IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received August 16, 2019, accepted September 6, 2019, date of publication September 10, 2019, date of current version September 24, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2940582

A Randomly Guided Firefly Algorithm Based on Elitist Strategy and Its Applications

CHUNFENG WANG[1] AND KUI LIU[2]
[1]College of Mathematics and Information Science, Henan Normal University, Xinxiang 453007, China
[2]Henan Engineering Laboratory for Big Data Statistical Analysis and Optimal Control, School of Mathematics and Information Sciences, College of Mathematics and Information Science, Henan Normal University, Xinxiang 453007, China

Corresponding author: Chunfeng Wang (wangchunfeng10@126.com)

This work was supported in part by the National Natural Science Foundation of China under Grant 11671122, in part by the Key Project of Henan Educational Committee under Grant 19A110021, and in part by the School Research Project under Grant 20180562.

ABSTRACT Firefly algorithm (FA) is one of the swarm intelligence algorithms, which is proposed by Yang in 2008. The standard FA has some disadvantages, such as high computational time complexity, slow convergence speed and so on. The main reason is that FA employs a full attracted model, which makes the oscillation of each firefly during its movement. To overcome these disadvantages, based on elitist strategy, a randomly guided firefly algorithm (ERaFA) is proposed. In this algorithm, for improving the convergence speed, an elitist attraction model is developed based on random selection from elite fireflies, which can lead the firefly to a right direction. To deal with the possible failure of the elite guidance, opposite learning strategy is adopted. Meanwhile, to strengthen the local search ability of our algorithm, and help our algorithm jump out a local optimum position, a new mechanism is proposed, which is similar to the crossover operator in GA. The performance of ERaFA is evaluated by some well-known test functions and applied to solve three constrained engineering problems. The results show that ERaFA is superior to FA and some other state-of-the-art algorithms in terms of the convergence speed and robustness.

INDEX TERMS Firefly algorithm, swarm intelligence, continuous optimization, elitist strategy, opposite learning.

I. INTRODUCTION

Since optimization problems often arise in engineering design, management science, economics and other fields, it is of great practical significance to present methods to solve these optimization problems. The method of solving optimization problems can be divided into deterministic methods and stochastic algorithms in general. The convergence of deterministic methods can usually be obtained, but these methods require the continuity, derivative and other information of the function, and can not get the global optimal solution in finite time. In contrast, stochastic algorithms do not require that functions are differentiable or continuous, so they can be used to solve a wide range of optimization problems. Swarm intelligence algorithms are a kind of stochastic algorithms, which can use the information sharing among groups to complete complex tasks, and attract the attention of researchers. Thus, more and more researchers pay attention to swarm intelligence algorithms, and presented many effective swarm intelligence algorithms, e.g. Particle Swarm Optimization(PSO) [1]–[3], Artificial Bee Colony (ABC) [4]–[6], Ant Colony Optimization(ACO) [7], [8], Differential Evolution (DE) [9], Harmony Search(HS) [10], Cuckoo Search(CS)[11], Genetic Algorithm (GA) [12], [13], Firefly Algorithm (FA) [14]–[16].

FA algorithm was first proposed by Yang in 2008 [14], which simulates the moving behavior of fireflies. Since FA was put forward, researchers have developed many variants of FA, and have applied them to solve many problems successfully appeared in many fields, including structure design [17], stock forecasting [18], and production scheduling [19], water resource [20] and cancer diagnosis [21], and so on. These variants of FA can be divided into the following categories.

■ Improvement based on modified strategy

In FA, parameters play a very important role, and how to adjust them is difficult. Two main mechanisms and five different strategies were proposed to adjust the control parameters in FA [22]. In order to improve the adaptability and overcome the shortcomings of FA, an adaptive firefly algorithm (AFA) was proposed in [23]. In this method, three strategies were presented. For solving continuous optimization problems, a modified MSA-FFA is developed based on the memetic

The associate editor coordinating the review of this manuscript and approving it for publication was Zhipeng Cai.

self-adaptive firefly algorithm (MSA-FFA) [24]. By selecting control parameters, the purpose of self-adaptation is achieved in this method. By studying the control parameters of FA, a modified FA called FA with adaptive control parameters (ApFA) was presented [25]. To jump out of the local optima and weaken the effects of the maximum iterations, a self-adaptive step firefly algorithm(SASFA) was developed [26]. The core idea of this method is to vary the step size with the number of iterations based on the information of individual and the current population. To improve the performance of FA, by dynamically adjusting the control parameters and employing an alternative search strategy, an adaptive FA with alternative search (AFAas) was proposed [27]. To stabilize the moving behavior of fireflies and increase convergence speed of FA, a new FA was proposed [28]. In this method, if there is no better fireflies in the vicinity of each firefly, a directed behavior that moves to the optimal solution of the current population is proposed. In addition, to increase convergence speed, it advise that all fireflies should move to global best in each iteration by using Gaussian distribution [28]. By using the Levy flights move strategy, a new meta-heuristic FA (LFA) was developed [29]. To escape from local minima, a modified FA was presented by combining FA and chaotic map, and applied to solve reliability-redundancy optimization [30]. To increase the global searching ability of FA, 12 different chaotic maps were introduced into FA (CFAs) [31]. Based on neighborhood search and dynamic parameter adjustment mechanism, a randomly attracted FA was proposed in 2017 [32]. By using Tidal Force formula, a modified firefly algorithm was proposed, which brought a new strategy into the optimization field [33].

■ Improvement based on hybrid strategy

By combing the advantages of FA and DE, a hybrid population-based algorithm, called hybrid firefly algorithm (HFA), was proposed in [34]. In this algorithm, to promote information sharing among the population, FA and DE are executed in parallel. For solving constrained numerical and engineering problems, a hybrid firefly algorithm was presented based on Rosenbrocks local search and Good-point-set method [35]. To strengthen the exploration and exploitation abilities of FA, a new FA variant (HMFA) was proposed. In this method, hybrid mutation strategies are employed [36]. Based on the combination of harmony search (HS) and firefly algorithm (FA), a hybrid approach, called HS/FA, was proposed [37]. This method utilized HS and FA to explore and exploit, respectively. Through combining FA with DE, a hybrid optimization method, named HEFA, was proposed [38], which can improve the searching precision and strengthen information sharing among the fireflies.

Although the aforementioned FA variants have a better performance than the classical FA, there is still room for improvement. For example, the time complexity of FA is relatively high, the reason is that each firefly $\boldsymbol{x}_i$ needs to be compared with all the other fireflies, and move to a firefly where its brightness is higher. Moreover, this movement may cause oscillations in the iteration. In addition, in the basic FA, it does not consider how to move $\boldsymbol{x}_i$ when $\boldsymbol{x}_i$ is better than the another firefly chosen to compare.

The aim of this paper is to propose an improved FA algorithm. The contributions of this paper are: (1) We assume that each firefly is guided by elitist firefly, which can reduce the time complexity, and improve the convergence rate. (2) To cope with the case that elitist firefly selected is worse than the firefly guided, the opposite learning strategy is adopted, which can help the corresponding firefly escape from a local position. (3) To enhance the local search ability of the proposed algorithm, a new mechanism, which is similar to the crossover operator in GA, is proposed.

The rest of the paper is organized as follows. FA algorithm is summarized in Section 2. In Section 3, the proposed algorithm ERaFA is developed. Benchmark problems and the corresponding experimental results are given in Section 4. Section 5 gives three practical problems. Finally, Section 6 concludes the paper.

II. BASIC FA ALGORITHM

FA is one of swarm intelligence algorithms. In FA, each firefly represents a point in the solution space. In initialization phase, the position of each firefly is randomly generated. After that, each firefly is compared with the rest of the firefly by their fitness value, and moves toward a firefly with relatively good fitness value, which is the phenomenon of attraction in FA.

Assume that D is the dimension of the problem, N is the population size, and $\boldsymbol{x}_i$ is the i-th firefly in the population, where $i = 1, 2, \cdots, N$. The attractiveness between two fireflies $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ is calculated as follows [14]:

$$\beta_{ij} = \beta_0 e^{-\gamma r_{ij}^2}, \tag{1}$$

where γ is the light absorption coefficient, r_{ij} is the distance between $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, which is computed by the following equation

$$r_{ij} = \| \boldsymbol{x}_i - \boldsymbol{x}_j \| = \sqrt{\sum_{d=1}^{D} (x_{id} - x_{jd})^2},$$

where x_{id} and x_{jd} are the d-th dimension of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, respectively.

In Eq. (1), β_0 is the attractiveness at $r = 0$. Through comparing the fitness values of $\boldsymbol{x}_i$ and the other fireflies $\boldsymbol{x}_j$, where $j = 1, 2, \cdots, N$ and $j \neq i$, the firefly $\boldsymbol{x}_i$ decides how to move. If $\boldsymbol{x}_j$ is brighter (better) than $\boldsymbol{x}_i$, that is $f(\boldsymbol{x}_j) < f(\boldsymbol{x}_i)$, then $\boldsymbol{x}_i$ will be attracted and move toward $\boldsymbol{x}_j$ by the following formula:

$$\boldsymbol{x}_i^{t+1} = \boldsymbol{x}_i^t + \beta_0 e^{-\gamma r_{ij}^2} (\boldsymbol{x}_j^t - \boldsymbol{x}_i^t) + \alpha \epsilon_i, \tag{2}$$

where $\epsilon_i \in [-0.5, 0.5]$ is a random number that obeys uniformly distributed, and $\alpha \in [0, 1]$ is a step factor.

The pseudo code description of the basic FA is given in algorithm 1, where *ItMax* is the maximum number of iterations.

Algorithm 1 Pseudo-Code of FA

Initialize the population size N, the maximum number of iterations $ItMax$.
while $t <= ItMax$ **do**
  for $i = 1$ to N **do**
    for $j = 1$ to N **do**
      if $f(\boldsymbol{x}_j) < f(\boldsymbol{x}_i)$ then
        Move $\boldsymbol{x}_i$ toward $\boldsymbol{x}_j$ according to (2).
        Compute the fitness value of the new $\boldsymbol{x}_i$.
      end
    end
  end
Rank the fireflies and find the current best
$t = t + 1$
end

As pointed out in [39], the basic FA is the full attraction model (see Fig. 1 (a)). In this model, each firefly is attracted by all other brighter fireflies. The advantage of this model is that it is likely to find better candidate solutions, but it should be pointed out that too many attractions may result in oscillation during the search process and high computational time complexity. Let $O(f)$ be the computational time complexity of the fitness evaluation function $f(\cdot)$. Reference [39] pointed out the computational time complexity of the full attraction model(basic FA) is $O(ItMax * N^2 * f)$, where $ItMax$ is the maximum number of generations.

