An improved evolution fruit fly optimization algorithm and its application

Article *in* Neural Computing and Applications · July 2020

DOI: 10.1007/s00521-019-04512-2

5 authors, including:

Yang Xuan
兰州大学
5 PUBLICATIONS **150** CITATIONS
SEE PROFILE

Weide Li
Lanzhou University
40 PUBLICATIONS **505** CITATIONS
SEE PROFILE

Ailing Yang
Lanzhou University
2 PUBLICATIONS **104** CITATIONS
SEE PROFILE

Neural Computing and Applications
https://doi.org/10.1007/s00521-019-04512-2

ORIGINAL ARTICLE

An improved evolution fruit fly optimization algorithm and its application

Xuan Yang[1] · Weide Li[1] · Lili Su[1] · Yaling Wang[1] · Ailing Yang[1]

Received: 27 March 2019 / Accepted: 21 September 2019

Abstract
Fruit fly optimization algorithm (FOA) is a kind of swarm intelligence optimization algorithm, which has been widely applied in science and engineering fields. The aim of this study is to design an improved FOA, namely evolution FOA (EFOA), which can overcome some shortcomings of basic FOA, including difficulty in local optimization, slow convergence speed, and lack of robustness. EFOA applies a few new strategies which adaptively control the search steps and swarm numbers of the fruit flies. The evolution mechanism used in EFOA can preserve dominant swarms and remove inferior swarms. Comprehensive comparison experiments are performed to compare EFOA with other swarm intelligence algorithms through 14 benchmark functions and a constrained engineering problem. Experimental results suggest that EFOA performs well both in global search ability and in robustness, and it can improve convergence speed.

Keywords Fruit fly optimization algorithm · Evolution mechanism · Swarm intelligence

1 Introduction

In recent years, meta-heuristic algorithms have attracted more and more attention from the academic and engineering circles. Meta-heuristic algorithms can help discover optimal solutions to functions or engineering problems. Therefore, some meta-heuristic algorithms are proposed, such as particle swarm optimization algorithm (PSO), fruit fly optimization algorithm (FOA), artificial bee colony algorithm (ABC), and gravitational search algorithm (GSA). The application of meta-heuristic algorithm in engineering field becomes more and more popular among many scholars. In the literature [1], firefly algorithm (FA) is employed to search for optimal proportional integral (PI) parameters of speed controller by minimizing the time domain objective function. FA algorithm is also used to optimize tuning of PI controllers for load frequency control of hybrid system composing of photovoltaic (PV) system and thermal generator [2]. Oshaba et al. [3] introduced BAT algorithm to search for optimal parameters of PI controller to solve the problem of maximum power point tracking (MPPT). ABC is introduced to optimize the parameters for hydrological models [4]. Chen et al. [5] combined PSO and back propagation artificial neural network to predict PKa values of neutral and alkaline drugs.

Of swarm intelligence algorithms, FOA has recently drawn much attention. FOA was first proposed by Pan [6, 7], which was inspired by fruit flies' foraging behavior to solve optimization problems. FOA is an optimization algorithm of swarm intelligence and has many advantages including simple principle, fewer adjusting parameters, and easy to understand, compared to other optimization algorithms [8–10]. Therefore, FOA has been successfully used in a wide range of applications including neural network parameter optimization [6, 11, 12], proportional–integral–derivative (PID), controller parameter

✉ Weide Li
weideli@lzu.edu.cn

Xuan Yang
yangxuan14@lzu.edu.cn

Lili Su
sull15@lzu.edu.cn

Yaling Wang
wangyaling14@lzu.edu.cn

Ailing Yang
yangal15@lzu.edu.cn

[1] School of Mathematics and Statistics, Lanzhou University, No. 222, TianShui Road (South), ChengGuan District, Lanzhou City 730000, Gansu Province, China

Published online: 03 October 2019

tuning [13], multidimensional knapsack problem [14], and many other fields. However, the basic FOA has its limitations in that it has some obstacles for successful solving of complex practical or high-dimensional continuous function optimization problems [15]. The insufficiency of the search efficiency, convergence speed, global search ability, and the arbitrariness of search direction of the fruit fly are all such obstacles. In addition, FOA is not efficient to get an optimal solution with high precision for the optimization problem with large-scale parameters, combinational optimization problems, and high-dimensional continuous functions [15, 16].

To further improve the expressive force of FOA, including fruit fly search efficiency, global search ability, and robustness, researchers have designed a variety of improved FOA methods. Shan et al. [17] proposed an improved FOA called LGMS-FOA (FOA based on linear generation mechanism of the candidate solution), in which the fruit fly individuals were generated by a linear mechanism. In 2012, another improved fruit fly optimization algorithm G-FOA was presented by Xu et al. [18] to determine the optimal value of a bi-variable nonlinear function, and the authors stated that their method can find the global optimum quicker and can get more accurate solution than the basic FOA and can avoid falling into a local situation. Wu et al. [19] described a new mechanism in SEDI-FOA (FOA based on intelligent selection of the evolutionary direction), in which the fruit fly has a correct direction for updating. The advantage of SEDI-FOA is that the search direction of population individuals is selected intelligently according to a mechanism, and there is a correct direction for the fruit fly to save time to reach the best optimization value. Other modifications of FOA have also been developed. Xiao et al. [20] adopted a cell communication mechanism to modify the basic FOA (CFOA) by including the global worst, the average, and the best solution into the search strategy to improve exploitation ability of fruit fly swarm. Yuan et al. [21] proposed a variation on naive FOA and MFOA (multi-swarm fruit fly optimization algorithm), using multi-swarm to optimize FOA. In MFOA, several sub-swarms move independently in the search space to simultaneously explore the global optimum at the same time and the local behaviors between sub-swarms are also considered. Wang et al. [16] proposed LP-FOA (level probability fruit fly optimization algorithm) in 2015 to solve high-dimensional optimization problem of continuous function and handle joint replenishment problems (JRPs). Meanwhile, a level probability policy and a new mutation parameter were introduced in LP-FOA to balance the diversity of population and stability. Wang et al. [22] presented an improved FOA (IFOA) that contains a length parameter to avoid the acquisition of a local optimal solution. Pan et al. [15] proposed IFFO (improved fruit fly optimization). In IFFO, a control parameter was introduced to adaptively tune the search scope around the swarm location and a solution-generating method was developed to enhance the accuracy and convergence rate of the algorithm. In [23], to further improve the searching performance of FOA and use it for aerodynamic optimization design, a new algorithm named improved fruit fly optimization algorithm (IFOA) is presented. The search step is modified by introducing an inertia weight function to IFOA, and the dynamical balance between the global search and the local search is satisfied. The searching efficiency and accuracy of the algorithm are integrally improved. Du et al. [24] proposed an improved FOA (DSLC-FOA) based on linear diminishing step and logistic chaos mapping to solve benchmark functions, unconstrained optimization problems, and constrained structural engineering design problems. This paper uses chaos mapping to deal with initial parameter and uses linear diminishing step to avoid far away from the optimal solution. This improvement makes DSLC-FOA more stable and effective. An improved fruit fly optimization algorithm (AE-LGMS-FOA) is proposed to be used in antenna array synthesis [25]. This improvement includes adding a new search mechanism to enhance the efficiency of algorithm for high-dimensional problems. Overall, the past studies have been extensive and the recently developed improved FOA methods are more superior in multiple aspects than the original FOA.

There are some shortcomings in the original FOA and later on improved FOAs, such as easy to generate local optimization solutions, slow convergence, and lack of robustness. When using the original FOA and some improved FOAs to address more complex optimization problems, we found significant limitations in accuracy or robustness. However, these optimization algorithms are needed to deal with some complex and difficult problems in scientific research and real life. Inspired by the FOAs themselves and their application flaws, a further improved FOA, evolution FOA (EFOA), is proposed in this paper. The motivation of EFOA is inspired by the elimination and evolution mechanism of biological competition. The number of fruit fly swarms is adaptively decreased with the number of iterations increasing. EFOA has higher probability jumping out of local optimum, because it keeps individual diversity and population diversity by generating multi-swarms in the early and middle phase searching. EFOA is not blind to increase the number of fruit fly swarms. The evolution mechanism selects the dominant fruit fly swarm and eliminates the inferior one in EFOA. What is more, a dynamical searching radius with the feature that from big to small (A big searching step equals to the distance of parameter range in first searching, which can improve the global search ability; a small searching step in end stage can improve the local search ability.) in iteration is adopted to overcome falling into local optimum. These modifications have made the EFOA algorithm more efficient with faster convergence, better robustness, and easier to find the global optimal solution as solving optimization problems. The motivation of this study can be answering as the following questions:

1. Whether multi-swarm mechanism is a good way to overcome being trapped in the local optimal problem?
2. How does it express about the converging rate and the global optimum extraction ability of EFOA compared with other meta-heuristic algorithms?
3. How does EFOA control the number of multi-swarms in iteration?
4. Whether EFOA can solve unconstrained and constrained optimization problems?

In this study, 14 benchmark testing functions and one constrained engineering problem are selected as testing functions to compare accuracy, robustness, and convergence speed in different methods. The efficiency and the global search ability of EFOA are verified by a simulation experiment. EFOA is not only compared with basic FOA but also with other conventional meta-heuristic methods like PSO, ABC, GSA and two improved FOA (LGMS-FOA and IFFO). The results prove that EFOA performs well. It is more precise and faster than the previous methods. What's more, EFOA functions better in finding the global optimum solution with higher probability than the earlier methods.

