Knowledge-Based Systems 114 (2016) 24–35

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

A novel multi-scale cooperative mutation Fruit Fly Optimization Algorithm

Yiwen Zhang[a,*], Guangming Cui[a], Jintao Wu[a], Wen-Tsao Pan[b], Qiang He[c]

[a] the Key Laboratory of Intelligent Computing & Signal Processing, Ministry of Education, Anhui University, Anhui 230031, China
[b] Department of Information Management, Oriental Institute of Technology, Taiwan, ROC
[c] School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, Australia

ARTICLE INFO

Article history:
Received 10 June 2016
Revised 26 September 2016
Accepted 27 September 2016
Available online 21 October 2016

Keywords:
Fruit fly optimization algorithm
Multi-scale cooperative mutation
Global optimization
Convergence

ABSTRACT

The Fruit Fly Optimization Algorithm (FOA) is a widely used intelligent evolutionary algorithm with a simple structure that requires only simple parameters. However, its limited search space and the swarm diversity weaken its global search ability. To tackle this limitation, this paper proposes a novel Multi-Scale cooperative mutation Fruit Fly Optimization Algorithm (MSFOA). First, we analyze the convergence of FOA theoretically and demonstrate that its convergence depends on the initial location of the swarm. Second, a Multi-Scale Cooperative Mutation (MSCM) mechanism is introduced that tackles the limitation of local optimum. Finally, the effectiveness of MSFOA is evaluated based on 29 benchmark functions. The experimental results show that MSFOA significantly outperforms the improved versions of FOA presented in recent literature, including IFFO, CFOA, and CMFOA, on most benchmark functions.

1. Introduction

Optimization calculation finds an optimal solution that fulfils a set of constraints and achieves an optimal goal. When solving complex optimization problems, such as those with high dimensions, severe constraints, and multi-polarity, traditional optimization techniques, such as integer programming [1], dynamic programming [2], linear programming [3], and graph algorithms [4,5] suffer from computational complexity. In recent years, Swarm Intelligence (SI) based optimization algorithms have become more and more popular, bringing new vitality to optimization calculation.

Compared to traditional optimization algorithms, such as Genetic Algorithm (GA) [7], SI features simplicity and effectiveness in solving complex optimization problems [8]. Many SI-based optimization algorithms have been proposed, e.g., Particle Swarm Optimization (PSO) and Ant Colony Optimization (ACO). PSO simulates the foraging behavior of bird swarms [8,20]. PSO depends strongly on parameter settings and is prone to converge into a local optimum. ACO models the foraging behavior of ants and is inspired by its information exchange mechanism [9]. However, ACO converges slowly due to stagnation. Such optimization algorithms find a near-optimal solution for a complex problem with a large number of variables and constraints [22]. In addition to SI based optimization algorithms that simulate animals behaviors, there are also optimization algorithms that simulate physical phenomena, e.g., Gravitational Search Algorithm (GSA) and Kinetic Gas Molecule Optimization (KGMO). GSA [10] simulates the characteristics of gravity. It is an intelligent algorithm that combines vector calculation with the law of universal gravitation. Kinetic Gas Molecule Optimization (KGMO) algorithm [11] simulates the dynamics of gas molecular. It models the variation of gaseous molecular kinetic energy with temperature and the irregular motion of gas molecules to seek the optimal value within the searching space. However, such algorithms are not very popular due to their high complexity.

The Fruit Fly Optimization Algorithm (FOA), proposed by Pan in 2011 [12], is a new SI based optimization algorithm inspired by the foraging behavior of fruit flies. Compared with other SI-based optimization algorithms, FOA is easy to understand and computationally inexpensive. As a novel optimization algorithm, FOA has attracted a lot of researchers attention and has been applied successfully in many areas in recent years. To name a few, J. Li et al. applied FOA to solving the hybrid flow-shop rescheduling problem in steelmaking systems and obtained convincing experimental results [13]. Y. Zhang et al. introduced FOA into service computing, and experimentally analyzed its performance [14,15]. H. Li et al. proposed a hybrid annual power load-forecasting model that incorporates FOA based on generalized regression neural network [16]. S. Lin et al. employed FOA to optimize an artificial neural network

* Corresponding author.
E-mail addresses: zhangyiwen@ahu.edu.cn (Y. Zhang), cgm133123@163.com (G. Cui), ahwjt@yahoo.com (J. Wu), teacherp0162@yahoo.com.tw (W.-T. Pan), qhe@swin.edu.au (Q. He).

http://dx.doi.org/10.1016/j.knosys.2016.09.027
0950-7051/© 2016 Elsevier B.V. All rights reserved.

model for improving logistics quality, service satisfaction classification and prediction [17]. X. Zheng et al. designed a new encoding scheme and multi-group techniques that improve the effectiveness and efficiency of solving the semiconductor final testing scheduling problem [23].

However, similar to other SI based optimization algorithms, FOA also has its limitations. One of its major limitations is that it is prone to converge into a local optimum because it often fails to traverse the entire solution space [21]. To address this issue, many researchers have attempted to improve FOA in recent years. For example, L. Wu et al. proposed an improved FOA named CMFOA based on cloud model, and evaluated its performance on 33 benchmark functions [6]. However, CMFOA still converges slowly. M. Mitis et al. proposed a chaotic FOA (CFOA) based on chaotic operations and systematically analyzed its performance [18]. The major limitation of CFOA is that the analysis was performed on only 6 benchmark functions, and thus failed to fully evaluate the performance of CFOA. Q. Plan et al. proposed an improved fruit fly optimization algorithm (IFFO) by improving the method of individual generation [19]. The performance of IFFO was analyzed comprehensively on 29 benchmark functions. However, its convergence is still very slow. J. Niu et al. proposed an improved FOA based on differential evolution (DFOA). They modified the expression of the smell concentration judgment value and introduced a differential vector to replace the stochastic search [24]. The convergence of DFOA was analyzed based on 12 benchmark functions. However, the influence on different differential vectors was not considered. X. Yuan et al. proposed an improved FOA based on multi-swarm (MFOA) [25]. The application of MFOA to several benchmark functions and parameter identification of synchronous generator shows an improvement in its performance over the original FOA technique. However, the parameters and the sub-swarm number of MFOA usually impact its search performance.

In order to tackle the limitation of FOA, this paper presents an improved FOA named MSFOA based on a Multi-Scale Cooperative Mutation (MSCM) mechanism with the following main contributions:

(1) The limitations and convergence of the original FOA are analyzed theoretically. We demonstrate that the convergence of FOA depends on the initial swarm location;
(2) An improved fruit fly optimization algorithm, named MSFOA, is developed. MSFOA can escape from local optimum and achieve much better global optimum by employing the MSCM mechanism to mutate the position of the swarm;
(3) Through extensive experiments, we evaluate the convergence, stability, and global optimization ability of MSFOA in comparison with the most recent improved versions of FOA, including IFFO, CFOA and CMFOA, based on 29 benchmark functions.

The remainder of the paper is organized as follows. In Section 2, FOA and its limitation are analyzed. The convergence and the convergence condition of FOA are analyzed in Section 3. Section 4 discusses the MSCM mechanism and MSFOA in detail. MSFOA are evaluated experimentally in Section 5. Finally, Section 6 concludes this paper.

2. Fruit Fly Optimization Algorithm

This section first introduces the original FOA. It then analyzes its limitations theoretically.

2.1. FOA basics

FOA was inspired by the foraging behaviors of fruit flies in the nature [12]. A fruit fly determines the location of food with its unique osphresis, its vision, and the smell concentration. The optimization process of FOA consists of the following 8 steps.