III. PROPOSED APPROACH

Since there are so many attractions in the search process of FA, the phenomenon of oscillation occurs during the moving process, and the time complexity is high. To reduce attractions, a random attraction FA (RaFA) was proposed in [40]. In RaFA (see Fig. (b)), for each firefly $\boldsymbol{x}_i$, a firefly $\boldsymbol{x}_j (j \neq i)$ is selected randomly from the current population firstly. Then, by comparing the fitness values of $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$, the movement of $\boldsymbol{x}_i$ is determined. If $\boldsymbol{x}_j$ is brighter than $\boldsymbol{x}_i$, $\boldsymbol{x}_i$ will move toward $\boldsymbol{x}_j$. Thus, the number of attractions for each firefly is not greater than 1. Although random attraction can effectively reduce the computational time complexity and accelerate the search, it may result in premature convergence. To overcome this problem, both the random and Cauchy mutation have been used in RaFA. Recently, to achieve a trade-off between full attraction and random attraction, a new FA variant called NaFA (see Fig 1. (c)) was developed [39], which employs a neighborhood attraction model inspired by the k-neighborhood concept [41].

In RaFA, since $\boldsymbol{x}_j$ is chosen randomly from the current population, it is not necessarily superior to $\boldsymbol{x}_i$, that is, $\boldsymbol{x}_j$ may not be a good guide for the movement of $\boldsymbol{x}_i$. In NaFA, k-neighbor concept was introduced, which can guide $\boldsymbol{x}_i$ better than RaFA, but it's not doubt that this method increased the time complexity. Moreover, in both two methods, the case is not considered that $\boldsymbol{x}_i$ how to move when $\boldsymbol{x}_i$ is brighter than

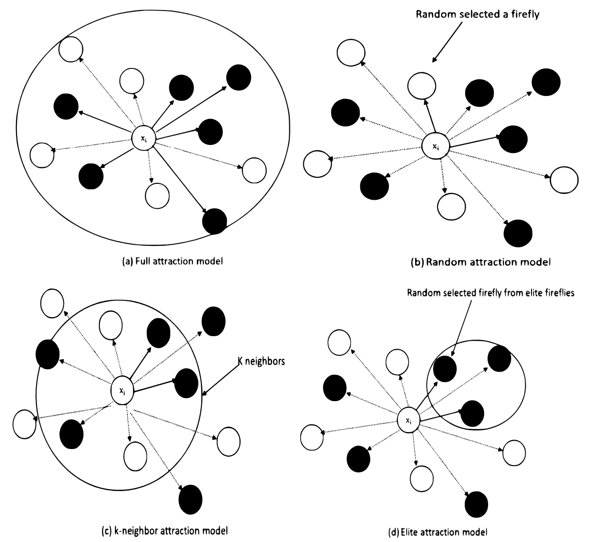

FIGURE 1. Different attraction models.

$\boldsymbol{x}_j$, that is they do not take full advantage of the information contained by $\boldsymbol{x}_i$.

In order to reduce the computational complexity, and well guide the movement of fireflies, an elitist attraction model (see Fig. 1(d)) is developed based on randomly selecting a firefly from elite fireflies, which lead each firefly to a better direction with greater probability. Meanwhile, if the firefly $\boldsymbol{x}_j$ selected randomly from elite fireflies is worse than $\boldsymbol{x}_i$, the opposite learning strategy is adopted to make better use of the information $\boldsymbol{x}_i$.

A. RANDOMLY GUIDED FA BASED ON ELITIST STRATEGY(ERAFA)

In ERaFA, we first give a proportional value ρ, which is used to determine the number of elite fireflies. Assume that N is the population size, then a firefly $\boldsymbol{x}_j$ is selected randomly from $[\rho * N]$ elite fireflies, and compared with $\boldsymbol{x}_i$, where $[\cdot]$ is a integral function. If $\boldsymbol{x}_j$ is brighter than $\boldsymbol{x}_i$, then $\boldsymbol{x}_i$ moves to $\boldsymbol{x}_j$, else $\boldsymbol{x}_i$ uses opposite learning strategy to move to a new position. The equation of motion is as follows:

$$\boldsymbol{x}_i^{t+1} = \begin{cases} \boldsymbol{x}_i^t + \beta_0 e^{-\gamma r_{ij}^2}(\boldsymbol{x}_j(t) - \boldsymbol{x}_i(t)) + \alpha\epsilon_i, & \text{if } f(\boldsymbol{x}_j) < f(\boldsymbol{x}_i), \\ \boldsymbol{l} + \boldsymbol{u} - \boldsymbol{x}_i^t, & \text{else}, \end{cases} \tag{3}$$

where $\boldsymbol{l}$ and $\boldsymbol{u}$ are the lower bound and upper bound of the search region, respectively.

B. ENHANCED LOCAL SEARCH ABILITY

To enhance the local search ability of our algorithm near the current optimal solution $\boldsymbol{x}^*$, a new mechanism is proposed, which is similar to the crossover operator in GA, and is used to generate new positions. This process is accomplished by crossing $\boldsymbol{x}^*$ with another feasible solution. In the early stage, $\boldsymbol{x}^*$ may have a long distance to the real optimal solution, while in the later stage, the distance between of $\boldsymbol{x}^*$ and the real optimal solution is getting closer and closer. Therefore, we hope that, the weight of $\boldsymbol{x}^*$ is smaller in the early stage, and become greater in the later stage. The process can ensure that our algorithm searches for a wide range in the early stage, and later concentrates on the neighborhood of $\boldsymbol{x}^*$. In addition, considering chaotic search has a stronger searching ability, chaotic search is used to generate the feasible solution for cross operation. Next, we give the details.

Let $\boldsymbol{x}^*$ be the best solution of the current iteration. Firstly, utilize the following equation (4) to generate chaotic variable σ_i:

$$\sigma_{i+1} = 4 * \sigma_i * (1 - \sigma_i), \quad 1 \leq i \leq k, \tag{4}$$

where k is the length of chaotic sequence, $\sigma_0 \in (0, 1)$ is a random number. Then map σ_i to a chaotic vector $\bar{\boldsymbol{x}}_i$ in the interval $[\boldsymbol{l}, \boldsymbol{u}]$:

$$\bar{\boldsymbol{x}}_i = \boldsymbol{l} + \sigma_i * (\boldsymbol{u} - \boldsymbol{l}), \quad i = 1, \cdots, k, \tag{5}$$

where $\boldsymbol{l}$ and $\boldsymbol{u}$ are the lower bound and upper bound of variable $\boldsymbol{x}$, respectively. Finally, a new candidate solution $\hat{\boldsymbol{x}}_i$ is obtained by the following equation:

$$\hat{\boldsymbol{x}}_i = \lambda * \boldsymbol{x}^* + (1 - \lambda) * \bar{\boldsymbol{x}}_i, \quad i = 1, \cdots, k, \tag{6}$$

where λ is a shrinking factor, which is defined as follows:

$$\lambda = \frac{t}{ItMax}, \tag{7}$$

where $ItMax$ is the maximum number of iterations, t is the number of iterations.

Based on the above discussion, the pseudo code of ERaFA is provided as follows:

Algorithm 2 Pseudo-Code of ERaFA

01:Initialize the population size N, the maximum number of iterations $ItMax$.
02: **while** $t <= ItMax$ **do**
03: Select $[\rho * N]$ elite fireflies from current population.
04: for $i = 1$ to N **do**
05: Select a firefly $\boldsymbol{x}_j$ from $[\rho * N]$ elite fireflies **do**
06: Move $\boldsymbol{x}_i$ according to (3).
07: Compute the fitness value of $\boldsymbol{x}_i$.
08: end
09: Rank the fireflies and find the current best.
10: By using (4)-(7), to search near $\boldsymbol{x}^*$, and update $\boldsymbol{x}^*$(if necessary).
11: $t = t + 1$
12: end

C. TIME COMPLEXITY

Let $O(f)$ be the computational time complexity of the fitness evaluation function $f(\cdot)$. For the standard FA, its time complexity is $O(ItMax * N^2 * f)$. For RaFA, its time complexity is $O(ItMax * N * f)$. For NaFA, its time complexity is $O(ItMax*k*N*f)$, where k is the number of neighbor. For our method ERaFA, its time complexity is $O(ItMax*(N+k)*f)$, here k is the number of the local search near the current optimal solution x^*. As can be seen, the time complexity of ERaFA and RaFA is not very different. On the whole, the time complexity of ERaFA is little higher than that of RaFA. Meanwhile, we can see that, the time complexity of ERaFA is much lower than that of FA and NaFA.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we have done a total of four experiments. In Experiments 1, 2 and 3, 32 benchmark functions are selected to test the performance of ERaFA. The detailed information of the test functions are displayed in Table 1. For these functions, $f_1 - f_8$ and $f_{20} - f_{21}$ are unimodal functions, f_9 is a discontinuous step function, f_{10} is noise function, $f_{11} - f_{19}$ are multimodal functions, $f_{22} - f_{23}$ are mis-scaled functions, $f_{24} - f_{29}$ are rotated functions, f_{30} is a shifted function, and $f_{31} - f_{32}$ are highly competitive problems, which are shifted and rotated functions. In Experiment 4, the proposed algorithm ERaFA is tested on some challenging benchmark functions selected from CEC 2015 [42]. This test suite includes different types of optimization problems, where f_1 and f_2 are unimodal functions, $f_3 - f_5$ are simple multimodal functions, $f_6 - f_8$ are hybrid functions, and $f_9 - f_{15}$ are composite functions. The detailed information about this test suit is given in Table 5. Among these four numerical experiments, the first one is to determine the parameter ρ, which has a great impact on the algorithm. The second one is to compare the performance of ERaFA with some other FAs, including FA, RaFA and NaFA. The third one is to comprehensively compare the performance of ERaFA with several state-of-the-art algorithms, including ApFA [25], CFA [31], NaFA [39], WSSFA [43], VSSFA [44], HPSOFF [45], FFPSO [46] and HFPSO [47]. The fourth one is to further test the performance of ERaFA. In this experiment, ERaFA is compared with some algorithms proposed recently, including ABC [48], SaDE [9], WWO [49], FWA-EI [50] and AEFA [51].

A. EXPERIMENT 1: THE DETERMINATION OF PARAMETER ρ

In ERaFA, the parameter ρ is used to change the proportion of the optimal solution, which is closely related to the rate of convergence of the algorithm. Thus, it is one of the key steps to select the value of parameter ρ in ERaFA. To determine the value of parameter ρ, we select 8 functions from Table 1: $f_2, f_4, f_6, f_7, f_8, f_{14}, f_{16}$ and f_{20}, and run ERaFA 30 times for each function with different values of ρ. In this experiment, the population size is 40, the maximum iterations($ItMax$) is set to 2500, the initial β_0, γ are set to 1, the dimensions of the test functions are set to 30. The statistical results including minimum, mean and standard deviation are given in Table 2.

From Table 2, we can see that the same results were obtained for f_2, f_6, f_8, f_{16} with different ρ; for f_4, when $\rho = 0.5$, the result is the worst; for f_7, the worst result was obtained at $\rho = 0.1$; for f_{14}, f_{20}, the best results were calculated at $\rho = 0.3$. Considering the above calculation results, $\rho = 0.3$ is the best choice. Therefore, in the following experiments, ρ is set to 0.3.