The main structure of this paper is organized as follows: Sect. 2 describes the basic FOA. Section 3 introduces EFOA in detail, and the differences between EFOA and FOA are analyzed. The simulation configuration, experimental results, and analysis of 14 benchmark testing functions are given in Sect. 4. In Sect. 5, EFOA is used to deal with a constraint engineering problem called pressure vessel design. The discussions and conclusions of this study are presented in Sect. 6. The key nomenclatures used in this paper are listed in Table 1.

Table 1 List of abbreviations

Abbreviations	The full name
FOA	Fruit fly optimization algorithm
EFOA	Evolution fruit fly optimization algorithm
LGMS-FOA	Fruit fly optimization algorithm based on linear generation mechanism of the candidate solution
IFFO	Improved fruit fly optimization
PSO	Particle swarm optimization algorithm
ABC	Artificial bee colony algorithm
GSA	Gravitational search algorithm

2 The basic FOA and analysis

2.1 The basic FOA process

The fruit fly optimization algorithm is a swarm intelligence algorithm, similar to the ant colony optimization algorithm (ACO) [26, 27], particle swarm optimization algorithm (PSO) [28], and others inspired by the foraging behavior of groups. The fruit fly may be more sensitive than other species, especially their senses of smell and vision. According to the previous studies [6], fruit flies are very sensitive to smells in the air and can smell 40 km from the source of food. Pan proposed the basic FOA based on these sophisticated osphresis and vision characteristics of fruit fly. The method of FOA requires the keen sense of osphresis to smell the food source so that all fly swarms can obtain the approximate location of food. Approaching the food location, the sensitive vision of flies allows sequentially finding food. In general, the method of FOA mainly includes the following four parts, global optimization problem initialization and algorithm parameters introduction, osphresis foraging phase, vision foraging phase, and population evaluation. According to previous works [6], the optimization procedure of FOA can be summarized as follows (Sects. 2.1.1–2.1.6).

2.1.1 Initialize the global optimization problem and then introduce algorithm parameters

Generally, the global optimization problem can be noted as Eq. 1:

$$\min_{x_j \in [\mathrm{LB_j}, \mathrm{UB_j}], j=1,2,\ldots,n} f(X). \tag{1}$$

In Eq. 1, $f(X)$ is the objective optimization function, $X = (x_1, x_2, \ldots, x_n)$ the decision variables, n the dimension of the X representing the number of decision variables. And LB_j and UB_j are the lower and upper bounds of the jth dimension value x_j, respectively.

The FOA method is a type of meta-heuristic algorithm, which generates a number of trial solutions to approach the optimal solution by iteration using a evolutionary mechanism. The population size (PS) (the number of trail solutions in each iteration) and the maximum number of iterations ($\mathrm{Iter}_{\max}$) are two important parameters, and they can influence FOA to get out of a local optimum and search for better solutions. In general, if the two parameters are initialized with large values in FOA , it will take more time to search. But more accurate result could be obtained.

2.1.2 Initialize the fruit fly swarm location

$\delta = (\delta_1, \delta_2, \ldots, \delta_n)$ is the randomly initialized swarm center location of the fruit flies in the search space, and the initialization formula can be written as Eq. 2:

$$\delta_j = \mathrm{LB}_j + (\mathrm{UB}_j - \mathrm{LB}_j) \times \mathrm{rand}(), j = 1, 2, \ldots, n. \quad (2)$$

where rand() is a function which returns a value from the uniform distribution [0,1].

2.1.3 Osphresis searching phase

In the osphresis searching phase, fruit fly groups are generated randomly around the current fruit fly swarm center location δ. Use $\{X_1, X_2, \ldots, X_{\mathrm{PS}}\}$ to present generated fruit fly groups, where $X_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,n})$, $(i = 1, 2, \ldots, \mathrm{PS})$ is the ith fruit fly individual, and it is yielded as Eq. 3:

$$x_{i,j} = \delta_j \pm \mathrm{rand}(), j = 1, 2, \ldots, n. \quad (3)$$

2.1.4 Vision searching phase

The vision searching phase is a key part for FOA. A sharp vision is required to find the best food source in this phase. X_{best} represents the individual which has the best performance [the minimum value of $f(X)$] in local fruit fly swarms, and it is described by Eq. 4:

$$X_{\mathrm{best}} = \operatorname*{argmin}_{i=1,2,\ldots,\mathrm{ps}} f(X_i). \quad (4)$$

The FOA can update the best food position in the next iteration. If X_{best} is better than the current fruit fly swarm center location δ, then this position will take the place of the swarm center location and become a new one in the next iteration, i.e., $\delta = X_{\mathrm{best}}$, if $f(X_{\mathrm{best}}) < f(\delta)$.

Algorithm 1: The FOA algorithm

```
    Input: input parameters PS and $Iter_{max}$
    Output: output result: solution $X^*$
 1  Initialization Iter=0, PS, $Iter_{max}$ and
      $\delta = (\delta_1, \delta_2, \cdots, \delta_n)$;//$\delta_j = LB_j + (UB_J - LB_j) \times rand(0,1), i = 1, 2, \cdots, n$;
 2  $X^* = \delta$
 3  repeat
 4      Osphresis searching phase                    /* Osphresis searching phase */
 5      for $i < PS$ do
 6          for $j < n$ do
 7              $x_{ij} = \delta_j + rand(-1, 1)$
 8              if $x_{ij} > UB_j$ then
 9                  $x_{ij} = UB_j$;
10              end
11              if $x_{ij} < LB_j$ then
12                  $x_{ij} = LB_j$;
13              end
14          end
15          $X_i = (x_{i1}, x_{i2}, \cdots, x_{in})$
16      end
17      //Vision searching phase
18      $X_{best} = \mathrm{argmin}_{i=1,2,\ldots,ps} f(X_i)$
19      if $f(X_{best}) < f(\delta)$ then
20          $\delta = X_{best}$
21      end
22      if $f(\delta) < f(X^*)$ then
23          $X^* = \delta$
24      end
25      Iter=Iter+1
26  until $Iter > Iter_{max}$;
```

2.1.5 Check stopping criterion

If the maximum number of iterations of the optimization is achieved, the optimization process is terminated. Otherwise, the osphresis searching phase and vision foraging phases will be continued.

2.1.6 Procedure of the basic FOA

The overall procedure of FOA is described in Algorithm 1. There, $\delta = (\delta_1, \delta_2, \ldots, \delta_n)$ is the center of the locations of fruit flies at certain time step and X^* is the final location which represents the optimal solution.

3 Improvement in the fruit fly optimization algorithm (EFOA)

In this part, we will provide the principle of the evolution fruit fly optimization algorithm (EFOA).

3.1 The generating method of parameters

The biggest difference between basic FOA and EFOA is that EFOA includes a step control parameter (λ) and a swarm evolution or elimination control parameter (EC). The advantage of the new control parameters can be described as follows.

3.1.1 The setting of step control parameter (λ)

In basic FOA algorithm, from the osphresis searching phase, all new fruit flies are generated around the center location of its swarm in the narrow range such as $[-1, 1]$, $[-2, 2]$ and $[-3, 3]$, which means when the narrow range is determined, the searching step cannot pass over the radius. The main drawback of this method appears in the number of iterations in which the algorithm needs to find an optimal solution. When the searched optimal solution is in a vast domain on the early stage of iterations, the fruit fly swarm (trail solutions) is often far away from the optimal solution, resulting the FOA has a week ability to approach the optimal solution because of the too small search radius (search step). In the final generations, the swarm location is close to an optimum or near an optimum solution; a very small scope is needed for the fine-tuning of solution vectors [15]. Therefore, a dynamical searching radius with the feature that from big to small (A big searching step in early stage can improve the global search ability; a small searching step in end stage can improve the local search ability.) in iteration is an appropriate way to overcome this drawback. λ as a nonlinear attenuation function to meet this feature was initially proposed by Pan et al. [15], and it can

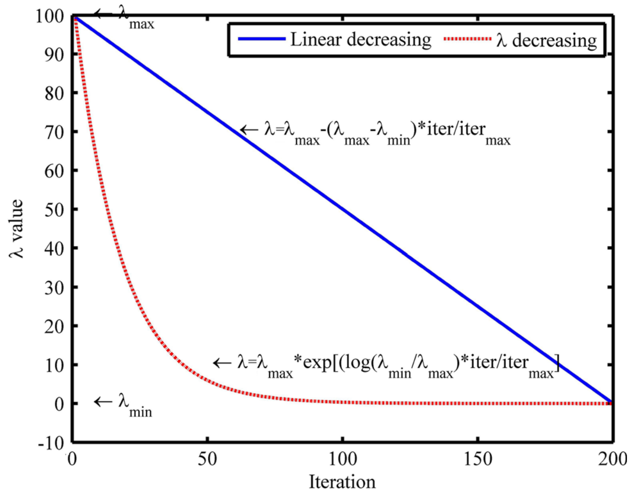

Fig. 1 Linear decreasing curve and λ decreasing curve, where $\lambda_{\max}$ is 100, $\lambda_{\min}$ 0.001, $\text{iter}_{\max}$ 200

decide the scale of step for each fruit fly adaptively. The parameter is influenced by iteration number, and it can be expressed as

$$\lambda = \lambda_{\max} \times \exp\left[\log\left(\frac{\lambda_{\min}}{\lambda_{\max}}\right) \cdot \frac{\text{Iter}}{\text{Iter}_{\max}}\right], \tag{5}$$

where λ is the search radius in each iteration and $\lambda_{\max}$ and $\lambda_{\min}$ represent the maximum and minimum radius, respectively. The changes of λ with the iteration and comparison result with another type of linear changing are shown in Fig. 1. During the initial search, each fruit fly will acquire a larger search step which can avoid falling in a local optimum value and improve the exploitation ability. At the middle and later periods of the iteration, λ decreases slower than the linear decreasing, which means that it biases toward the local accuracy search ability.