Step 1: Initialization the location of fruit fly swarm.

$$\begin{cases} x\_axis = rand(LR) \\ y\_axis = rand(LR) \end{cases} \tag{1}$$

where LR represents the location parameter of the initial swarm.

Step 2: Generation of the location of individual fruit flies in the swarm.

$$\begin{cases} x_i = x\_axis + rand(V) \\ y_i = y\_axis + rand(V) \end{cases} \tag{2}$$

where V represents the range parameter generated by the swarm, x and y represent the location coordinates.

Step 3: Calculation of the distance between individual fruit fly and origin.

$$Dist_i = \sqrt{x_i^2 + y_i^2} \tag{3}$$

Step 4: Calculation of the smell concentration judgment value of each individual fruit fly.

$$S_i = \frac{1}{Dist_i} \tag{4}$$

Step 5: Calculation of the smell concentration value of each individual fruit fly.

$$Smell_i = Smell\_function(S_i) \tag{5}$$

Step 6: Identification of the optimal smell concentration value in the swarm, denoted by the maximum value.

$$[bestSmell \quad bestindex] = max(Smell_i) \tag{6}$$

Step 7: Reservation of the identified optimal smell concentration value and replacement of the swarm location.

$$\begin{cases} Smellbest = bestSmell \\ x\_axis = x_{bestindex} \\ y\_axis = y_{bestindex} \end{cases} \tag{7}$$

Step 8: Termination of the algorithm if the maximum number of generation is reached; otherwise, go back to **Step 2**.

2.2. Analysis of FOA

Compared with other SI based optimization algorithms, FOA has the advantages of simple structure and low computational complexity. However, FOA cannot solve complex optimization problems efficiently, due to its following limitations.

Limitation 1. FOA is prone to converge into a local optimum because it cannot traverse the entire solution space.

Proof. The calculation of distance ($Dist_i$) in FOA is $Dist_i = \sqrt{x_i^2 + y_i^2}$ and the smell concentration judgment value is $S_i = \frac{1}{Dist_i}$. Therefore, $Dist_i$ and S_i are always greater than 0. Thus, the search space cannot reach the negative threshold. □

Limitation 2. Fruit fly individual values are relatively monotonous and tend towards zero.

Proof. Step 2 to 4 presented in Section 2.1 describe the calculation of the distance and the smell concentration judgment value of individual fruit flies. The calculation process is as follow.

According to Eqs. (3) and (4), when the problem has a large search space and the individual value of x or y are large, S_i always tends toward zero, driving the algorithm to converge into a local optimum. While the algorithm has an initial advantage at the extreme points $X^* = 0$, it can result in weak swarm variation and limited global optimization capacity.

To conclude, FOA is prone to converge into a local optimum inherently, except for the $X^* = 0$ benchmark functions. □

3. Convergence analysis of FOA

In FOA, the swarm generation follows Eq. (2). If the coordinates of each fruit fly are convergent in the swarm, FOA is also convergent. Thus, Eq. (2) can be used to analyze the convergence and global optimization ability of FOA. Based on Eq. (2), the generation of individuals can be rewritten as follows:

$$\begin{cases} x_i = x\_axis + rand_1(V) \\ y_i = y\_axis + rand_2(V) \end{cases} \tag{8}$$

If we consider only the position change of the swarm and suppose that $rand_1() = \varphi_1, rand_2() = \varphi_2$, there is $rand_1(V) = V\varphi_1, rand_2(V) = V\varphi_2$. We then have the following analysis.

(I) When $V\varphi_1$ and $V\varphi_2$ are constant, i.e., the change of each random value is not considered, we only consider the variety of swarm location, i.e., (x_n, y_n) represents the nth swarm location (x_axis, y_axis), Eq. (8) can be simplified as:

$$\begin{cases} x_n = x_{n-1} + V\varphi_1 \\ y_n = y_{n-1} + V\varphi_2 \end{cases} \tag{9}$$

Eq. (9) can be further transformed:

$$\begin{cases} x_n = x_0 + nV\varphi_1 \\ y_n = y_0 + nV\varphi_2 \end{cases} \tag{10}$$

There is:

$$\begin{cases} \lim\limits_{n\to+\infty} x_n = \lim\limits_{n\to+\infty}(x_0 + nV\varphi_1) = \lim\limits_{n\to+\infty} x_0 + \lim\limits_{n\to+\infty} nV\varphi_1 \\ \lim\limits_{n\to+\infty} y_n = \lim\limits_{n\to+\infty}(y_0 + nV\varphi_2) = \lim\limits_{n\to+\infty} y_0 + \lim\limits_{n\to+\infty} nV\varphi_2 \end{cases} \tag{11}$$

In Eq. (11), where $V\varphi_1$ and $V\varphi_2$ are constant, FOA does not converge and thus does not guarantee a global optimum.

(II) When $V\varphi_1$ and $V\varphi_2$ are not a constant, i.e., random change occurs during the iteration process, $V\varphi_1$ and $V\varphi_2$ will be recorded as $V\varphi_{1,t}$ and $V\varphi_{2,t}$, indicating the random values generated by the first t iteration. In this case, Eq. (8) can be simplified as follows:

$$\begin{cases} x_n = x_{n-1} + V\varphi_{1,n-1} \\ y_n = y_{n-1} + V\varphi_{2,n-1} \end{cases} \tag{12}$$

Eq. (12) can be further converted:

$$\begin{cases} x_n = x_0 + V\sum\limits_{i=1}^{n}\varphi_{1,i} \\ y_n = y_0 + V\sum\limits_{i=1}^{n}\varphi_{2,i} \end{cases} \tag{13}$$

From Eq. (13), we obtain that

$$\begin{aligned} \lim_{n\to+\infty} x_n &= \lim_{n\to+\infty}(x_0 + V\sum_{i=1}^{n}\varphi_{1,i}) = \lim_{n\to+\infty} x_0 \\ &\quad + \lim_{n\to+\infty} V\sum_{i=1}^{n}\varphi_{1,i} = x_0 + \lim_{n\to+\infty} V\sum_{i=1}^{n}\varphi_{1,i} \end{aligned} \tag{14}$$

Similarly,

$$\begin{aligned} \lim_{n\to+\infty} y_n &= \lim_{n\to+\infty}(y_0 + V\sum_{i=1}^{n}\varphi_{2,i}) = \lim_{n\to+\infty} y_0 \\ &\quad + \lim_{n\to+\infty} V\sum_{i=1}^{n}\varphi_{2,i} = y_0 + \lim_{n\to+\infty} V\sum_{i=1}^{n}\varphi_{2,i} \end{aligned} \tag{15}$$

where $\varphi_{1,i}, \varphi_{2,i} \in [-1, 1]$, indicate random values within the range of [-1, 1]. Hence, $\varphi_{1,\,i}, \varphi_{2,\,i}$ can be simplified as φ_i. We need only to analyze the convergence property of $\lim\limits_{n\to+\infty}\sum_{i=1}^{n}\varphi_i$ in Eqs. (14) and (15).

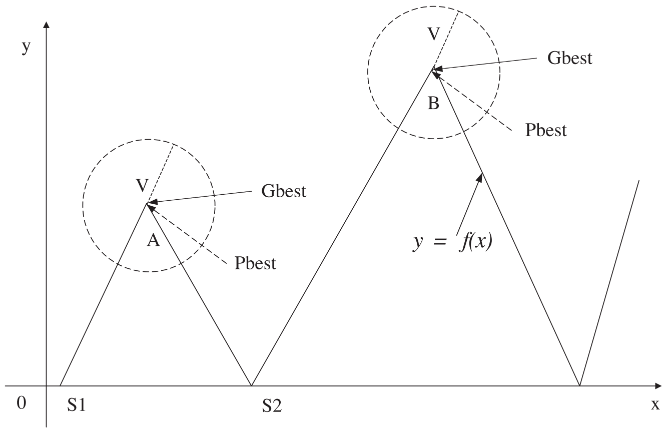

Fig. 1. Effect of the swarm position on global optimization ability.

