	Std.	mean	Std.	mean	Std.
F1	3.45e−5	3.60e−5	367.25	107.98	0.8935	1.9443	600.98	2.39e+3	200.64	1.41e+3
F2	7.60e−9	1.84e−9	1.58e−5	1.85e−5	3.53e−7	1.47e−6	8.28e−9	1.52e−9	9.82e−9	1.6e−10
F3	2.38e+3	902.67	1.18e+4	3.08e+3	3.67e+3	2.77e+3	1.56e+4	1.17e+4	8.60e+3	5.56e+3
F4	9.1375	2.4565	22.076	2.0802	19.462	9.9705	43.274	10.057	31.087	7.2463
F5	119.58	176.36	2.92e+4	1.43e+4	4.74e+3	1.74e+4	2.23e+4	3.84e+4	52.359	42.805
F6	1.67e−4	0.0011	0.0021	0.0020	0.0119	0.0107	4.69e−9	2.91e−9	5.02e−9	2.94e−9
F7	0.0346	0.1665	5.4986	0.5121	10.949	10.027	8.5060	9.0879	7.8654	5.5808
F8	4.64e−5	1.50e−4	0.0341	0.0300	19.022	9.0278	8.7767	20.464	116.28	46.840
F9	0.0122	0.0116	4.3230	0.9763	0.5468	0.3182	4.3414	17.816	4.3054	17.788
F10	0.1963	0.2997	6.0087	1.7194	21.620	112.14	2.4586	1.6277	5.1198	3.8319
F11	0.0116	0.0171	421.31	737.18	7.15e+03	4.85e+4	2.7496	2.1155	19.649	9.5225
F12	1.72e−5	6.46e−5	3.73e−4	3.43e−4	0.0022	0.0026	4.92e−9	2.69e−9	5.22e−9	2.75e−9
F13	2.05e−4	4.96e−4	4.02e−4	4.26e−4	5.89e−4	4.98e−4	5.05e−9	3.10e−9	4.95e−9	2.84e−9
F14	7.11e−9	2.26e−9	0.0795	0.0522	6.62e−9	2.23e−9	6.41e−9	2.61e−9	6.54e−9	2.63e−9
F15	5.32e−9	2.97e−9	0.0449	0.0291	4.29e−9	3.12e−9	0.0013	0.0090	0.0013	0.0090
F16	4.89e−9	2.82e−9	0.0026	0.0026	4.53e−9	2.69e−9	5.04e−9	2.72e−9	4.93e−9	3.00e−9
F17	5.28e−9	2.86e−9	0.0027	0.0029	4.53e−7	1.06e−6	5.00e−9	3.20e−9	5.09e−9	2.92e−9
F18	5.47e−9	2.44e−9	3.43e−5	4.51e−5	4.29e−9	2.73e−9	4.82e−9	3.18e−9	4.61e−9	3.15e−9
F19	4.77e−9	3.11e−9	5.56e−5	7.20e−5	0.0016	0.0013	4.55e−9	2.92e−9	5.52e−9	2.97e−9
F20	4.61e−4	2.45e−4	0.0022	0.0046	6.15e−4	3.80e−4	5.32e−4	2.24e−4	0.0058	0.0090
F21	5.07e−9	2.84e−9	1.50e−4	2.70e−4	1.84e−5	1.57e−5	4.21e−9	2.99e−9	5.39e−9	2.93e−9
F22	4.99e−9	2.96e−9	1.43e−4	2.06e−4	0.0076	0.0071	5.09e−9	2.71e−9	5.35e−9	3.07e−9
F23	4.75e−9	2.90e−9	0.0024	0.0078	6.05e−6	7.26e−6	4.71e−9	2.88e−9	4.68e−9	2.82e−9
F24	5.90e−9	2.97e−9	1.73e−5	3.23e−5	0.0070	0.0030	6.23e−9	2.32e−9	5.67e−9	2.38e−9
F25	5.14e−9	2.69e−9	0.0037	0.0062	4.21e−4	4.41e−4	5.12e−9	2.88e−9	4.88e−9	2.80e−9
Fun.	DE		DE/BBO		WOA		IWOA		IWOA$^+$	
	mean	Std.	mean	Std.	mean	Std.	mean	Std.	mean	Std.
F1	8.54e−9	1.09e−9	8.90e−9	8.4e−10	7.28e−9	1.92e−9	8.34e−9	1.15e−9	8.27e−9	1.32e−9
F2	6.93e−9	2.04e−9	7.37e−9	2.26e−9	6.60e−9	2.55e−9	6.25e−9	2.21e−9	6.25e−9	2.14e−9
F3	1.07e−6	4.88e−6	4.19e+3	1.47e+3	2.87e+3	3.00e+3	0.0445	0.0597	5.02e−7	8.53e−7
F4	0.3339	0.3452	0.6614	0.2328	40.197	0.0469	0.0028	0.0037	1.99e−5	4.12e−5
F5	1.0243	1.6075	45.247	28.841	26.049	0.3093	9.8439	10.359	8.0839	2.2693
F6	4.10e−4	6.13e−4	0.0053	0.0049	1.50e−6	2.48e−6	4.98e−9	3.23e−9	4.33e−9	2.86e−9
F7	4.0082	1.3400	9.67e−9	1.34e−9	8.29e−9	1.46e−9	8.96e−9	7.2e−10	8.91e−9	7.9e−10
F8	13.574	17.626	5.09e−4	4.56e−4	0.2216	1.5563	4.11e−9	3.01e−9	5.05e−9	2.95e−9
F9	0.0265	0.0328	0.0035	0.0048	0.0024	0.0083	0.0010	0.0035	0.0019	0.0061
F10	1.7117	2.4478	0.0145	0.0757	3.79e−4	7.64e−4	9.34e−9	8.9e−10	9.39e−9	5.8e−10
F11	1.3308	1.6914	2.19e−4	0.0016	0.0139	0.0246	2.59e−8	3.11e−8	1.30e−8	6.93e−9
F12	0.0013	0.0011	5.88e−4	6.41e−4	2.21e−7	3.87e−7	1.92e−5	1.34e−4	4.78e−9	3.13e−9
F13	7.41e−4	7.04e−4	5.36e−4	5.11e−4	4.40e−7	7.93e−7	1.93e−4	5.09e−4	4.98e−9	3.06e−9
F14	5.08e−9	2.23e−9	6.40e−9	2.03e−9	7.43e−9	2.10e−9	4.57e−9	2.30e−9	4.85e−9	2.56e−9
F15	0.0153	0.0275	0.0026	0.0126	0.0115	0.0247	0.0051	0.0175	3.75e−9	2.82e−9
F16	3.88e−9	2.62e−9	4.31e−9	2.62e−9	5.40e−9	3.10e−9	3.48e−9	3.01e−9	3.18e−9	2.83e−9
F17	5.73e−9	2.56e−9	5.30e−9	2.76e−9	5.19e−9	2.88e−9	4.87e−9	2.68e−9	4.19e−9	2.64e−9
F18	3.60e−9	2.88e−9	4.93e−9	3.01e−9	5.79e−9	2.80e−9	3.32e−9	2.58e−9	3.38e−9	2.54e−9
F19	4.46e−9	2.91e−9	5.26e−9	2.89e−9	5.19e−9	2.86e−9	4.06e−9	3.18e−9	4.10e−9	2.89e−9
F20	0.0053	0.0088	1.28e−4	3.20e−4	3.24e−4	3.63e−4	0.0020	0.0098	1.19e−4	3.04e−4
F21	4.03e−9	2.90e−9	4.25e−9	2.95e−9	5.32e−9	2.88e−9	3.94e−9	3.01e−9	3.16e−9	2.63e−9
F22	4.33e−9	3.03e−9	4.85e−9	2.91e−9	5.23e−9	3.07e−9	4.16e−9	3.16e−9	4.08e−9	2.81e−9
F23	4.05e−9	2.81e−9	4.83e−9	2.64e−9	9.52e−7	3.05e−6	3.90e−9	2.69e−9	3.64e−9	2.88e−9
F24	4.70e−9	2.75e−9	5.37e−9	2.64e−9	2.79e−4	3.56e−4	4.65e−9	2.42e−9	5.24e−9	2.64e−9
F25	4.02e−9	3.46e−9	4.08e−9	2.91e−9	6.42e−8	1.65e−7	3.50e−9	2.56e−9	3.88e−9	2.89e−9

and IWOA especially on multimodal functions. From Fig. 1 we can see that IWOA$^+$ can improve its solution quality for a long run. In general, IWOA$^+$ converges faster than IWOA, and IWOA shows a better performance than WOA.

Table 4, shows the average and standard deviation of *NFCs* and *SRs*. From Table 4, it can be found that our proposed algorithms need less NFCs than other comparative algorithms. By carefully looking at Table 4, we can find that overall *SR* of our proposed algorithms are better than other eight comparative algorithms. For example, IWOA$^+$ reaches *VTR* before the halting criterion is satisfied for 18 test functions. However, it fails to solve F4 and F5.

Generally, by comparing *AR* in Table 5 we can conclude that IWOA and IWOA$^+$ are faster than other comparative algorithms, except for fixed-dimension test function where the DE is the fastest.

In order to test statistical significance, a Wilcoxon signed-rank test (Derrac, García, Molina, & Herrera, 2011) is carried out to comparatively evaluate the performance of the mean results of the algorithms shown in Table 4. To meet this purpose, the following hypotheses are proposed:

- H0: There is no difference among the performance of the IWOA$^+$ and other algorithms.
- H1: A difference exists among the performance of the IWOA$^+$ and other algorithms.

