Complementary Differential Evolution-based Whale Optimization Algorithm for Function Optimization

Qiang Qu, Yi-Han Huang, Xiao-Li Wang, and Xue-Bo Chen

***Abstract*—The whale optimization algorithm (WOA) is a metaheuristic search algorithm for solving the problem of function optimization. However, in the later stage of iterations, WOA suffers from premature convergence because the search agents are attracted by the elite vector. In this paper, a hybrid WOA based on complementary differential evolution, called CDEWOA, is proposed. First, a novel uniform initialization strategy is employed to enhance the diversity of initial population. Second, the differential evolution with a complementary mutation operator is embedded in the WOA to improve search accuracy and speed. Third, the introduction of a local peak avoidance strategy enables CDEWOA to jump out local optimum. Finally, the proposed CDEWOA is tested with 14 mathematical optimization problems. The test results illustrate that CDEWOA has better performance than IWOA, WOA, CDE, DE, and PSO in terms of convergence speed and convergence accuracy.**

***Index Terms*—whale optimization algorithm, differential evolution, complementary mutation operator, uniform initialization strategy**

I. Introduction

MANY practical problems in engineering and other fields are modeled as problems of function optimization . The optimal solution of an engineering problem under certain constraints is often the minimum or maximum value of the index function. Traditional function optimization problems are generally deterministic problems with fixed structures and parameters. One can obtain analytical solutions by solving equations or approximate solutions by derivative-based algorithm. However, it is difficult for traditional optimization algorithms to obtain the optimal solution of a complex function, due to many local optimums, a complex search space, and model uncertainty, especially for high-dimensional problems. Some researchers have solved those complex high-dimension problems by mimicking the unique behaviors of some animals, plants, or physical phenomena, and have obtained better solutions even without any characteristics of the objective function. These metaheuristic algorithms include the genetic algorithm (GA) [1], differential evolution (DE) algorithm [2], simulated annealing (SA) algorithm [3], central force optimization (CFO) algorithm [4], particle swarm optimization (PSO) algorithm [5], ant colony optimization (ACO) algorithm [6], gravitational search-based PSO algorithm [7], improved flower pollination algorithm [8], and the meme grouping strategy based krill herd algorithm [9].

The novel metaheuristic algorithm called the whale optimization algorithm (WOA), was proposed by Griffith et al. in 2016 [10]. Due to advantages such as simple definition, few control parameters, and a wide range of applications, WOA has attracted widespread attention from researchers. However, it also inherits the drawbacks of metaheuristic algorithms. WOA is easy to fall into local minimum, and the convergence accuracy is not ideal.

Researchers have tried to find a better parameter update method for WOA to improve its performance. By changing the convergence factor from linear decreasing to adaptive variation, researchers proposed adaptive whale optimization (AWOA) [11]. AWOA balances exploration and exploitation of WOA and obtains better performance. Sun et al. combined the nonlinear dynamic strategy based on a cosine function, Lévy-flight strategy, and quadratic interpolation method to present a modified WOA that offered superior performance on large-scale global optimization problems [12]. Kaur et al. introduced chaos theory into the optimization process of WOA and proposed the chaotic whale optimization algorithm (CWOA) [13]. Various chaotic maps are considered in the CWOA to balance exploration and exploitation, and the results of simulation prove that the chaotic maps (especially the tTent Map) are able to improve the performance of WOA.

The advantages of several optimization algorithms are combined to form a hybrid optimization algorithm. To enhance the exploitation performance of WOA, Korashy et al. used the leadership hierarchy of the gray wolf optimizer (GWO) to find the best optimum solution of WOA and proposed the hybrid WOA/GWO algorithm [14]. To enhance the exploitation, Mafarja et al. used the SA algorithm to form two hybrid models: the low-level teamwork hybrid (LTH) and the high-level hybrid (HRH). In the LTH model, the SA

Manuscript received April 12, 2020; revised August 28, 2020. This work was supported in part by National Natural Science Foundation of China (Grant No. 71571091 and 71371092).

Qiang Qu is with the School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, 114051, PR China (e-mail:askdqq@163.com).

Yi-Han Huang is with the School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, 114051, PR China (e-mail: yihan_huang@qq.com).

Xiao-Li Wang is with the Anshan Food Supervision and Law Enforcement Detachment, Anshan, 114001, PR China (e-mail: 494853271 @qq.com)

Xue-Bo Chen is with the School of Electronic and Information Engineering, University of Science and Technology Liaoning, Anshan, 114051, PR China (e-mail: xuebochen@126.com).

algorithm is used as a component of WOA to ensure the local optimum. In the HRH model, the SA algorithm is employed in a pipeline mode to preserve the diversity of the algorithm [15]. Abdel-Basset et al. incorporated WOA with Nawaz-Enscore-Ham (NEH) and proposed the hybrid whale algorithm (HWA) to improve the performance of WOA [16]. To enhance local optimum avoidance ability, a modified differential evolution operator with strong exploration capability is embedded in WOA with the aid of a lifespan mechanism. Further, an asynchronous model is employed to accelerate WOA's convergence [17]. Bozorgi et al. combined exploitation of WOA with exploration of DE to present the improved whale optimization algorithm (IWOA) [18].

The main aim of this paper is to solve the problems of weak diversity and precocity of WOA, and its highlights are listed below.

Complementary differential evolution based WOA (CDEWOA) uses a novel uniform initialization strategy to increase the probability of finding a good solution during the exploration phase. The strategy consists of two steps: uniformly sampling the search space to obtain candidate initial solution sets and randomly permuting the candidate solutions to generate the initial search agents.

On the basis of circular encircling and spiral attacking the prey, CDEWOA embeds the complementary differential evolution strategy into WOA to increase the probability of obtaining an optimal value.

CDEWOA introduces a mechanism to detect whether the algorithm is trapped in the local optimum. On determining that CDEWOA falls into the local peak, the corresponding escaping strategy is performed immediately to recover the global search capability of CDEWOA.

The rest of the paper is structured as follows. Section II introduces the principle of the WOA and analyzes the defects of the algorithm. Section III describes the principle of CDEWOA. Section IV describes the use of 14 benchmark functions to validate the performance of CDEWOA by comparing it with IWOA, WOA, complementary differential evolution (CDE), DE, and PSO. Section V summarizes the main findings of this study.

II. Whale Optimization Algorithm

WOA is inspired by the intelligent hunting method of whales. The predation process of whales can be said to consist of three stages: searching for the prey, encircling the prey, and bubble-net attacking of the prey.

A. Searching for the Prey

The process of searching for the prey can be denoted as

$$\boldsymbol{D} = \left|\boldsymbol{C} \cdot \boldsymbol{X}_{rand}(t) - \boldsymbol{X}(t)\right| \tag{1}$$

$$\boldsymbol{X}(t+1) = \boldsymbol{X}_{rand}(t) - \boldsymbol{A} \cdot \boldsymbol{D} \tag{2}$$

where $\boldsymbol{X}_{rand}(t)$ is a random vector selected from the current population, $\boldsymbol{X}(t)$ represents the current position vector to be updated, $\boldsymbol{A}$ and $\boldsymbol{C}$ are the coefficient vectors, and $\cdot$ is an element-by-element multiplication. All of the variables and coefficients are updated as the iteration increases. $\boldsymbol{A}$ and $\boldsymbol{C}$ can be calculated as follows:

$$\boldsymbol{A}=2a \cdot \boldsymbol{r} - a \tag{3}$$

$$\boldsymbol{C} = 2 \cdot \boldsymbol{r} \tag{4}$$

$$a = 2 - \frac{2t}{T_{\max}} \tag{5}$$

where the convergence factor a can be calculated by (5), its value linearly decreases from 2 to 0 as the iteration increases, and $\boldsymbol{r}$ is a random vector in [0,1].