Because $\varphi_i \in [-1, 1]$ is a random value, different values of φ_i appear with the same probability P. The expectation of φ_i is

$$\begin{aligned} E_\varphi &= \int_{-1}^{1} P \times f(\varphi)d\varphi = \int_{-1}^{1} P \times \varphi d\varphi \\ &= P\int_{-1}^{1}\varphi d\varphi = P \times 0.5 \times \varphi^2|_{-1}^{1} = \frac{P}{2}(1-1) = 0 \end{aligned} \tag{16}$$

Therefore, given $\lim\limits_{n\to+\infty}\sum\limits_{i=1}^{n}\varphi_i = 0$, there is

$$\begin{cases} \lim\limits_{n\to+\infty} x_n = x_0 + 0 = x_0 \\ \lim\limits_{n\to+\infty} y_n = y_0 + 0 = y_0 \end{cases} \tag{17}$$

Eq. (17) shows that the coordinates of the fruit fly swarm converge to the initial coordinates. Since the initial coordinates are randomly generated, FOA does not converge and thus does not guarantee a global optimum. According to Eq. (17), the convergence position of the swarm depends strongly on the initial position of the swarm. Hence, the initial location of the swarm affects the ability of FOA to achieve a global optimum. The influence relation is illustrated in Fig. 1.

As demonstrated in Fig. 1, if the initial position of the swarm is S_1, point A will be found as the extreme point. If the initial position of the swarm is S_2, point B will be found as the extreme point. Point A is obviously not the global extremum. When the iteration process reaches the extremum point A, the swarm position must be adjusted with multi-scale variation in order to ensure that it escapes the local optimum.

4. MSFOA

This section presents MSFOA, our improved fruit fly optimization algorithm based on a multi-scale cooperative mutation mechanism. MSFOA overcomes FOA's limitation of local optimum caused by its dependence of the initial location of the swarm. When the algorithm reaches a local extreme value, it mutates the position of the swarm to escape from the local optimum. Based on the new position of the swarm, it then searches for a new extreme value iteratively to find the global optimum.

4.1. Multi-scale cooperative mutation

In each iteration of MSFOA, the location of the escape is determined using the uniform mutation scale. The mutation scale allows MSFOA to escape from the swarm location of the last generation. However, the distance between the local extrema of the given benchmark function cannot be predicted. Thus, an appropriate mutation scale cannot be determined in advance. In order for MSFOA

Table 1
Parameter settings.

Algorithm	Parameter	Value	Significance
FOA	V	1	The range size of fly swarm
CFOA	$\cos(i\cos^{-1}(x_i))$	Chebyshev	Chaotic map function
CMFOA	En_{max}	$(Ub-Lb)/4$	Maximum entropy
	En	$En_{max}\times(1-\frac{t}{iter})^{\alpha}$	Entropy
	He	$0.1En$	Hyper Entropy
	Ex	X_axis	Expectation
IFFO	λ_{max}	$(Ub-Lb)/2$	Maximum radius
	λ_{min}	10^{-5}	Minimum radius
MSFOA	n	0.005	Search coefficient
	ω_0	1	Initial weight
	α	0.95	Weight coefficient
	M	5	Scale number

to find an optimal solution efficiently, the following issues must be addressed when choosing an appropriate mutation scale.

(1) If the mutation scale is too large, some extreme points may be skipped, the global extremum included.
(2) If the mutation scale is too small, a large number of iterations may be required to traverse the entire search space. This may significantly slow down the convergence of the algorithm.

To address the above issues, MSFOA employs a *Gaussian* mutation operator with differently scaled variances to escape from local optimum.

4.1.1. Gaussian *mutation operator*

Suppose that there are a total of M mutation scales. The variance of the *Gaussian* mutation operator is initialized:

$$\delta^{t_0}=(\delta_1^{t_0},\delta_2^{t_0},\ldots,\delta_M^{t_0}) \tag{18}$$

where $\delta_i^{t_0}$ represents the variance of the ith scale in the t_0 iteration, and its initial value is defined as the entire solution space. During the iteration process, the variance is adjusted as follows: 1) the particles in the swarm are sorted according to the fitness value; 2) the sorted particles are grouped to generate M subgroups, with $P=N/M$ particles in each subgroup with the fitness value of each subgroup calculated as follows:

$$Fit_m^t=\sum_{i=1}^{P}f(x_i^m)/P, m=1,2,\ldots,M \tag{19}$$

where Fit_m^t is the fitness value of the mth subgroup in the tth iteration and $f(x_i^m)$ indicates the fitness value of the ith particle in the mth subgroup. Then, according to the fitness value of the subgroup, the standard deviation of each subgroup is adjusted dynamically. The adjustment process is described as follows:

(1) The maximum and minimum values of all subgroups are obtained, denoted as Fit_{max}, Fit_{min}, respectively.
(2) The fitness value of each subgroup is set in the tth iteration according to the Eqs. (20)–(22):

$$Fit_{max}=max(Fit_1^t,Fit_2^t,\ldots,Fit_M^t) \tag{20}$$

$$Fit_{min}=min(Fit_1^t,Fit_2^t,\ldots,Fit_M^t) \tag{21}$$

$$\delta_m^t=\delta_m^{t-1}exp\left(\frac{M\times Fit_m^t-\sum_{i=1}^{M}Fit_i^t}{Fit_{max}-Fit_{min}}\right) \tag{22}$$

As the algorithm iterates, the mutation operator may become very large. Therefore, the standard deviation of the mutation operator is required to normalize the mutation operator:

$$\delta_m^t=|W/4-\delta_m^t|,(\delta_m^t>W/4) \tag{23}$$

where W is the definition domain of the given optimization problem. This is repeated until $\delta_m^t>W/4$.

(3) According to the standard deviation of each subgroup, the swarm location is defined as follows:

$$x_i=\begin{cases}x_i+rand(0,1)\times\delta_i^t, & conditional\\ x_i+rand(0,1)\times W, & others\end{cases} \tag{24}$$

where *conditional* represents $f(x_i+rand(0,1)\times\delta_i^t)<f(x_i+rand(0,1)\times W)$ and $f(x_i+rand(0,1)\times\delta_i^t)=min(f(x_i+rand(0,1)\times\delta_j^t))$, $j=1,2,\ldots,M$, $rand(0,\ 1)$ is a random value between zero and one.

4.1.2. *MSCM algorithm*

According to the description of the *Gaussian* mutation operator above, the MSCM algorithm is described in Algorithm 1.

Algorithm 1 *MSCM*()

```
Input:
    the fitness value $f(x_i)$ of each particle in swarm
Output:
    the position of swarm
1: $sort(f(x_i)), i=1,2,\ldots,sizepop$
2: $SubGroup_j=Separation(f(x_i)), j=1,2,\ldots,M$
3: Calculate the fitness of each subgroup according to Eq. (19)
4: Calculate the standard deviation of each subgroup according to Eq. (20)–Eq. (22)
5: while $\delta_m^t>W/4, m=1,2,\ldots,M$ do
6:     $\delta_m^t=|W/4-\delta_m^t|$
7: end while
8: Determine the location of the new swarm according to (Eq. 24)
9: Return the swarm position
```

Where δ represents the variance of the *Gaussian* mutation operator at each mutation scale.