The results of Wilcoxon signed-rank test are shown in Table 6.

5.3. Influence of population size

The size of population has a direct influence on the performance of population-based algorithms specially DE. Since increasing population size increases the diversity and for DE it means that more direction can be explored (Gong et al., 2011). In this sub-section, the influence of population size on the performance of the five comparative algorithms, namely WOA, DE, DE/BBO, IWOA and

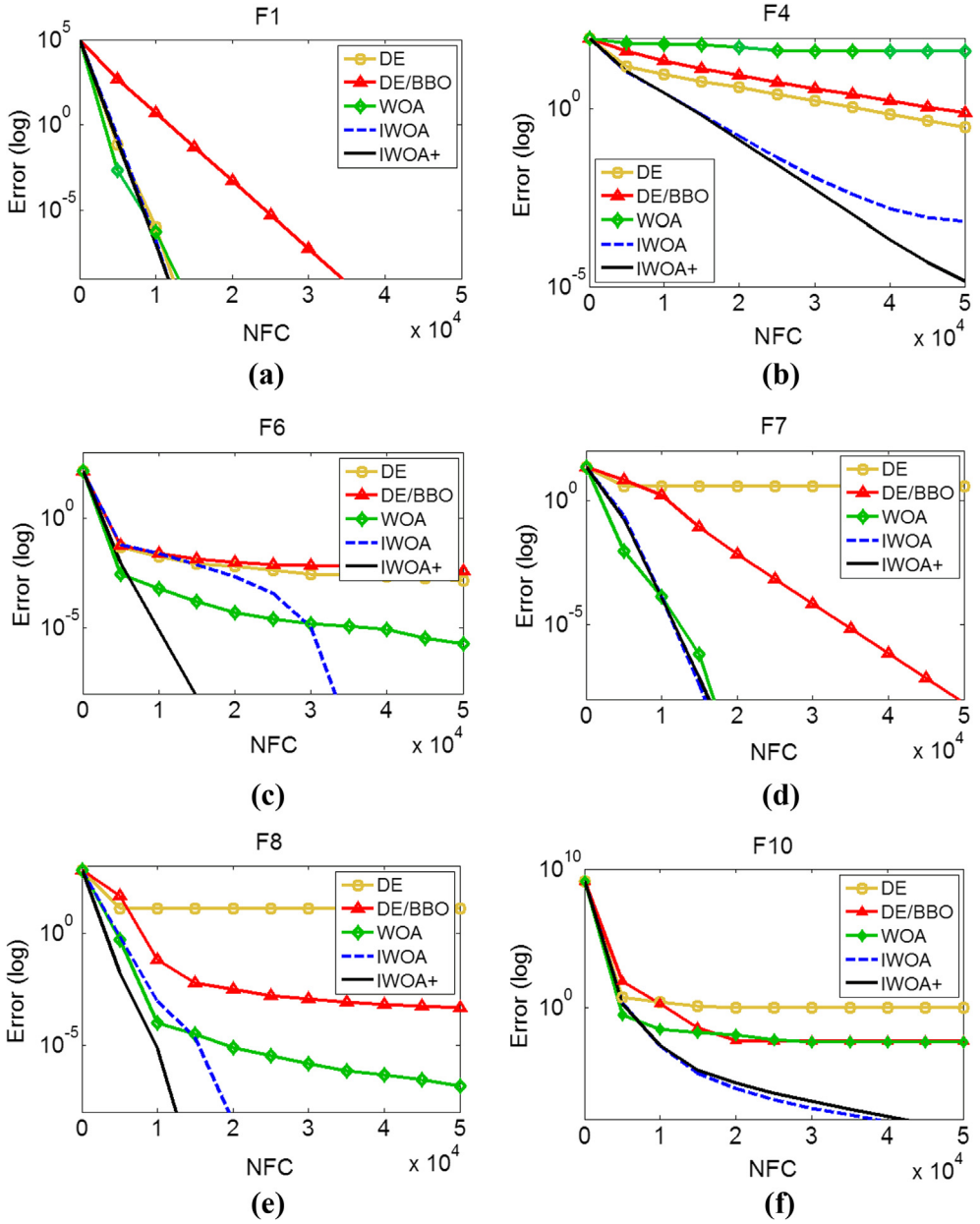

Fig. 1. Convergence curves of DE, DE/BBO, WOA, IWOA and IWOA$^+$ for some selected benchmark functions. (a) F01. (b) F04. (c) F06. (d) F07. (e) F08. (f) F10.

Table 4
Comparison of the average and the standard deviation of *NFC* values of the 10 comparative algorithms when *VTR* is 10^{-8}. *SR* shows the number of successful runs.

Fun.	PSO			BBO			SCA			MFO			PSO/GSA		
	mean	Std.	*SR*	mean	Std.	*SR*	mean	Std.	*SR*	mean	Std.	*SR*	mean	Std.	*SR*
F1	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0	4.10e+4	1.18e+4	19
F2	2.94e+4	1.76e+3	50	NA	NA	0	3.92e+4	8.3e+3	35	3.56e+4	3.85e+3	50	2.05e+4	4.14e+3	50
F3	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0
F4	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0
F5	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0
F6	3.85e+4	7.14e+3	37	NA	NA	0	NA	NA	0	1.20e+4	2.64e+3	50	3.53e+4	2.41e+3	50
F7	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0
F8	4.65e+4	4.59e+3	21	NA	NA	0	NA	NA	0	4.16e+4	6.87e+3	34	NA	NA	0
F9	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0
F10	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0
F11	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0
F12	4.09e+4	7.53e+3	31	NA	NA	0	NA	NA	0	1.23e+4	2.63e+3	50	3.24e+4	3.86e+3	50
F13	4.02e+4	6.91e+3	35	NA	NA	0	NA	NA	0	1.26e+4	2.46e+3	50	3.27e+4	2.68e+3	50
F14	2.55e+4	740.87	50	NA	NA	0	9.41e+3	1.27e+3	50	8.99e+3	536.64	50	3.44e+4	1.41e+3	50
F15	1.80e+4	1.92e+3	50	NA	NA	0	5.34e+3	1.50e+3	50	7.15e+3	6.32e+3	48	1.53e+4	1.38e+3	48
F16	1.44e+4	1.31e+3	50	NA	NA	0	3.55e+3	917.84	50	4.76e+3	674.67	50	4.39e+3	945.12	50
F17	2.74e+4	3.79e+3	50	NA	NA	0	4.97e+4	1.92e+3	1	8.60e+3	1.84e+3	50	1.58e+4	2.70e+3	50
F18	1.47e+4	1.07e+3	50	NA	NA	0	3.25e+3	810.15	50	3.79e+3	520.47	50	1.17e+4	1.59e+3	50
F19	1.62e+4	1.98e+3	50	NA	NA	0	NA	NA	0	3.88e+3	664.24	50	1.45e+4	1.90e+3	50
F20	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0	4.08e+4	1.14e+4	23
F21	1.57e+4	1.53e+3	50	NA	NA	0	NA	NA	0	4.65e+3	617.19	50	1.34e+4	1.16e+3	50
F22	2.90e+4	5.57e+3	50	NA	NA	0	NA	NA	0	5.11e+3	873.25	50	1.35e+4	1.49e+3	50
F23	1.65e+4	1.42e+3	50	NA	NA	0	NA	NA	0	4.73e+3	465.37	50	17,514	1.69e+3	50
F24	1.8e+4	935.1	50	4.9e+4	2.6e+3	2	NA	NA	0	5.68e+3	541.5	50	1.7e+4	1.05e+3	50
F25	1.68e+4	1.20e+3	50	NA	NA	0	NA	NA	0	4.61e+3	539.42	50	1.41e+4	1.38e+3	50
Fun.	DE			DE/BBO			WOA			IWOA			IWOA+		
	mean	Std.	*SR*	mean	Std.	*SR*	mean	Std.	*SR*	mean	Std.	*SR*	mean	Std.	*SR*
F1	1.0e+4	574.8	50	3.1e+4	592.6	50	1.0e+4	953.1	50	1.0e+4	331.2	50	1.0e+4	402.9	50
F2	2.5e+3	278.3	50	6.9e+3	743.1	50	2.5e+3	476.3	50	3.0e+3	336.7	50	2.7e+3	327.1	50
F3	4.9e+4	471.4	4	NA	NA	0	NA	NA	50	NA	NA	0	4.9e+4	281.8	1
F4	NA	NA	0	NA	NA	0	NA	NA	50	NA	NA	0	NA	NA	0
F5	NA	NA	0	NA	NA	0	NA	NA	50	NA	NA	0	NA	NA	0
F6	NA	NA	0	NA	NA	0	4.9e+4	2.8e+3	1	2.7e+4	5.5e+3	50	1.0e+4	2.4e+3	50
1F7	NA	NA	0	4.90e+4	725.39	42	15,994	497.0	50	1.54e+4	422.75	50	1.62e+4	368.16	50
F8	NA	NA	0	NA	NA	0	4.7e+4	5.6e+3	18	1.6e+4	1.6e+3	50	8.7e+3	1.0e+3	50
F9	4.1e+4	1.6e+4	11	3.9e+4	8.2e+3	31	14,134	1.07e+4	46	1.5e+4	1.1e+4	45	1.7e+4	1.3e+4	43
F10	4.58e+4	1.13e+4	9	3.0e+4	5.0e+3	47	NA	NA	0	3.66e+4	3.26e+3	50	4.07e+4	2.0e+3	50
F11	4.48e+4	1.21e+4	8	3.14e+4	2.80e+3	48	NA	NA	0	4.81e+4	2.84e+3	19	4.95e+4	939.55	24
F12	NA	NA	0	NA	NA	0	4.95e+4	1.45e+3	5	3.09e+4	9.51e+3	48	1.03e+4	5.15e+3	50
F13	NA	NA	0	NA	NA	0	4.96e+4	1.44e+3	3	3.67e+4	1.08e+4	37	1.29e+4	6.66e+3	50
F14	2.61e+3	107.38	50	7.18e+3	329.62	50	9.71e+3	1.47e+3	50	2.63e+3	96.822	50	2.67e+3	109.96	50
F15	1.36e+4	2.06e+4	38	8.69e+3	8.57e+3	48	1.19e+4	1.81e+4	41	6.09e+3	1.30e+4	46	3.47e+3	4.78e+3	50
F16	2.14e+3	183.40	50	5.72e+3	586.55	50	2.28e+3	1.05e+3	50	2.14e+3	171.16	50	2.11e+3	264.46	50
F17	4.29e+3	698.15	50	1.91e+4	5.67e+3	50	1.27e+4	5.63e+3	50	4.41e+3	871.29	50	4.18e+3	672.86	50
F18	1.12e+3	83.697	50	3.27e+3	283.84	50	2.50e+3	999.01	50	1.13e+3	90.671	50	1.16e+3	89.809	50
F19	1.3e+3	141.2	50	3.21e+3	505.04	50	1.19e+4	4.41e+3	50	1.35e+3	159.97	50	1.44e+3	204.53	50
F20	2.11e+4	2.28e+4	31	3.02e+4	8.72e+3	43	NA	NA	0	1.32e+4	1.86e+4	42	1.10e+4	1.60e+4	43
F21	1.20e+3	118.12	50	3.32e+3	334.12	50	8.15e+3	4.25e+3	50	1.29e+3	97.874	50	1.30e+3	117.73	50
F22	1336	147.71	50	3.55e+3	563.14	50	1.51e+4	6.00e+3	50	1.44e+3	229.08	50	1.50e+3	198.79	50
F23	1.36e+3	91.645	50	4.31e+3	375.27	50	4.40e+4	1.48e+4	7	1.39e+3	93.826	50	1.45e+3	116.04	50
F24	1301	76.073	50	3.63e+3	276.66	50	NA	NA	0	1.45e+3	96.348	50	1.51e+3	107.42	50
F25	1.38e+3	99.359	50	3.51e+3	313.42	50	3.13e+4	1.77e+4	27	1.42e+3	99.559	50	1.50e+3	121.06	50