B. Encircling the Prey

When a search agent finds the prey, the other search agents can move closer to the prey. As the location of the prey is not known a priori, WOA assumes the current best location as the location of the prey. Therefore, the behavior of encircling the prey can be modeled as

$$\boldsymbol{D} = \left|\boldsymbol{C} \cdot \boldsymbol{X}^*(t) - \boldsymbol{X}(t)\right| \tag{6}$$

$$\boldsymbol{X}(t+1) = \boldsymbol{X}^*(t) - \boldsymbol{A} \cdot \boldsymbol{D} \tag{7}$$

where $\boldsymbol{X}^*(t)$ represents the location of the current best individual, $\boldsymbol{X}(t)$ represents the current position vector to be updated, and $\boldsymbol{A}$ and $\boldsymbol{C}$ are the coefficient vectors. The whales disperse and explore the search space when $|\boldsymbol{A}| > 1$; otherwise, the whales approach the prey.

C. Bubble-net Attacking of the Prey

Another unique behavior of whales is that of bubble-net attacking of the prey, as shown in Fig. 1.

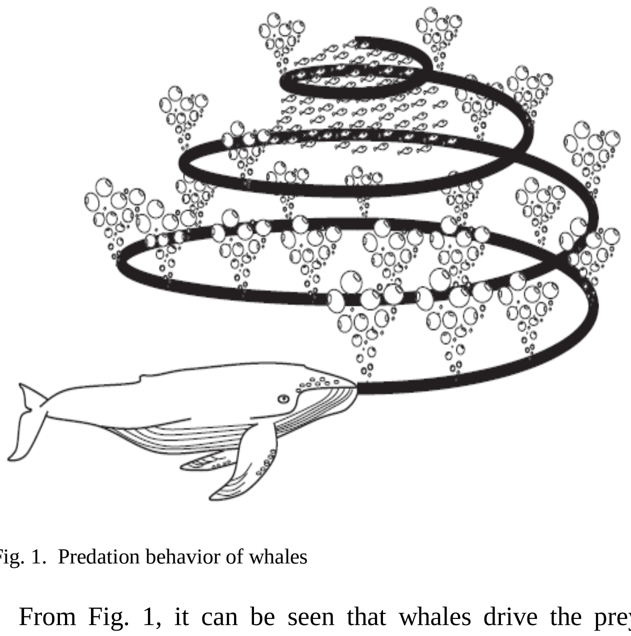


Fig. 1. Predation behavior of whales

From Fig. 1, it can be seen that whales drive the prey upward from the bottom with spiral spitting, thereby forcing the prey to the surface of the spiral center where it will eventually be swallowed. The above behavior can be expressed as

$$\boldsymbol{D}^{'} = \left|\boldsymbol{X}^*(t) - \boldsymbol{X}(t)\right| \tag{8}$$

$$\boldsymbol{X}(t+1) = \boldsymbol{D}^{'} \cdot e^{bl} \cdot \cos(2\pi l) + \boldsymbol{X}^*(t) \tag{9}$$

where $\boldsymbol{D}'$ indicates the distance from the current individual to the best individual, b is a constant for defining the shape of the logarithmic spiral, l represents a variable that varies randomly in the range of [–1,1], $\boldsymbol{X}^*(t)$ indicates the location of current best individual, and $\boldsymbol{X}(t+1)$ represents the next position of the search agent.

As the whale's bubble-net attacks the prey while encircling the prey, we can assume that there is a 50% probability of choosing whether to encircle or attack the prey. This behavior of whales can be expressed as

$$\boldsymbol{X}(t+1)=\begin{cases}\boldsymbol{D}'\cdot e^{bl}\cdot\cos(2\pi l)+\boldsymbol{X}^*(t) & \text{if } p\geq 0.5\\ \boldsymbol{X}^*(t)-\boldsymbol{A}\cdot\boldsymbol{D} & \text{if } p<0.5\end{cases} \tag{10}$$

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

D. Limitations of WOA

In WOA, the linearly reduced random vector $\boldsymbol{A}$ is used to balance the capabilities of global search and local search. In the first half of the iterations, $\boldsymbol{A}$ tends to meet $|\boldsymbol{A}|\geq 1$, and WOA has strong global search ability. In the latter half of the iterations, $\boldsymbol{A}$ tends to meet $|\boldsymbol{A}|<1$, and the algorithm has strong local search ability. Meanwhile, it can be known from (10) that the search agent only updates its position according to the elite individual, so that the diversity of the population drops rapidly, and the possibility of falling into the local minimum increases. Moreover, once WOA falls into the local optimum, it lacks an effective mechanism to jump out the local peak.

III. CDEWOA

The population-based optimization algorithm primarily consists of two processes: exploration and exploitation. A better balance between exploration and exploitation will improve the performance of WOA. By improving convergence speed of WOA and preventing it from falling into local optimum for WOA, we have made corresponding improvements in three areas: uniform population initialization strategy, complementary differential evolution strategy, and jumping out local optimum strategy.

A. Uniform Population Initialization Strategy

As a uniform population initialization has a positive effect on solving optimization problems, the chaotic mapping function has been used in population initialization [19]. Although chaotic mapping can enhance the initial diversity, it requires a certain number of iterative operations, and it is difficult for the iterative result to guarantee the uniformity distribution. Therefore, this paper introduces a novel population initialization strategy.

Assuming that the number of search agents is N and the search space is M dimensions, the uniform initialization process can be described as follows:

First, we uniformly take N points in each dimension of search space as a candidate solution set for the initialization position. Then, the jth sample in the ith dimension search space can be expressed as

$$X_i^j(0)=X_i^L+(j-0.5)(X_i^U-X_i^L)/N \tag{11}$$

where $i=1,2,\cdots,M$, $j=1,2,\cdots,N$, $X_i^j(0)$ presents the jth uniform sample of the ith dimension variable, and X_i^U and X_i^L are the upper bound and lower bound, respectively, for the jth dimension variable.

Second, the initialization candidate solution set is randomly permuted, and can be written as

$$\bar{\boldsymbol{X}}_i(0)=\text{randperm}(\boldsymbol{X}_i(0)) \tag{12}$$

where $\boldsymbol{X}_i(0)=[X_i^1(0),X_i^2(0),\cdots,X_i^N(0)]$.

Lastly, we can obtain the uniform initialization search agent by

$$\boldsymbol{X}^j(0)=[\bar{X}_1^j(0),\bar{X}_2^j(0),\cdots,\bar{X}_M^j(0)]^{\mathrm{T}} \tag{13}$$

where $\boldsymbol{X}^j(0)$ is the jth initialization individual.

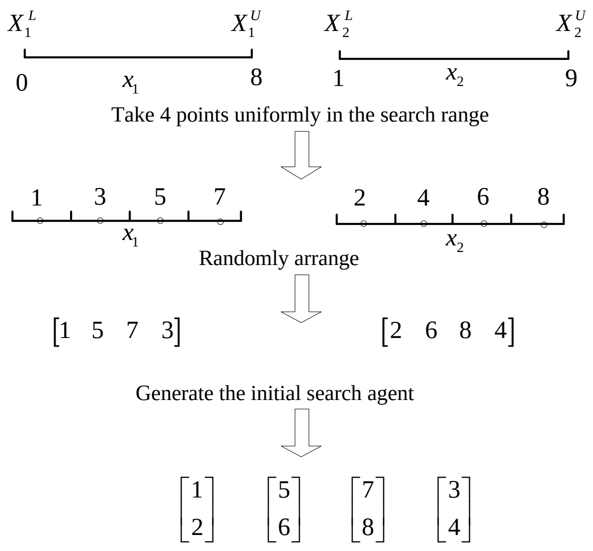


Fig. 2. Example of uniform population initialization.