TABLE 1. Benchmark test functions.

Functions	Range	Optimal value
$f_1 = \sum_{i=1}^{n} x_i^2$	[-100,100]	0
$f_2 = \sum_{i=1}^{D} \mid x_i \mid + \prod_{i=1}^{D} \mid x_i \mid$	[-10,10]	0
$f_3 = \sum_{i=1}^{D} (\sum_{j=1}^{i} x_j)^2$	[-100,100]	0
$f_4 = \max_i\{\mid x_i \mid, 1 \leq i \leq n\}$	[-100,100]	0
$f_5 = \sum_{i=1}^{D} i x_i^2$	[-10,10]	0
$f_6 = \sum_{i=1}^{D} i x_i^4$	[-1.28,1.28]	0
$f_7 = \sum_{i=1}^{D} \mid x_i \mid^{(i+1)}$	[-1,1]	0
$f_8 = \sum_{i=1}^{D} 10^{6\frac{i-1}{D-1}} x_i$	[-100,100]	0
$f_9 = \sum_{i=1}^{D} (\lfloor x_i + 0.5 \rfloor)^2$	[-1.28,1.28]	0
$f_{10} = \sum_{i=1}^{D} i x_i^4 + random[0,1)$	[-1.28,1.28]	0
$f_{11} = \sum_{i=1}^{D} (x_i^2 - 10\cos(2\pi x_i) + 10)$	[-5.12,5.12]	0
$f_{12} = -20\exp(-0.2 * \sqrt{\sum_{i=1}^{D} x_i^2/D}) - \exp(\sum_{i=1}^{D} \cos(2\pi x_i/D) + 20 + e$	[-32,32]	0
$f_{13} = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} cos(\frac{x_i}{\sqrt{i}}) + 1$	[-600,600]	0
$f_{14} = 0.5 + \frac{sin(\sqrt{\sum_{i=1}^{D} x_i^2})^2 - 0.5}{(1+0.001\sum_{i=1}^{D} x_i^2)^2}$	[-100,100]	0
$f_{15} = \frac{1}{D}\sum_{i=1}^{D} (x_i^4 - 16x_i^2 + 5x_i)$	[-5,5]	-78.332
$f_{16} = \sum_{i=1}^{D} \mid x_i sin(x_i) + 0.1x_i \mid$	[-10,10]	0
$f_{17} = \sum_{i=1}^{D} (y_i^2 - 10\cos(2\pi y_i) + 10), if \mid x_i \mid < \frac{1}{2}, y_i = x_i; else\ y_i = \frac{round(2x_i)}{2}$	[-5.12,5.12]	0
$f_{18} = \frac{1}{D}[10sin^2(\pi * y_1) + \sum_{i=1}^{D-1}(y_i - 1)^2(1 + 10sin^2(\pi * y_{i+1})) + (y_D - 1)^2]$ $+ \sum_{i=1}^{D} u(x_i, 10, 100, 4)$, where $y_i = 1 + \frac{1}{4}(x_i + 1)$ $u(x_i, a, k, m) = \begin{cases} k(x_i - a)^m, & x_i > a, \\ 0, & -a \leq x_i \leq a, \\ k(-x_i - a)^m, & x < -a. \end{cases}$	[-50,50]	0
$f_{19} = 418.9829 * D - \sum_{i=1}^{D}(x_i * sin(\sqrt{\mid x_i \mid}))$	[-500,500]	0
$f_{20} = \sum_{i=1}^{D-1}(100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2)$	[-2.048,2.048]	0
$f_{21} = -\exp(-0.5 * \sum_{i=1}^{D} x_i^2)$	[-1,1]	-1
$f_{22} = \sum_{i=1}^{D-1}((a_i x_i)^2 - 10cos(2\pi a_i x_i) + 10)$ $a_i = 10^{\frac{i-1}{D-1}}$	[-5.12,5.12]	0
$f_{23} = \sum_{i=1}^{D-1}((a_i x_i)^2 - 10cos(2\pi a_i x_i) + 10)$ $a_i = 1000^{\frac{i-1}{D-1}}$	[-5.12,5.12]	0
$f_{24} = \sum_{i=1}^{D} z_i^2,\ z = x * M$	[-500,500]	0
$f_{25} = \max_i\{\mid z_i \mid, 1 \leq i \leq D\}, z = x * M$	[-10,10]	0
$f_{26} = -20\exp(-0.2 * \sqrt{\sum_{i=1}^{D} z_i^2/D})$ $- \exp(\sum_{i=1}^{n} \cos(2\pi z_i/D) + 20 + e, z = x * M$	[-32,32]	0
$f_{27} = \frac{1}{4000}\sum_{i=1}^{D} z_i^2 - \prod_{i=1}^{D} cos(\frac{z_i}{\sqrt{i}}) + 1\ , z = x * M$	[-600,600]	0
$f_{28} = \sum_{i=1}^{D}(20^{\frac{i-1}{D-1}} z_i)^2, z = x * M$	[-100,100]	
$f_{29} = \sum_{i=1}^{D}(1000 * z_1)^2 + \sum_{i=2}^{D} z_i^2,\ z = x * M$	[-100,100]	0
$f_{30} = \sum_{i=1}^{D}(z_i^2 - 10\cos(2\pi z_i) + 10),\ z = x - o$	[-5.12,5.12]	0
$f_{31} = z_1^2 + \sum_{i=2}^{D} z_i^2 + 100,\ z = (x - o) * M$	[-100,100]	100
$f_{32} = \sum_{i=1}^{D}(\sum_{k=1}^{kmax}[a^k cos(2\pi b^k(z_i + 0.5))]) - D\sum_{k=1}^{kmax}[a^k cos(2\pi b^k \cdot 0.5)] + 300,\ z = (x - o) * M, a = 0.5, b = 3, kmax = 20$	[-100,100]	300

TABLE 2. The comparison for different value of parameter ρ.

Functions	$\rho = 0.1$			$\rho = 0.2$			$\rho = 0.3$			$\rho = 0.4$			$\rho = 0.5$		
	Min	Mean	SD	Min	Mean	SD	Min	Mean	SD	Min	Mean	SD	Min	Mean	SD
f_2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_4	0	0	0	0	0	0	0	0	0	0	0	0	0	4.26e-015	9.54e-015
f_6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_7	0	4.43e-111	1.08e-110	0	0	0	0	0	0	0	0	0	0	0	0
f_8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_{14}	5.51e-017	1.925e-007	4.30e-003	5.51e-017	9.99e-017	3.04e-017	5.51e-017	6.61e-017	2.48e-017	5.51e-017	7.71e-017	3.04e-017	5.51e-017	8.81e-017	3.04e-017
f_{16}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f_{20}	2.75e-007	8.74e-007	7.86e-007	2.20e-007	4.76e-006	8.74e-006	4.59e-009	1.74e-008	1.36e-008	4.23e-009	2.78e-006	2.92e-006	1.92e-007	8.22e-006	1.21e-005

B. EXPERIMENT 2: COMPARISON OF ERAFA, FA, RAFA AND NAFA

In this subsection, to test the performance of ERaFA, it is compared with FA, RaFA, NaFA. The parameters are the same as that in the experiment 1. All algorithms run 30 times. The comparison results are given in Table 3, and the best solutions obtained by algorithms are marked in boldface. The t test values were used to determine whether the results of ERaFA are statistically different from the results of other algorithms, where significant level is set to 0.05. In this test, "+" indicates that the performance of ERaFA is statistically significantly better than of its competitor, "=" means that the performance of the competitor is statistically comparable to that of ERaFA, "−" implies that the performance of the competitor is statistically significantly better than that of ERaFA.

From Table 3, we can see that, for functions $f_1 - f_8$, $f_{10} - f_{12}$, $f_{14} - f_{17}$, $f_9 - f_{26}$ and $f_{28} - f_{32}$, the accuracy of the results obtained by ERaFA is better than that of the other algorithms. For function f_9, all the algorithms have the same accuracy, and can find the global optimal values. For function f_{13}, both FA and ERaFA found the optimal results, which are better than that of RaFA and NaFA. For function f_{27}, ERaFA and NaFA have the best results with the same accuracy, which is superior to that of the other two algorithms. For function f_{18}, the accuracy of the result obtained by ERaFA is lower than that of FA and NaFA. From these results, we can see that ERaFA is only defeated by other algorithms on function f_{18}. Further analysis, we find that, ERaFA almost can the optimal values of all the unimodal functions, which means that it has a strong search ability on unimodal functions. For multimodal functions, ERaFA beat the other algorithms on almost all these functions, except for function f_{18}. This implies that ERaFA is not easy to fall into local optimum. In addition, for mis-scaled, rotated, shifted and rotated functions $f_{22} - f_{32}$, the results show that ERaFA has superior search ability, because ERaFA can obtain the results with better accuracy than that of the other algorithms, except for f_{27}. In summary, the accuracy of the results obtained by ERaFA are better than that of the other algorithms for over 90% of all test functions. The main reason is that ERaFA has a better balance between global and local search ability.

By t test results, we can see that the performance of ERaFA is superior or equal to the other algorithms on all the functions, except for the functions f_9, f_{18}, and f_{27}, which is consistent with the above analysis.

In order to compare the convergence speed of these algorithms, the convergence curves of all algorithms for functions $f_1 - f_{30}$ are given in Figure 2. From Figure 2, it can be seen that, for most test functions, the convergence rate of ERaFA is very fast. For function f_{17}, although ERaFA and NaFA have the same solution accuracy, the convergence rate of ERaFA slightly better than that of NaFA.

In conclusion, the performance of ERaFA is better than FA, RaFA and NaFA, and ERaFA can obtain the best results for most functions.

C. EXPERIMENT 3: COMPARISON OF ERAFA WITH OTHER FA VARIANTS

In this subsection, 14 functions are selected from Table 1 to further test the performance of ERaFA. We compared the performance of ERaFA with eight other recently proposed FA variants, which include CFA, WSSFA, VSSFA, RaFA ApFA, HPSOFF, FFPSO and HFPSO. The detailed information of these algorithms are presented as follows.

−CFA(FA with chaos), Gandomi et al.(2013)
−WSSFA(Wise step strategy FA), Yu et al.(2014)
−VSSFA(Variable step size FA), Yu et al.(2015)
−RaFA(FA with random attraction), Wang et al.(2016)
−ApFA(FA with adaptive control parameters), Wang et al.(2017)
−HPSOFF(Hybrid PSO and FA), Arunachalam et al.(2015)
−FFPSO(Hybrid FA and PSO), Kora et al.(2016)
−HFPSO(Hybrid FA and PSO), Aydilek (2018)
−ERaFA, Our approach.