3.1.2 The swarm evolution control parameter (EC)

Pan et al. [15] added a step control parameter λ into IFFO, but only based on single swarm of fruit fly in their modification. A main shortcoming of the modification is obvious. Once single swarm is dropped into a local optimal, IFFO loses the ability to flee it because the number of flies is not enough or the searching step is too small. There is an easy way to use the multi-swarm strategy to cover the shortage, which is the idea of parallel searching. In the improved FOA (MFOA) [21], the multi-swarm strategy is used in MFOA which has more chance to generate dominant swarms. However, it also generates a lot of insignificance inferior swarms which influence the efficiency of searching in each iteration. Inspired by MFOA, we eliminate the number of fly swarms dynamically with iteration number to overcome the drawback of inferior swarms. And both the searching ability and the efficiency are improved.

EC, as an evolution or elimination parameter, is inspired by natural selection in which the inferior swarm is eliminated and the dominant swarm is preserved. In EFOA, the evolution mechanism can be considered as an elimination mechanism, and the changing curve of parameter EC is inspired by λ. In this paper, we use both terms, elimination and evolution. There is a simple transform relationship between EC and the elimination coefficient (ELC), which can be expressed as

$$\mathrm{EC} = 1 - \mathrm{ELC}. \tag{6}$$

ELC can be obtained as

$$\mathrm{ELC} = \mathrm{ELC}_{\max} \times \exp\left[\log\left(\frac{\mathrm{ELC}_{\min}}{\mathrm{ELC}_{\max}}\right) \cdot \frac{\mathrm{Iter}}{\mathrm{Iter}_{\max}}\right]. \tag{7}$$

When the search starts, many swarms will be removed because of their bad performance and the remaining advanced fly swarms will generate a new population. Repetition of the process of swarm elimination will lead to preservation of only a few swarms. The elimination procedure offers the advantage of letting EFOA jump out of the local extremum to find a better global optimum.

3.2 The complete procedure of EFOA

The implementation procedure of the proposed EFOA is illustrated in Algorithm 2, and the meaning of corresponding parameters is given in Table 2.

When using the method of EFOA, the main process can be illustrated as follows:

Step1. Randomly generate NS swarms' center locations.
Step2. Generate NS new swarms, each swarm with PS fruit flies according to the update rule of osphresis searching stage.
Step3. Select the optimal fruit fly in each swarm as a new center location of this swarm by the vision searching stage, according to the fitness function value (*f*).
Step4. Sort all new swarm center locations in ascending order according to their *f* value.
Step5. Adaptively eliminate a certain number of inferior swarms, and the remaining dominant swarms will become the next iteration swarm center locations according to the coefficient of ELC and the number of swarm locations at present.
Step6. Repeat the process of Steps 2–5 until the termination condition is satisfied. If the optimized process is terminated, the global optimum will be obtained.

Table 2 Explanation of the parameters

Parameters	PS, NS, λ, $\lambda_{\max}$, $\lambda_{\min}$, Iter, $\mathrm{Iter}_{\max}$, EC, ELC, $\mathrm{ELC}_{\max}$, $\mathrm{ELC}_{\min}$
PS	The population size of fruit fly
NS	The number of swarm
λ	The search radius in each iteration
$\lambda_{\max}$	The maximum radius
$\lambda_{\min}$	The minimum radius
Iter	The iteration number
$\mathrm{Iter}_{\max}$	The maximum iteration number
EC	The evolution coefficient
ELC	The elimination coefficient
$\mathrm{ElC}_{\max}$	The maximum eliminate coefficient
$\mathrm{ELC}_{\min}$	The minimum eliminate coefficient

Algorithm 2: The improved EFOA algotithm

```
Input: input parameters NS, PS, $Iter_{max}$, $\lambda_{max}$, $\lambda_{min}$, $ELC_{max}$, $ELC_{min}$
Output: output result: solution $X^*$
1  // Initialization
2  Set NS, PS, $Iter_{max}$, $\lambda_{max}$, $\lambda_{min}$, $ELC_{max}$, $ELC_{min}$
3  For $i = 1, 2, \ldots,$ NS
4  // Generate multiple swarm center location $X = (X_1, X_2, \ldots, X_{NS})$
5  $x_{i,j} = LB_j + (UB_j - LB_j) \times rand()\ j = 1, 2, \ldots, n$
6  End For
7  $Iter = 1$
8  $X^* = NULL$
9  Repeat
10 $\lambda = \lambda_{max} \times exp[log(\frac{\lambda_{min}}{\lambda_{max}}) \cdot \frac{Iter}{Iter_{max}}]$
11 $ELC = ELC_{max} \times exp[log(\frac{ELC_{min}}{ELC_{max}}) \cdot \frac{Iter}{Iter_{max}}]$
12 For $i = 1, 2, \ldots,$ NS
13 // Generate the new i-th population $X_i = (X_{i1}, X_{i2}, \ldots X_{iPS})$
14 // Osphresis foraging phase
15 For $j = 1, 2, \ldots,$ PS
16 d=a random integer in the range of [1,PS]
17 // Generate fruit fly $X_{i,j} = (X_{i,j,1}, X_{i,j,2}, \ldots X_{i,j,n})$, i is the i-th swarm, j is the j-th fruit fly, n is the
     maximum dimension
18 $$x_{i,j,k} = \begin{cases} x_{i,j,k} \pm \lambda \cdot rand() & \text{if k=d} \\ x_{i,j,k} & \text{otherwise} \end{cases}$$
19 if $x_{i,j,k} > UB_k$ then $x_{i,j,k} = UB_k$
20 if $x_{i,j,k} < LB_k$ then $x_{i,j,k} = LB_k$
21 End For
22 // Vision foraging phase
23 $X_{ibest} = arg(min(f(X_{ij}))),\ j = 1, 2, \ldots, PS$
24 if $X_{ibest} < f(X_i)$ then $X_i = X_{ibest},\ BestSmell(i) = f(X_{ibest})$
25 if $X_{ibest} > f(X_i)$ then $X_i = X_i,\ BestSmell(i) = f(X_i)$
26 End For
27 // Sort BestSmell in ascending order
28 $[Index, Sort\_BestSmell] = Sort(BestSmell)$
29 // Generate the NS in the next iteration
30 $NS = round[NS \times (1 - ELC)]$
31 For $i = 1, 2, \ldots,$ NS
32 $X_i = X(Index(i))$
33 End For
34 $X^* = X_1$
35 $Iter \Leftarrow Iter + 1$
36 Until $Iter = Iter_{max}$
```

3.3 The differences between EFOA and other FOA methods

The key difference between these methods lies on the application of parameters and the adoption of searching strategies. FOA adopts a conservative step which limits the searching efficiency. IFFO uses the step control parameter λ, which can moderately speed up search efficiency in the earlier stage of searching. However, EFOA not only adopts λ, and but also uses EC to adaptively weed out the inferior swarms. As the process of natural selection, only the dominant swarms can survive. Because of using parameters

EC and λ, EFOA can promote the probability of finding the global optimal value than the method of IFFO. The second difference is performed in searching mode. λ in IFFO is the adaptive control step size, and the evolution in IFFO is only based on single swarm. However, EFOA adopts multi-swarm searching mode and allows adaptive evolution and elimination of weakness swarms. Although MFOA is based on the idea of multi-swarms, MFOA does not have the elimination mechanism, which means that it may spend more time than EFOA.

4 Experimental configuration and results analysis

4.1 Experimental configuration

To test the effectiveness and efficiency, six methods (the basic FOA, LGMS-FOA, IFFO, PSO, ABC, and GSA) were adopted to compare with EFOA for 14 benchmark functions. To make comparisons more easily for the simulation experiments, some common parameters were used for all comparison algorithms: population size of 100 and the maximal number of iterations of 1000. The value of parameters following used on compared models is cited from the related published papers. Apart from the above two common parameters, the other parameters of IFFO [15] are $\lambda_{\max} = (\mathrm{UB} - \mathrm{LB})/2$ and $\lambda_{\min} = 10^{-4}$, of LGMS-FOA [17] are $\alpha = 0.95$, of PSO [29] are $c1 = 1.5$ and $c2 = 1.5$ (which are better than $c1 = 2$ and $c2 = 2$ in our testing work). In ABC, the percentages of onlooker bees and employ bees on colony are 50%, respectively, and the number of scout bees is one [30]. In GSA, G_0 is set to 100 and α is set to 20 [31]. In EFOA, $\mathrm{ELC}_{\max} = 0.1$, $\mathrm{ELC}_{\min} = 0.05$, $\lambda_{\max} = (\mathrm{UB} - \mathrm{LB})/2$ and $\lambda_{\min} = 10^{-4}$ are used. (This is not the best parameter set in EFOA, but this set of parameters is given for consideration to the efficiency and accuracy.) Each simulation was repeated 100 times independently, and the mean and standard deviation (SD) were selected as the judgment criteria.

Fourteen benchmark functions that are commonly used in the literature [10, 15–17, 19, 21, 22, 30, 31] were selected to test the accuracy and robustness of EFOA. The detailed information of the 14 functions is listed in Table 3. Functions 1 and 2 are unimodal functions (UF) and the remaining are multimodal functions (MF). One MF has more than one local optimal values, so MF is often used to test the ability of algorithm whether it can reach at the global optimal value. All test functions contain 30 dimensions with extreme value 0 except four functions; they are f6 which only has 2 dimensions with extreme value of -1, and f12 which has 2 dimensions with extremum of -1.0316, and f13 which has 100 dimensions with extremum of 0.398. Besides, f11 has a minimum value of $-418.9829 \times n$, where n is the tested dimension.