Fig. 2 provides an example of taking four samples uniformly from a two-dimensional (2D) search space, where the range of the first-dimension search space is [0, 8] and the range of the second-dimension search space is [1, 9]. According to (11), we can obtain the uniform sample as

$$\boldsymbol{X}_1(0)=[1,3,5,7] \tag{14}$$

$$\boldsymbol{X}_2(0)=[2,4,6,8] \tag{15}$$

Then, randomly arranging $\boldsymbol{X}_1(0)$ and $\boldsymbol{X}_2(0)$ obtains

$$\bar{\boldsymbol{X}}_1(0)=[1,5,7,3] \tag{16}$$

$$\bar{\boldsymbol{X}}_2(0)=[2,6,8,4] \tag{17}$$

Finally, we take the initial search agent as

$$\boldsymbol{X}(0) = [X^1(0), X^2(0), X^3(0), X^4(0)] = \begin{bmatrix} 1 & 5 & 7 & 3 \\ 2 & 6 & 8 & 4 \end{bmatrix} \quad (18)$$

As the candidate solution set for the initialization position comes from uniform sampling for each dimension of input space, this "relatively uniform" initialization permits the initial search agents to be more evenly distributed throughout the search space. As a result, it can increase the initial diversity of the population and obtain better exploration ability.

B. Complementary Differential Evolution Strategy

Motivation

"Learning from each other" is a simple and effective way to improve the performance of a metaheuristic algorithm. The differential evolution (DE) algorithm is an efficient global optimization algorithm. DE is highly able to perform global searching in the early stage, and avoids falling into local optimum. This ability can remedy the problem of premature convergence in WOA. Therefore, an improved DE algorithm based on a complementary DE strategy is employed to alleviate the problem of premature convergence in WOA.

Principle of Complementary Differential Evolution

Like the genetic algorithm, the DE algorithm uses mutation, crossover and selection to find the optimum of functions. According to different mutation methods, DE can be divided into

DE/rand/1

$$\boldsymbol{H}^{rand}(t) = \boldsymbol{X}^{r1}(t) + F\cdot(\boldsymbol{X}^{r2}(t) - \boldsymbol{X}^{r3}(t)) \quad (19)$$

DE/best/1/

$$\boldsymbol{H}^{best}(t) = \boldsymbol{X}^{*}(t) + F\cdot(\boldsymbol{X}^{r1}(t) - \boldsymbol{X}^{r2}(t)) \quad (20)$$

DE/current-to-best/1

$$\begin{aligned}\boldsymbol{H}^{current}(t) &= \boldsymbol{X}(t) + F\cdot(\boldsymbol{X}^{*}(t) - \boldsymbol{X}(t)) \\ &+ F\cdot(\boldsymbol{X}^{r1}(t) - \boldsymbol{X}^{r2}(t))\end{aligned} \quad (21)$$

where $\boldsymbol{X}^{r1}(t)$, $\boldsymbol{X}^{r2}(t)$, and $\boldsymbol{X}^{r3}(t)$ are three individuals randomly selected from the population, $r1 \neq r2 \neq r3$, $F \in (0,1)$ represents the scaling factor used to control the influence of differential vectors, and $\boldsymbol{X}^{*}(t)$ represents the position of the current best individual.

The different mutation strategies have different effects on the performance of DE when solving global optimization problems. DE/best/1 and DE/current-to-best/1 have faster convergence speed and better performance when solving single-peak problems. The ability of local search is stronger than global search, so it is easy to fall into local optimum and lead to precocity when solving multi-peak problems. In contrast, although DE/rand/1 has a slow convergence rate, its strong global search ability can effectively prevent premature convergence. To balance exploration and exploitation, we combine the DE/best/1 and DE/rand/1 through a weight factor to form a novel donor vector, which is called the complementary differential mutation. That is,

$$\boldsymbol{H}(t) = w\cdot\boldsymbol{H}^{rand}(t) + (1-w)\boldsymbol{H}^{best}(t) \quad (22)$$

$$w = (t\text{-}1)/T_{\max} \quad (23)$$

where $T_{\max}$ indicates the maximum number of iterations.

After generating the donor vector, introducing the crossover operator can enhance the diversity of population. The crossover operation can be written as

$$V_j(t) = \begin{cases} H_j(t) & \text{if } rand(0,1) \le CR \text{ or } j = \mathrm{rnbr}(i) \\ X_j(t) & \text{if } rand(0,1) > CR \text{ and } j \neq \mathrm{rnbr}(i) \end{cases} \quad (24)$$

where $H_j(t)$, $X_j(t)$, and $V_j(t)$ are the jth element of the donor vector $\boldsymbol{H}(t)$, the current individual $\boldsymbol{X}(t)$, and the trail vector $\boldsymbol{V}(t)$, respectively; $CR \in [0,1]$ is the crossover rate, and $\mathrm{rnbr}(i) \in [1, 2, \cdots, D]$ is a randomly chosen index that ensures that $\boldsymbol{V}(t)$ obtains at least one parameter from $\boldsymbol{H}(t)$.

Decide which of $\boldsymbol{X}(t)$ and $\boldsymbol{V}(t)$ to enter the next generation, according to the value of the fitness. That is,

$$\boldsymbol{X}^{DE}(t+1) = \begin{cases} \boldsymbol{V}(t) & \text{if } f(\boldsymbol{V}(t) > f(X(t) \\ \boldsymbol{X}(t) & \text{else} \end{cases} \quad (25)$$

Pseudocode of Complementary Differential Evolution Strategy

The pseudocode of complementary differential evolution strategy is shown in Fig. 3.

```
Algorithm 1 Complementary differential evolution strategy
Initialize scaling factor $F$, crossover rate $CR$
Calculate complementary mutation factor $w$ by (23)
Generate a donor vector by (22)
Use crossover operator to generate the trail vector by (24)
Use greedy selection to generate current search agent by (25)
```

Fig. 3. Framework of CDE strategy.

Improved Attacking Strategy

So far, we have described three ways to update the whale's position: encircling the prey, bubble-net attacking of the prey, and CDE strategy. To take full advantage of each update method, we assign equal probability to these update methods. That is,

$$\boldsymbol{X}(t+1) = \begin{cases} \boldsymbol{X}^{DE} & \text{if } p \ge 0.66 \\ \boldsymbol{D}^{'}\cdot e^{bl}\cdot\cos(2\pi l) + \boldsymbol{X}^{*}(t) & \text{if } 0.33 \le p < 0.66 \\ \boldsymbol{X}^{*}(t) - \boldsymbol{A}\cdot\boldsymbol{D} & \text{if } p < 0.33 \end{cases} \quad (26)$$

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

C. Jumping out Local Optimum Strategy

Using the CDE strategy can prevent WOA from falling into a local optimum to a certain extent, but once WOA is trapped in the local optimum, it is still difficult to escape from the local peak. Therefore, it is necessary to introduce a mechanism to detect whether the algorithm is trapped in the local optimum. Once we find WOA to fall into the local peak, the corresponding escaping strategy is performed immediately to recover the global search capability of WOA.