In this experiment, for functions $f_1 - f_4$, $f_9 - f_{13}$ and $f_{18} - f_{20}$, ERaFA is compared with VSSFA, WSSFA, CFA, RaFA, ApFA. The population size is set to 20. The maximum value of the function value is the termination condition, which is set to $5e5$ and is consistent with the comparison literature. For functions f_{31} and f_{32}, ERaFA is compared with FFPSO, HPSOFF and HFPSO, the population size is set to 30. The maximum value of the function value is the termination condition, which is set to $1.5e3$ and is consistent with the comparison literature. The other parameters are the same as that in the experiment 1. The results are taken from [39] and

TABLE 3. Computational results of FA, RaFA, NaFA, and ERaFA.

Functions	Algorithm	Min	Mean	Std
f_1	FA	2.001e-040	$2.119e-040^{+}$	1.131e-041
	RaFA	2.484e-040	$3.257e-040^{+}$	7.383e-041
	NaFA	5.925e-041	$6.389e-041^{+}$	7.126e-042
	ERaFA	**0**	**0**	**0**
f_2	FA	5.958e-021	$6.011e-021^{+}$	5.372e-023
	RaFA	7.797e-021	$8.049e-021^{+}$	3.418e-022
	NaFA	3.010e-021	$3.220e-021^{+}$	2.369e-022
	ERaFA	**0**	**0**	**0**
f_3	FA	7.1878e-040	$7.8727e-040^{+}$	6.4396e-041
	RaFA	8.831e+001	$1.804e+002^{+}$	1.213e+002
	NaFA	1.234e+000	$1.817e+000^{+}$	3.132e-001
	ERaFA	**0**	**0**	**0**
f_4	FA	5.421e-021	$5.840e-021^{+}$	4.169e-022
	RaFA	5.037e-003	$4.050e-001^{+}$	3.268e-001
	NaFA	3.052e-021	$3.259e-021^{+}$	2.948e-022
	ERaFA	0	**0**	**0**
f_5	FA	2.325e-041	$2.796e-041^{+}$	6.869e-042
	RaFA	7.030e-001	$9.750e-001^{+}$	2.008e-001
	NaFA	2.065e-035	$2.298e-034^{+}$	3.420e-034
	ERaFA	**0**	**0**	**0**
f_6	FA	1.113e-087	$1.263e-087^{+}$	1.310e-088
	RaFA	5.551e-087	$7.100e-047^{+}$	1.229e-046
	NaFA	7.554e-089	$1.190e-088^{+}$	4.031e-089
	ERaFA	**0**	**0**	**0**
f_7	FA	4.688e-009	$8.170e-009^{+}$	3.341e-009
	RaFA	6.530e-008	$1.119e-007^{+}$	4.459e-008
	NaFA	1.376e-009	$1.023e-008^{+}$	7.677e-009
	ERaFA	**0**	**0**	**0**
f_8	FA	6.345e+005	$8.700e+005^{+}$	2.510e+005
	RaFA	3.941e+006	$4.610e+006^{+}$	1.024e+006
	NaFA	4.294e+005	$7.477e+005^{+}$	2.925e+005
	ERaFA	**0**	**0**	**0**
f_9	FA	**0**	**0**$^{=}$	**0**
	RaFA	**0**	**0**$^{=}$	**0**
	NaFA	**0**	**0**$^{=}$	**0**
	ERaFA	**0**	**0**	**0**
f_{10}	FA	3.560e-003	6.137e-003^{+}	2.421e-003
	RaFA	1.072e-002	1.405e-002^{+}	4.033e-003
	NaFA	2.742-003	3.443e-003^{+}	1.059e-003
	ERaFA	**1.014e-006**	**2.632e-006**	**1.603e-006**
f_{11}	FA	2.526e+001	4.177e+001^{+}	1.406e+001
	RaFA	1.521e+001	1.832e+001^{+}	3.149e+000
	NaFA	1.044e+001	1.782e+001^{+}	8.401e+000
	ERaFA	**0**	**0**	**0**
f_{12}	FA	1.332e-014	2.042e-014^{+}	7.105e-015
	RaFA	6.217e-015	1.332e-014^{+}	7.105e-015
	NaFA	2.042e-014	2.279e-014^{+}	4.102e-015
	ERaFA	**-8.8818e-016**	**-8.8818e-016**	**0**
f_{13}	FA	**0**	**0**$^{=}$	**0**
	RaFA	9.964e-001	9.988e-001^{+}	2.261e-003
	NaFA	6.772e-002	2.959e-001^{+}	3.518e-001
	ERaFA	**0**	**0**	**0**
f_{14}	FA	3.728e-002	5.063e-002^{+}	2.372e-002
	RaFA	1.797e-001	2.549e-001^{+}	6.386e-002
	NaFA	3.719e-002	3.866e-002^{+}	3.2174e-017
	ERaFA	**5.551e-017**	**7.401e-017**	**2.72e-17**
f_{15}	FA	-72.677	-72.049^{+}	5.441e+000
	RaFA	-73.353	-72.588^{+}	8.139e-001
	NaFA	-75.505	-72.363^{+}	2.879e+000
	ERaFA	-78.3323	**-78.3323**	**1.76e-009**
f_{16}	FA	1.722e-022	2.035e-016^{+}	3.525e-016
	RaFA	4.960e-003	1.067e-002^{+}	7.005e-003
	NaFA	9.228e-023	1.063e-022^{+}	1.396e-023
	ERaFA	**0**	**0**	**0**
f_{17}	FA	5.865e+001	6.396e+001^{+}	4.128e+000
	RaFA	1.868e+001	2.156e+001^{+}	4.726e+000
	NaFA	2.311e+001	2.556e+001^{+}	2.304e+000
	ERaFA	**0**	**0**	**0**
f_{18}	FA	**1.570e-032**	**1.570e-032**$^{-}$	**0**
	RaFA	5.430e-008	2.322e-003^{+}	4.035e-003
	NaFA	**1.570e-032**	**1.570e-032**$^{-}$	**0**
	ERaFA	1.246e-008	3.383e-007	2.840e-007

TABLE 3. ***(Continued.)*** **Computational results of FA, RaFA, NaFA, and ERaFA.**

f_{19}	FA	2.653e+003	3.328e+003^{+}	6.390e+002
	RaFA	8.121e+003	8.218e+003^{+}	1.080e+002
	NaFA	4.875e+003	5.371e+003^{+}	5.822e+002
	ERaFA	**1.041e-008**	**8.093e-005**	**9.185e-005**
f_{20}	FA	1.903e+001	2.079e+001^{+}	1.122e+000
	RaFA	2.708e+001	8.226e+002^{+}	9.135e+002
	NaFA	2.628e+001	4.593e+001^{+}	3.396e+001
	ERaFA	**8.0927e-006**	**2.719e-004**	**4.561e-004**
f_{21}	FA	0	0^{+}	0
	RaFA	0	0^{+}	0
	NaFA	0	0^{+}	0
	ERaFA	**-1**	**-1**	**0**
f_{22}	FA	3.412e+001	4.014e+001^{+}	6.943e+000
	RaFA	4.278e+001	5.178e+001^{+}	1.643e+001
	NaFA	2.451e+001	2.630e+001^{+}	1.452e+000
	ERaFA	**0**	**0**	**0**
f_{23}	FA	3.174e+003	3.591e+003^{+}	6.478e+002
	RaFA	4.341e+003	1.008e+004^{+}	5.126e+003
	NaFA	8.002e+002	1.340e+003^{+}	7.057e+002
	ERaFA	**0**	**0**	**0**
f_{24}	FA	1.582e+005	1.686e+005^{+}	1.461e+004
	RaFA	9.041e-039	9.950e-039^{+}	7.934e-040
	NaFA	5.791e-039	6.623e-039^{+}	7.314e-040
	ERaFA	**0**	**0**	**0**
f_{25}	FA	2.906e+000	3.180e+000^{+}	2.421e-001
	RaFA	6.946e-005	1.382e-003^{+}	1.735e-003
	NaFA	6.293e-022	6.666e-022^{+}	3.437e-023
	ERaFA	**0**	**0**	**0**
f_{26}	FA	1.356e+001	1.405e+001^{+}	5.447e-001
	RaFA	1.332e-014	1.569e-014^{+}	4.102e-015
	NaFA	1.332e-014	1.569e-014^{+}	4.102e-015
	ERaFA	**-8.881e-016**	**-8.881e-016**	**0**
f_{27}	FA	9.980e-001	9.994e-001^{+}	7.717e-004
	RaFA	7.824e-001	9.261e-001^{+}	1.261e-001
	NaFA	**0**	**0**$^{=}$	**0**
	ERaFA	**0**	**0**	**0**
f_{28}	FA	2.559e+005	2.696e+005^{+}	1.338e+004
	RaFA	4.565e+003	8.292e+003^{+}	5.278e+003
	NaFA	4.706e+001	2.238e+002^{+}	1.664e+002
	ERaFA	0	**0**	**0**
f_{29}	FA	2.790e+005	2.939e+005^{+}	1.765e+004
	RaFA	3.316e+003	6.725e+003^{+}	3.734e+003
	NaFA	1.325e+002	4.540e+002^{+}	3.740e+002
	ERaFA	**0**	**0**	**0**
f_{30}	FA	1.660e+002	1.711e+002^{+}	4.522e+000
	RaFA	1.442e+002	1.73e+002^{+}	3.002e+001
	NaFA	1.614e+001	1.857e+001^{+}	1.519e+000
	ERaFA	**9.970e-007**	**5.091e-006**	**3.577e-006**
f_{31}	FA	3.3864e+005	4.9946e+005^{+}	1.4901e+005
	RaFA	6.3523e+008	1.0860e+009^{+}	5.1039e+008
	NaFA	1.2023e+010	1.5527e+010^{+}	2.6890e+009
	ERaFA	**1.0947e+002**	**1.5519e+002**	**6.4974e+001**
f_{32}	FA	3.2947e+002	3.3130e+002^{+}	1.6782e+000
	RaFA	3.2752e+002	3.2867e+002^{+}	1.4068e+001
	NaFA	3.2397e+002	3.2850e+002^{+}	4.3458e+000
	ERaFA	**3.0000e+002**	**3.0000e+002**	**7.1020e-002**

[46] directly, except for that of our algorithm ERaFA. The comparison results are shown in Table 4.

The results in Table 4 show that, the mean values obtained by VSSFA and WVSSFA are worse than those of the rest of algorithms for all the test functions. The results obtained by ERaFA are better than those of the other algorithms except for function f_{18}. For function f_{18}, as seen from Table 4, the result obtained by ApFA is the best, followed by CFA, and ERaFA in third palce. In addition, for functions f_{31} and f_{32}, the accuracy of the results obtained by ERaFA is better than that of the other algorithms. Based on the above analysis, it is clear that the overall performance of ERaFA is the best.

D. EXPERIMENT 4: COMPARISON OF ERAFA WITH SOME STATE-OF-THE-ART ALGORITHMS

In this experiment, we compare ERaFA with the following state-of-art algorithm on CEC 2015 benchmark set:

–Artificial bee colony (ABC), Karaboga et al.(2007)
–The self-adaptive DE (SaDE), Qin et al.(2009)
–The Water Wave Optimization (WWO), Zheng et al.(2015)
– The FWA-EI, Zhang et al.(2017)
–AEFA, Sajwan et al.(2019)
–ERaFA, Our approach.