4.2 Experimental results and analysis for five optimization algorithms

Table 4 reports the experimental results for the above 7 methods on the 14 testing functions in terms of the Mean (mean value) and Std (standard deviation) which are obtained from 100 repeated simulations. The Mean represents the average precision, and the Std reflects the robustness of the algorithm to some degree [17].

As shown in Table 4, EFOA shows better performance of Mean and Std than the remaining methods on the testing functions of f2, f3, f4, f10, and f14. In f8, EFOA, ABC, IFFO, and GSA have the optimal criteria value of Mean = 0 and Std = 0. For f9, EFOA, GSA, and LGMS-FOA exhibit the best performance. For f1, the performance of LGMS-FOA is superior to other methods, and EFOA performs better than the other three methods except LGMS-FOA, ABC, and GSA. PSO shows the best performance on f6, f12, and f13, but EFOA and ABC also perform good in those functions. The performance of PSO and LGMS-FOA is not as good as EFOA and IFFO on most of the functions. The poorest performance comes from the basic FOA algorithm, which shows FOA is the worst method on all tested functions. IFFO is one of the good optimization methods after EFOA, because IFFO has better results on f1, f2, f3, f4, f5, f7, and f10 when it is compared with FOA, LGMS-FOA, and PSO. ABC and GSA are good searching methods (eg., ABC is the only one method better than EFOA in f11, and GSA is better than EFOA in f1, f5, and f10, but in the rest of testing functions EFOA performs better than GSA.) In general, EFOA is more superior than the other six methods for most of the testing functions. Therefore, EFOA has the best performance at finding the optimal value as it is shown in Table 4.

According to the results in Table 4, to compare further the performance, we ranked each model's score from high to low in each testing function and the corresponding results are shown in Table 5. It is obvious that EFOA's score is almost more than 6, expect that its score is 4 in f1, which conveys that EFOA method has a strong robustness. From the SUM score, we can know that the highest is EFOA with 91, much higher than that of all other methods, and ABC, GSA, and IFFO are the second good group methods with the scores 78, 74, and 74, followed by LGMS-FOA and PSO with 58 and 57, respectively. The

Table 3 List of benchmark testing functions

Number	Test functions	Types	Dimensions	Range	Extreme value
1	$f(x)=\sum_{i=1}^{n} x_i^2$	UF	30	[– 100, 100]	0
2	$f(x)=\sum_{i=1}^{n-1}[100(x_{i+1}-x_i^2)^2+(x_i-1)^2]$	UF	30	[– 30, 30]	0
3	$f(x)=\sum_{i=1}^{n}[x_i^2-10\cos(2\pi x_i)+10]$	MF	30	[– 600, 600]	0
4	$f(x)=\frac{1}{4000}\sum_{i=1}^{n}[x_i^2]-\prod_{i=1}^{n}\cos\frac{x}{\sqrt{i}}+1$	MF	30	[– 600, 600]	0
5	$f(x)=20+e-20\exp\left(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n}x_i^2}\right)-\exp\left(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i)\right)$	MF	30	[– 100, 100]	0
6	$f(x)=\frac{\sin^2\sqrt{x_1^2+x_2^2}-0.5}{[1+0.001(x_1^2+x_2^2)]^2}-0.5$	MF	2	[– 100, 100]	– 1
7	$f(x)=\sum_{i=1}^{n}(10^6)^{\frac{i-1}{n-1}x_i^2}$	MF	30	[– 100, 100]	0
8	$f(x)=\sum_{i=1}^{n}(\mathrm{floor}(x_i+0.5))^2$	MF	30	[– 100, 100]	0
9	$f(x)=1-\cos\left(2\pi\left(\sqrt{\sum_{i=1}^{n}x_i^2}\right)+0.1\sqrt{\sum_{i=1}^{n}x_i^2}\right)$	MF	30	[– 100, 100]	0
10	$f(x)=\sum_{i=1}^{n}\lvert x_i\sin x_i+0.1x_i\rvert$	MF	30	[– 10, 10]	0
11	$f(x)=\sum_{i=1}^{n}-x_i\sin(\sqrt{\lvert x_i\rvert})$	MF	30	[– 500, 500]	– 12,569.487
12	$f(x)=4x_1^2-2.1x_1^4+x_1^6/3+x_1x_2-4x_2^2+4x_2^4$	MF	2	[– 5, 5]	– 1.0316
13	$f(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\cos x_1+10\right)$	MF	30	[– 10, 10]	0.398
14	$f(x)=0.1\left\{\sin^2(3\pi x_1)+\sum_{i=1}^{n}(x_i-1)^2[1+\sin^2(3\pi x_i+1)]+(x_n^2-1)[1+\sin^2(2\pi x_n)]\right\}+\sum_{i=1}^{n}u(x_i,5,100,4)$	MF	100	[– 50, 50]	0

In function 14, $u(x, a, k, m)$ is a function and has the following expression $u(x,a,k,m)=k.*((x-a).^m).*(x>a)+k.*((-x-a).^m).*(x<(-a))$

Table 4 Comparison of FOA, LGMS-FOA, IFFO, EFOA, PSO, ABC, and GSA on 14 benchmark functions

Function	States	FOA	LGMS-FOA	IFFO	EFOA	PSO	ABC	GSA
f1	Mean	6.50	4.12E−40	5.72E−08	6.55E−12	6.06	3.74E−12	3.86E−18
	Std	0.79	7.34E−41	2.22E−08	1.41E−12	11.45	3.31E−12	9.59E−19
f2	Mean	1.58E+03	3.29E+02	47.31	1.05	1.59E+02	4.19	26.61
	Std	9.88E+02	5.66E+02	35.59	1.91	2.97E+02	2.97	6.24
f3	Mean	3.60E+05	1.56E+02	1.66E−05	1.70E−09	1.73E+02	0.16	4.84E+03
	Std	7.99E+04	47.15410719	6.55E−06	3.36E−10	4.56E+02	0.35	2.16E+03
f4	Mean	93.55	1.14E−02	1.03E−02	5.18E−13	0.36	1.28E−07	1.79
	Std	18.16	1.12E−02	1.20E−02	1.47E−13	0.53	1.02E−06	0.58
f5	Mean	21.09	20.00	0.20	1.89E−06	19.40	20.00	1.56E−09
	Std	0.06	3.94E−05	2.00	2.40E−07	3.43	5.10E−09	1.76E−10
f6	Mean	− 0.83	− 9.97E−01	− 9.82E−01	− 1.00E+00	− 1	− 1	− 9.90E−01
	Std	0.11	4.66E−03	1.57E−02	1.67E−03	2.52E−05	1.87E−09	2.55E−03
f7	Mean	3.24E+06	7.00E+06	2.34E−03	9.44E−08	1.45E+05	1.22E−08	2.49E+04
	Std	9.28E+05	3.15E+06	1.86E−03	3.70E−08	2.16E+05	1.28E−08	1.45E+04
f8	Mean	9.13	2.57	0	0	6.82	0	0
	Std	1.80	1.99	0	0	16.13	0	0
f9	Mean	9.27E−08	0	1.15E−13	0	1.27E−09	5.41E−10	0
	Std	1.95E−07	0	2.10E−13	0	1.96E−09	1.50E−09	0
f10	Mean	31.12	5.08	3.66E−04	1.47E−06	0.81	3.09	1.00E−09
	Std	4.51	2.43	3.20E−04	6.27E−07	1.27	0.82	1.35E−10
f11	Mean	− 7.25E+03	− 7.67E+03	− 1.11E+04	− 1.16E+04	− 7.67E+03	− 1.23E+04	− 2.94E+03
	Std	6.79E+02	6.96E+02	2.85E+02	1.45E+02	8.68E+02	1.06E+02	3.66E+02
f12	Mean	− 1.01	− 1.03	− 1.03	− 1.03	− 1.03	− 1.03	− 1.03
	Std	0.12	1.55E−15	4.59E−11	1.18E−15	1.52E−06	9.40E−13	1.54E−15
f13	Mean	0.40	0.40	0.40	0.40	0.40	3.00	0.40
	Std	2.78E−04	1.06E−15	5.19E−11	1.06E−15	1.98E−05	1.42E−03	1.06E−15
f14	Mean	1.76E+02	4.37E+05	4.74E−07	2.20E−10	54.70	1.65E−02	23.69
	Std	1.20E+02	4.67E+05	1.48E−07	2.55E−11	1.22E+02	1.65E−02	7.75

worst performance method is basic FOA, because it is limited by its own searching strategy. All the ability scores can demonstrate that EFOA, as a powerful optimization algorithm, has the good performance not only in optimal solution searching ability, but also for most functions. EFOA can give a high-level solution and has the ability of generalization.