Table 5
Comparing of *AR* of the proposed algorithms with other eight algorithms for benchmark functions. (∞ means the proposed algorithm is faster than other algorithm).

Function	IWOA v							
	PSO	BBO	SCA	MFO	PSOGSA	DE	WOA	DE/BBO
F1	∞	∞	∞	∞	4.1000	1.0000	1.0000	3.1000
F2	9.8000	∞	13.0667	11.8667	6.8333	0.8333	0.8333	2.3000
F3	NA	NA	NA	NA	NA	0	NA	NA
F4	NA	NA	NA	NA	NA	NA	NA	NA
F5	NA	NA	NA	NA	NA	NA	NA	NA
F6	1.4259	∞	∞	0.4444	1.3074	∞	1.8148	∞
F7	∞	∞	∞	∞	∞	∞	1.0386	3.1818
F8	2.9063	∞	∞	2.6000	∞	∞	2.9375	∞
F9	∞	∞	∞	∞	∞	2.7333	0.9423	2.6000
F10	∞	∞	∞	∞	∞	1.2514	∞	0.8197
F11	∞	∞	∞	∞	∞	0.9314	∞	0.6528
F12	1.3236	∞	∞	0.3981	1.0485	∞	1.6019	∞
F13	1.0954	∞	∞	0.3433	0.8910	∞	1.3515	∞
F14	9.6958	∞	3.5779	3.4183	13.0798	0.9924	3.6920	2.7300
F15	2.9557	∞	0.8768	1.1741	2.5123	2.2332	1.9540	1.4269
F16	6.7290	∞	1.6589	2.2243	2.0514	1.0000	1.0654	2.6729
F17	6.2132	∞	11.2698	1.9501	3.5828	0.9728	2.8798	4.3311
F18	13.0088	∞	2.8761	3.3540	10.3540	0.9912	2.2124	2.8938
F19	12.0000	∞	∞	2.8741	10.7407	0.9630	8.8148	2.3778
F21	∞	∞	∞	∞	3.0909	1.5985	∞	2.2879
F22	12.1705	∞	∞	3.6047	10.3876	0.9302	6.3178	2.5736
F23	20.1389	∞	∞	3.5486	9.3750	0.9278	10.4861	2.4653
F24	11.8705	∞	∞	3.4029	12.6000	0.9784	31.6547	3.1007
F25	2.4138	33.7931	∞	3.9172	11.7241	0.8972	∞	2.5034
Function	IWOA$^+$ v							
	PSO	BBO	SCA	MFO	PSOGSA	DE	WOA	DE/BBO
F1	∞	∞	∞	∞	4.1000	1.0000	1.0000	3.1000
F2	10.8889	∞	14.5185	13.1852	7.5926	0.9259	0.9259	2.5556
F3	∞	∞	∞	∞	∞	1.0000	∞	∞
F4	NA	NA	NA	NA	NA	NA	NA	NA
F5	NA	NA	NA	NA	NA	NA	NA	NA
F6	3.8500	∞	∞	1.2000	3.5300	∞	4.9000	∞
F7	∞	∞	∞	∞	∞	∞	0.9873	3.0247
F8	5.3448	∞	∞	4.7816	∞	∞	5.4023	∞
F9	∞	∞	∞	∞	∞	2.4118	0.8314	2.2941
F10	∞	∞	∞	∞	∞	1.1253	∞	0.7371
F11	∞	∞	∞	∞	∞	0.9051	∞	0.6343
F12	3.9709	∞	∞	1.1942	3.1456	∞	4.8058	∞
F13	3.1163	∞	∞	0.9767	2.5349	∞	3.8450	∞
F14	9.5506	∞	3.5243	3.3670	12.8839	0.9775	3.6367	2.6891
F15	5.1873	∞	1.5389	2.0605	4.4092	3.9193	3.4294	2.5043
F16	6.8246	∞	1.6825	2.2559	2.0806	1.0142	1.0806	2.7109
F17	6.5550	∞	11.8900	2.0574	3.7799	1.0263	3.0383	4.5694
F18	12.6724	∞	2.8017	3.2672	10.0862	0.9655	2.1552	2.8190
F19	11.2500	∞	∞	2.6944	10.0694	0.9028	8.2639	2.2292
F21	∞	∞	∞	∞	3.7091	1.9182	∞	2.7455
F22	12.0769	∞	∞	3.5769	10.3077	0.9231	6.2692	2.5538
F23	19.3333	∞	∞	3.4067	9.0000	0.8907	10.0667	2.3667
F24	11.3793	∞	∞	3.2621	12.0786	0.9379	30.3448	2.9724
F25	11.9205	32.4503	∞	3.7616	11.2583	0.8616	∞	2.4040

Table 6
The results of Wilcoxon signed-rank test for all compared algorithms.

Model	Average of best results	Std-dev	Base model for comparison	Z value	Pr(>|Z|)
PSO	100.358925	475.525370	IWOA$^+$	−4.372373	0.000012
BBO	1673.065668	6198.838026		−4.372373	0.000012
SCA	625.301010	1794.497320		−4.372373	0.000012
MFO	1542.843524	5328.790036		−4.372373	0.000012
PSO/GSA	361.492508	1716.962642		−4.345466	0.000014
DE	0.881298	2.789598		−3.511352	0.000446
WOA	117.459855	573.522521		−4.237839	0.000023
DE/BBO	169.437451	837.665941		−4.372373	0.000012
IWOA	0.395980	1.968337		−2.257143	0.023999
IWOA$^+$	0.323438	1.616763			