30-dimensional test functions were used and all results were obtained from 25 independent runs. In each run, the pop-

TABLE 4. Computational results of ERaFA and five other FA variants.

Function	Algorithm	Mean
f_1	VSSFA	5.84e+004
	WSSFA	6.34e+004
	CFA	3.27e-006
	RaFA	5.36e-184
	ApFA	2.02e-044
	ERaFA	**0**
f_2	VSSFA	1.13e+002
	WSSFA	1.35e+002
	CFA	8.06e-004
	RaFA	8.76e-005
	ApFA	1.83e-012
	ERaFA	**0**
f_3	VSSFA	1.16e+005
	WSSFA	1.10e+005
	CFA	1.24e-005
	RaFA	4.91e+002
	ApFA	1.01e+001
	ERaFA	**0**
f_4	VSSFA	8.18e+001
	WSSFA	7.59e+001
	CFA	8.98e-004
	RaFA	2.43e+000
	ApFA	1.30e-007
	ERaFA	**0**
f_9	VSSFA	5.48e+004
	WSSFA	6.18e+004
	CFA	**0**
	RaFA	**0**
	ApFA	**0**
	ERaFA	**0**
f_{10}	VSSFA	4.43e+001
	WSSFA	3.24e-001
	CFA	9.03e-002
	RaFA	5.47e-002
	ApFA	2.76e-003
	ERaFA	**3.27e-006**
f_{11}	VSSFA	3.12e+002
	WSSFA	3.61e+002
	CFA	5.27e+001
	RaFA	2.69e+001
	ApFA	1.21e+001
	ERaFA	**0**
f_{12}	VSSFA	2.03e+001
	WSSFA	2.05e+001
	CFA	4.02e-004
	RaFA	3.61e-014
	ApFA	2.55e-014
	ERaFA	**5.32e-016**
f_{13}	VSSFA	5.47e+002
	WSSFA	6.09e+002
	CFA	7.91e-006
	RaFA	**0**
	ApFA	3.33e-016
	ERaFA	**0**
f_{18}	VSSFA	3.99e+008
	WSSFA	6.18e+008
	CFA	8.28e-009
	RaFA	4.50e-005
	ApFA	**1.23e-016**
	ERaFA	4.79e-007
f_{19}	VSSFA	1.07e+004
	WSSFA	1.06e+004
	CFA	4.36e+003
	RaFA	5.03e+002
	ApFA	6.42e+003
	ERaFA	**1.46e-005**
f_{20}	VSSFA	2.16e+008
	WSSFA	2.49e+008
	CFA	2.06e+001
	RaFA	2.92e+001
	ApFA	2.81e+001
	ERaFA	**2.45e-003**
f_{31}	FFPSO	1.62873e+010
	HPSOFF	4.8387e+007
	HFPSO	1.3768e+007
	ERaFA	**2.1702e+002**
f_{32}	FFPSO	3.1455e+002
	HPSOFF	3.0845e+002
	HFPSO	3.0671e+002
	ERaFA	**3.0012e+002**

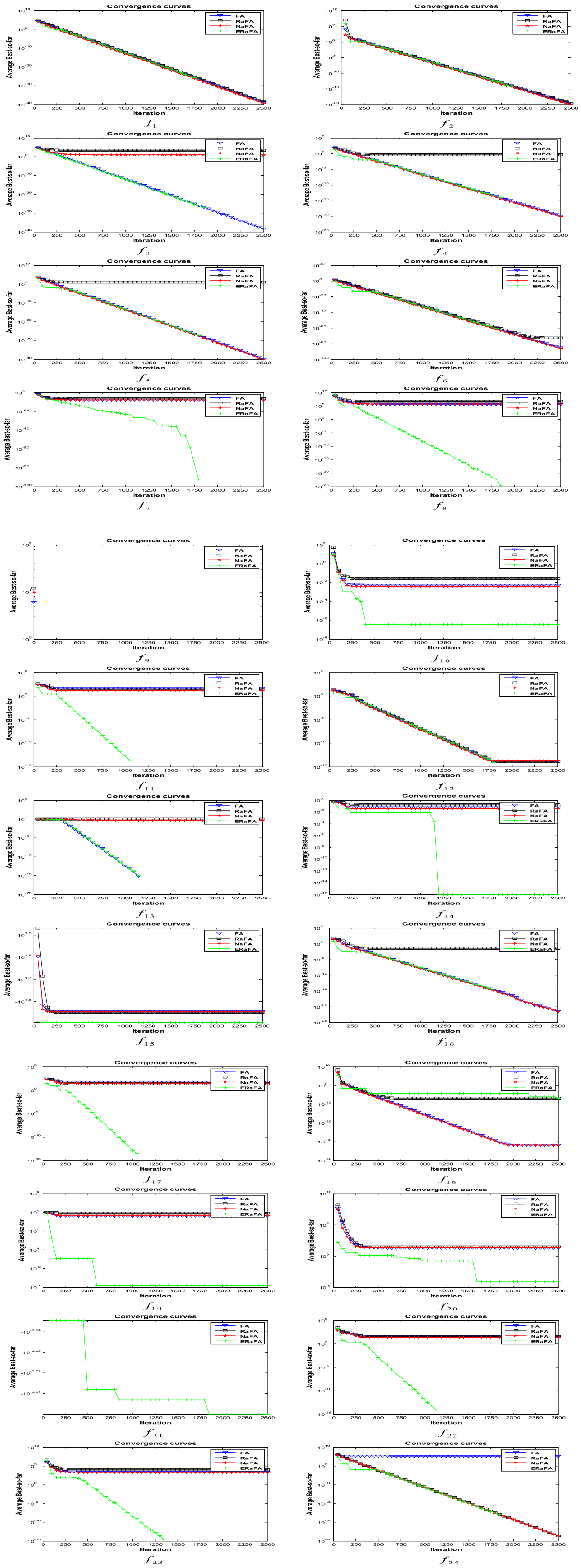

FIGURE 2. Convergence curves of f_1-f_{30}.

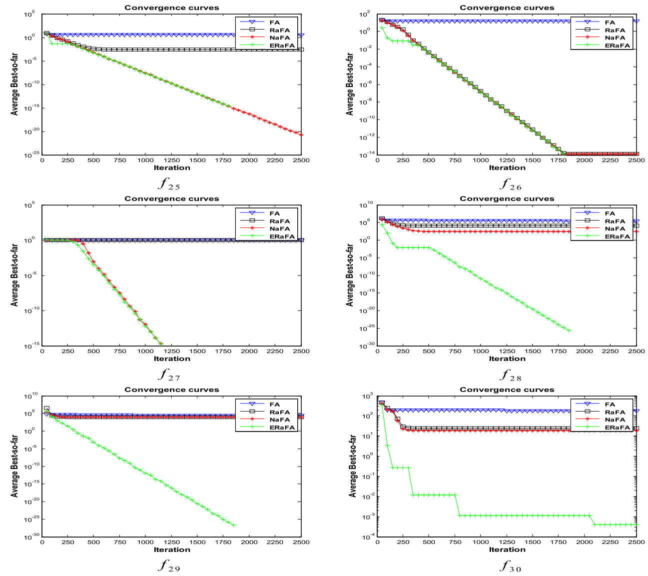

FIGURE 2. *(Continued.)* Convergence curves of f_1-f_{30}.

ulation size and the maximum number of function evaluations (MaxFES) are 20 and 10000×30 respectively, which is consistent with the comparison literature. These results are taken from [50] directly, except for that of the proposed algorithm ERaFA. The comparison results are given in Table 4, including the best, worst and standard deviation. In Table 4, the best fitness and median value among the algorithms are marked in bold.

From Table 6, it can be seen that ERaFA, except for F_7 and F_{13}, ERaFA outperformed or equally performed in comparison with all the other existing algorithms. The detailed comparison results are as follows:

(1) Unimodal functions $F_1 - F_2$. Compared with the five other algorithms, ERaFA can achieve the best performance on F_1. Regarding F_2, ERaFA, SaDE and FWA-EF have the same minimum and mean, which are superior to the results of the other algorithms. However, the deviation of SaDE is too large.

(2) Simple multimodal functions $F_3 - F_5$. ERaFA obtains better results than the other algorithms on F_3 and F_4 in terms of the minimum and mean. For F_5, the results of ERaFA are slightly better than that of ABC, but much better than that of the rest algorithms.

(3) Hybrid functions $F_6 - F_8$. Except for function F_7, ERaFA performs better than the other approaches according to the minimum and mean. For F_7, the results of ERaFA are slightly worse than that of SaDE and AEFA, but better than that of WWO and FWA-EF.

(4) Composition functions $F_9 - F_{15}$. Except for function F_{13}, ERaFA has better or equal performance among these algorithms. For function F_{13}, the performance of ERaFA is slightly worse than SaDE, WWO, FWA-EF and AEFA, but better than ABC. For function F_9, ERaFA is slightly better than ABC, but much better than the other algorithms. For function F_{12}, ERaFA, SaDE, WWO and AEFA have the same

TABLE 5. IEEE CEC15 learning based benchmark test suite [42], with search range = [−100, 100] D and $f_{\min}$ is minimum fitness value.

Nature of Functions	Function No.	Function Name	f_{min}
Unimodal Functions	F_1	Rotated High Conditioned Elliptic Function	100
	F_2	Rotated Cigar Function	200
Simple Multimodal Functions	F_3	Shifted and Rotated Ackleys Function	300
	F_4	Shifted and Rotated Rastrigins Function	400
	F_5	Shifted and Rotated Schwefels Function	500
Hybrid Functions	F_6	Hybrid Function 1 ($N = 3$)	600
	F_7	Hybrid Function 2 ($N = 4$)	700
	F_8	Hybrid Function 3 ($N = 5$)	800
Composition Functions	F_9	Composition Function 1 ($N = 3$)	900
	F_{10}	Composition Function 2 ($N = 3$)	1000
	F_{11}	Composition Function 3 ($N = 5$)	1100
	F_{12}	Composition Function 4 ($N = 5$)	1200
	F_{13}	Composition Function 5 ($N = 5$)	1300
	F_{14}	Composition Function 6 ($N = 7$)	1400
	F_{15}	Composition Function 7 ($N = 10$)	1500

TABLE 6. Comparative results of objective function values for CEC15 30D.