For comparison and analysis of the converging process, the corresponding convergence curves for the 7 optimization algorithms on the 1–14 benchmark functions are given in Figs. 2 and 3. To show the result clearly, all of the searching results in iteration 1–200 are demonstrated in Fig. 2, while Fig. 3 displays the detailed results in iteration 151–160. The basic FOA has the slowest decrease rate, especially on f1, f3 f4, f6, f8, and f10. LGMS-FOA has better converging rate than FOA, ABC, and GSA in early stage, but the converging rate in the latter stage can not reach the level which ABC and GSA can in almost all of testing functions. When comparing the IFFO and EFOA, we can see that EFOA is better than IFFO in f1–f14, although they use the same step control parameters. IFFO performs better in comparison with FOA, LGMS-FOA, and PSO in most of the testing functions. EFOA has the best converging rate in all seven algorithms in the ending stage, as shown in Fig. 2. Although PSO method is faster than EFOA during the preliminary searching stage, in the middle and later periods PSO becomes trapped in the local optimal solution. But EFOA performs well until the iteration is over. ABC and GSA have weak searching ability in the starting stage in almost all of benchmark functions, but in the end of searching stage, they perform better than FOA, LGMS-FOA, and PSO. The converging ability of EFOA only lags behind PSO, but outperforms the remaining methods during the initial search. On the other hand, in iteration 151–160, EFOA can search more accurate solution than all other methods in almost all testing

Table 5 Ranking of FOA, LGMS-FOA, IFFO, EFOA, PSO, ABC, and GSA on 14 benchmark functions according to their performance

Function	FOA	LGMS-FOA	IFFO	EFOA	PSO	ABC	GSA
f1	2	7	3	4	1	5	6
f2	1	2	4	7	3	6	5
f3	1	4	6	7	3	5	2
f4	1	4	5	7	3	6	2
f5	1	3	4	6	5	3	7
f6	1	6	4	7	7	7	5
f7	2	1	5	6	3	7	4
f8	4	5	7	7	6	7	7
f9	3	7	6	7	4	5	7
f10	1	2	5	6	4	3	7
f11	3	2	5	6	2	7	4
f12	6	7	7	7	7	7	7
f13	6	7	7	7	7	5	7
f14	3	1	6	7	2	5	4
SUM	35	58	74	91	57	78	74
Ranking	6	4	3	1	5	2	3

functions. In fact, EFOA is obviously located on a good level, especially in f2, f5, f7, f10, and f11.

In Fig. 4, we compare the distributions of optimized results in 100 times for the seven methods on 14 testing functions. As is shown, FOA has the worst results on all testing functions because it has high mean value and wide distribution box. LGMS-FOA has poor performance on f2, f5, f6, f7, f8, f10, f11, and f14, and PSO has poor ability on f1, f2, f7, f8, f10, and f11. The robustness of IFFO and EFOA performs well in almost all testing functions, but EFOA is outstanding on f1, f2, f5, f6, f11, f13, and f14 obviously. ABC and GSA do not have good results in some testing functions, such as in f2, f5, f7, f10, f11, and f14. Hence, from the vision of robustness, EFOA is a good optimization algorithm, which has been verified on the 14 testing functions.

EFOA exhibits good performance in terms of accuracy, convergence, and robustness. The parameters of λ and ELC play an important role in the process of searching for the optimal value. Controlling the searching step and the number of swarms adaptively is essential for effective searching for EFOA.

From the experimental results, we can see that EFOA typically reaches the optimization values with a higher accuracy and shows stronger robustness than the other improved algorithm FOA, PSO, ABC, and GSA. These results indicate that EFOA obviously outperforms the original FOA, OLGMS-FOA, IFFO, PSO, ABC, and GSA for more functions with satisfactory precision.

4.3 Comparison and analysis on different evolution coefficients

To evaluate the influence on different ELC ranges, another experiment is designed to compare the performances on different setting ranges of ELC. The change of swarm numbers under various ELC ranges is shown in Fig. 5. The curve suggests that a small value of ELC can preserve more dominant swarms on iterations and a big value of ELC will remove more inferior swarms. The Mean and Std of the ten testing functions (f1–f10) on different evolution coefficients for EFOA are shown in Table 6, from which we can draw the following conclusion:

1. EFOA on ELC range (0.1–0.05) consistently performs better than other ranges in almost of all testing functions.
2. EFOA on all ELC ranges achieves the best solution in f8. Thus, the evolution coefficient does not help on function f8.
3. EFOA on range (0.1–0.05) shows a dramatic difference on f4. It may indicate that EFOA on the remaining four ranges is trapped in local extremum.

To conclude, on the different evolution coefficients, the smaller the ELC is, the better performance EFOA has.

5 The application of EFOA in engineering problem

In this section, EFOA is used to deal with an engineering problem called pressure vessel design, which belongs to a classical type of constrained optimization problem. In this study, the method about building an efficient optimization object function in constrained optimization problem is to convert equality and inequality constraints as penalty term appending to optimization object function [32].

In this case, the goal is to minimize the total cost (f(x)), including the cost of material, forming, and welding. There are four design variables: thickness of the shell (T_s, x_1), thickness of the head (T_h, x_2), inner radius (R, x_3), and length of the cylindrical section of the vessel, not including the head (L, x_4) [33]. Among these four variables, $0.0625 \leqslant x_1, x_2 \leqslant 99$ and $10 \leqslant x_3, x_4 \leqslant 200$ are continuous variables. Four parameters of the pressure vessel design problem are depicted in Fig. 6.

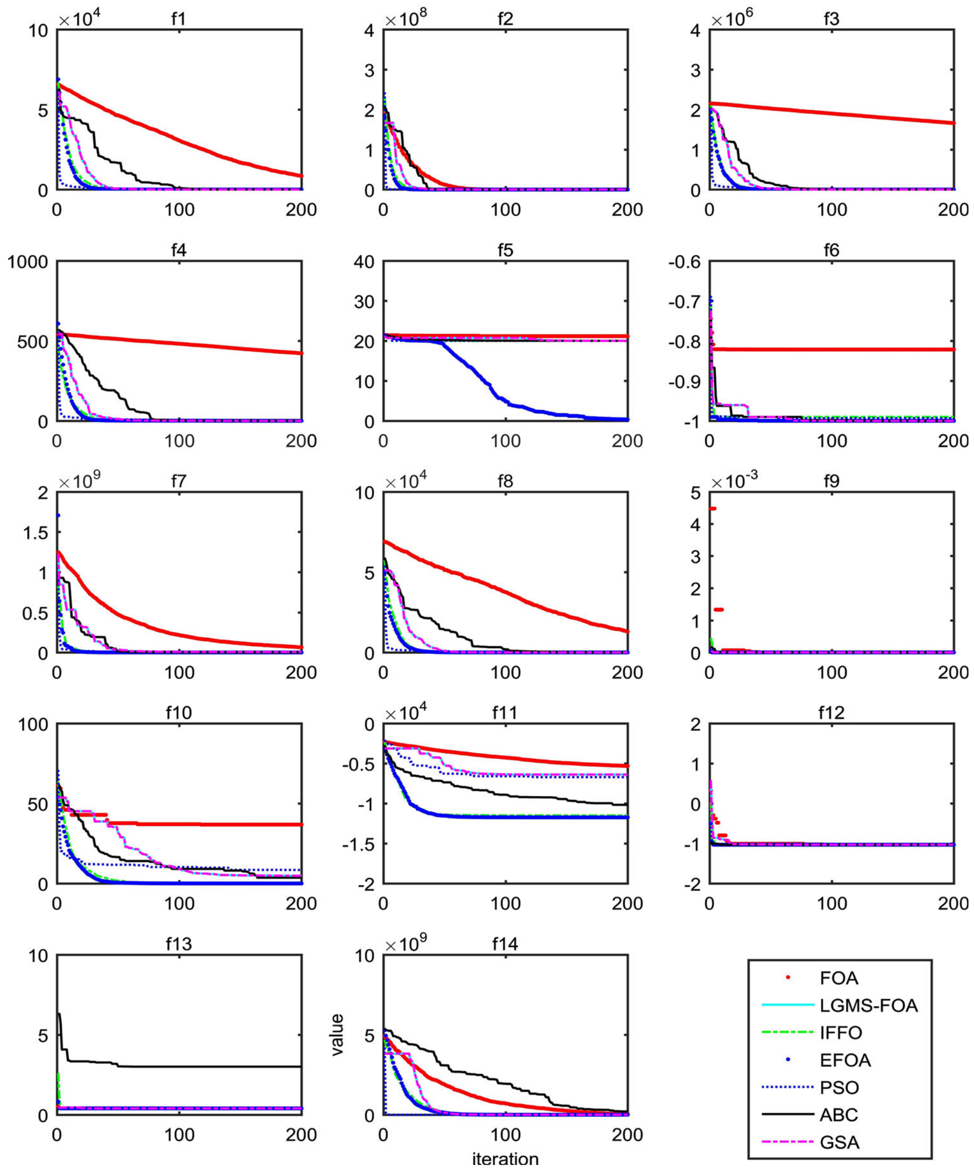

Fig. 2 Optimization process curves for f1–f14. To read the results clearly, we only display the results of Iter = 200. The function value is on the vertical axis. The number of iteration is on the horizontal axis

The mathematical formulation of this problem can be described as follows:

Minimize

$$f(x) = 0.6224x_1x_3x_4 + 1.7781x_2x_3^2 + 3.1661x_1^2x_4 + 19.84x_1^2x_3, \tag{8}$$

subject to :

$$\begin{aligned} g_1(x) &= -x_1 + 0.0193x_3 \leqslant 0,\\ g_2(x) &= -x_2 + 0.00954x_3 \leqslant 0,\\ g_3(x) &= -\pi x_3^2x_4 - \frac{4}{3}\pi x_3^3 + 1{,}296{,}000 \leqslant 0,\\ g_4(x) &= x_4 - 240 \leqslant 0. \end{aligned} \tag{9}$$

The results of EFOA on the problem are shown in Tables 7 and 8. EFOA method is executed by 100 times using randomized independent experiment. The best solutions obtained by EFOA and other previous researchers' approach are listed in Table 7. It can be observed that the best solution of EFOA is better than that obtained from the previous literatures in pressure vessel design problem. What the amazing is that the statistic index of "best," "mean," and "worst" of EFOA is superior to all comparison methods in Table 8. EFOA is also suitable for constrained optimization problem to sum up the above arguments, and it can be easy to jump out of local optimum which has been verified by the data experiment.