The strategy for detecting to fall into local peak can be described as

$$Is\_local = \begin{cases} \text{true} & \text{if } \sum_{u=t-s}^{t} \Delta f^*(u) \le \delta \\ \text{false} & \text{else} \end{cases} \tag{27}$$

Algorithm 2 CDEWOA

```
Use uniform initialization strategy to initialize whale population
Initialize scaling factor $F$, and crossover rate $CR$
Calculate the fitness of each search agent, and update the best agent $\boldsymbol{X}^*$
while (t < Maximum iterations)
  for each search agent
    Update $a$, $\boldsymbol{A}$, $\boldsymbol{C}$, $l$, and $p = rand()$
    if $p < 0.33$
      if $|\boldsymbol{A}| \ge 1$
        Update the position of current search agent by (2)
      else if $|\boldsymbol{A}| < 1$
        Update the position of current search agent by (7)
      end if
    else if $0.33 \le p < 0.66$
      Update the position of current search agent by (9)
    else if $p \ge 0.66$
      Update the position of current search agent by (25)
    end if
    Calculate $\Delta_S = \sum_{u=t-S}^{t} \Delta f^*(u)$
    if $\Delta_S \le \delta$
      Update the position of current search agent by Eq. (28)
    end if
   Check if the position of the current search agent goes beyond search space and amend it
   Calculate the fitness of each search agent, and update the best agent $\boldsymbol{X}^*$
  end for
  $t = t + 1$
end while
return $\boldsymbol{X}^*$
```

Fig. 4. Framework of CDEWOA.

where $\Delta f^*(u) = f\left(\boldsymbol{X}^*(u)\right) - f\left(\boldsymbol{X}^*(u-1)\right)$ is the variable quantity of optimum at the uth iteration, $\boldsymbol{X}^*(u)$ is the optimum at the uth iteration, and δ is a constant. When the accumulation of variable quantity of optimum in successive s generations is less than δ, WOA is considered to fall into the local optimum, and a jumping out strategy is executed. In this paper, the DE/current-to-best/1 mutation strategy is used to jump out the local optimum. This process can be expressed as

$$\boldsymbol{X}(t+1) = \begin{cases} \boldsymbol{X}(t) & \text{if } Is\_local = \text{false} \\ \boldsymbol{X}(t) + F \cdot (\boldsymbol{X}^*(t) - \boldsymbol{X}(t)) & \text{else} \\ \quad + F \cdot (\boldsymbol{X}_{r1}(t) - \boldsymbol{X}_{r2}(t)) & \end{cases} \tag{28}$$

D. Pseudocode of CDEWOA

The pseudocode of CDEWOA is shown in Fig. 4.

IV. Simulation and Analysis

To prove the effectiveness of the CDEWOA proposed in this paper, we tested its performance using 14 benchmark functions. Table I shows the unimodal benchmark functions F1-F6, which can test the search ability and the convergence rate of the metaheuristic algorithm.

TABLE I
DESCRIPTION OF UNIMODAL BENCHMARK FUNCTIONS

Functions	V_no	Range	F$_{min}$
$F_1(x) = \sum_{i=1}^{n} x_i^2$	30	[–100,100]	0
$F_2(x) = \sum_{i=1}^{n} \lvert x_i \rvert + \prod_{i=1}^{n} \lvert x_i \rvert$	30	[–10,10]	0
$F_3(x) = \sum_{i=1}^{n} (\sum_{j-1}^{i} x_j)^2$	30	[–100,100]	0
$F_4(x) = \max_i \{\lvert x_i \rvert, 1 \le i \le n\}$	30	[–100,100]	0
$F_5(x) = \sum_{i=1}^{n-1} \left[100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2\right]$	30	[–30,30]	0
$F_6(x) = \sum_{i=1}^{n} ([x_i + 0.5])^2$	30	[–100,100]	0

Table II shows functions F7–F11, which are multimodal benchmark functions. As these functions own many local optimums, they are better for testing the ability of escaping from local optimums.

TABLE II
DESCRIPTION OF MULTIMODAL BENCHMARK FUNCTIONS

Functions	V_no	Range	F$_{min}$
$F_7(x) = \sum_{i=1}^{n} \left[x_i^2 - 10\cos(2\pi x_i) + 10\right]$	30	[–5.12,5.12]	0
$F_8(x) = -20\exp(-20\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2}) - \exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)) + 20 + e$	30	[–32,32]	0
$F_9(x) = \frac{1}{4000}\sum_{i=1}^{n} x_i^2 - \prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}}) + 1$	30	[–600,600]	0
$F_{10}(x) = \frac{\pi}{n}\left\{10\sin(\pi y1) + \sum_{i=1}^{n-1}(y_i - 1)^2\left[1 + 10\sin^2(\pi y_i + 1)\right] + (y_n - 1)^2\right\} + \sum_{i=1}^{n} u(x_i, 10, 100, 4)$, $y_i = 1 + \frac{x_i + 1}{4}$, $u(x_i, a, k, m) = \begin{cases} k(x_i - a)m, x_i > a \\ 0, -a < x_i < a \\ k(-x_i - a)m, x_i < -a \end{cases}$	30	[–50,50]	0
$F_{11}(x) = 0.1\left\{\sin^2 \sum_{i=1}^{n}(x_i - 1)^2\left[1 + \sin^2(3\pi x_i + 1)\right] + (x_n - 1)^2\left[1 + \sin^2(2\pi x_n)\right]\right\} + \sum_{i=1}^{n} u(x_i, 5, 100, 4)$	30	[–100,100]	0

TABLE III
DESCRIPTION OF FIXED-DIMENSION MULTIMODAL BENCHMARK FUNCTIONS

Functions	V_no	Range	Fmin
$F_{12}(x)=(\frac{1}{500}+\sum_{j=1}^{25}\frac{1}{j+\sum_{j=1}^{2}(x_i-a_{ij})^6})^{-1}$ where $\boldsymbol{a}=\begin{bmatrix}-32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32 & -32 & -16 & 0 & 16 & 32\\ -32 & -32 & -32 & -32 & -32 & -16 & -16 & -16 & -16 & -16 & 0 & 0 & 0 & 0 & 0 & 16 & 16 & 16 & 16 & 16 & 32 & 32 & 32 & 32 & 32\end{bmatrix}$	2	[–65,65]	1
$F_{13}(x)=\sum_{i=1}^{11}\left[a_i-\frac{x_1(b_i^2+b_ix_2)}{b_i^2+b_ix_3+x_4}\right]^2$, $\boldsymbol{b}=[4,2,1,1/2,1/4,1/6,1/8,1/10,1/12,1/14,1/16]$, $\boldsymbol{a}=[[0.1957, 0.1947, 0.1735, 0.16, 0.0844, 0.0627, 0.0456, 0.0342, 0.0323, 0.0235, 0.0246]]$	4	[–5,5]	0.00030
$F_{14}(x)=\left[1+(x_1+x_2+1)^2(19-14x_1+3x_1^2-14x_2+6x_1x_2+3x_2^2)\right]\times\left[30+(2x_1-3x_2)^2\times(18-32x_1+12x_1^2+48x_2-36x_1x_2+27x_2^2)\right]$	2	[–2,2]	3

The difference between the fixed dimensional multimodal function and the multimodal function is the ability to define the desired number of design variables. Table III shows functions F12–F14, which are fixed dimensional multimodal functions. Although the fixed dimensional multimodal function does not allow us to modify the number of design variables, it provides a different search space than the multimodal function.

We compared the performance of CDEWOA with that of IWOA [18], WOA [10], CDE [20], DE [2], and PSO [5] based on the 14 benchmark functions. Each algorithm was run 50 times independently. The general control parameters of algorithms such as the maximum number of iterations ($T_{\max}$) and the population size (M), were set to the same value, that is, $T_{\max}=500$ and $M=30$. The simulation environment was MATLAB R2016a running on a 64-bit Windows 10 computer with 8 GB of RAM. The other control parameters are shown in Table IV.