4.2. *Improved Fruit Fly Optimization Algorithm based on MSCM*

The original FOA does not traverse the entire search space and thus lacks the ability to find the global optimal solution. To overcome this limitation, MSFOA employs a linear generation mechanism based on MSCM. The pseudo code of MSFOA is presented in Algorithm 2.

In Algorithm 2, ω_0, α and n are the search coefficient, initial weight, and weight coefficient, respectively, function *Get*(R_{min}, R_{max}) obtains the definition domain boundary of the benchmarks function, function *Get*(*Pbest, Pi, Gbest, Gi*) obtains the local extreme value *Pbest*, the global extreme value *Gbest*, and the corresponding particle number *Pi, Gi*.

Table 2
Benchmark functions.

ID	Function name	Equation	Global optimum	Domain
F1	Axis parallel hyperellipsoid	$f(x)=\sum_{i=1}^{n} ix_i^2$	$x^*=0$ and $f(x^*)=0$	$-5.12 \le x_i \le 5.12$
F2	Dixon-price	$f(x)=\sum_{i=2}^{n} i\times(2x_i^2-x_{i-1})^2+(x_i-1)^2$	$x^*=0$ and $f(x^*)=0$	$-10 \le x_i \le 10$
F3	Exponential problem	$f(x)=-exp(-0.5\sum_{i=1}^{n}(x_i^2))$	$x^*=0$ and $f(x^*)=-1$	$-1 \le x_i \le 1$
F4	High conditioned elliptic	$f(x)=\sum_{i=1}^{n}(10^6)^{\frac{i-1}{n-1}} x_i^2$	$x^*=0$ and $f(x^*)=0$	$-10 \le x_i \le 10$
F5	Quartic	$f(x)=\sum_{i=1}^{n} ix_i^4+rand()$	$x^*=0$ and $f(x^*)=0$	$-1.28 \le x_i \le 1.28$
F6	Rosebrock	$f(x)=\sum_{i=1}^{n-1}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)$	$x^*=(1,1,\dots,1)$ and $f(x^*)=0$	$-30 \le x_i \le 30$
F7	Schwefels problem	$f(x)=\sum_{i=1}^{n}(\sum_{j=1}^{i} x_j)^2$	$x^*=0$ and $f(x^*)=0$	$-100 \le x_i \le 100$
F8	Schwefels problem 2.21	$f(x)=\max\|x_i\|, 1\le i\le n$	$x^*=0$ and $f(x^*)=0$	$-100 \le x_i \le 100$
F9	Schwefels problem 2.22	$f(x)=\sum_{i=1}^{n}\|x_i\|+\prod_{i=1}^{n}\|x_i\|$	$x^*=0$ and $f(x^*)=0$	$-100 \le x_i \le 100$
F10	Sphere	$f(x)=\sum_{i=1}^{n} x_i^2$	$x^*=0$ and $f(x^*)=0$	$-100 \le x_i \le 100$
F11	Step	$f(x)=\sum_{i=1}^{n}(\lfloor x_i+0.5\rfloor)^2$	$x^*=0$ and $f(x^*)=0$	$-100 \le x_i \le 100$
F12	Sum of different power	$f(x)=\sum_{i=1}^{n}\|x_i\|^{i+1}$	$x^*=0$ and $f(x^*)=0$	$-1 \le x_i \le 1$
F13	Sum squares	$f(x)=\sum_{i=1}^{n} ix_i^2$	$x^*=0$ and $f(x^*)=0$	$-1 \le x_i \le 1$
F14	Shifted sphere	$f(x)=\sum_{i=1}^{n} z_i^2+f\_bias_1$	$z=x-o, o=$ $(o_1,o_2,\dots,o_n), x^*=o$ and $f(x^*)=f\_bias_1=-450$	$-100 \le x_i \le 100$
F15	Shifted Schwefels problem 1.2	$f(x)=\sum_{i=1}^{n}(\sum_{j=1}^{i} z_j)^2+f\_bias_2$	$z=x-o, o=$ $(o_1,o_2,\dots,o_n), x^*=o$ and $f(x^*)=f\_bias_2=-450$	$-100 \le x_i \le 100$
F16	Ackley	$f(x)=-20exp(-0.2\sqrt{\frac{1}{n}\sum_{i=1}^{n} x_i^2})$ $-exp(\frac{1}{n}\sum_{i=1}^{n}\cos(2\pi x_i))+20+e$	$x^*=0$ and $f(x^*)=0$	$-32 \le x_i \le 32$
F17	Alpine	$f(x)=\sum_{i=1}^{n}\|x_i\sin x_i+0.1x_i\|$	$x^*=0$ and $f(x^*)=0$	$-10 \le x_i \le 10$
F18	Expansion of F10	$f(x)=f_{10}(x_1,x_2)+\dots+f_{10}(x_{i-1},x_i)+f_{10}(x_n,x_1)$	$x^*=0$ and $f(x^*)=0$	$-100 \le x_i \le 100$
F19	Expanded Scaffer	$f(x)=f_s(x_1,x_2)+f_s(x_2,x_3)+\dots+f_s(x_n,x_1)$ $f_s(x,y)=0.5+\frac{\sin^2(\sqrt{x^2+y^2})-0.5}{(1+0.001(x^2+y^2))^2}$	$x^*=0$ and $f(x^*)=0$	$-100 \le x_i \le 100$
F20	Generalized penalized function 1	$f(x)=$ $\frac{\pi}{n}\{10\sin^2(\pi y_1)+\sum_{i=1}^{n-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]+$ $(y_n-1)^2\}+\sum_{i=1}^{n}\mu(x_i,10,100,4),\ y_i=$ $1+0.25(x_i+1)$ $\mu(x_i,\alpha,k,m)=\begin{cases}k(x_i-\alpha)^m, x_i>\alpha\\ 0, -\alpha\le x_i\le\alpha\\ k(-x_i-\alpha)^m, x_i<-\alpha\end{cases}$	$x^*=(-1,\dots,-1)$ and $f(x^*)=0$	$-50 \le x_i \le 50$
F21	Griewank	$f(x)=\frac{1}{4000}\sum_{i=1}^{n} x_i^2-\prod_{i=1}^{n}\cos(\frac{x_i}{\sqrt{i}})+1$	$x^*=0$ and $f(x^*)=0$	$-600 \le x_i \le 600$
F22	Inverted cosine wave	$f(x)=-\sum_{i=1}^{n-1}(exp(-\frac{x_i^2+x_{i+1}^2+0.5x_ix_{i+1}}{8}))$ $\times\cos(4\sqrt{x_i^2+x_{i+1}^2+0.5x_ix_{i+1}}))$	$x^*=0$ and $f(x^*)=1-n$	$-5 \le x_i \le 5$
F23	Neumaier 3 problem	$f(x)=\sum_{i=1}^{n}(x_i-1)^2-\sum_{i=2}^{n} x_ix_{i-1}$	$x^*=i(n+1-i)$ and $f(x^*)=-\frac{n(n+4)(n-1)}{6}$	$-n^2 \le x_i \le n^2$
F24	Pathologic	$f(x)=\sum_{i=1}^{n-1}(0.5+\frac{\sin^2\sqrt{100x_i^2+x_{i+1}^2}-0.5}{1+0.001(x_i^2-2x_ix_{i+1}+x_{i+1}^2)^2})$	$x^*=0$ and $f(x^*)=0$	$-100 \le x_i \le 100$
F25	Rastrigin	$f(x)=\sum_{i=1}^{n}(x_i^2-10\cos(2\pi x_i)+10)$	$x^*=0$ and $f(x^*)=0$	$-5.12 \le x_i \le 5.12$
F26	Non-continuous rastrigin	$f(x)=\sum_{i=1}^{n}(y_i^2-10\cos 2\pi y_i+10)$ $y_i=\begin{cases}x_i,\ \|x_i\|<0/5\\ round(2x_i)/2,\ \|x_i\|\ge 0.5\end{cases}$	$x^*=0$ and $f(x^*)=0$	$-5.12 \le x_i \le 5.12$
F27	Salomon	$f(x)=1-\cos(2\pi\sqrt{\sum_{i=1}^{n} x_i})+0.1\sum_{i=1}^{n} x_i^2$	$x^*=0$ and $f(x^*)=0$	$-100 \le x_i \le 100$
F28	Weierstrass	$f(x)=\sum_{i=1}^{n}\{\sum_{k=0}^{k_{\max}}[a^k\cos(2\pi b^k(x_i+0.5))]\}$ $-n\sum_{k=0}^{k_{\max}} a^k\cos(2\pi b^k\times 0.5),\ a=0.5, b=3, k_{\max}=30$	$x^*=0$ and $f(x^*)=0$	$-0.5 \le x_i \le 0.5$
F29	Whitley	$f(x)=\sum_{k=1}^{n}\sum_{j=1}^{n}(\frac{y_{jk}^2}{4000}-\cos y_{jk}+1)$ $y_{jk}=100(x_k-x_j^2)^2+(1-x_j^2)^2$	$x^*=(1,\dots,1)$ and $f(x^*)=0$	$-100 \le x_i \le 100$