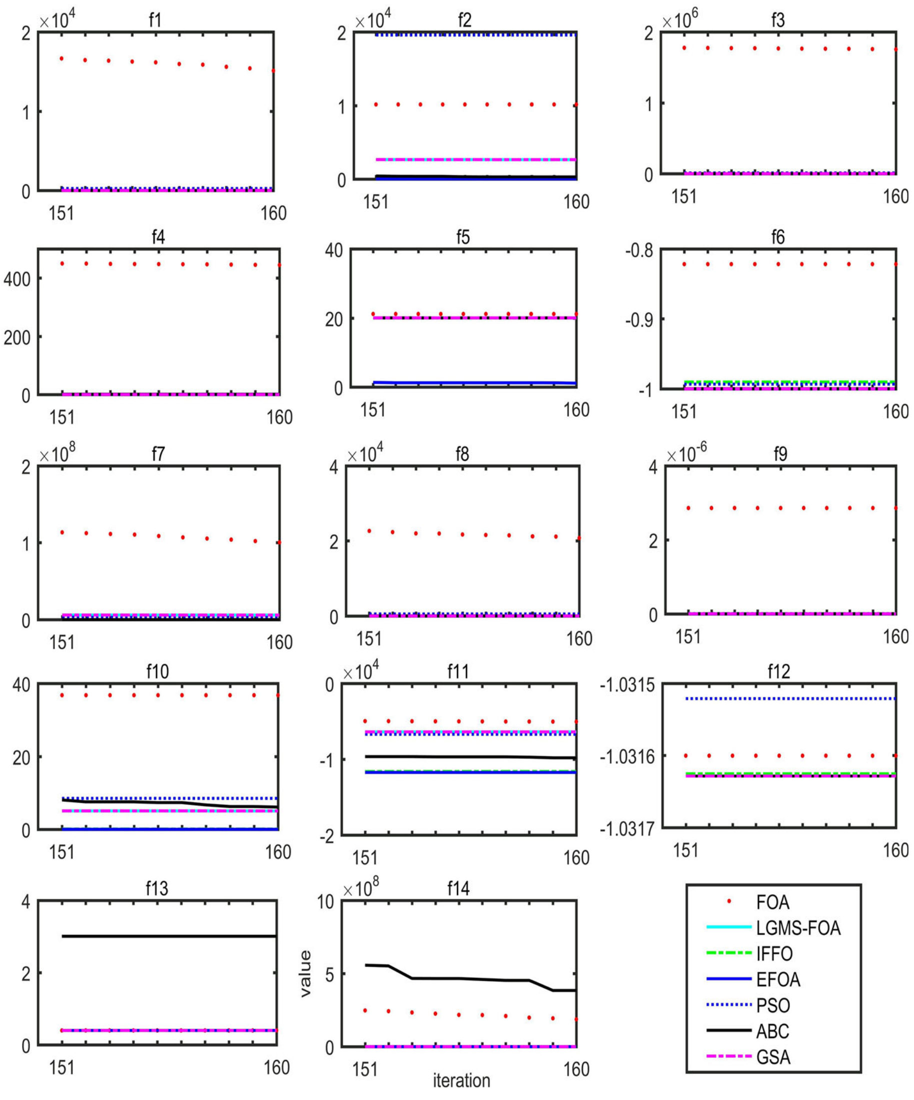

Fig. 3 Optimization process curves for f1–f14. To read the results clearly, we display the results of Iter in range of [151,160]. The function value is on the vertical axis. The number of iteration is on the horizontal axis

6 Conclusion and discussion

From the experimental data results of 14 benchmark functions and a constrained engineering problem, EFOA is verified that it can quickly find out the better solution and jump out local optimum with a much higher possibility.

The reasons of EFOA which can jump out of local optimum and quickly find out the better solution are list as follows:

1. In searching process, a dynamical searching radius with the feature that is from big to small (A big searching step equals to the distance of parameter range in first searching, which can improve the global search ability; a small searching step in end stage can improve the local search ability.) in iteration is an efficient way to overcome falling into local optimum.
2. EFOA belongs to multi-swarms simultaneously iteration search mode; therefore, it can keep individual diversity and swarms diversity in searching process. A searching mode with the feature of individual diversity and swarms diversity has more chance to approach the globe optimum area and jump out of local optimum than single swarm iteration search mode. We can regard some inferior fruit fly swarms as local optimum area. These inferior swarms will be eliminate with iteration in EFOA. Then, a few dominant swarms as elites continue searching.

Some conclusions in this study can be listed as follows:

1. EFOA with the feature of individual diversity and swarms diversity has more chance to approach the globe optimum area and jumps out of local optimum than single swarm iteration search mode.

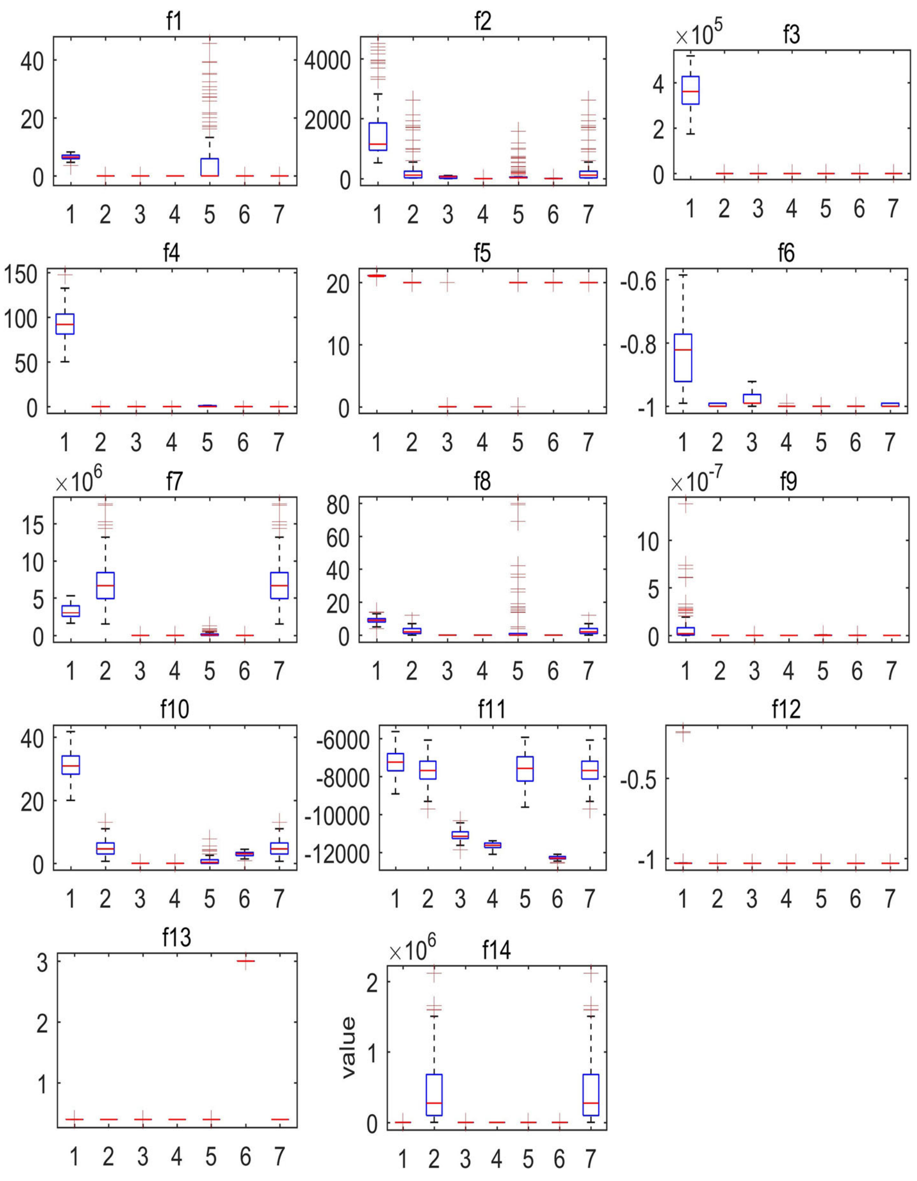

Fig. 4 Comparison of seven optimization algorithms performed for 100 times. The distributions of optimized results for the 14 testing functions are shown in f1–f14. The blue boxes represent the 100 times optimization result distributions, and the red line in boxes is the average value of 100 times optimization values. In each subgraph, the numbers 1–7 in x axis represent the optimization algorithm (1: FOA, 2, LGMS-FOA, 3: IFFO, 4: EFOA, 5: PSO, 6: ABC, 7: GSA). The final searched function value is on the vertical axis

2. EFOA has the best converging rate in all seven algorithms in the ending stage, as shown in Fig. 2. Although PSO method is faster than EFOA during the preliminary searching stage, in the middle and later periods PSO becomes trapped in the local optimal solution. But EFOA performs well until the iteration is over.
3. EFOA controls the number of multi-swarms in iteration by the swarm evolution control parameter (EC) or ELC.
4. EFOA can solve both unconstrained and constrained optimization problems.
5. Multi-swarm mechanism is a good way to overcome being trapped in the local optimal problem.

The contribution of this research is that we proposed an improved FOA algorithm called EFOA, which adopts the step iteration update and the swarm evolution strategies. The special parameter of ELC increases the probability for finding global optimal solution and improves the acceleration of converging rate and enhances the robustness of the method. All simulation results from 14 testing benchmark functions and a constrained engineering verified that the performance of EFOA is superior. EFOA has better performance than the other methods because the parameter ELC can preserve the dominant swarm, the parameter λ can improve the exploring competence of individuals, and multi-swarm simultaneous iteration search mode can keep individual diversity and swarm diversity. EFOA allows adaptive evolution and elimination of a variety of swarms which are fixed in FOA. Overall, the proposed EFOA can become a powerful tool to solve global optimization problems.