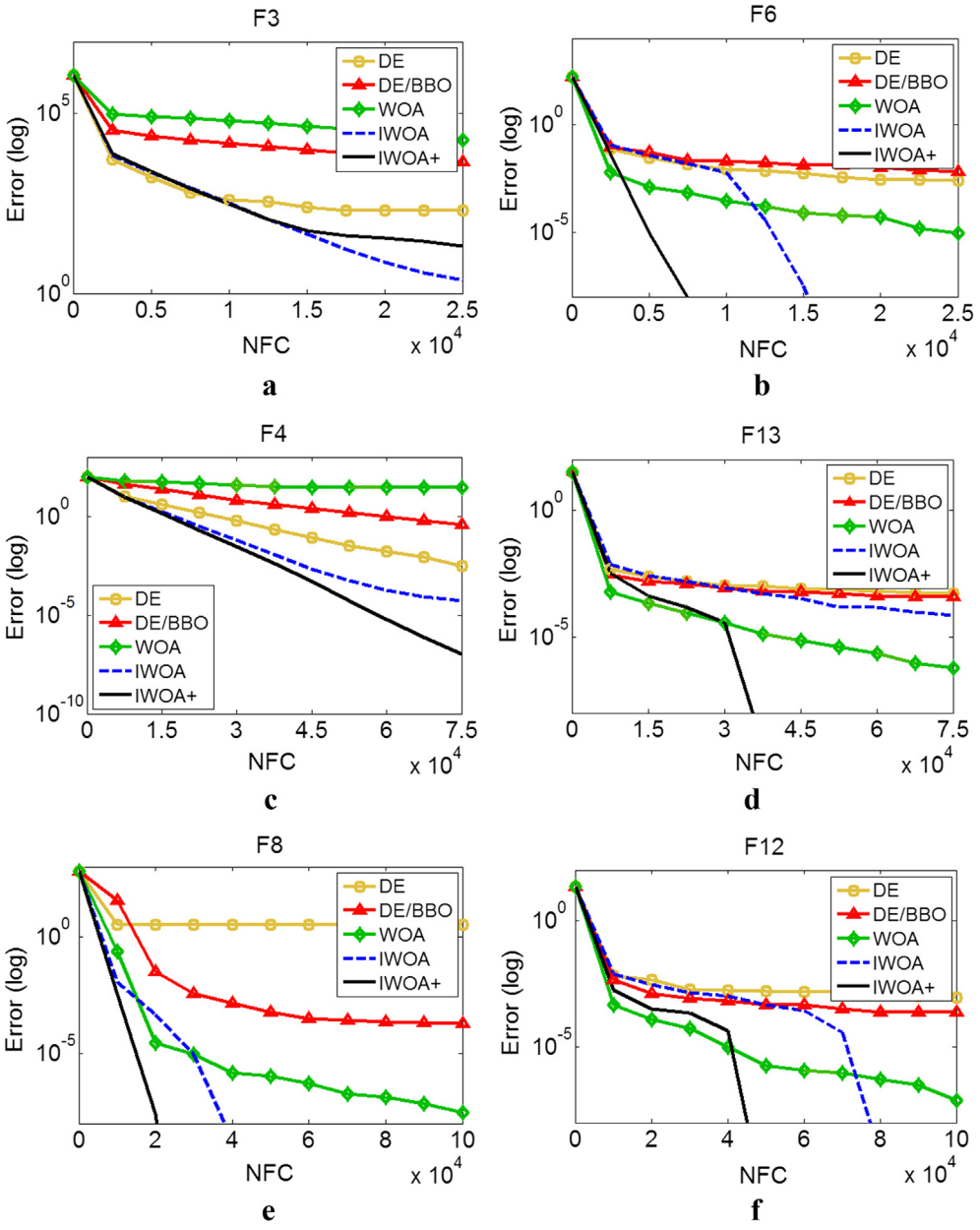

Fig. 2. Convergence curves of DE, DE/BBO, WOA, IWOA and IWOA$^+$ for some selected benchmark functions (Max Iteration = 500, n = 30). (a) F03 (NP = 50). (b) F06 (NP = 50). (c) F04 (NP = 150). (d) F13 (NP = 150). (e) F08 (NP = 200). (f) F12 (NP = 200).

Table 7
Mean and standard deviation of Error for benchmark functions. (NP = 50, *Max-NFCs* = 25000, n = 30).

Fun.	DE		DE/BBO		WOA		IWOA		IWOA$^+$	
	mean	Std.	mean	Std.	mean	Std.	mean	Std.	mean	Std.
F1	8.34e−9	2.17e−9	8.86e−9	9.9e−10	7.32e−9	2.25e−9	8.16e−9	1.22e−9	8.32e−9	1.07e−9
F2	7.88e−9	1.39e−9	7.42e−9	1.99e−9	6.36e−9	2.46e−9	6.17e−9	2.40e−9	6.20e−9	2.30e−9
F3	200.07	989.7	4.57e+3	1.33e+3	1.39e+4	9.88e+3	6.3058	8.3167	5.64e−4	9.64e−4
F4	16.657	5.3144	2.6073	1.2569	50.0021	26.1392	0.0601	0.0404	0.0019	0.0018
F5	24.931	40.469	59.290	38.130	26.9589	0.3870	20.8195	13.0362	15.8586	9.2086
F6	0.0018	0.0018	0.0076	0.0071	6.87e−6	1.24e−5	4.54e−9	3.01e−9	4.93e−9	2.78e−9
F7	8.7769	2.5803	0.0373	0.1844	8.00e−9	1.50e−9	9.03e−9	7.13e−10	9.07e−9	5.9e−10
F8	67.4647	36.5025	6.94e−4	0.0010	0.0575	0.4065	4.49e−9	3.09e−9	4.36e−9	3.09e−9
F9	0.3430	1.2369	0.0075	0.0072	0.0064	0.0156	0.0028	0.0052	0.0029	0.0079
F10	3.7821	4.8955	0.0373	0.0931	0.0109	0.0302	0.0041	0.0205	0.0021	0.0147
F11	2.8782	4.4954	0.0326	0.2258	0.1284	0.1026	0.0127	0.0377	0.0127	0.0320
F12	0.0022	0.0020	9.96e−4	9.08e−4	1.44e−6	3.22e−6	1.85e−5	1.23e−4	4.82e−9	2.84e−9
F13	0.0019	0.0018	7.74e−4	8.19e−4	2.23e−6	5.46e−6	2.74e−4	6.21e−4	4.85e−9	2.95e−9

Table 8
Mean and standard deviation of Error for benchmark functions. (NP = 150, *Max-NFCs* = 75000, n = 30).

Fun.	DE		DE/BBO		WOA		IWOA		IWOA$^+$	
	mean	Std.	mean	Std.	mean	Std.	mean	Std.	mean	Std.
F1	8.02e−9	1.40e−9	8.75e−9	9.9e−10	7.49e−9	2.18e−9	8.16e−9	1.28e−9	7.96e−9	1.09e−9
F2	6.84e−9	2.05e−9	7.08e−9	1.80e−9	6.19e−9	2.72e−9	6.13e−9	2.41e−9	6.30e−9	2.26e−9
F3	9.42e−9	4.5e−10	3.61e+3	1.08e+3	563.49	697.02	0.0015	0.0019	9.58e−9	1.78e−9
F4	0.0011	0.0014	0.4248	0.1834	34.517	27.501	1.13e−4	1.38e−4	6.99e−7	1.26e−6
F5	1.2764	1.8781	39.625	27.735	24.844	3.5831	5.0584	14.475	3.4434	1.9723
F6	0.0011	0.0013	0.0028	0.0028	2.50e−7	5.01e−7	5.30e−9	3.18e−9	4.13e−9	3.01e−9
F7	1.9390	1.1466	9.34e−9	5.4e−10	7.93e−9	1.40e−9	8.85e−9	8.2e−10	9.03e−9	6.2e−10
F8	5.3182	10.205	4.21e−4	4.34e−4	0.3453	2.4416	4.67e−9	3.11e−9	4.60e−9	3.02e−9
F9	0.0162	0.0200	0.0045	0.0061	0.0018	0.0046	0.0027	0.0065	0.0034	0.0072
F10	1.1936	2.1396	0.0021	0.0147	3.40e−4	0.0020	9.31e−9	4.0e−10	9.43e−9	5.0e−10
F11	0.5384	0.9399	4.39e−4	0.0022	0.0038	0.0052	9.63e−9	8.6e−10	9.71e−9	2.5e−10
F12	8.21e−4	8.56e−4	5.85e−4	4.91e−4	1.88e−7	2.84e−7	1.46e−8	7.06e−8	4.42e−9	2.71e−9
F13	4.90e−4	4.34e−4	3.24e−4	2.84e−4	1.62e−7	2.87e−7	4.56e−5	3.13e−4	4.60e−9	3.21e−9

Table 9
Mean and standard deviation of Error for benchmark functions. (NP = 200, *Max-NFCs* = 10,0000, n = 30).

Fun.	DE		DE/BBO		WOA		IWOA		IWOA$^+$	
	mean	Std.	mean	Std.	mean	Std.	mean	Std.	mean	Std.
F1	8.16e−9	1.19e−9	8.70e−9	8.5e−10	7.75e−9	1.79e−9	8.02e−9	1.36e−9	7.82e−9	1.27e−9
F2	6.47e−9	1.94e−9	6.69e−9	2.63e−9	6.21e−9	2.63e−9	6.17e−9	2.49e−9	5.88e−9	2.59e−9
F3	9.45e−9	4.6e−10	3.13e+3	877.84	209.15	266.10	1.04e−4	9.49e−5	9.16e−9	7.7e−10
F4	6.38e−6	1.54e−5	0.2472	0.0902	25.066	20.159	6.49e−6	1.40e−5	1.34e−8	1.39e−8
F5	1.1163	1.8082	38.230	27.541	24.931	0.3601	1.3775	2.0639	3.0327	10.3503
F6	8.67e−4	0.0012	0.0021	0.0027	1.24e−7	1.45e−7	4.38e−9	2.44e−9	4.56e−9	3.08e−9
F7	1.4425	1.0275	9.48e−9	4.1e−10	8.48e−9	1.32e−9	8.69e−9	7.9e−10	8.83e−9	7.0e−10
F8	2.2781	7.1333	2.27e−4	2.62e−4	1.09e−8	1.39e−8	4.33e−9	2.95e−9	4.87e−9	2.80e−9
F9	0.0118	0.0124	0.0058	0.0076	0.0026	0.0087	0.0022	0.0048	0.0023	0.0064
F10	0.6121	1.0524	0.0021	0.0147	3.16e−5	8.02e−5	9.30e−9	8.7e−10	9.51e−9	5.7e−10
F11	0.2239	0.6555	2.19e−4	0.0016	0.0045	0.0081	9.27e−9	1.19e−9	9.70e−9	2.6e−10
F12	8.74e−4	7.02e−4	2.64e−4	3.16e−4	1.05e−7	1.84e−7	2.51e−8	1.45e−7	4.19e−9	2.65e−9
F13	2.95e−4	2.53e−4	2.50e−4	3.26e−4	1.33e−7	2.71e−7	2.59e−5	8.19e−5	4.77e−9	2.88e−9

Table 10
Mean and standard deviation of NFCs for benchmark functions. (NP = 50, *Max-NFCs* = 25,000, n = 30).