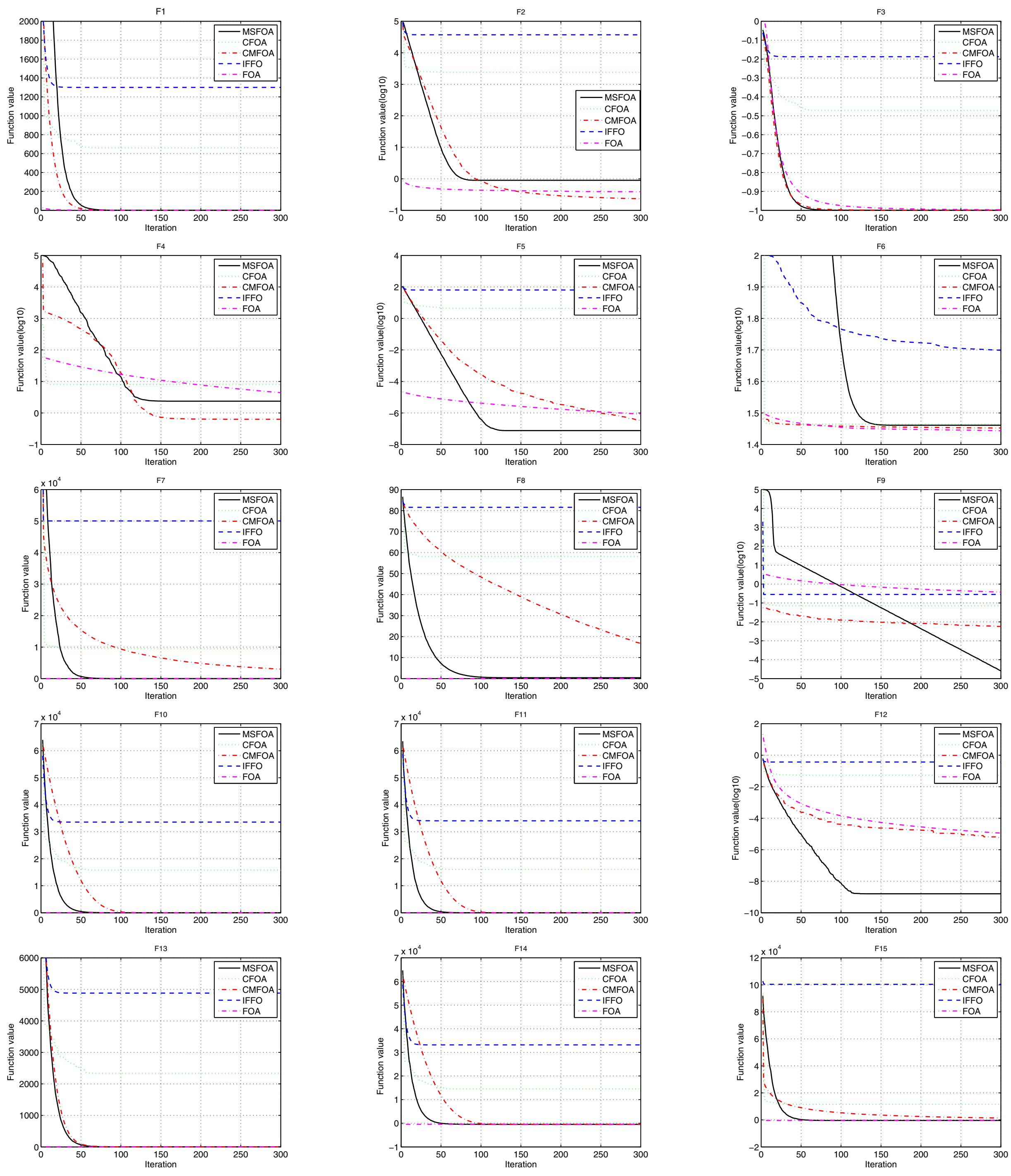

Fig. 2. Results of unimodal functions on $D = 30$.

5. Experimental evaluations

This section compares MSFOA with FOA and 3 existing improved version of FOA, including IFFO, CFOA and CMFOA.

5.1. Experimental setup and parameter configuration

The experiments were conducted on a machine running Windows7 64 bit with Intel Core i7 3.6 GHz and 16 G memory. The parameters of each algorithm are listed in Table 1.

5.2. Benchmark functions

We implemented MSFOA and evaluated the its performance on 29 scalable benchmark functions with 30 and 50 dimensions, which have been widely used for evaluating global optimization algorithms [16,18,19]. Table 2 summarizes those functions. Among these functions, the first 15 ones are unimodal and the other are multimodal. Using these functions, we can comprehensively evaluate the performance of MSFOA.