F	Metric	ABC	SaDE	WWO	FWA-EF	AEFA	ERaFA
F_1	best	1.21e+002	2.43e+003	2.57e+005	7.10e+005	2.02e+002	**1.000e+002**
	worst	5.16e+010	6.73e+004	2.94e+006	1.75e+006	1.46e+003	1.001e+002
	median	3.58e+003	2.55e+004	1.28e+006	1.03e+006	5.57e+002	**1.000e+002**
	std	1.71e+009	2.21e+004	7.14e+005	3.97e+005	3.13e+001	3.482e-002
F_2	best	3.38e+004	**2.00e+002**	**2.00e+002**	**2.00e+002**	2.05e+002	**2.000e+002**
	worst	6.82e+004	1.36e+005	2.89e+002	2.00e+002	1.25e+004	2.001e+002
	median	3.38e+004	**2.00e+002**	2.08E+02	**2.00E+02**	5.89e+003	**2.000e+002**
	std	7.28e-011	3.75e+004	2.40e+001	2.67e-003	2.70e+003	4.365e-002
F_3	best	3.25e+002	3.20e+002	3.20e+002	3.20e+002	3.20e+002	**3.000e+002**
	worst	3.46e+002	3.21e+002	3.20e+002	3.20e+002	3.20e+002	3.000e+002
	median	3.25e+002	3.20e+002	3.20e+002	3.20e+002	3.20e+002	**3.000e+002**
	std	1.12e+000	5.43e-002	6.46e-006	3.65e-005	1.44e-002	9.562e-004
F_4	best	4.41e+003	4.15e+002	4.39e+02	5.64e+02	4.07e+002	**4.000e+002**
	worst	8.26e+003	5.64e+002	5.30e+002	7.66e+002	4.07e+002	4.000e+002
	median	4.63e+003	4.26e+002	4.93e+002	6.28e+002	4.07e+002	**4.000e+002**
	std	1.55e+002	2.91e+001	1.93e+001	4.55e+001	1.37e-003	4.391e-006
F_5	best	5.02e+002	4.01e+003	3.89e+003	2.63e+003	6.25e+002	**5.000e+002**
	worst	5.06e+002	7.69e+003	5.63e+003	4.83e+003	6.26e+002	5.000e+002
	median	5.02e+002	5.12e+003	3.08e+003	3.86e+003	0.25e+002	**5.000e+002**
	std	1.06e-001	3.18e+002	3.97e+002	4.94e+002	3.68e-002	6.637e-004
F_6	best	6.01e+002	8.96e+002	4.26e+003	1.83e+004	8.49e+002	**6.000e+002**
	worst	6.05e+002	1.83e+004	1.84e+005	1.22e+005	8.49e+002	6.004e+002
	median	6.01e+002	6.34e+003	6.74e+004	5.10e+004	8.49e+002	**6.001e+002**
	std	2.26e-001	5.64e+003	4.41e+004	3.50e+004	3.43e-002	2.251e-001
F_7	best	7.02e+002	**7.01e+002**	7.08e+002	7.10e+002	**7.01e+002**	7.080e+002
	worst	7.98e+002	7.06e+002	7.15e+002	7.81e+002	7.01e+002	7.081e+002
	median	7.02e+002	7.02e+002	7.13e+002	7.13e+002	**7.01e+002**	7.080e+002
	std	3.20e+000	1.29e+000	7.83e+00	1.77e+01	1.32e-004	3.800e-002
F_8	best	8.18e+002	8.11e+002	1.27e+003	5.18e+003	8.63e+002	**8.055e+002**
	worst	6.67e+006	7.00Ee+003	8.74e+004	8.01e+004	8.63e+002	8.056e+002
	median	8.19e+002	1.12e+003	4.11e+004	3.11e+004	8.63e+002	**8.055e+002**
	std	2.45e+004	1.47e+003	2.59e+004	1.00e+003	9.69e-004	4.700e-003
F_9	best	9.12e+002	**1.00e+003**	**1.00e+003**	**1.00e+003**	**1.00e+003**	**9.100e+003**
	worst	9.14e+002	1.00e+003	1.00e+003	1.00e+003	1.00e+003	1.000e+003
	median	9.12e+002	**1.00e+003**	**1.00e+003**	**1.00e+003**	**1.00e+003**	**9.100e+003**
	std	8.98e-002	0	3.26e-002	3.05e-001	2.76e-001	0
F_{10}	best	2.66e+006	1.22e+003	1.21e+003	1.24e+003	1.27e+003	**1.1045e+003**
	worst	1.15e+008	2.39e+003	1.98e+003	2.35e+003	1.27e+003	1.1045e+003
	median	2.66e+006	1.48e+000	1.44e+003	1.41e+003	1.27e+003	**1.1045e+003**
	std	2.20e+006	4.27e+002	2.29e+002	3.23e+002	1.13e-001	3.690e-002
F_{11}	best	1.14e+003	1.40e+003	1.40e+003	1.40e+003	1.40e+03	**1.100e+003**
	worst	1.46e+003	1.58e+003	1.41e+003	2.40e+003	1.40e+03	1.100e+003
	median	1.14e+003	1.50e+003	1.41e+003	1.41e+003	1.40e+03	**1.100e+003**
	std	7.50e+00	5.68e+001	2.27e+000	3.23e+002	5.78e-004	1.607e-013
F_{12}	best	1.73e+003	**1.30e+003**	**1.30e+003**	1.31e+003	**1.30e+003**	**1.300e+003**
	worst	4.27e+003	1.31e+003	1.31e+003	1.31e+003	1.30e+003	1.300e+003
	median	1.75e+003	**1.30e+003**	**1.30e+003**	1.31e+003	**1.30e+003**	**1.300e+003**
	std	6.81e+001	7.35e-001	5.47e-001	1.35e+000	4.61e-001	1.690e-009
F_{13}	best	1.65e+003	**1.30e+003**	**1.30e+003**	**1.30e+003**	**1.30e+003**	1.313e+003
	worst	2.59e+003	1.30e+003	1.30e+003	1.30e+003	1.30e+003	1.314e+003
	median	1.66e+003	**1.30e+003**	**1.30e+003**	**1.30e+003**	**1.30e+003**	1.314e+003
	std	1.93e+001	2.42e-003	7.86e-003	5.79e-001	1.32e-003	1.609e-001
F_{14}	best	1.65e+003	3.26e+004	3.27e+004	3.26e+004	**1.50e+003**	**1.500e+003**
	worst	1.89e+003	3.50e+004	3.60e+004	3.67e+004	1.51e+003	1.506e+003
	median	1.65e+003	3.37e+004	3.38e+004	3.36e+004	1.51e+003	**1.500e+003**
	std	6.20e+000	8.19e+002	1.02e+003	1.37e+003	2.47e+000	3.061e-001
F_{15}	best	2.63e+003	**1.60e+003**	**1.60e+003**	**1.60e+003**	**1.60e+003**	**1.600e+003**
	worst	3.18e+003	1.60e+003	1.60e+003	1.60e+003	1.60e+003	1.600e+003
	median	2.66e+003	**1.60e+003**	**1.60e+003**	**1.60e+003**	**1.60e+003**	**1.600e+003**
	std	7.18e+000	6.44e-013	2.23e-013	2.12e-004	2.10e-002	1.607e-013

performance, which is better than that of ABC and FWA-EF. For F_{15}, the performance of ERaFA is the same as that of SaDE, WWO, FWA-EF and AEFA, but is much better than that of ABC.

In a word, the comparison results indicate that ERaFA is superior or comparable to the other algorithms for most of the problems.

V. APPLICATION OF ERAFA

In this section, ERaFA is used to solve three practical problems: Three-bar truss design, I-beam design problem and Welded beam design problem. In order to further prove the performance of ERaFA, we compare the results obtained by ERaFA with other algorithms's. In this paper, Deb's rules is utilized to solve constraint conditions for practical problems. The detailed description of Deb's rules is given as follows [52]:

(1) Between a feasible solution and a infeasible solution, The feasible solution is preferred.

(2) The infeasible solution is regarded as a feasible solution, when the infeasible solution violates the constraints very rarely.

(3) For two feasible solutions, the solution with better objective function value is better.

(4) For two infeasible solutions, the solution violating constraints very little is better.

A. THREE-BAR TRUSS DESIGN

Three-bar truss design(see Fig. 3) is a structual optimization problem. To minimize the weight subject to stress, deflection, and buckling constraints, the two parameters $A_1(x_1)$ and $A_2(x_2)$ should be optimized. Up to now, it has been studied by many scholars. Likewise, in order to solve this problem, Chen and Xu [53] proposed the balanced variant of WOA, that is BWOA. Gandomi *et al.* [11] applied CS to solve it. Zhang *et al.* [54] proposed an improved DE(DEDS). Sadollah *et al.* [55] utilized Mine blast algorithm(MBA) to solve it.

The optimization problem can be written as follows:

$$\min f(x) = (2\sqrt{2}x_1 + x_2) \times l$$

subject to

$$\begin{aligned} g_1(x) &= P(\sqrt{2}x_1 + x_2)/(\sqrt{2}x_1^2 + 2x_1x_2) - \sigma \leq 0, \\ g_2(x) &= Px_2/(\sqrt{2}x_1^2 + 2x_1x_2) - \sigma \leq 0, \\ g_3(x) &= P/(\sqrt{2}x_2 + x_1) - \sigma \leq 0, \end{aligned}$$

where

$$\begin{aligned} &0 \leq x_1 \leq 1, \\ &0 \leq x_2 \leq 1, \\ &l = 100cm, \quad P = 2kN/cm^2, \ \sigma = 2kN/cm^2. \end{aligned}$$

The results obtained by the above-mentioned algorithms and ERaFA are shown in Table 7. Observing the Table 7, the best result obtained by these algorithm is 263.8958433 when x_1 and x_2 are set as 0.788675594564431, 0.408246989474874, respectively, which is obtained by ERaFA. And the results obtained by other algorithms are all worse than ERaFA's.

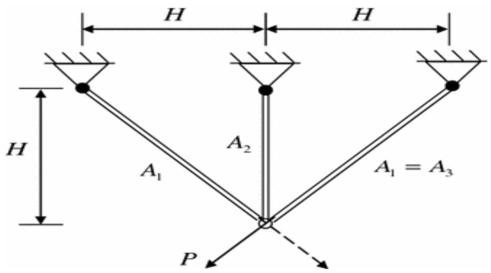

FIGURE 3. Three-bar truss design.

B. I-BEAM DESIGN PROBLEM

For I-beam design problem(see Fig. 4), its aim is to minimize the vertical deflection of an I-beam. Meanwhile, The cross-sectional area and stress constraints should be satisfied. there are 4 variables: length(b), height(h), and two thick-nesses of this problem(t_w, t_f). For convenience, we set [b, h, t_w, t_f] = [x_1, x_2, x_3, x_4].