Fun	DE			DE/BBO			WOA			IWOA			IWOA$^+$		
	mean	Std.	*SR*	mean	Std.	*SR*	mean	Std.	*SR*	mean	Std.	*SR*	mean	Std.	*SR*
F1	1.01e+4	1.0e+3	50	1.71e+4	429.8	50	6.4e+3	582.2	50	6.20e+3	261.9	50	6.35e+3	159.1	50
F2	1.76e+3	282.8	50	3.69e+3	380.2	50	1.73e+3	480.6	50	1.81e+3	237.1	50	1.67e+3	215.6	50
F3	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0
F4	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0
F5	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0
F6	NA	NA	0	NA	NA	0	NA	NA	0	1.23e+4	2.6e+3	50	4.95e+3	1.62e+3	50
F7	NA	NA	0	NA	NA	0	8.44e+3	335.29	50	9.00e+3	219.5	50	9.51e+3	313.8	50
F8	NA	NA	0	NA	NA	0	2.42e+4	2.1e+3	9	8.9e+3	1.0e+3	50	5.04e+3	567.6	50
F9	2.29e+4	5.05e+3	7	2.06e+4	3.5e+3	31	7.66e+3	4.4e+3	47	1.15e+4	8.0e+3	37	1.11e+4	7.49e+3	39
F10	2.37e+4	3.0e+3	8	1.86e+4	3.34e+3	40	NA	NA	0	NA	NA	0	24,891	270.19	13
F11	2.46e+4	1.44e+3	4	1.78e+4	2.21e+3	46	NA	NA	0	NA	NA	0	NA	NA	0
F12	NA	NA	0	NA	NA	0	2.49e+4	82.590	1	1.75e+4	4.5e+3	45	6.46e+3	4.22e+3	50
F13	NA	NA	0	NA	NA	0	2.49e+4	489.97	2	20,551	4.5e+3	30	8.61e+3	4.99e+3	50

IWOA$^+$ is investigated. The parameters of the five algorithms are the same as in Section 5.1 except for population size which is set to 50, 150 and 200. Results are given in Fig. 2 and Tables 7–9. Tables 10–12 shows the mean and standard deviation of NFC on 25 benchmark functions Table 7 shows the results when NP = 50. It can be seen that:

- IWOA performs better than DE for all test functions.
- IWOA outperforms DE/BBO.
- In comparison with the basic WOA, IWOA achieves better performance except for F12 and F13.

According to Table 7, we can conclude that IWOA$^+$ reaches better performance than the other three algorithms generally. Tables 8 and 9 indicate the results when the population size increases to 150 and 200 respectively. The results show that the overall performance of IWOA and IWOA$^+$ improves in comparison with the other algorithms. From Tables 8 and 9 it can be seen that IWOA$^+$ outperforms DE, DE/BBO and WOA on all test functions. Also, IWOA performs better than DE, DE/BBO and WOA on all test functions except for function F13.

Tables 10–12 present the average and standard deviation of *NFCs*. Furthermore, they show the *SR* values of the five comparative

Table 11
Mean and standard deviation of NFCs for benchmark functions. (NP = 150, *Max-NFCs* = 75,000, n = 30).

Fun	DE			DE/BBO			WOA			IWOA			IWOA$^+$		
	mean	Std.	*SR*	mean	Std.	*SR*	mean	Std.	*SR*	mean	Std.	*SR*	mean	Std.	*SR*
F1	1.29e+4	441.29	50	4.61e+4	835.84	50	1.44e+4	1.5e+3	50	1.47e+4	361.06	50	1.45e+4	338.13	50
F2	3.40e+3	340.21	50	1.01e+4	812.27	50	2.99e+3	537.70	50	4.27e+3	452.03	50	3.80e+3	461.65	50
F3	6.63e+4	3.54e+3	50	NA	NA	0	NA	NA	0	NA	NA	0	7.13e+4	2.37e+3	47
F4	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0	7.49e+4	457.23	2
F5	7.27e+4	4.57e+3	14	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0
F6	NA	NA	0	NA	NA	0	7.35e+4	4.12e+3	9	3.36e+4	5.58e+3	50	1.17e+4	3.09e+3	50
F7	6.63e+4	2.01e+4	8	7.09e+4	948.83	50	2.26e+4	893.20	50	2.12e+4	488.33	50	2.20e+4	635.79	50
F8	7.38e+4	8.45e+3	1	NA	NA	0	6.72e+4	1.07e+4	36	2.43e+4	2.29e+3	50	1.26e+4	2.24e+3	50
F9	5.64e+4	2.86e+4	15	5.94e+4	1.34e+4	29	1.98e+4	1.65e+4	46	2.71e+4	2.27e+4	41	2.90e+4	2.47e+4	39
F10	6.14e+4	2.44e+4	12	4.27e+4	4.83e+3	49	NA	NA	0	4.25e+4	3.98e+3	50	4.93e+4	2.49e+3	50
F11	5.64e+4	2.62e+4	17	4.60e+4	6.04e+3	48	NA	NA	0	5.61e+4	6.48e+3	49	6.38e+4	3.03e+3	50
F12	NA	NA	0	NA	NA	0	7.42e+4	2.52e+3	6	4.20e+4	1.08e+4	48	1.32e+4	4.05e+3	50
F13	NA	NA	0	NA	NA	0	7.22e+4	6.79e+3	13	5.13e+4	1.51e+4	44	1.87e+4	9.30e+3	50

Table 12
Mean and standard deviation of NFCs for benchmark functions. (NP = 200, *Max-NFCs* = 100,000, n = 30).

Fun	DE			DE/BBO			WOA			IWOA			IWOA$^+$		
	mean	Std.	*SR*	mean	Std.	*SR*	mean	Std.	*SR*	mean	Std.	*SR*	mean	Std.	*SR*
F1	1.57e+4	610.12	50	5.98e+4	1.09e+3	50	1.72e+4	2.12e+3	50	1.84e+4	494.01	50	1.84e+4	534.59	50
F2	4.29e+3	420.66	50	1.30e+4	923.67	50	3.46e+3	495.56	50	5.47e+3	549.46	50	4.81e+3	538.87	50
F3	7.78e+4	3.93e+3	50	NA	NA	0	NA	NA	0	NA	NA	0	8.90e+4	3.1e+3	50
F4	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0	9.42e+4	4.89e+3	38
F5	9.23e+4	7.74e+3	33	NA	NA	0	NA	NA	0	NA	NA	0	NA	NA	0
F6	NA	NA	0	NA	NA	0	9.85e+4	6.33e+3	6	4.35e+4	9.05e+3	50	1.52e+4	3.67e+3	50
F7	8.18e+4	3.25e+4	12	9.25e+4	1.36e+3	50	2.93e+4	1.10e+3	50	2.67e+4	465.06	50	2.76e+4	593.70	50
F8	NA	NA	0	NA	NA	0	8.08e+4	1.99e+4	37	3.25e+4	3.07e+3	50	1.58e+4	2.16e+3	50
F9	6.81e+4	4.10e+4	19	8.13e+4	1.83e+4	26	3.04e+4	2.87e+4	43	3.60e+4	3.22e+4	40	3.49e+4	3.08e+4	41
F10	7.00e+4	3.88e+4	19	5.56e+4	6.61e+3	49	NA	NA	0	4.70e+4	2.78e+3	50	5.86e+4	3.53e+3	50
F11	62,670	4.09e+4	23	5.88e+4	6.08e+3	49	NA	NA	0	6.16e+4	4.95e+3	50	7.42e+4	3.70e+3	50
F12	NA	NA	0	NA	NA	0	9.75e+4	5.47e+3	13	5.45e+4	1.72e+4	49	1.65e+4	7.71e+3	50
F13	NA	NA	0	NA	NA	0	9.46e+4	1.28e+4	15	6.95e+4	2.08e+4	40	2.21e+4	1.02e+4	50

Table 13
Mean and standard deviation of Error for benchmark functions. (NP = 100, *Max-NFCs* = 50,000, n = 10).

Fun.	DE		DE/BBO		WOA		IWOA		IWOA$^+$	
	mean	Std.	mean	Std.	mean	Std.	mean	Std.	mean	Std.
F1	6.75e−9	2.02e−9	8.12e−9	1.25e−9	7.79e−9	1.68e−9	6.54e−9	2.46e−9	6.32e−9	1.97e−9
F2	5.23e−9	2.65e−9	5.78e−9	2.30e−9	6.11e−9	2.51e−9	4.97e−9	2.96e−9	4.89e−9	2.66e−9
F3	7.74e−9	1.89e−9	2.57e−7	3.79e−7	9.57e−4	0.0042	7.50e−9	1.70e−9	6.96e−9	1.84e−9
F4	8.28e−9	1.60e−9	8.62e−9	1.16e−9	1.6102	5.2905	8.32e−9	1.37e−9	8.17e−9	1.09e−9
F5	0.8776	1.5471	1.7341	0.9717	7.5268	19.191	0.4784	1.6108	0.7973	1.8081
F6	6.67e−5	1.15e−4	0.0016	0.0016	3.57e−7	6.04e−7	4.53e−9	2.96e−9	4.39e−9	2.81e−9
F7	0.2242	0.5779	8.72e−9	1.02e−9	8.36e−9	1.34e−9	8.01e−9	1.43e−9	7.88e−9	1.26e−9
F8	2.68e−6	4.56e−6	1.81e−4	2.99e−4	2.71e−8	6.55e−8	4.27e−9	2.66e−9	4.53e−9	2.80e−9
F9	0.1162	0.0643	0.0290	0.0234	0.0256	0.0798	0.0090	0.0138	0.0066	0.0133
F10	0.0933	0.3096	7.81e−9	1.74e−9	1.76e−4	0.0012	7.46e−9	1.96e−9	7.50e−9	1.60e−9
F11	8.79e−4	0.0030	7.78e−9	1.59e−9	5.24e−5	5.86e−5	7.64e−9	1.73e−9	7.44e−9	1.99e−9
F12	0.0012	0.0013	4.66e−4	4.63e−4	3.73e−7	1.09e−6	4.67e−9	3.02e−9	5.20e−9	2.79e−9
F13	5.74e−4	7.32e−4	3.86e−4	3.71e−4	2.32e−7	6.54e−7	5.06e−9	3.37e−9	5.01e−9	2.88e−9