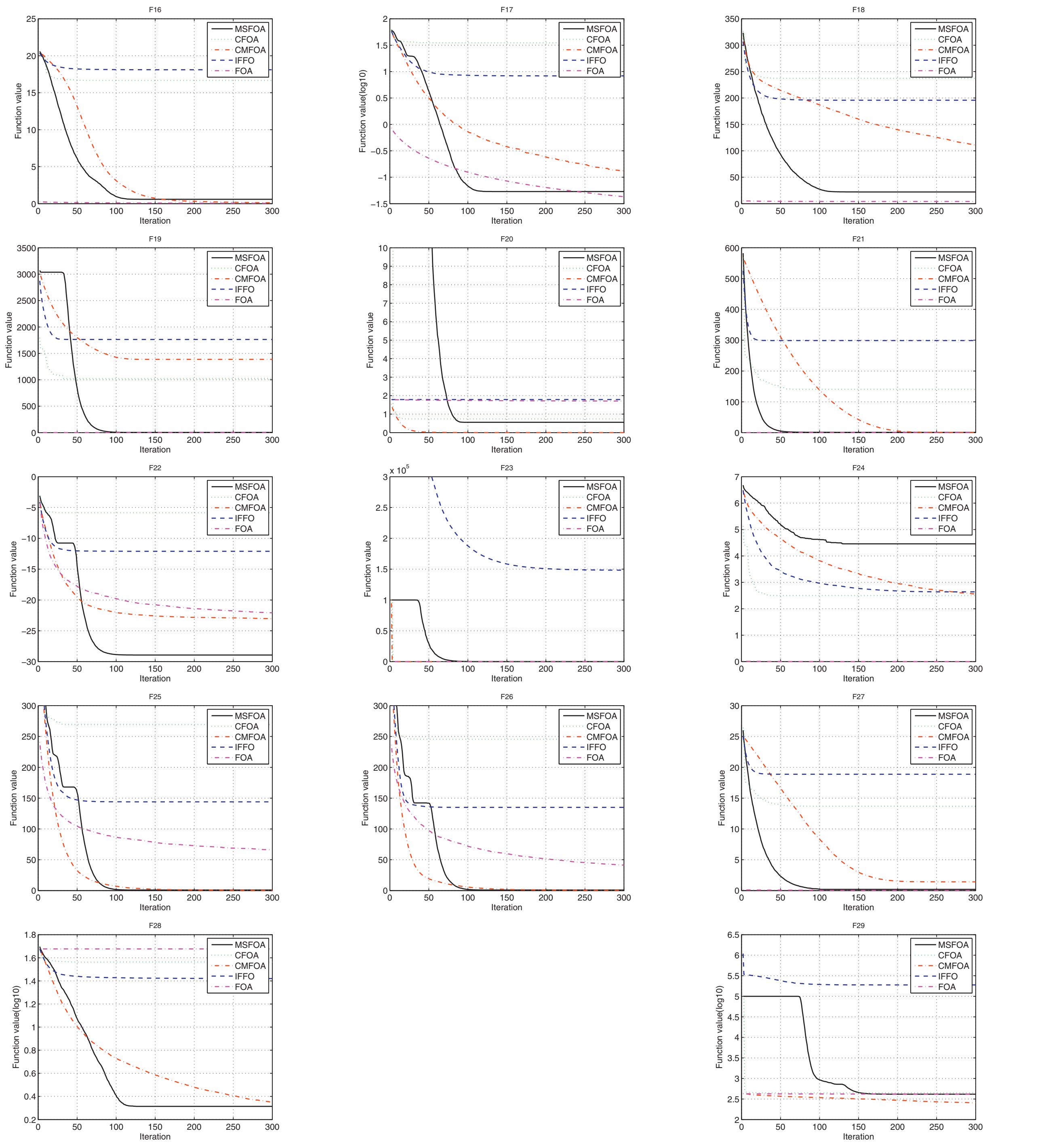

Fig. 3. Results of multimodal functions on $D = 30$.

5.3. Experimental results

To compare the algorithm in a fair and objective manner, we used a same set of randomly generated initial swarms for all algorithms. The swarm size is 50 and the number of iteration is 300. The algorithms stop after achieving constant results from 50 runs and the results are averaged. The root mean square error (RMSE) is calculated for each experimental result x_i and the minimum x_{opt} given in Table 2 according to Eq. (25), where n is the total number of runs.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(x_i - x_{opt})^2}{n}} \tag{25}$$

To evaluate the stability and the global optimization ability of the algorithms, the experiments were conducted with dimensions $D = 30$ and $D = 50$, Tables 3–6 present the experimental results, where the *fit* is the average fitness value of the experimental results, *RMSE* measures the stability of the results averaged across 50 runs and *minfit* measures the optimal results of the 50 runs.

Table 3 presents many interesting findings. Among the 15 unimodal benchmark functions with $D = 30$, MSFOA finds 9 signifi-

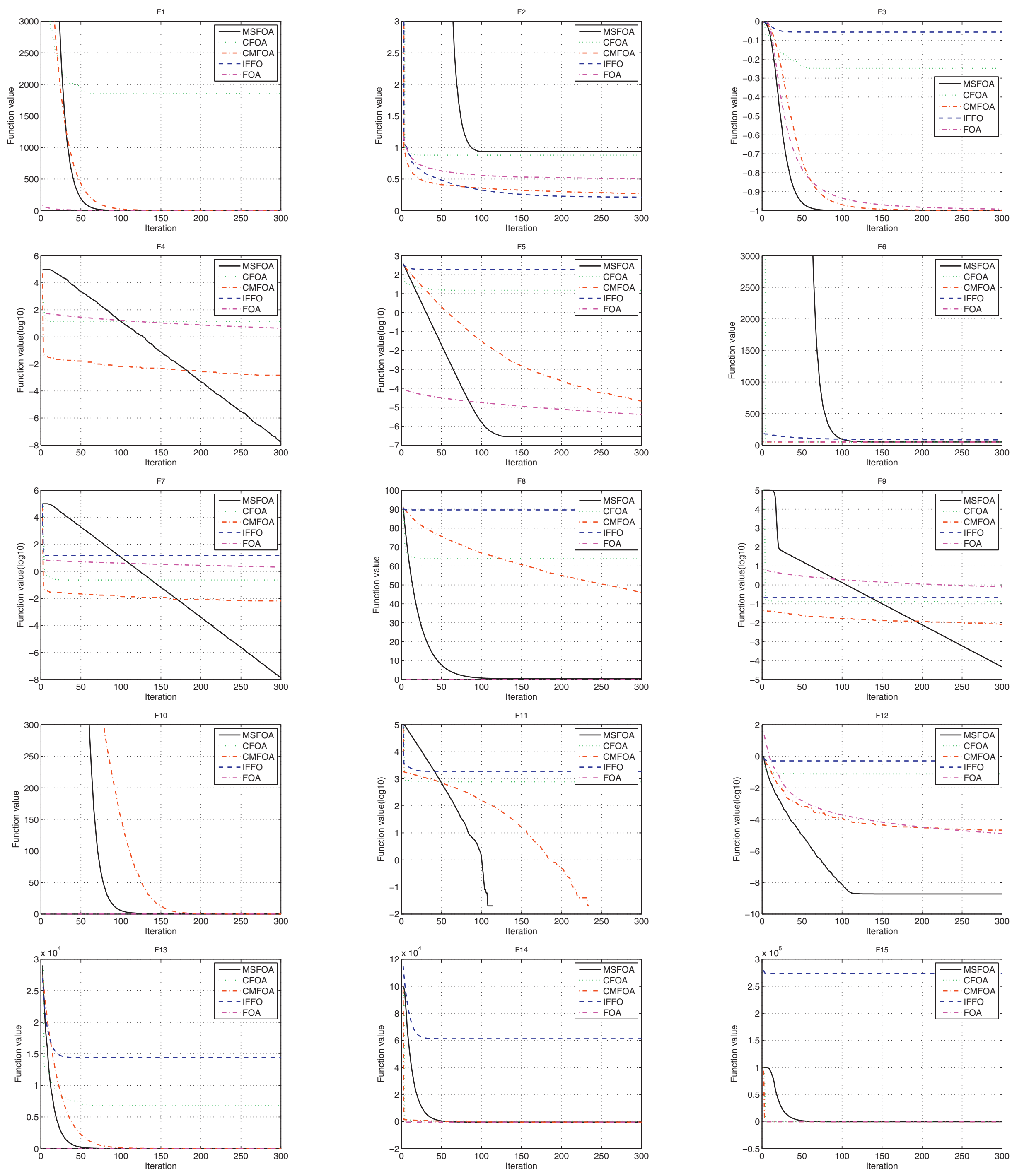

Fig. 4. Results of unimodal functions on $D = 50$.