The optimization problem can be written as follows:

$$\min f(x) = 5000/(x_3(x_2 - 2x_4)/12 + x_1x_4^3/6 + 2x_1x_4((x_2 - x_4)/2)^2)$$

subject to

$$\begin{aligned} g_1(x) &= 2x_1x_3 + x_3(x_2 - 2x_4 - 300 \leq 0, \\ g_2(x) &= 18x_2 \times 10^4/(x_3(x_2 - 2x_4)^3 + 2x_1x_3(4x_4^2 \\ &\quad + 3x_2(x_2 - 2x_4))) + 15x_1 \times 10^3/((x_2 - 2x_4)x_3^3 \\ &\quad + 2x_3x_1^3) - 56 \leq 0, \end{aligned}$$

where

$$\begin{aligned} 10 &\leq x_1 \leq 50, \\ 10 &\leq x_2 \leq 80, \\ 0.9 &\leq x_3 \leq 5, \\ 0.9 &\leq x_4 \leq 5. \end{aligned}$$

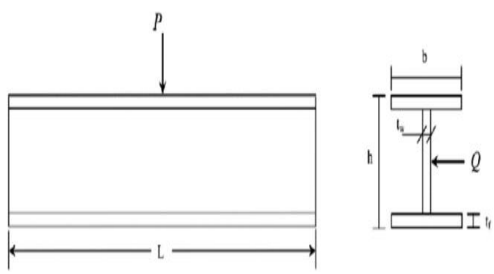

FIGURE 4. I-beam design problem.

The results obtained by CS [11], MFO [56], WOA [57], BWOA [53] and ERaFA are shown in Table 8, analysing the statistical data shown in Table 8, the optimal value obtained

TABLE 7. Comparison the best solution obtained by different algorithms for three-bar truss design.

Optimum variables	BWOA	CS	DEDS	MBA	ERaFA
x_1	0.788666327	0.78867	0.78867513	0.7885650	**0.78867559**
x_2	0.408273202	0.40902	0.40824828	0.4085597	**0.40824698**
$f(x)$	263.8958435	263.9716	263.8958434	263.8958522	**263.8958433**

TABLE 8. Comparison the best solution obtained by different algorithms for I-beam design problem.

Optimum variables	CS	MFO	WOA	BWOA	ERaFA
x_1	50	50	49.99799	**50**	**50**
x_2	80	80	80	**80**	**80**
x_3	0.9	1.7647	1.7647477	**1.76470588**	**1.76470588**
x_4	2.321675	5	5	**5**	**5**
$f(x)$	0.0130747	0.0066259	0.00662619	**0.00625958**	**0.00625958**

TABLE 9. Comparison the best solution obtained by different algorithms for welded beam design problem.

Optimum variables	CPSO	RO	HGA	BWOA	ERaFA
x_1	0.202369	0.205700	0.203687	0.205829	**0.205729**
x_2	3.544214	3.470500	3.528467	3.251922	**3.253120**
x_3	9.048210	9.036600	9.004233	9.034556	**9.036623**
x_4	0.205723	0.205700	0.207241	0.205829	**0.205729**
$f(x)$	1.728024	1.724852	1.735344	1.695620	**1.695247**

by ERaFA is 0.00625958, which is the same as that of BWOA, but is much better than others'.

C. WELDED BEAM DESIGN PROBLEM

Welded beam design problem was firstly proposed by Coello [58], and it aims at minimizing manufacturing cost of the welded beam, which is constrained on shear stress(τ), end deflection of the beam(δ), buckling load on the bar (P_c), and bending stress(σ). Moreover, there are four design parameters: $h(x_1)$, $l(x_2)$, $t(x_3)$, $b(x_4)$.

The optimization problem can be written as follows:

$$\min f(x) = 1.10471x_1^2\, x_2 + 0.04811\, x_3x_4(14.0 + x_2)$$

subject to

$$\begin{aligned}
g_1(x) &= \tau(x) - \tau_{max} \le 0,\\
g_2(x) &= \sigma(x) - \sigma_{max} \le 0,\\
g_3(x) &= x_1 - x_4 \le 0,\\
g_4(x) &= 0.10471x_1^2 + 0.04811x_3x_4(14.0 + x_2) - 5.0 \le 0,\\
g_5(x) &= 0.125 - x_1 \le 0,\\
g_6(x) &= \delta(x) - \delta_{max} \le 0,\\
g_7(x) &= P - P_c \le 0,
\end{aligned}$$

where

$$\begin{aligned}
\tau(x) &= \sqrt{(\tau')^2 + 2\tau'\tau''\frac{x_2}{2R} + (\tau'')^2},\\
\tau' &= \frac{P}{\sqrt{2}x_1x_2},\\
\tau'' &= \frac{MR}{J},\\
M &= P(L + \frac{x_2}{2}),\\
R &= \sqrt{\frac{x_2^2}{4} + (\frac{x_1 + x_3}{2})^2},\\
J &= 2[\sqrt{2}x_1x_2(\frac{x_2^2}{12} + (\frac{x_1 + x_3}{2})^2)],\\
\sigma(x) &= \frac{6PL}{x_4x_3^2},\\
\delta(x) &= \frac{4PL^3}{Ex_3^3x_4},\\
P_c &= \frac{4.013E\sqrt{\frac{x_3^2x_4^6}{36}}}{L^2}(1 - \frac{x_3}{2L}\sqrt{\frac{E}{4G}}),\\
P &= 6000lb, \quad L = 14in, \ E = 30e6psi,\\
G &= 12e6psi, \quad \tau_{max} = 13600psi,\\
\sigma_{max} &= 3000psi, \quad \delta_{max} = 0.25in,\\
&0.1 \le x_1 \le 2.0, \quad 0.1 \le x_2 \le 10.0,\\
&0.1 \le x_3 \le 10.0, \quad 0.1 \le x_4 \le 2.0.
\end{aligned}$$

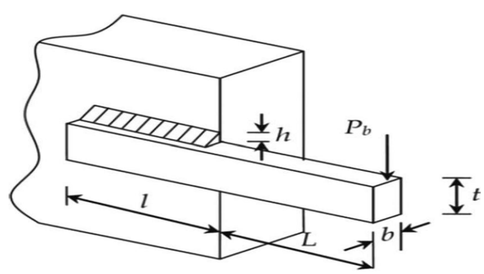

FIGURE 5. Structure design of welded beam design problem.

For this problem, ERaFA is compared with BWOA [53], CPSO [59], RO [60] and HGA [61], and their results are shown in Table 9. Observing the statistical data in Table 9, we know that the best solution is 1.695247, and its corresponding four variables are 0.205729, 3.253120, 9.036623 and 0.205729. which are obtained by ERaFA.

VI. CONCLUSION

In this paper, in order to enhance the optimization accuracy of FA, speed up the convergence, reduce computational time complexity and avoid oscillation in the iteration, an improved firefly algorithm (ERaFA) was presented. It mainly used an elitist strategy, an opposite leaning strategy, and a local search ability. Comparison with the standard FA and some other FA variants show that the performance of ERaFA is superior to the others on most benchmark test functions. Besides, ERaFA is applied to three practical problem: Three-bar truss design, I-beam design problem and Welded beam design problem. And the results show that the ERaFA is efficient.

With the time going by, multi-objective optimization problems become more popular. In the future, ERaFA can be used to deal with them. And from above simulation results, we can see that, for some functions, ERaFA can not find their optimal values. And ERaFA lacks knowledge of mathematical theory. Thus, there are many works that we will do.