algorithms. From Table 10, it is obvious that the IWOA and IWOA$^+$ need less *NFCs* to reach *VTR* for the majority of the benchmark functions. According to the results, we can find that our proposed algorithms can improve the performance of WOA in term of *SR*. For example, from Table 10 we can see that IWOA$^+$ fails to reach the *VTR* only for four test functions while DE can't reach to *VTR* for eight benchmark functions. In addition, IWOA and IWOA$^+$ can reach *VTR* for five and seven test functions in 50 out of 50 runs respectively while WOA can reach the *VTR* only for three test functions on all 50 runs. Moreover, the reported results in Tables 11 and 12 confirm the superior performance of our proposed algorithms.

5.4. Effect of dimensionality

In this sub-section the impact of increasing the problem dimension on the performance of proposed algorithms and three other algorithms are investigated. Experimental results for function F1-F13 when dimension is set to 10, 50 and 100 are recorded in Tables 13–15 respectively. It is important to mention that the other parameters are not changed as mentioned in Section 5.1. From Tables 13–15, we can conclude that our proposed algorithms can maintain their performance better than the other comparative algorithms and they are better than the other algorithms for the majority of test functions. For example:

Table 14
Mean and standard deviation of Error for benchmark functions. (NP = 100, *Max-NFCs* = 50,000, n = 50).

Fun.	DE		DE/BBO		WOA		IWOA		IWOA$^+$	
	mean	Std.	mean	Std.	mean	Std.	mean	Std.	mean	Std.
F1	8.77e−9	1.47e−9	1.01e−8	2.54e−9	7.58e−9	1.75e−9	8.38e−9	9.9e−10	8.80e−9	1.03e−9
F2	7.16ez−9	1.83e−9	7.40e−9	1.86e−9	6.23e−9	2.52e−9	6.72e−9	2.18e−9	6.87e−9	2.20e−9
F3	114.44	705.97	4.37e+4	8.21e+3	4.01e+4	2.07e+4	673.23	504.67	1.37e+3	2.42e+3
F4	19.760	3.8784	10.657	2.3385	68.510	22.061	0.4228	0.2278	0.0850	0.0486
F5	43.276	34.354	84.165	48.238	46.450	0.3768	37.174	13.440	36.943	20.125
F6	0.0033	0.0036	0.0057	0.0054	1.22e−6	2.04e−6	7.54e−6	4.68e−5	4.54e−9	3.09e−9
F7	8.1994	2.2154	2.09e−5	7.49e−6	8.26e−9	1.32e−9	8.69e−9	9.1e−10	9.36e−9	5.0e−10
F8	111.12	50.650	9.44e−4	0.0010	5.47e−8	1.00e−7	5.41e−9	2.67e−9	5.33e−9	3.23e−9
F9	0.0833	0.1612	0.0034	0.0047	0.0040	0.0126	8.86e−4	0.0034	0.0010	0.0029
F10	2.1575	3.0858	0.0311	0.0976	0.0012	0.0021	0.0051	0.0272	0.0050	0.0226
F11	2.5941	4.1077	0.0011	0.0033	0.1048	0.0934	0.0074	0.0274	0.0162	0.0397
F12	0.0020	0.0014	9.07e−4	0.0010	6.76e−7	1.21e−6	5.07e−5	1.79e−4	4.42e−9	2.94e−9
F13	6.56e−4	5.88e−4	5.52e−4	4.86e−4	2.07e−7	2.74e−7	1.76e−4	5.54e−4	4.40e−9	2.77e−9

Table 15
Mean and standard deviation of Error for benchmark functions. (NP = 100, *Max-NFCs* = 50,000, n = 100).

Fun.	DE		DE/BBO		WOA		IWOA		IWOA$^+$	
	mean	Std.	mean	Std.	mean	Std.	mean	Std.	mean	Std.
F1	0.3460	2.1554	0.0145	0.0059	7.06e−9	2.34e−9	8.17e−9	1.23e−9	8.98e−9	8.7e−10
F2	8.02e−9	1.58e−9	7.44e−9	1.66e−9	6.89e−9	2.04e−9	7.30e−9	2.26e−9	7.27e−9	1.87e−9
F3	1.88e+4	8.05e+3	2.29e+5	2.55e+4	4.90e+5	7.83e+4	4.54e+4	1.21e+4	7.91e+4	1.75e+4
F4	36.315	2.9274	36.468	3.3745	79.401	18.512	8.9163	2.5570	6.8796	6.1015
F5	295.52	87.604	455.43	109.35	96.658	0.4841	88.243	1.9621	88.314	2.5341
F6	0.0043	0.0044	0.0097	0.0092	4.23e−6	1.59e−5	8.97e−4	0.0042	4.66e−9	2.97e−9
F7	14.410	1.2857	0.4281	0.5463	8.23e−9	1.51e−9	8.78e−9	1.05e−9	9.44e−9	3.9e−10
F8	428.18	76.574	0.0022	0.0023	1.09e−7	2.20e−7	4.65e−9	2.92e−9	4.76e−9	3.07e−9
F9	0.6274	1.1056	0.0126	0.0072	0.0016	0.0067	7.88e−4	0.0052	0.0011	0.0037
F10	8.4911	3.2854	3.9769	2.2234	0.0024	9.51e−4	0.1958	0.4360	0.1126	0.3348
F11	75.554	19.898	21.398	16.512	0.3800	0.2462	1.1607	0.5020	1.1724	0.4139
F12	0.0018	0.0018	8.59e−4	8.56e−4	8.94e−7	1.77e−6	2.24e−4	5.76e−4	4.15e−9	2.48e−9
F13	0.0011	0.0014	9.87e−4	8.40e−4	9.44e−7	2.09e−6	1.90e−4	4.45e−4	4.57e−5	4.70e−4

- From Table 13, we can see IWOA$^+$ outperforms DE, DE/BBO and WOA on all test functions.
- In addition, Table 14 shows that IWOA$^+$ is better than DE, DE/BBO and WOA, on 12, 12 and 10 benchmark functions respectively.
- From Table 15 we can observe that IWOA$^+$ is better than WOA for 8 out of 13 test functions.
- From Tables 13–15, we can find that the performance of IWOA$^+$ is better than IWOA for the majority of functions.

Some of illustrative convergence graphs are shown in Fig. 3. From Fig. 3, we can conclude that our modifications on WOA are effective and can improve WOA's performance on high dimensionality.

5.5. Comparison the diversity preservation

In this section we want to consider and compare the population diversity of DE, DE/BBO, WOA, IWOA and IWOA$^+$. The population diversity is calculated according to Eq. (10):

$$\text{Diversity} = \frac{1}{NP}\sum_{i=1}^{NP}\sqrt{\sum_{j=1}^{n} x_i^2(j) - \bar{x}^2(j)} \tag{10}$$

where $\bar{x}$ shows the mean position in population (El-abd, 2017). Fig. 4 shows some representative diversity curves of the five algorithms. By looking carefully at Fig. 4, we can find that the behavior of IWOA$^+$ changes according to the type of test functions. According to Section 4.2, IWOA$^+$ changes the *search mode* parameter and finally uses the re-initialization when the population can't achieve a better solution for the problem. From Fig. 4, we can find that for unimodal functions (4, 17) the diversity of IWOA$^+$ decreases quickly. However, for multimodal functions (8, 12, 13, 21) the IWOA$^+$'s diversity is maintained at high level for a big part of the search process which helps to escape from local optima. So, IWOA$^+$ is able to adapt the search according to different landscapes. Table 16 illustrates the average number of changing the search mode and using re-initialization which are related to the type and complexity of test functions. According to Table 16, it is obvious that with increasing the complexity of functions the changing rate and number of using re-initialization are increased.

5.6. The effectiveness of the exploration and the exploitation parts in IWOA

According to Section 4.1, IWOA includes two main parts which are exploration part and exploitation part. IWOA employs these two parts adaptively. In this experiments, we investigate the influence of these two parts on the performance of IWOA. Fig. 5, shows the results on F04, F08, F10 and F15. In Fig. 5, IWOA1 means IWOA without employing the exploitation part and IWOA2 means IWOA without employing exploration part. From Fig. 5, it is clear that IWOA surpasses IWOA1 and IWOA2. From graphical results of Fig. 5, it is clear that IWOA$^+$ with utilizing re-initialization and adaptive control strategy and also with the help of exploration and exploitation parts can obtain better performance than IWOA.