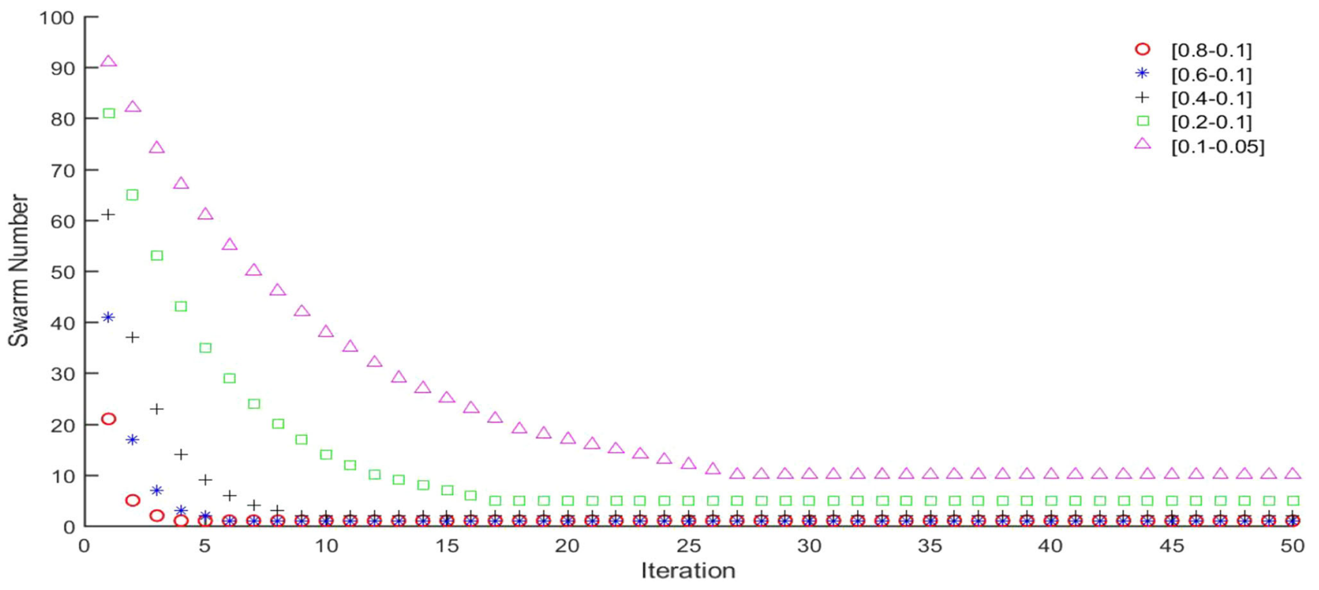

Fig. 5 Change of swarm numbers under various ELC ranges [$ELC_{max} - ELC_{min}$]

Table 6 Comparison of EFOA performance on different evolution coefficients (minimum mean values are in bold. All the iterations were repeated 100 times)

Function	EFOA (0.8–0.1)	EFOA (0.6–0.1)	EFOA (0.4–0.1)	EFOA (0.2–0.1)	EFOA (0.1–0.05)
f1					
Mean	1.96E−11	1.82E−11	1.54E−11	1.25E−11	**1.04E−11**
Std	6.85E−12	5.50E−12	5.78E−12	2.98E−12	2.22E−12
f2					
Mean	54.95	49.61	37.91	15.42	**4.37**
Std	32.84	38.46	34.27	23.98	5.20
f3					
Mean	3.80E−09	3.48E−09	2.85E−09	2.45E−09	**2.08E−09**
Std	1.21E−09	1.30E−09	8.37E−10	7.20E−10	4.32E−10
f4					
Mean	9.01E−03	9.94E−03	2.96E−03	9.86E−04	**6.57E−13**
Std	1.13E−02	1.27E−02	5.95E−03	2.77E−03	2.99E−13
f5					
Mean	3.17E−06	3.10E−06	2.77E−06	2.53E−06	**2.27E−06**
Std	6.59E−07	4.77E−07	5.08E−07	3.21E−07	2.75E−07
f6					
Mean	− 9.87E−01	− 9.87E−01	− 9.91E−01	− 9.94E−01	**− 9.97E−01**
Std	1.43E−02	1.12E−02	5.46E−03	4.71E−03	4.65E−03
f7					
Mean	9.20E−07	6.83E−07	3.75E−07	2.70E−07	**1.65E−07**
Std	1.05E−06	6.47E−07	2.61E−07	1.38E−07	7.51E−08
f8					
Mean	**0**	**0**	**0**	**0**	**0**
Std	0	0	0	0	0
f9					
Mean	9.77E−17	6.44E−17	**0**	**0**	**0**
Std	4.03E−16	1.82E−16	0	0	0
f10					
Mean	9.63E−06	9.02E−06	5.60E−06	3.11E−06	**2.21E−06**
Std	7.11E−06	6.09E−06	3.45E−06	1.75E−06	1.32E−06

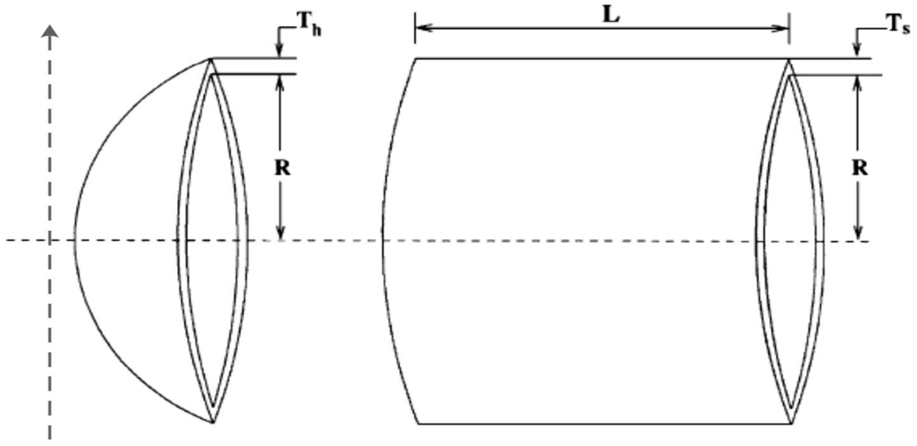

Fig. 6 Design of pressure vessel problem [33]

Table 7 Comparison of the best solution of pressure vessel design problem with different methods

Method	Design variables				$f(x)$
	$x1$	$x2$	$x3$	$x4$	
Sandgren [34]	1.125000	0.625000	47.700000	117.701000	8129.1036
Zhang and Wang [35]	1.125000	0.625000	58.290000	43.693000	7197.7000
Kannan and Kramer [36]	1.125000	0.625000	58.291000	43.690000	7198.0428
Deb [32]	0.937500	0.500000	48.329000	112.679000	6410.3811
Coello [37]	0.812500	0.437500	40.323900	200.000000	6288.7445
Coello and Montes [37]	0.812500	0.437500	42.097398	176.654050	6059.946
Hu et al. [38]	0.812500	0.437500	42.098450	176.636600	6059.131296
Gandomi et al. [39]	0.812500	0.437500	42.0984456	176.6365958	6059.7143348
He et al. [38]	0.812500	0.437500	42.098445	176.6365950	6059.7143
Lee and Geem [40]	1.125000	0.625000	58.278900	43.754900	7198.433
Montes et al. [41]	0.812500	0.437500	42.098466	176.6360470	6059.701660
He and Wang [33]	0.812500	0.437500	42.091266	176.746500	6061.0777
Montes and Coello [42]	0.812500	0.437500	42.098087	176.640518	6059.7456
Cagnina et al. [43]	0.812500	0.437500	42.098445	176.6365950	6059.714335
Kaveh et al. [44]	0.812500	0.437500	42.103566	176.573220	6059.0925
Kaveh et al. [45]	0.812500	0.437500	42.098353	176.637751	6059.7258
Coelho [46]	0.812500	0.437500	42.098400	176.6372000	6059.7208
Akay et al. [47]	0.812500	0.437500	42.098446	176.636596	6059.714339
Mazhoud et al. [48]	NA	NA	NA	NA	6059
Long et al. [49]	0.812500	0.437500	42.098440	176.636600	6059.714300
Liu et al. [50]	0.812500	0.437500	42.098445	176.636595	6059.714335
EFOA	0.780955183609562	0.386026882099020	40.4639584864676	198.000396067761	5890.11939272369

While EFOA really has improvement for solving global optimization problems, it has some limitations:

1. In this study, the best evolution direction of fruit fly is not directly given.
2. EFOA may be more time-consuming than basic FOA, relatively speaking, when more swarms are used in the early stage.
3. Additional initialization parameters need to be given.
4. Communication between swarms is not considered in EFOA so far.