REFERENCES

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Netw. (ICNN)*, vol. 4, Nov./Dec. 1995, pp. 1942–1948.
[2] M. Taherkhani and R. Safabakhsh, "A novel stability-based adaptive inertia weight for particle swarm optimization," *Appl. Soft. Comput.*, vol. 38, pp. 281–295, Jan. 2016.
[3] B. Jiao, Z. Lian, and X. Gu, "A dynamic inertia weight particle swarm optimization algorithm," *Chaos, Solitons Fractals*, vol. 37, no. 3, pp. 698–705, Aug. 2008.
[4] M. S. Kıran, H. Hakli, M. Gunduz, and H. Uguz, "Artificial bee colony algorithm with variable search strategy for continuous optimization," *Inf. Sci.*, vol. 300, pp. 140–157, Apr. 2015.
[5] J. Luo, Q. Wang, and X. H. Xiao, "A modified artificial bee colony algorithm based on converge–onlookers approach for global optimization," *Appl. Math. Comput.*, vol. 219, no. 20, pp. 10253–10262, Jun. 2013.
[6] D. Karaboga, B. Gorkemli, C. Ozturk, and N. Karaboga, "A comprehensive survey: Artificial bee colony (ABC) algorithm and applications," *Artif. Intell. Rev.*, vol. 42, no. 1, pp. 21–57, 2014.
[7] T. J. Liao, T. Stützleb, M. A. M. de Oca, and M. Dorigo, "A unified ant colony optimization algorithm for continuous optimization," *Eur. J. Oper. Res.*, vol. 234, no. 3, pp. 597–609, May 2014.
[8] U. M. Diwekar and B. H. Gebreslassie, "Efficient ant colony optimization (EACO) algorithm for deterministic optimization," *Int. J. Swarm Intell. Evol. Comput.*, vol. 5, no. 2, pp. 1–10, Mar. 2016.
[9] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Trans. Evol. Comput.*, vol. 13, no. 2, pp. 398–417, Sep. 2009.
[10] D. Manjarres, I. Landa-Torres, S. Gil-Lopez, J. Del Ser, M. N. Bilbao, S. Salcedo-Sanz, and Z. W. Geem, "A survey on applications of the harmony search algorithm," *Eng. Appl. Artif. Intell.*, vol. 26, no. 8, pp. 1818–1831, 2013.
[11] A. H. Gandomi, X.-S. Yang, and A. H. Alavi, "Cuckoo search algorithm: A metaheuristic approach to solve structural optimization problems," *Eng. Comput.*, vol. 29, no. 1, pp. 17–35, Jan. 2013.
[12] K. Deep and M. Thakur, "A new mutation operator for real coded genetic algorithms," *Appl. Math. Comput.*, vol. 193, no. 1, pp. 211–230, Oct. 2007.
[13] L. Manuel, M. Laguna, R. Martí, F. J. Rodríguez, and C. García-Martínez, "A genetic algorithm for the minimum generating set problem," *Appl. Soft Comput.*, vol. 48, pp. 254–264, Nov. 2016.
[14] X. S. Yang, *Nature-Inspired Metaheuristic Algorithms*. London, U.K.: Luniver Press, 2008.
[15] X.-S. Yang, "Multiobjective firefly algorithm for continuous optimization," *Eng. Comput.*, vol. 29, no. 2, pp. 175–184, 2013.
[16] N. J. Cheung, X.-M. Ding, and H.-B. Shen, "A non-homogeneous firefly algorithm and its convergence analysis," *J. Optim. Theory Appl.*, vol. 170, no. 2, pp. 616–628, Aug. 2016.
[17] A. H. Gandomi, X. S. Yang, and A. H. Alavi, "Mixed variable structural optimization using firefly algorithm," *Comput. Struct.*, vol. 89, nos. 23–24, pp. 2325–2336, Dec. 2013.
[18] A. Kazem, E. Sharifi, F. K. Hussain, M. Saberi, and O. K. Hussain, "Support vector regression with chaos-based firefly algorithm for stock market price forecasting," *Appl. Soft Comput.*, vol. 13, no. 2, pp. 947–958, 2013.
[19] M. K. Marichelvam, T. Prabaharan, and X. S. Yang, "A discrete firefly algorithm for the multi-objective hybrid flowshop scheduling problems," *IEEE Trans. Evol. Comput.*, vol. 18, no. 2, pp. 301–305, Apr. 2014.
[20] H. Wang, W. Wang, Z. Cui, X. Zhou, J. Zhao, and Y. Li, "A new dynamic firefly algorithm for demand estimation of water resources," *Inf. Sci.*, vol. 438, pp. 95–106, Apr. 2018.
[21] R. Sawhney, P. Mathur, and R. Shankar, "A firefly algorithm based wrapper-penalty feature selection method for cancer diagnosis," in *Computational Science and Its Applications* (Lecture Notes in Computer Science), vol. 10960, O. Gervasi, B. Murgante, S. Misra, E. Stankova, C. M. Torre, A. M. A. C. Rocha, D. Taniar, B. O. Apduhan, E. Tarantino, and Y. Ryu, Eds. Cham, Switzerland: Springer, 2008.
[22] N. J. Cheung, X.-M. Ding, and H.-B. Shen, "Adaptive firefly algorithm: Parameter analysis and its application," *PLoS ONE*, vol. 9, no. 11, Nov. 2014, Art. no. e112634.
[23] X. H. Yan, Y. L. Zhu, J. W. Wu, and H. Chen, "An improved firefly algorithm with adaptive strategies," *Adv. Sci. Lett.*, vol. 16, no. 1, pp. 249–254, Sep. 2012.
[24] A. Gálvez and A. Iglesias, "New memetic self-adaptive firefly algorithm for continuous optimisation," *Int. J. Bio-Inspired Comput.*, vol. 8, no. 5, pp. 300–317, Jan. 2016.
[25] H. Wang, X. Zhou, H. Sun, X. Yu, J. Zhao, H. Zhang, and L. Cui, "Firefly algorithm with adaptive control parameters," *Soft Comput.*, vol. 21, no. 17, pp. 5091–5102, Sep. 2016.
[26] S. H. Yu, S. L. Yang, and S. B. Su, "Self-adaptive step firefly algorithm," *J. Appl. Math.*, vol. 2013, Sep. 2013, Art. no. 832718.
[27] H. Wang, W. Wang, H. Sun, J. Zhao, X. Yu, L. Lv, and H. Zhu, "Adaptive firefly algorithm with alternative search," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Vancouver, BC, Canada, Jul. 2016, pp. 1779–1785.
[28] S. M. Farahani, A. Abshouri, B. Nasiri, and M. Meybodi, "A Gaussian firefly algorithm," *Int. J. Mach. Learn. Comput.*, vol. 1, no. 5, pp. 448–454, Jan. 2011.
[29] X. S. Yang, "Firefly algorithm, Levy flights and global optimization," in *Research and Development in Intelligent Systems*, M. Bramer, R. Ellis, and M. Petridis, Eds. London, U.K.: Springer, 2010, pp. 209–218.
[30] L. D. S. Coelho, D. L. de Andrade Bernert, and V. C. Mariani, "A chaotic firefly algorithm applied to reliability-redundancy optimization," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, vol. 18, Jun. 2013, pp. 517–521.
[31] A. H. Gandomi, X-S. Yang, S. Talatahari, and A. H. Alavi, "Firefly algorithm with chaos," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 18, no. 1, pp. 89–98, Jan. 2013.
[32] H. Wang, Z. Cui, H. Sun, S. Rahnamayan, and X.-S. Yang, "Randomly attracted firefly algorithm with neighborhood search and dynamic parameter adjustment mechanism," *Soft Comput.*, vol. 21, no. 18, pp. 5325–5339, 2017.
[33] A. Yelghi and C. Köse, "A modified firefly algorithm for global minimum optimization," *Appl. Soft Comput.*, vol. 62, pp. 29–44, Jan. 2018.
[34] L. N. Zhang, L. Q. Liu, X. S. Yang, and Y. T. Dai, "A novel hybrid firefly algorithm for global optimization," *PLoS ONE*, vol. 11, no. 9, Sep. 2016, Art. no. e0163230.
[35] W. Long and T. B. Wu, "A hybrid firefly algorithm for constrained optimization and engineering application," in *Proc. Int. Conf. Electron. Sci. Automat. Control (ESAC)*, 2015, pp. 158–162.
[36] F. Tang and J. Tang, "Firefly algorithm with hybrid mutation strategies," *Int. J. Wireless Mobile Comput.*, vol. 11, no. 2, pp. 166–170, Jan. 2016.
[37] L. H. Guo, G. G. Wang, H. Q. Wang, and D. N. Wang, "An effective hybrid firefly algorithm with harmony search for global numerical optimization," *Sci. World J.*, vol. 2013, Sep. 2013, Art. no. 125625.
[38] A. Abdullah, S. Deris, M. S. Mohamad, and S. Z. M. Hashim, "A new hybrid firefly algorithm for complex and nonlinear problem," in *Distributed Computing and Artificial Intelligence*, vol. 151. Berlin, Germany: Springer 2012, pp. 673–680.
[39] H. Wang, W. Wang, X. Zhou, H. Sun, J. Zhao, X. Yu, and Z. Cui, "Firefly algorithm with neighborhood attraction," *Inf. Sci.*, vols. 382–383, pp. 374–387, Mar. 2017.

[40] H. Wang, W. Wang, H. Sun, and S. Rahnamayan, "Firefly algorithm with random attraction," *Int. J. Bio-Inspired Comput.*, vol. 8, no. 1, pp. 33–41, Feb. 2016.
[41] S. Das, A. Abraham, U. K. Chakraborty, and A. Konar, "Differential evolution using a neighborhood-based mutation operator," *IEEE Trans. Evol. Comput.*, vol. 13, no. 3, pp. 526–553, Jun. 2009.
[42] J. Liang, B. Qu, and P. Suganthan, "Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization," Comput. Intell. Lab., Zhengzhou Univ., Zhengzhou, China, Tech. Rep. 201411A, 2014.
[43] S. Yu, S. Su, Q. Lu, and L. Huang, "A novel wise step strategy for firefly algorithm," *Int. J. Comput. Math.*, vol. 91, no. 12, pp. 2507–2513, May 2014.
[44] S. Yu, S. Zhu, Y. Ma, and D. Mao, "A variable step size firefly algorithm for numerical optimization," *Appl. Math. Comput.*, vol. 263, pp. 214–220, Jul. 2015.
[45] S. Arunachalam, T. AgnesBhomila, and M. R. Babu, "Hybrid particle swarm optimization algorithm and firefly algorithm based combined economic and emission dispatch including valve point effect," in *Proc. Int. Conf. Swarm, Evol., Memetic Comput.*, Lecture Notes in Computer Science. Cham, Switzerland: Springer, 2015, pp. 647–660.
[46] P. Kora and K. S. R. Krishna, "Hybrid firefly and particle swarm optimization algorithm for the detection of bundle branch block," *Int. J. Cardiovascular Acad.*, vol. 2, no. 1, pp. 44–48, Mar. 2016.
[47] I. B. Aydilek, "A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems," *Appl. Soft Comput.*, vol. 66, pp. 232–249, May 2018.
[48] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: Artificial bee colony (ABC) algorithm," *J. Global Optim.*, vol. 39, no. 3, pp. 459–471, Apr. 2007.
[49] Y. H. Zheng, L. Xia, and Q. C. Yu, "A method for identifying three-dimensional rock blocks formed by curved fractures," *Comput. Geotechn.*, vol. 65, pp. 1–11, Apr. 2015.
[50] B. Zhang, Y.-J. Zheng, M.-X. Zhang, and S.-Y. Chen, "Fireworks algorithm with enhanced fireworks interaction," *IEEE/ACM Trans. Comput. Biol. Bioinf.*, vol. 14, no. 1, pp. 42–55, Jan. 2017.
[51] A. Sajwan and A. Yadav, "AEFA: Artificial electric field algorithm for global optimization," *Swarm Evol. Comput.*, vol. 48, pp. 93–108, Aug. 2019.
[52] E. Mezura-Montes and C. A. C. Coello, "An empirical study about the usefulness of evolution strategies to solve constrained optimization problems," *Int. J. Gen. Syst.*, vol. 37, no. 4, pp. 443–473, Jul. 2008.
[53] H. Chen, Y. Xu, M. Wang, and X. Zhao, "A balanced whale optimization algorithm for constrained engineering design problems," *Appl. Math. Model.*, vol. 71, pp. 45–59, Jul. 2019.
[54] M. Zhang, W. Luo, and X. Wang, "Differential evolution with dynamic stochastic selection for constrained optimization," *Inf. Sci.*, vol. 178, no. 15, pp. 3043–3074, Aug. 2008.
[55] A. Sadollah, A. Bahreininejad, H. Eskandar, and M. Hamdi, "Mine blast algorithm: A new population based algorithm for solving constrained engineering optimization problems," *Appl. Soft Comput.*, vol. 13, no. 5, pp. 2592–2612, May 2013.
[56] S. Mirjalili, "Moth-flame optimization algorithm: A novel nature-inspired heuristic paradigm," *Knowl.-Based Syst.*, vol. 89, pp. 228–249, Nov. 2015.
[57] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Adv. Eng. Softw.*, vol. 95, pp. 51–67, May 2016.
[58] C. A. C. Coello, "Use of a self-adaptive penalty approach for engineering optimization problems," *Comput. Ind.*, vol. 41, no. 2, pp. 113–127, Mar. 2000.
[59] R. A. Krohling and L. D. S. Coelho, "Coevolutionary particle swarm optimization using Gaussian distribution for solving constrained optimization problems," *IEEE Trans. Syst., Man, Cybern. B, Cybern.*, vol. 36, no. 6, pp. 1407–1416, Dec. 2006.
[60] A. Kaveh and M. Khayatazad, "A new meta-heuristic method: Ray optimization," *Comput. Struct.*, vols. 112–113, pp. 283–294, Dec. 2012.
[61] Q. Yuan and F. Qian, "A hybrid genetic algorithm for twice continuously differentiable NLP problems," *Comput. Chem. Eng.*, vol. 34, pp. 36–41, Jan. 2010.

CHUNFENG WANG was born in Kaifeng, Henan, China, in 1978. He received the M.S. degree from Henan Normal University, in 2006, and the Ph.D. degree from Xidian University, in 2012. He has published over 40 articles in domestic and foreign academic journals, applied for one national invention patents, hosted and participated national natural science foundation four projects, and hosted provincial two projects. His research interests include swarm intelligent algorithms, optimization theory and its applications, and Bayesian network learning.

KUI LIU was born in Jiaozuo, China, in 1980. He received the M.S. and Ph.D. degrees from Xidian University, in 2008 and 2013, respectively. He has published over 20 articles in domestic and foreign academic journals, applied for one national invention patents, hosted and participated two national natural science foundation projects, and hosted one provincial project.