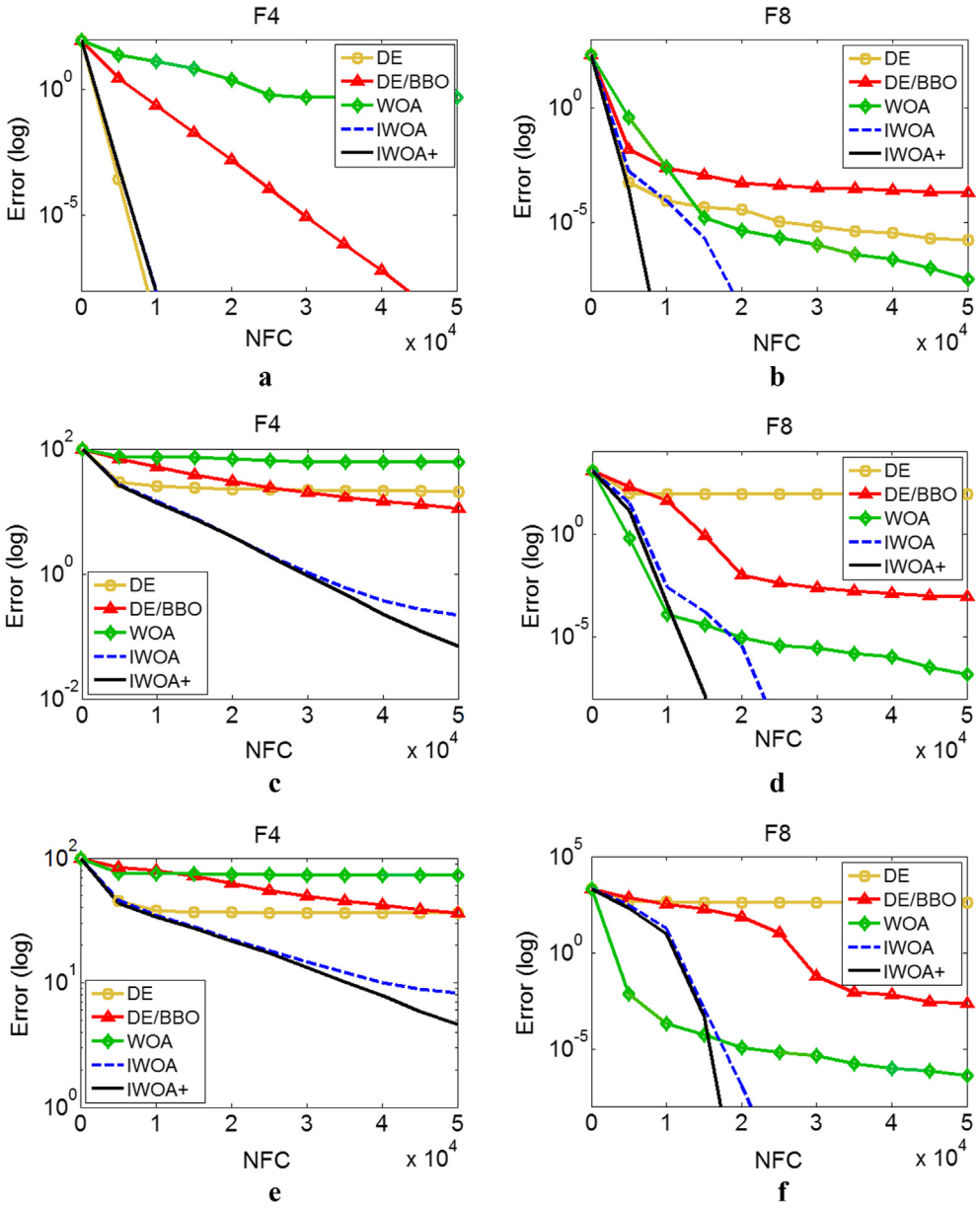

Fig. 3. Convergence curves of DE, DE/BBO, WOA, IWOA and IWOA$^+$. (Max Iteration = 500, NP = 100). (a) F04 (n = 10). (b) F08 (n = 10). (c) F04 (n = 50). (d) F08 (n = 50). (e) F04 (n = 100). (f) F08 (n = 100).

5.7. Discussion

To increase the exploration ability of WOA, in this paper, a hybrid algorithm is introduced. Since the exploration power of DE's mutation is good, it is integrated in WOA operators. Results verify that IWOA can improve the performance of WOA, especially in multimodal test functions. Although IWOA converges slower than WOA, it obtains better performance or results on multimodal functions. In addition, IWOA$^+$ via control parameter *search mode* can switch between exploration and exploitation phases automatically. Furthermore, we apply the initialization in IWOA$^+$ which helps to improve the diversity of the population. As a result, IWOA$^+$ has the ability of balancing exploration and exploitation effectively. The results indicate the effectiveness of IWOA$^+$. For example, Table 3, shows the ability of IWOA and specially IWOA$^+$ on multimodal test functions. Results prove that IWOA and IWOA$^+$ can achieve a better balance between exploration and exploitation.

6. Conclusion

In order to balance the exploration and exploitation abilities of recently developed population-based algorithm, WOA, the two improvements, IWOA and IWOA$^+$, are presented in this paper. In IWOA, the DE's mutation operator which has a good exploration ability is integrated into WOA. IWOA$^+$ is presented through adding

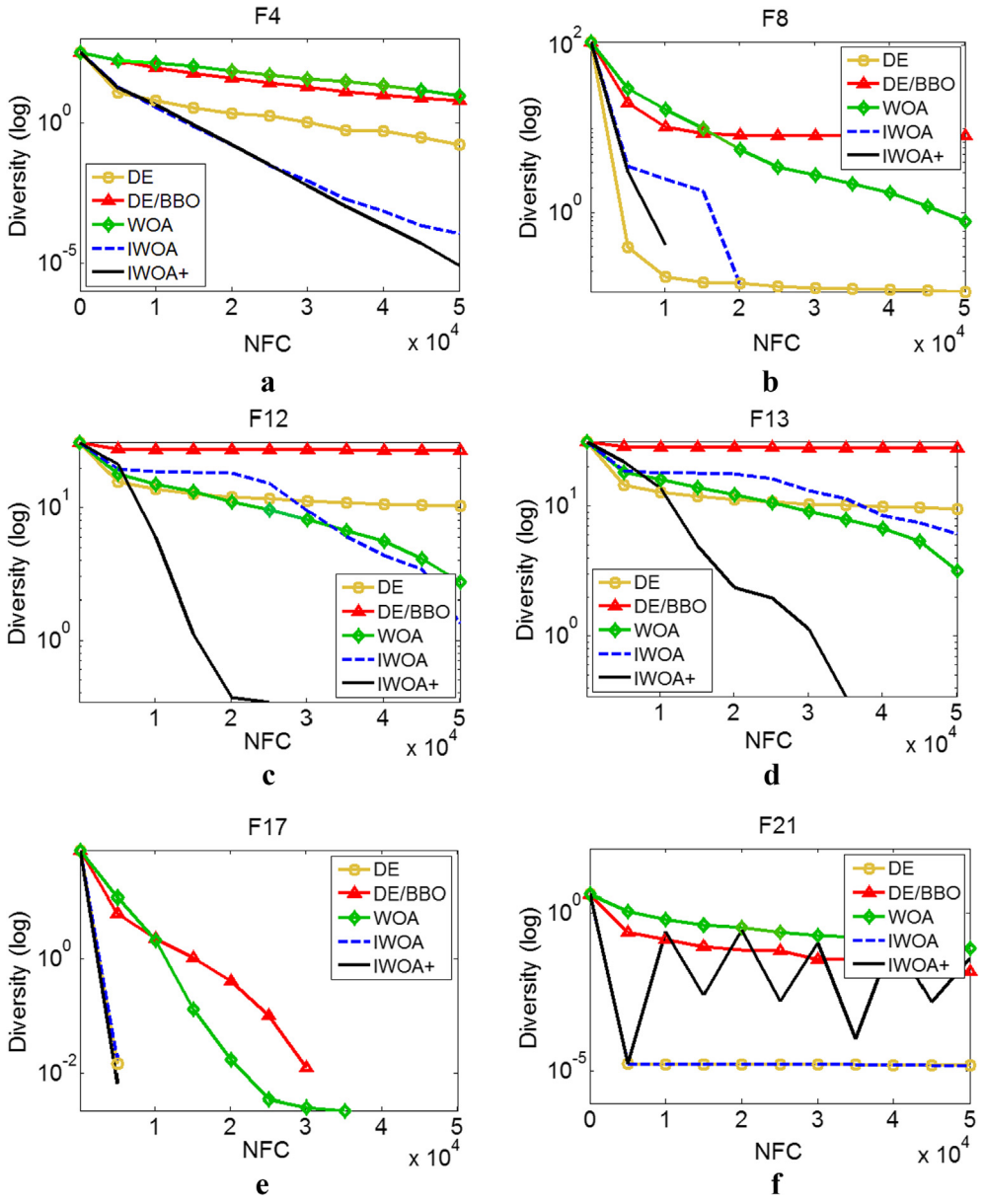

Fig. 4. Population diversity of DE, DE/BBO, WOA, IWOA and IWOA$^+$ for a sample of benchmark functions. (a) F04. (b) F08. (c) F12. (d) F13. (e) F17. (f) F21.

Table 16
Average number of changing the search mode and using re-initialization of IWOA$^+$ over 50 independent runs.

Fun	Dim.	change to exploit mode	Re-initialization
F4	30	0	0
F8	30	1	0
F12	30	1.6	0.6
F13	30	1.9	0.9
F17	2	0.14	0
F21	2	15.9	15

the search mode parameter to IWOA for changing the active part (exploration or exploitation) adaptively in IWOA. In addition, we included re-initialization to improve IWOA$^+$ performance. These two proposed algorithms are evaluated on 25 optimization benchmark functions and they are compared with WOA and the other seven nature-inspired algorithms. Experimental results show that our proposed algorithms can improve the performance of WOA. Furthermore, results illustrate that our proposed algorithms are better than other eight comparative algorithms. The influence of problem dimension and population size are also investigated. In this work we use 25 test functions to evaluate our proposed algorithms. We will investigate the performance of IWOA and IWOA$^+$ on the real-world applications. Our other future work direction will be to extend IWOA and IWOA$^+$ for multi-objective optimization or to apply for training neural networks. In addition, future work can consist on investigating the effect of applying chaos theory on the performance of our proposed algorithms.