cantly better solutions than CFOA, CMFOA and IFFO, whose benchmark functionsare F3/5/6/7/8/9/12/13/15. In addition, with benchmark function F11, MSFOA and CMFOA both obtain the optimal solution, which is superior to CFOA and IFFO. FOA demonstrates a different pattern. Although it is able to find the optimal solution easily in specific scenarios where $X^* = 0$, it limitations significantly weakens its ability, as analyzed in Section 2.2. Generally, the success rate of MSFOA outperforms FOA, CMFOA and IFFO in unimodal benchmark functions. For non-optimal benchmark functions, functions F1/2/10/14 find near-optimal solutions. The success rate of MSFOA in finding near-optimal solutions is $4/5 = 80\%$. Table 4 demonstrates that out of the 14 multimodal benchmark functions with $D = 30$, MSFOA obtains better solutions for functions F17/18/19/22/23/27/28. The solutions found by MSFOA for the other multimodal benchmark function with $D = 30$ are near-optimal, including functions F16/20/21/25/26/29. It can be seen that for the unimodal benchmark functions with $D = 30$, the success rate of MSFOA outperforms CFOA, CMFOA and IFFO by $7/14 = 50\%$ on average. As for the non-optimized benchmark functions, the success rate of MSFOA in finding the near-optimal solution is $6/7 = 85.7\%$. Overall, with $D = 30$, MSFOA outperforms the other algorithms by $17/29 = 58.6\%$. For the non-optimized benchmark functions, the success rate of MSFOA finding the near-optimal solution is $10/12 = 83.3\%$.

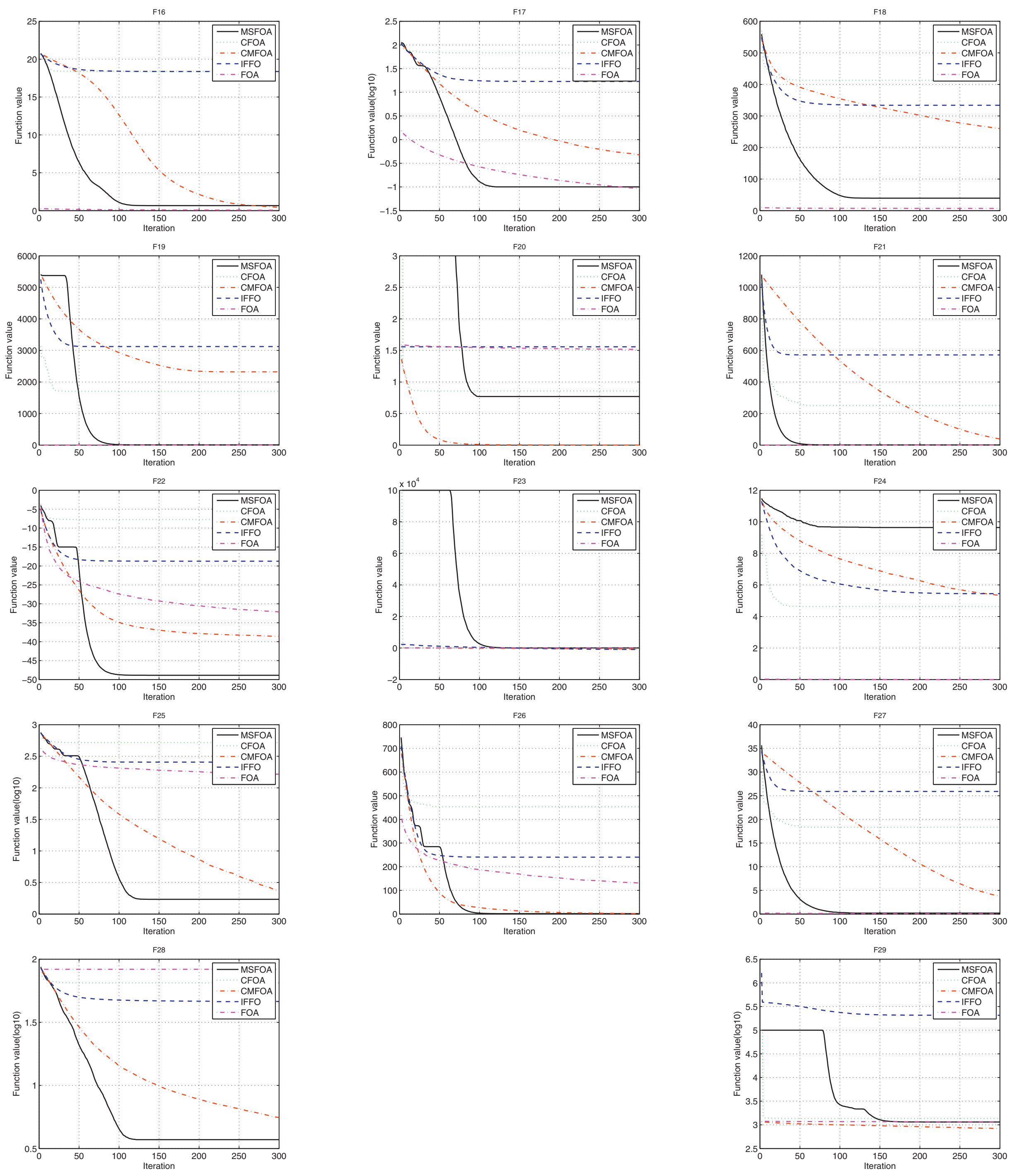

Fig. 5. Results of multimodal functions on $D = 50$.

Table 5 presents the experimental results for unimodal functions with $D = 50$. Among the 15 unimodal benchmark functions, MSFOA finds 8 significantly better solutions than CFOA, CMFOA, IFFO and FOA, including functions F3/5/6/7/8/9/12/13/15, and 5 near-optimal solutions for functions F1/6/10/13/14. Table 6 shows that MSFOA finds the optimal solution, which is 10 times better than the other algorithms for unimodal benchmark functions, including F16/17/18/19/21/22/25/26/27/28. For functions F20/29, MSFOA finds near-optimal solutions. Overall, MSFOA outperforms the other algorithms by $10/14 = 71.4\%$ with $D = 50$. In addition, for the other non-optimized benchmark functions, the success rate of MSFOA in finding the near-optimal solution is $2/4 = 50\%$. Based on discussion above, with $D = 50$, MSFOA can increase the success rate by $19/29 = 65.5\%$ based on the experimental results. Besides, the success rate of MSFOA in finding near-optimal solutions is $7/10 = 70\%$. The experimental results show that, as the number of dimensions increases, MSFOA is more likely to outperform the other algorithms.

5.4. Convergence analysis

To evaluate ability of MSFOA to escape local optima, Figs. 2–5 illustrate the convergence curves for obtained mean values from 300 iterations of each algorithm in different sets of experiments.

Table 3
Results based on unimodal functions with $D = 30$.