TABLE IV
PARAMETER SETTINGS OF THE ALGORITHM

Algorithm	Parameter setting
PSO	$w(t)=w_{\max}-t\cdot(w_{\max}-w_{\min})/T_{\max}$, $w_{\max}=0.9$, $w_{\min}=0.2$, $c_1=c_2=2$
DE	$F_0=0.5$, $CR=0.9$, $F=F_0\cdot 2^{\lambda}$, $\lambda=\exp(1-T_{\max}/(T_{\max}+1-t))$
CDE	$F_0=0.5$, $CR=0.9$, $F=F_0\cdot 2^{\lambda}$, $\lambda=\exp(1-T_{\max}/(T_{\max}+1-t))$, $w=1-0.8\cdot t/T_{\max}$
WOA	$a=2(1-t/T_{\max})$, $b=1$, $l=(a_2-1)*rand+1$, $a_2=-1-t/T_{\max}$
IWOA	$a=2(1-t/T_{\max})$, $b=1$, $l=(a_2-1)\cdot rand+1$, $a_2=-1-t/T_{\max}$, $F=[0.2,0.8]$, $CR=0.9$
CDEWOA	$a=2(1-t/T_{\max})$, $b=1$, $l=(a_2-1)\cdot rand+1$, $a_2=-1-t/T_{\max}$, $F=0.5$, $CR=0.9$, $w=(t-1)/T_{\max}$, $\delta=0.1$

Tables V and VI display the simulation results obtained by each algorithm, where *Ave* and *Std* represent the average and standard deviation of the best fitness, respectively, and *Best* represents the best optimal fitness.

As can be seen from Table V, CDEWOA is very competitive in optimizing the unimodal function, compared with the IWOA, WOA, CDE, DE, and PSO. It is the best optimizer for functions F1, F2, F5 and F6. Although its performance in functions F3 and F4 is slightly lower than that of DE and CDE, CDEWOA still can obtain relatively ideal results. These results also confirm that CDEWOA has better exploitation capability.

From Table VI, we can also see that CDEWOA shows excellent optimization performance for optimizing the multimodal functions and fixed dimensional multimodal functions. CDEWOA obtains the best average optimal value on functions F7, F9, F10, F11, F12, F13, and F14. At the same time, except for function F12, the best optimal values obtained by CDEWOA are better than those obtained by the IWOA, WOA, CDE, DE, and PSO. As the multi-peak functions are mainly used to detect the global search capability of the metaheuristic algorithm, it can be seen that, by introducing complementary difference evolution strategy and jumping out local optimal optimum strategy, the global search capability of the CDEWOA algorithm can be improved and the diversity of the population can be increased.

The convergence curves of CDEWOA, IWOA, WOA, CDE, DE and PSO for 14 benchmark functions are shown in Fig. 5. The CDEWOA algorithm shows three different convergence behaviors.

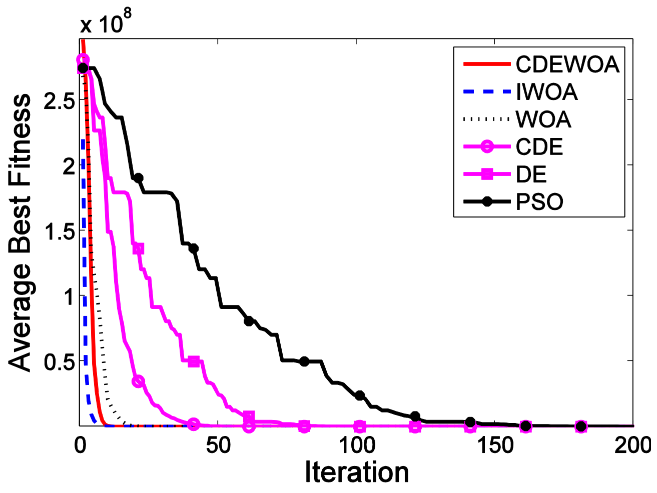
First, compared with the DE, CDE, and PSO, the convergence speed and convergence accuracy of the three WOA-based algorithms (WOA, IWOA, and CDEWOA) are relatively high. This is mainly because at the initial stage of iterations, these three algorithms complete the exploration by moving around the prey positions randomly selected. However, in the later stage of iterations, the WOA-based algorithms adopt circular encircling or spiral attacking strategies to enhance the exploitation of algorithms. Put another way, the CDE, DE, and PSO use the same equation to update the position of search agents, which increases the possibility of falling into a local minimum; however, the WOA-based algorithms separately consider the exploration and exploitation of algorithms and use the different update strategies, both of which help them improve their convergence performance.

TABLE V
SIMULATION RESULTS FOR F1-F6

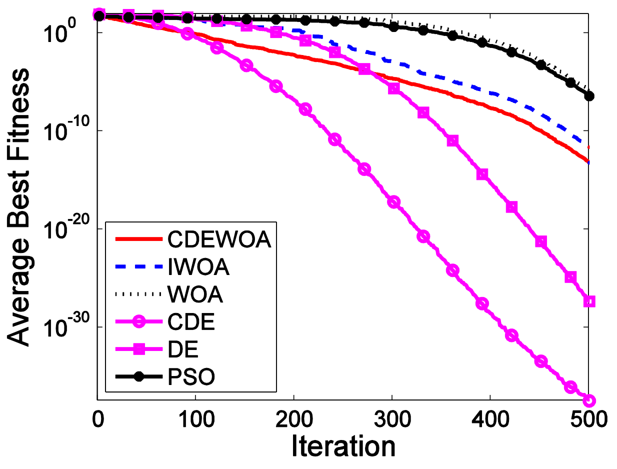
Algorithms		F1	F2	F3	F4	F5	F6
CDEWOA	Ave	**3.14E-85**	**6.69E-50**	2.40E-05	1.71E-13	**1.56E+01**	**0**
	Std	**2.01E-85**	**3.70E-49**	6.53E-06	9.74E-13	**3.09E+00**	**0**
	Best	**1.28E-90**	**6.66E-63**	2.92E-07	5.56E-25	**1.21E+00**	**0**
IWOA	Ave	1.60E-78	3.09E-48	5.16E-05	3.90E-10	2.76E+01	**0**
	Std	6.99E-78	1.88E-48	4.97E-06	6.81E-11	1.96E+01	**0**
	Best	9.47E-84	4.01E-51	5.80E-06	6.35E-13	2.07E+01	**0**
WOA	Ave	2.43E-73	4.39E-49	4.68E-04	4.46E-08	2.80E+01	**0**
	Std	1.65E-72	2.96E-48	1.68E-04	2.65E-08	4.70E+00	**0**
	Best	5.41E-85	5.25E-59	1.15E-04	4.02E-12	2.70E+01	**0**
CDE	Ave	1.34E-84	3.05E-45	**2.58E-56**	**1.28E-37**	2.82E+01	**0**
	Std	7.60E-84	4.73E-45	**1.82E-55**	**3.01E-37**	3.82E+00	**0**
	Best	8.64E-90	2.32E-47	**1.11E-66**	**5.06E-40**	2.72E+01	**0**
DE	Ave	6.39E-66	1.27E-36	6.68E-35	2.49E-28	2.89E+01	**0**
	Std	1.38E-65	1.99E-37	3.32E-34	3.34E-28	6.01E+00	**0**
	Best	8.81E-69	7.51E-39	6.39E-41	4.48E-30	2.87E+01	**0**
PSO	Ave	2.86E-23	2.48E+00	4.59E-05	1.98E-08	5.15E+01	1.95E+01
	Std	9.52E-22	5.37E+00	1.83E-05	3.81E-09	2.95E+01	2.26E+00
	Best	9.38E-22	1.28E+00	9.78E-06	1.21E-13	6.91E+01	1.31E+00