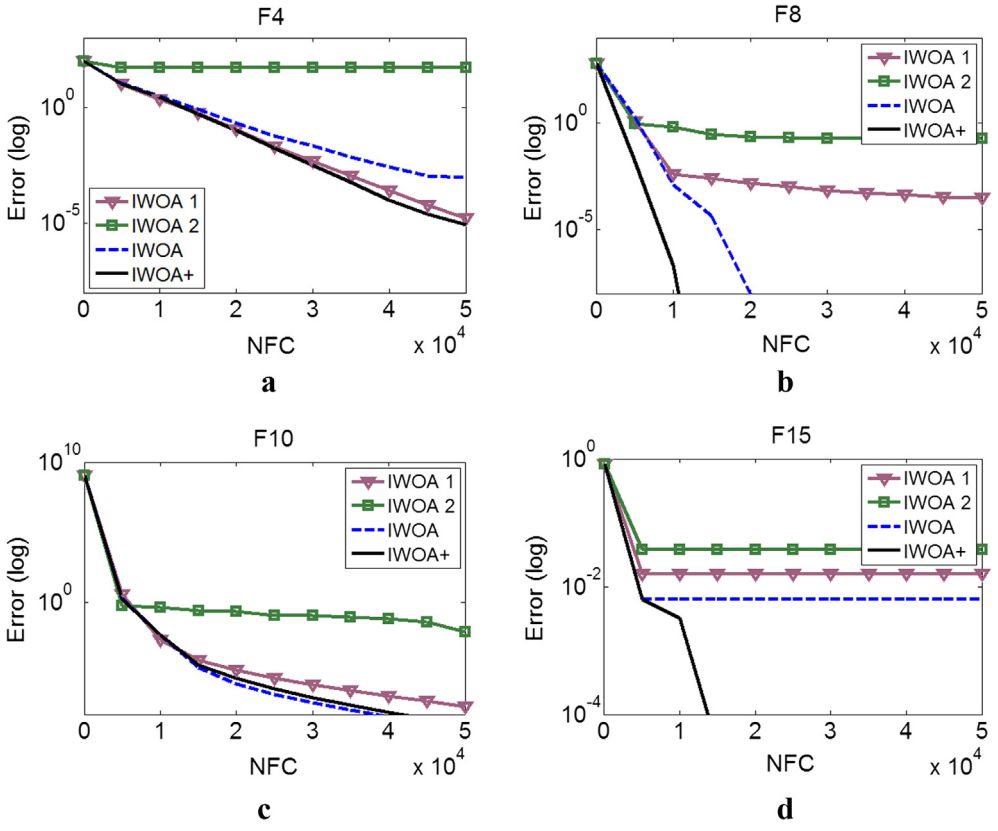

Fig. 5. Convergence curves of IWOA 1, IWOA 2, IWOA and IWOA* for the selected benchmark functions. (a) F04. (b) F08. (c) F10. (d) F15.

Conflict of interest

The authors declare that there is not any conflict of interest in this paper.

Acknowledgment

The authors express deep gratitude to Dr. Esmaiel Hadavandi for many constructive suggestions.

References

Aljarah, I., Faris, H., & Mirjalili, S. (2018). Optimizing connection weights in neural networks using the whale optimization algorithm. *Soft Computing, 22*(1).

Aziz, M. A. E., Ewees, A. A., & Ella, A. (2017). Whale Optimization Algorithm and Moth-Flame Optimization for multilevel thresholding image segmentation. *Expert Systems with Applications, 83*, 242–256.

ben oualid Medani, K., Sayah, S., & Bekrar, A. (2018). Whale optimization algorithm based optimal reactive power dispatch: A case study of the Algerian power system. *Electric Power Systems Research, 163*, 696–705.

Bentouati, B., Chaib, L., & Chettih, S. (2016). A hybrid whale algorithm and pattern search technique for optimal power flow problem. *2016 8th international conference on modelling, identification and control (ICMIC), Algiers* (pp. 1048–1053).

Brest, J., Zumer, V., & Maucec, M. S. (2006). Self-adaptive differential evolution algorithm in constrained real-parameter optimization. *2006 IEEE int. conf. evol. comput.* (pp. 215–222).

Cheng, M.-Y., & Lien, L.-C. (2012). Hybrid artificial intelligence-based PBA for benchmark functions and facility layout design optimization. *Journal of Computing in Civil Engineering, 26*, 612–624.

Clerc, M., & Kennedy, J. (2002). The particle swarm - Explosion, stability, and convergence in a multidimensional complex space. *IEEE Transactions on Evolutionary Computation, 6*(1), 58–73.

Das, S., Abraham, A., & Konar, A. (2008). Swarm intelligence algorithms in bioinformatics. *Studies in Computational Intelligence, 94*(2008), 113–147.

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.

Eberhart, R., & Shi, Y. (2004). Guest editorial special issue on particle swarm optimization. *IEEE Transactions on Evolutionary Computation, 8*(3), 201–203.

El-abd, M. (2017). Global-best brain storm optimization algorithm. *Swarm and Evolutionary Computation, 37*, 27–44.

Engelbrecht, A. P. (2007). *Computational intelligence: An introduction* (2nd ed.). Wiley Publishing.

Findler, N. S. V., Lo, C., & Lo, R. (1987). Pattern search for optimization. *Mathematics and Computers in Simulation, 29*(1), 41–50.

Gandomi, A. H., & Alavi, A. H. (2012). Krill herd: A new bio-inspired optimization algorithm. *Communications in Nonlinear Science and Numerical Simulation, 17* (12), 4831–4845.

Ghasemi, M., Ghavidel, S., Rahmani, S., Roosta, A., & Falah, H. (2014). A novel hybrid algorithm of imperialist competitive algorithm and teaching learning algorithm for optimal power flow problem with non-smooth cost functions. *Engineering Applications of Artificial Intelligence, 29*, 54–69.

Gong, W., Cai, Z., & Ling, C. X. (2011). DE/BBO: A hybrid differential evolution with biogeography-based optimization for global numerical optimization. *Soft Computing, 15*(4), 645–665.

Hadavandi, E., Mostafayi, S., & Soltani, P. (2018). A Grey Wolf Optimizer-based neural network coupled with response surface method for modeling the strength of siro-spun yarn in spinning mills. *Applied Soft Computing Journal, 72*, 1–13.

Hafezi, R., Shahrabi, J., & Hadavandi, E. (2015). A bat-neural network multi-agent system (BNNMAS) for stock price prediction: Case study of DAX stock price. *Applied Soft Computing Journal, 29*, 196–210.

Jamil, M., & Blekinge, X. Y. (2013). A literature survey of benchmark functions for global optimization problems. *International Journal of Mathematical Modelling and Numerical Optimisation, 4*(2), 150–194.

Kaur, G., & Arora, S. (2017). Chaotic whale optimization algorithm. *Journal of Computational Design and Engineering* (December 2017).

Lin, C.-L., Mimori, A., & Chen, Y.-W. (2012). Hybrid particle swarm optimization and its application to multimodal 3D medical image registration. *Computational Intelligence and Neuroscience, 2012*, 561406.

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

Lucas, C., Nasiri-Gheidari, Z., & Tootoonchian, F. (2010). Application of an imperialist competitive algorithm to the design of a linear induction motor. *Energy Conversion and Management, 51*(7), 1407–1411.

Mafarja, M. M., & Mirjalili, S. (2017). Hybrid Whale Optimization Algorithm with simulated annealing for feature selection. *Neurocomputing*, 1–11.

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

Mirjalili, S. (2016). SCA: A Sine Cosine Algorithm for solving optimization problems. *Knowledge-Based Systems, 96*, 120–133.

Mirjalili, S., & Hashim, S. Z. M. (2010). A new hybrid PSOGSA algorithm for function optimization. *Proceedings of ICCIA 2010 - 2010 international conference on computer and information application, no. 1* (pp. 374–377).
Mirjalili, S., & Lewis, A. (2016). The whale optimization algorithm. *Advances in Engineering Software, 95*, 51–67.
Mirjalili, S., Mirjalili, S. M., & Lewis, A. (Mar. 2014). Grey wolf optimizer. *Advances in Engineering Software, 69*, 46–61.
Simon, D., & Member, S. (2008). Biogeography-based optimization. *IEEE Transactions on Evolutionary Computation, 12*(6).
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.
Wu, J., Wang, H., Li, N., Yao, P., Huang, Y., & Yang, H. (2018). Path planning for solar-powered UAV in urban environment. *Neurocomputing, 275*, 2055–2065.
Yang, X.-S., & Deb, S. (2009). Cuckoo search via levy flights. *2009 World congress on nature & biologically inspired computing (NaBIC)* (pp. 210). 214.
Yao, X., Liu, Y., & Lin, G. (1999). Evolutionary programming made faster. *IEEE Transactions on Evolutionary Computation, 3*(2), 82–102.
Yu, Y., Wang, H., Li, N., Su, Z., & Wu, J. (2017). Automatic carrier landing system based on active disturbance rejection control with a novel parameters optimizer. *Aerospace Science and Technology, 69*, 149–160.