Future research will be required for further modifications of EFOA to develop a self-adaptive FOA (join in communication between swarms) and multi-swarm FOA with better convergence, accuracy, and robustness and also less time-consuming during the process of evolution. At the same time, generalization of EFOA will be needed to

Table 8 Statistical index results of different approaches for pressure vessel problem (NA means not available)

Method	Statistic index			
	Best	Mean	Worst	SD
Sandgren [34]	8129.1036	NA	NA	NA
Zhang and Wang [35]	7197.7000	NA	NA	NA
Kannan and Kramer [36]	7198.0428	NA	NA	NA
Deb [32]	6410.3811	NA	NA	NA
Coello [37]	6288.7445	6293.8432	6308.1497	7.4133
Coello and Montes [37]	6059.9463	6177.2533	6469.3220	130.9297
Hu et al. [38]	6059.131296	NA	NA	NA
Gandomi et al. [39]	6059.714	6447.7360	6495.3470	502.693
He et al. [38]	6059.7143	6289.92881	NA	305.78
Lee and Geem [40]	7198.433	NA	NA	NA
Montes et al. [41]	6059.701660	NA	NA	NA
He and Wang [33]	6061.0777	6147.1332	6363.8041	86.4545
Montes and Coello [42]	6059.7456	6850.0049	7332.8798	426.0000
Cagnina et al. [43]	6059.714335	6092.0498	NA	12.1725
Kaveh and Talatahari [44]	6059.0925	6075.2567	6135.3336	41.6825
Kaveh and Talatahari [45]	6059.7258	6081.7812	6150.1289	67.2418
Coelho [46]	6059.7208	6440.3786	7544.4925	448.4711
Akay and Karaboga [47]	6059.7143	6245.3081	NA	205
Mazhoud et al. [48]	6059.7143	6291.1231	6820.4101	288.4550
Long et al. [49]	6059.7143	6087.3225	6137.4069	2.21e−02
Liu et al. [50]	6059.714335	6121.420389	6410.024261	23.811197
EFOA	5887.69136876857	5923.29327049569	5979.38306645097	21.6074408884499

solve more complex combinatorial and discrete optimization problems and other practical problems.

Acknowledgements This study is supported by National Nature Science Foundation of China (Grant No. 41571016) and the National Key Research and Development Program of China (Grant No. 2018YFC0406606). The authors like to thank Prof. Yujing Lu, an English professor of Lanzhou University, for her proofreading. The authors are also thankful to anonymous referees for the valuable and constructive suggestions.

Compliance with ethical standards

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Ali ES (2015) Speed control of DC series motor supplied by photovoltaic system via firefly algorithm. Neural Comput Appl 26(6):1321–1332
2. Abd-Elazim SM, Ali ES (2018) Load frequency controller design of a two-area system composing of PV grid and thermal generator via firefly algorithm. Neural Comput Appl 30(2):607–616
3. Oshaba AS, Ali ES, Elazim SMA (2017) Pi controller design for MPPT of photovoltaic system supplying SRM via bat search algorithm. Neural Comput Appl 28(4):651–667
4. Huo J, Liu L (2018) Application research of multi-objective artificial bee colony optimization algorithm for parameters calibration of hydrological model. Neural Comput Appl 31(9):4715–4732
5. Chen B, Zhang H, Li M (2019) Prediction of pk(a) values of neutral and alkaline drugs with particle swarm optimization algorithm and artificial neural network. Neural Comput Appl. https://doi.org/10.1007/s00521-018-3956-5
6. Pan WT (2012) A new fruit fly optimization algorithm: taking the financial distress model as an example. Knowl Based Syst 26(2):69–74
7. Pan WT (2013) Using modified fruit fly optimisation algorithm to perform the function test and case studies. Connect Sci 25(2–3):151–160
8. Duan Q, Mao M, Duan P, Hu B (2016) An improved artificial fish swarm algorithm optimized by particle swarm optimization algorithm with extended memory. Kybernetes 45(2):210–222
9. Jovanovic R, Tuba M, Vo S (2015) An ant colony optimization algorithm for partitioning graphs with supply and demand. Comput Sci 209(3):207–212
10. Sharma H, Bansal JC, Arya KV (2013) Opposition based levy flight artificial bee colony. Memet Comput 5(3):1–15
11. Chen PW, Lin WY, Huang TH, Pan WT (2013) Using fruit fly optimization algorithm optimized grey model neural network to perform satisfaction analysis for e-business service. Appl Math Inf Sci 7(2L):459–465
12. Li HZ, Guo S, Li CJ, Sun JQ (2013) A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm. Knowl Based Syst 37(2):378–387

13. Sheng W, Bao Y (2013) Fruit fly optimization algorithm based fractional order fuzzy-pid controller for electronic throttle. Nonlinear Dyn 73(1–2):611–619
14. Wang L, Zheng XL, Wang SY (2013) A novel binary fruit fly optimization algorithm for solving the multidimensional knapsack problem. Knowl Based Syst 48(2):17–C23
15. Pan QK, Sang HY, Duan JH, Gao L (2014) An improved fruit fly optimization algorithm for continuous function optimization problems. Knowl Based Syst 62(5):69–83
16. Wang L, Liu R, Liu S (2016) An effective and efficient fruit fly optimization algorithm with level probability policy and its applications. Knowl Based Syst 97(C):158–174
17. Shan D, Cao GH, Dong HJ (2013) LGMS-FOA: an improved fruit fly optimization algorithm for solving optimization problems. Math Probl Eng 2013(7):1256–1271
18. Xu F, Tao Y (2014) The improvement of fruit fly optimization algorithm. Int J Autom Comput 10(03):227–241
19. Wu L, Xiao W, Zhang L, Liu Q, Wang J (2016) An improved fruit fly optimization algorithm based on selecting evolutionary direction intelligently. Int J Comput Intell Syst 9(1):80–90
20. Xiao C, Hao K, Ding Y (2015) An improved fruit fly optimization algorithm inspired from cell communication mechanism. Math Probl Eng 2015:1–15
21. Yuan X, Dai X, Zhao J, He Q (2014) On a novel multi-swarm fruit fly optimization algorithm and its application. Appl Math Comput 233(3):260–271
22. Wang L, Shi Y, Liu S (2015) An improved fruit fly optimization algorithm and its application to joint replenishment problems. Expert Syst Appl 42(9):4310–4323
23. Tian X, Jie LI, S. O. Aeronautics, N. P. University (2017) An improved fruit fly optimization algorithm and its application in aerodynamic optimization design. Acta Aeronaut Astronaut Sin 38(4)
24. Du TS, Ke XT, Liao JG, Shen YJ (2017) DSLC-FOA: an improved fruit fly optimization algorithm application to structural engineering design optimization problems. Appl Math Model. S0307904X17305310
25. Darvish A, Ebrahimzadeh A (2018) Improved fruit-fly optimization algorithm and its applications in antenna arrays synthesis. IEEE Trans Antennas Propag PP(99):1–1
26. Dorigo M, Di CG, Gambardella LM (1999) Ant algorithm for discrete optimization. Arti Life 5(2):137–172
27. Dorigo M, Gambardella LM (1997) Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans Evolut Comput 1(1):53–66
28. Kennedy J, Eberhart R (1995) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, vol 4, pp 1942–1948
29. Kennedy J, Eberhart R (2002) Particle swarm optimization. In: Proceedings of IEEE international conference on neural networks, vol 4, pp 1942–1948
30. Karaboga D, Basturk B (2007) A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm. Kluwer, Dordrecht
31. Rashedi E, Nezamabadi-Pour H, Saryazdi S (2012) Gsa: a gravitational search algorithm. Inf Sci 4(6):390–395
32. Deb K (1997) GeneAS: a robust optimal design technique for mechanical component design. Springer, Berlin
33. He Q, Wang L (2007) An effective co-evolutionary particle swarm optimization for constrained engineering design problems. Eng Appl Artif Intell 20(1):89–99
34. Sandgren E (1990) Nonlinear integer and discrete programming in mechanical design. J Mech Des 112(2):223–229
35. Zhang C, Wang H-PB (1993) Mixed-discrete nonlinear optimization with simulated annealing. Eng Optim 21(4):277–291
36. Kannan BK, Kramer SN (1994) An augmented Lagrange multiplier based method for mixed integer discrete continuous optimization and its applications to mechanical design. J Mech Des 116(2):405–411
37. Coello CAC (2000) Use of a self-adaptive penalty approach for engineering optimization problems. Comput Ind 41(2):113–127
38. Hu X, Eberhart RC, Shi Y (2003) Engineering optimization with particle swarm. In: Swarm intelligence symposium
39. Gandomi AH, Yang X, Alavi AH (2013) Cuckoo search algorithm: a metaheuristic approach to solve structural optimization problems. Eng Comput 29(1):17–35
40. Lee KS, Geem ZW (2005) A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice. Comput Methods Appl Mech Eng 194(3638):3902–3933
41. Mezuramontes E, Coello CAC, Velazquezreyes J, Munozdavila L (2007) Multiple trial vectors in differential evolution for engineering design. Eng Optim 39(5):567–589
42. Mezuramontes E, Coello CAC (2008) An empirical study about the usefulness of evolution strategies to solve constrained optimization problems. Int J Gen Syst 37(4):443–473
43. Cagnina L, Esquivel SC, Coello CAC (2008) Solving engineering optimization problems with the simple constrained particle swarm optimizer. Informatica (Lith Acad Sci) 32(3):319–326
44. Kaveh A, Talatahari S (2009) Engineering optimization with hybrid particle swarm and ant colony optimization. Asian J Civ Eng (Build Hous) 10(6):611–628
45. Kaveh A, Talatahari S (2010) An improved ant colony optimization for constrained engineering design problems. Eng Comput 27(1):155–182
46. Coelho LDS (2010) Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems. Expert Syst Appl 37(2):1676–1683
47. Akay B, Karaboga D (2012) Artificial bee colony algorithm for large-scale problems and engineering design optimization. J Intell Manuf 23(4):1001–1014
48. Mazhoud I, Hadjhamou K, Bigeon J, Joyeux P (2013) Particle swarm optimization for solving engineering problems: a new constraint-handling mechanism. Eng Appl Artif Intell 26(4):1263–1273
49. Long W, Liang X, Huang Y, Chen Y (2014) An effective hybrid cuckoo search algorithm for constrained global optimization. Neural Comput Appl 25(3):911–926
50. Liu J, Wu C, Wu G, Wang X (2015) A novel differential search algorithm and applications for structure design. Appl Math Comput 268:246–269