TABLE VI
SIMULATION RESULTS FOR F7-F14

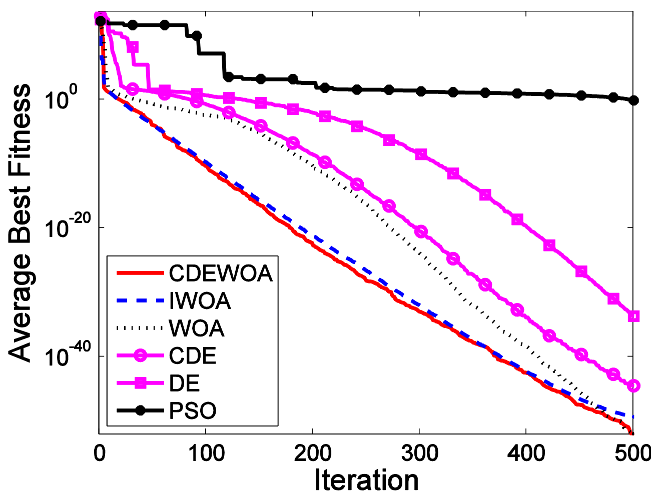
Algorithms		F7	F8	F9	F10	F11	F12	F13	F14
CDEWOA	Ave	**5.02E-20**	2.17E-15	**0**	**6.43E-06**	**3.02E-01**	**1.59E+00**	**2.96E-04**	**3.00E+00**
	Std	**7.58E-21**	1.77E-15	**0**	**5.95E-07**	3.39E-02	**1.98E-01**	**1.69E-05**	**2.28E-15**
	Best	**1.62E-21**	**8.88E-16**	**0**	**7.31E-08**	**9.98E-02**	1.39E+00	**3.00E-04**	**3.00E+00**
IWOA	Ave	7.65E-06	1.20E-14	2.31E-03	6.66E-03	7.40E-01	2.58E+00	7.86E-04	8.28E+00
	Std	1.31E-06	5.02E-15	6.67E-04	2.46E-03	4.70E-01	3.32 E-01	1.45E-04	3.78E+00
	Best	2.69E-07	4.44E-15	**0**	1.09E-03	**9.98E-02**	**1.07E+00**	2.99E-04	**3.00E+00**
WOA	Ave	3.91E-03	4.51E-15	**0**	3.24E-02	3.09 E-01	6.08E+00	6.46E-04	**3.00E+00**
	Std	2.59E-04	2.54E-15	**0**	5.13E-03	**2.29 E-02**	4.72E-01	2.46E-04	7.62E-05
	Best	6.04E-04	**8.88E-16**	**0**	1.95E-03	**9.98E-02**	3.09E+00	**3.00E-04**	**3.00E+00**
CDE	Ave	2.15E-03	**1.88E-15**	**0**	6.96E-03	3.95 E+00	5.91E+00	3.08E-04	**3.00E+00**
	Std	1.33E-04	**1.61E-15**	**0**	5.67E-03	3.39 E+00	1.11E+00	8.52E-05	3.82E-10
	Best	2.96E-06	**8.88E-16**	**0**	4.61E-04	**9.98E-02**	3.07E+00	**3.00E-04**	**3.00E+00**
DE	Ave	5.94E+00	4.01E-15	**0**	9.88E-01	5.01E+00	6.02E+00	5.60E-03	8.69E+00
	Std	0.36E+00	2.17E-15	**0**	1.65E-01	2.43 E+00	5.64E-01	5.96E-03	3.62E+00
	Best	4.86E+00	**8.88E-16**	**0**	6.37E-02	1.00 E+00	4.80E+00	2.77E-04	**3.00E+00**
PSO	Ave	2.84E-05	1.14E-03	2.82E-03	6.75E-03	1.95 E+00	8.02E+00	9.47E-04	**3.00E+00**
	Std	9.67E-06	9.50E-04	8.16E-04	1.49E-03	1.97 E+00	1.14 E+00	2.85E-04	6.49E-15
	Best	1.08E-06	8.84E-05	**0**	8.03E-04	**9.98 E-02**	3.07E+00	2.96E-04	**3.00E+00**



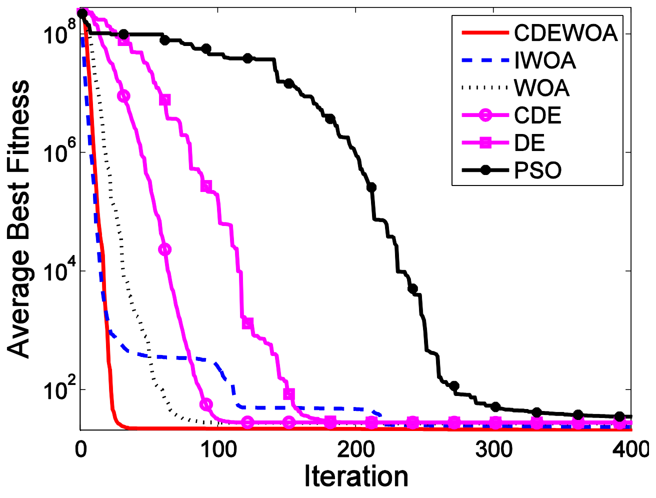
(a) F1



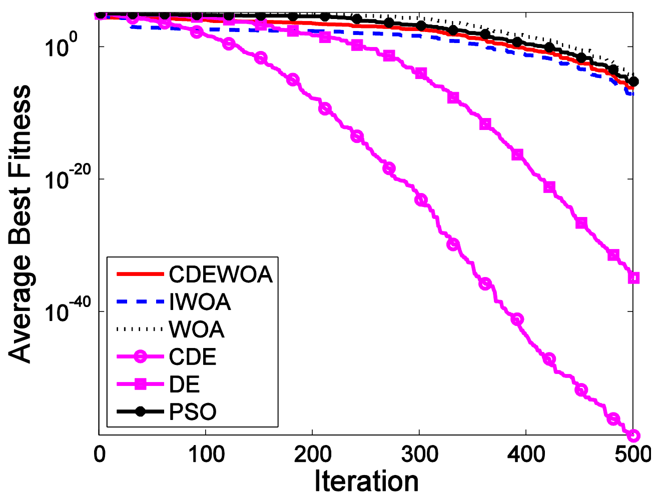
(d) F4



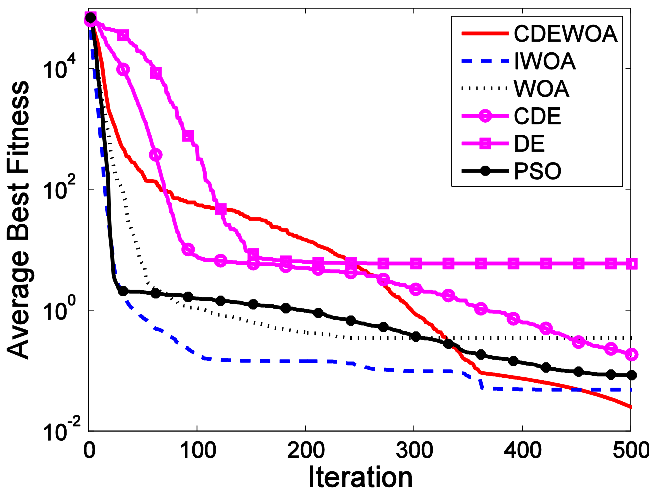
(b) F2



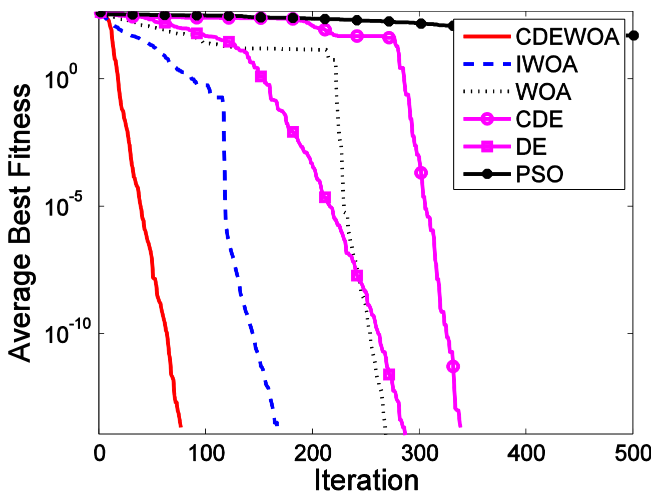
(e) F5



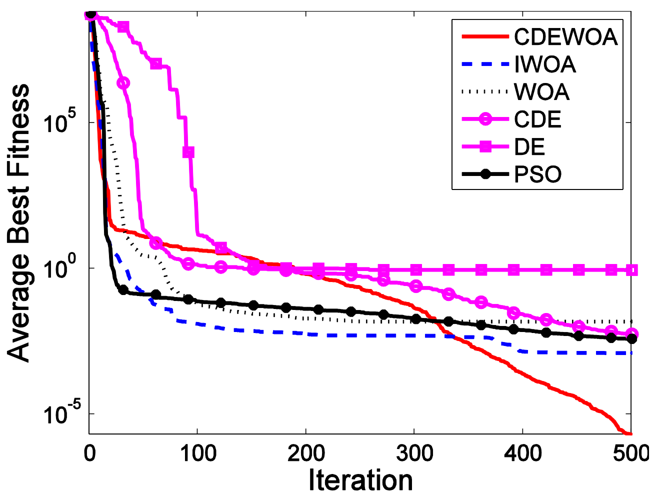
(c) F3



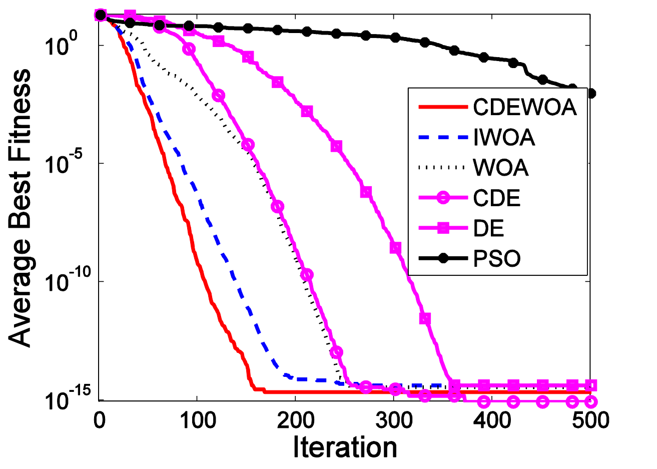
(f) F6



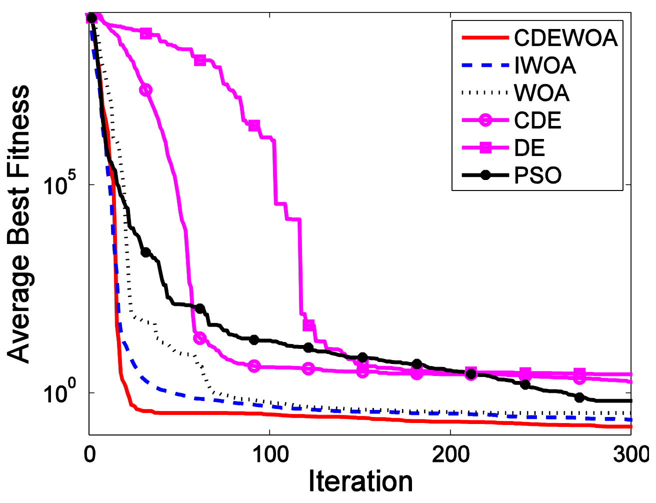
(g) F7



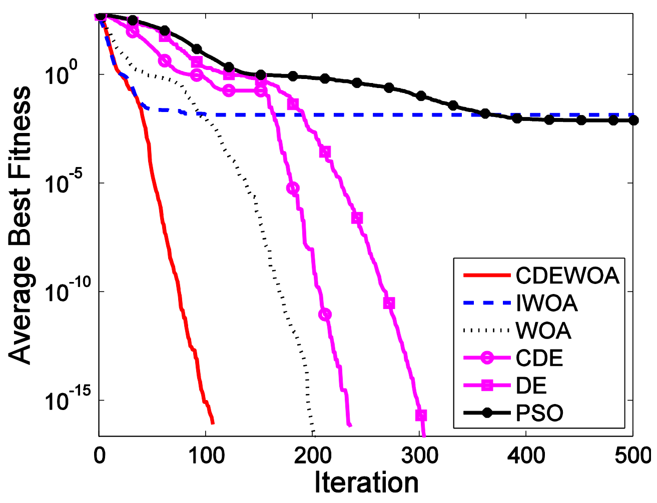
(j) F10



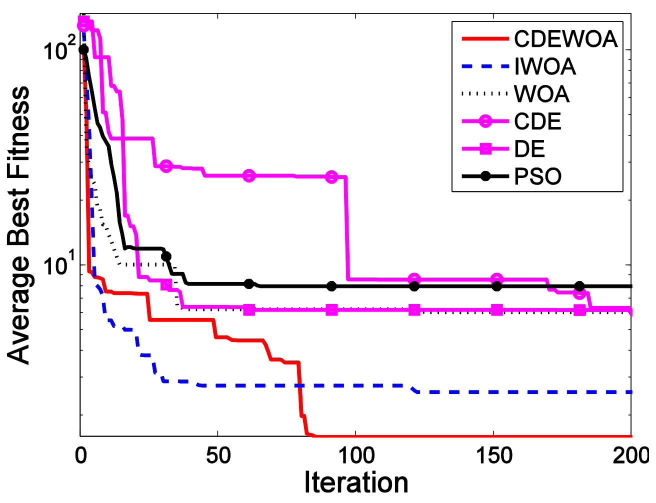
(h) F8



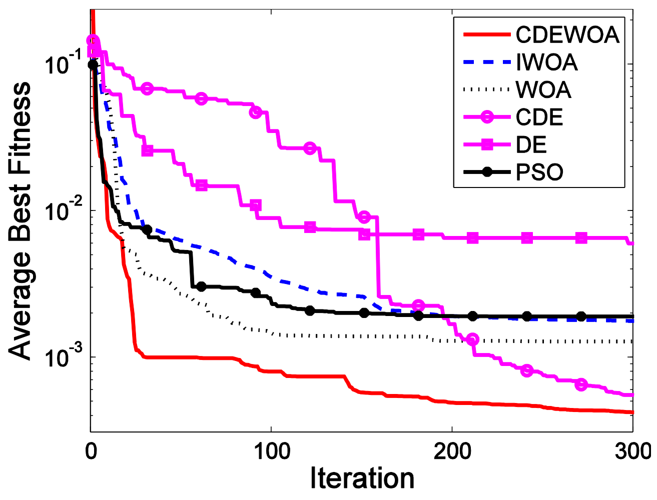
(k) F11



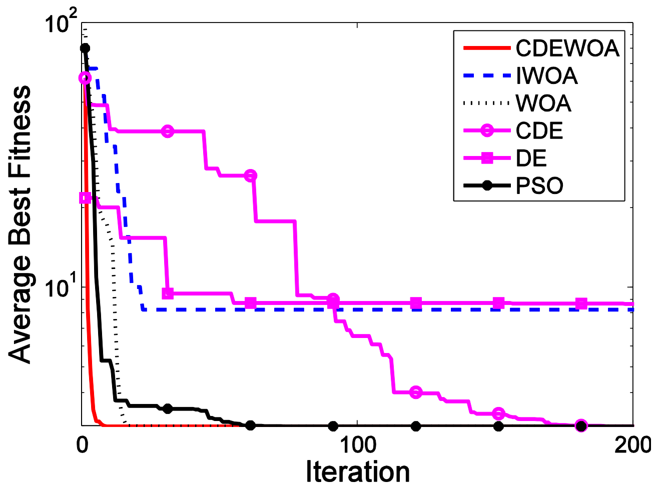
(i) F9



(l) F12



(m) F13



(n) F14

Fig. 5. Convergence curves of functions F1-F14

Second, in the exploration phase, the convergence speed of CDEWOA is faster than that of IWOA and WOA, especially for functions F4, F5, F7, F8, F9, F11, F13 and F14. On the one hand, it is because the uniform initialization strategy used in CDEWOA improves the probability of finding a good solution in the exploration stage. On the other hand, during the exploration phase, CDEWOA extended the complementary differential evolution strategy to circular encircling and spiral attacking prey. The combination of multiple search methods is also conducive to finding more effective solutions.

Finally, in the exploitation phase, the convergence accuracy of the CDEWOA algorithm tends to increase as the iteration increases. This behavior is evident in functions F2, F3, F4, F6, F7, F9, F10, F11, and F13. This is mainly due to the jumping out local optimal optimum strategy proposed for CDEWOA that enables it to escape the local extreme value and increases its local search ability.

V. Conclusion

Uniform rather than random initialization enhances the initial diversity of the population to some extent. A complementary differential mutation operator can help CDEWOA to maintain a good diversity of populations in the search process. The strategy of jumping out local optimum decreases the probability of falling into precocity when solving multi-peak problems. Simulation results show that the performance of CDEWOA proposed in this paper is better than that of other algorithms, such as IWOA, WOA, CDE, DE, and PSO.

References

[1] M. Srinivas, and L. M. Patnaik, "Genetic algorithms: a survey," *Computer*, vol.27, no.6, pp.17-26, 2002.

[2] R. Storn, and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341-359, 1997.

[3] S. Kirkpatrick, C. D. Gelatt, and M.P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

[4] T. A. Central, "Force optimization: a new nature inspired computational framework for multidimensional search and optimization", *Natural Inspired Cooperative Strategies for Optimization*, vol. 129, pp.221-238, 2008.

[5] R. Poli, J. Kennedy, and T. Blackwell, "Particle swarm optimization an overview," *Swarm Intelligence*, no.1, pp.33-57, 2007.

[6] M. Dorigo, and G. D. Caro, "Ant colony optimization: a new meta-heuristic," *IEEE in Proceedings of the 1999 Congress on Evolutionary Computation*,1999.

[7] J. S. Wang, and J. D. Song, "A hybrid algorithm based on gravitational search and particle swarm optimization algorithm to solve function optimization problems," *Engineering Letters*, vol. 25, no. 1, pp. 22-29, 2017.

[8] X. X. Ma, and J. S. Wang, "An improved flower pollination algorithm to solve function optimization problem," *IAENG International Journal of Computer Science*, vol. 45, no. 3, pp. 364-370, 2018.

[9] Y. F. Sun, J. S. Wang, and X. X. Ma, "An improved krill herd algorithm based on meme grouping strategy of shuffled frog leaping algorithm," *IAENG International Journal of Computer Science*, vol. 46, no. 2, pp. 156-162, 2019.

[10] S. Mirjalili, and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51-67, 2016.

[11] N. Thakur, Y. K. Awasthi, M. Hooda, and A. S. Siddiqui, " Adaptive whale optimization for intelligent multi-constraints power quality improvement under deregulated environment," *Journal of Engineering Design and Technology*, vol. 17, no.3, pp. 490-514, 2019.

[12] Y. J. Sun, X. Wang, Y. Chen, and Z. J. Liu, "A modified whale optimization algorithm for large-scale global optimization problems," *Expert Systems with Applications*. Vol. 114: pp. 563-577, 2018.

[13] G. Kaur, and S. Arora, "Chaotic whale optimization algorithm," *Journal of Computational Design and Engineering*, vol.5, no.3, pp. 275-284, 2018.

[14] A. Korashy, S. Kamel, F. Jurado, and A. R. Youssef, "Hybrid whale optimization algorithm and grey wolf optimizer algorithm for optimal coordination of direction overcurrent relays," *Electric Power Components and Systems*, Vol. 47, no. 6-7, pp.1-15, 2019.

[15] M. M. Mafarja, and S. Mirjalili, "Hybrid whale optimization algorithm with simulated annealing for feature selection," *Neurocomputing*, vol. 260, pp. 302-312, 2017.

[16] A. B. Mohamed, M. Gunasekaran, E. S. Doaa, and M. Seyedali, "A hybrid whale optimization algorithm based on local search strategy for the permutation flow shop scheduling problem," *Future Generation Computer Systems*, vol. 85, pp. 129-145, 2018.

[17] A. B. Mohamed, E. S. Doaa, and A.K. Sangaiah, "A modified nature inspired meta-heuristic whale optimization algorithm for solving 0-1 knapsack problem," *International Journal of Machine Learning and Cybernetics*, vol.10, no.3, pp. 495-514, 2019.

[18] M. B. Seyed, and Y. Samaneh, "IWOA: an improved whale optimization algorithm for optimization problems," *Journal of Computational Design and Engineering*, vol.6, no.3, pp.243-259, 2019.

[19] M. Abd Elaziz, and S. Mirjalili, "A hyper-heuristic for improving the initial population of whale optimization algorithm," *Knowledge based Systems*, vol. 172, pp. 42-63, 2019.

[20] X. Y. Ding, and S. H. Li, "An improved differential evolution algorithm," *Journal of Shanxi Normal University*, vol.44, no.1, pp.1-6, 2016.

Qiang Qu received the M. S. degree in control science and engineering from University of Science and Technology Liaoning, China in 2002, and the Ph. D. degree in signal and information processing from Dalian University of Technology, China in 2010. He is currently a professor and Master's supervisor in School of Electronic and Information Engineering, University of Science and Technology Liaoning. His main research interests include modeling of complex industry process, intelligent control and swarm intelligence.

Yi-han Huang received the B. S. degree in electronic information engineering and M. S. degree in control science and engineering from University of Science and Technology Liaoning, China in 2017, and 2020, respectively. Her main research interests include swarm intelligence and intelligent control.

Xiao-li Wang received the B. S. degree in computing Science from Shenyang Normal University, China in 1999. Her main research interests include swarm intelligence and intelligent control.

Xue-bo Chen received the M. S. degree from University of Science and Technology Liaoning, China in 1986, and received the Ph. D. degree in technical science from University of Belgrade, Yugoslavia in 1994. He is currently a professor and Ph. D. supervisor in University of Science and Technology Liaoning. His research interests include pattern recognition, large complex system and intelligent control.