	MSFOA			CFOA			CMFOA			IFFO			FOA		
	fit	*RMSE*	*minfit*	*fit*	*RMSE*	*minfit*	*fit*	*RMSE*	*minfit*	*fit*	*RMSE*	*minfit*	*fit*	*RMSE*	*minfit*
F1	0.2159	0.2204	0.1343	660.2475	678.5074	373.4377	**0.0269**	0.0296	0.0105	1300.736	1369.589	584.1743	0.0747	0.0747	0.0697
F2	0.8978	0.8983	0.8104	2379.321	5255.79	0.745	**0.2319**	0.2394	0.041	37160.19	45964.49	0.16033	0.3886	0.3898	0.3255
F3	**−0.9999**	9.03E−05	−0.9999	−0.4796	0.5342	−1.6222	−0.9998	0.0001	−0.9999	−0.1875	0.8159	−0.4079	−0.997	0.0029	−0.9972
F4	2.3677	2.6623	5.30E−09	8.0325	8.2833	3.7178	0.6291	4.4461	1.1973	**0**	0	0	4.378	4.3793	4.1928
F5	**7.60E−08**	7.96E−08	2.10E−08	4.3621	4.98	1.3566	3.35E−07	4.17E−07	3.48E−08	63.7886	6.90E+01	5.5663	8.37E−07	8.43E−07	6.56E−07
F6	28.9084	28.9084	28.8697	29.1372	29.1374	28.8973	**28.3267**	28.3273	27.7986	49.9864	67.943	19.3741	**27.7722**	27.773	27.1817
F7	**3.0875**	4.0459	4.2744	9506.081	1.74E+04	0.0175	2995.407	3954.645	0	50038.07	62948.34	0	**0.334**	0.3342	0.2929
F8	**0.4171**	0.4179	0.341	58.1084	58.3499	44.5589	14.6682	15.3765	3.4493	81.5304	81.8814	53.6711	**0.01**	0.01	0.009
F9	**2.46E−05**	2.46E−05	2.15E−05	0.071	7.10E−02	0.0558	0.0057	0.0078	4.13E−05	0.2819	0.417	0.0146	0.3767	0.3767	0.3685
F10	1.7005	1.7212	1.0415	15773.3	16318.49	9295.579	**0.0928**	0.1	0.0357	33563.36	35286.16	10000	**0.0012**	0.0012	0.0011
F11	**0**	0	0	16032.82	1.65E+04	8272	**0**	0	0	34054.14	35127.86	10000	**0**	0	0
F12	**1.60E−09**	2.49E−09	1.63E−12	0.0528	8.14E−02	0.004	6.04E−06	1.56E−05	2.84E−08	0.3682	4.21E−01	0.0178	1.12E−05	1.12E−05	1.05E−05
F13	0.2514	2.56E−01	0.1837	2336.439	2.42E+03	1489.376	**0.0593**	0.0629	0.0229	4881.062	5121.91	2376.668	**0.0676**	0.0676	0.0641
F14	−448.26	1.8145	−448.675	14580.82	1.55E+04	8442.303	−449.916	0.0896	−449.967	33165.12	35250.1	10000	**−449.998**	0.0012	−449.999
F15	**−447.014**	3.1573	−448.479	11500.54	1.95E+04	−449.985	1388.562	3068.412	−450	100353	107688.3	10000	**−449.666**	0.3342	−449.707

Table 4
Results based on multimodal functions with $D = 30$.

	MSFOA			CFOA			CMFOA			IFFO			FOA		
	fit	*RMSE*	*minfit*	*fit*	*RMSE*	*minfit*	*fit*	*RMSE*	*minfit*	*fit*	*RMSE*	*minfit*	*fit*	*RMSE*	*minfit*
F16	0.5921	5.98E−01	0.3303	16.6651	1.68E+01	11.8807	**0.1205**	0.1235	0.0584	18.1089	18.12	16.3887	**0.047**	0.047	0.0454
F17	**0.0536**	5.42E−02	0.0378	35.1172	3.55E+01	24.8742	0.1313	0.1536	0.0116	8.297	9.1585	1.7174	**0.0426**	0.04266	0.0416
F18	**22.1398**	2.22E+01	20.2797	237.2154	2.38E+02	188.7896	110.3349	117.4436	25.1415	195.7356	197.3435	133.5441	**4.1584**	4.4621	3.8426
F19	**3.5532**	3.634	1.5463	1023.277	1.08E+03	2.53	1386.112	1386.339	1312.543	1764.08	1767.256	1481.51	**0.0025**	0.0025	0.0021
F20	**0.5585**	5.68E−01	0.2763	0.7418	7.66E−01	0.377	**0.0002**	0.0004	5.66E−05	1.7906	1.7943	1.6718	1.7081	1.7081	1.706
F21	0.9661	9.67E−01	0.8753	140.4681	1.48E+02	83.1875	0.2594	0.2759	0.0802	298.8661	309.5331	85.2268	**5.84E−06**	5.85E−06	5.01E−06
F22	**−28.9297**	7.15E−02	−28.961	−5.8429	2.32E+01	−9.2334	−23.0247	6.1269	−26.5912	−12.0985	17.0523	−17.6643	−22.0844	8.457	−28.8958
F23	**15.9872**	4.95E+03	0.2277	**15.5299**	4.95E+03	1.3874	827.128	4103.259	953.333	148346.8	182416.9	10000	28.5484	4958.548	27.8234
F24	4.4591	4.4884	2.8701	2.5011	2.6854	2.2566	2.5451	2.5777	1.3086	2.6405	2.6745	1.7569	**0.0005**	0.0005	0.0004
F25	**0.882**	9.00E−01	0.5586	269.4565	2.72E+02	57.4233	**0.3156**	0.3368	0.1033	143.975	148.2346	62.425	65.8715	78.3757	8.0881
F26	**0.9228**	9.45E−01	0.5585	245.5428	2.47E+02	171.8963	**0.2949**	0.3393	0.1093	134.9764	138.4007	80.3198	41.2627	57.4151	2.5246
F27	**0.1965**	1.97E−01	0.1029	13.6886	1.38E+01	10.0998	1.4198	1.4512	0.5998	18.8718	19.0456	12.8998	**0.0289**	0.0289	0.0249
F28	**2.0606**	2.065	1.7414	36.5615	3.67E+01	25.6295	2.2385	2.2534	1.7347	26.3136	26.6626	18.091	47.5008	47.5375	41.82
F29	**413.9529**	413.9529	416.9529	445.7681	448.7133	309.0915	**255.5357**	257.1731	198.8975	189084.8	731768.4	57.3266	415.8849	415.8901	410.1574

Table 5
Results based on unimodal functions with $D = 50$.

	MSFOA			CFOA			CMFOA			IFFO			FOA		
	fit	*RMSE*	*minfit*	*fit*	*RMSE*	*minfit*	*fit*	*RMSE*	*minfit*	*fit*	*RMSE*	*minfit*	*fit*	*RMSE*	*minfit*
F1	0.7774	0.7886	0.4438	1851.076	1920.386	1230.201	**0.3307**	0.3487	0.136	3876.2845	3984.373	2012.9023	0.3373	0.3374	0.3231
F2	0.9339	0.934	0.9141	0.8788	0.8792	0.8139	0.2677	0.2715	0.1918	**0.2136**	0.2233	0.16	0.502	0.5034	0.4045
F3	−**0.9998**	0.0001	−0.9998	−0.2486	0.7546	−0.4004	−0.9991	0.0008	−0.9995	−0.0569	0.9437	−0.1819	−0.9919	0.008	−0.9923
F4	**1.44E−08**	1.64E−08	6.60E−09	14.114	1.44E+01	7.221	0.0014	0.004	2.48E−08	**0**	0	0	4.3658	4.3672	4.1956
F5	**2.82E−07**	2.91E−07	1.36E−07	14.9252	1.72E+01	5.6223	2.11E−05	2.85E−05	2.57E−06	1.94E+02	2.01E+02	9.29E+01	4.07E−06	4.11E−06	3.06E−06
F6	48.9021	48.9021	48.8121	49.3252	49.3254	49.0511	48.4185	48.4192	47.7928	82.6729	106.5196	44.3654	47.9556	47.9564	47.5122
F7	**1.24E−08**	2.09E−08	1.33E−08	0.2305	3.19E−01	0.0539	0.0065	0.0218	0	15.0457	92.15	0	2.0564	2.058	1.8584
F8	**0.4507**	4.51E−01	0.3945	63.948	6.41E+01	53.3332	45.9334	46.2545	32.3353	89.6021	89.6765	76.4056	**0.0122**	0.0122	0.0109
F9	**4.41E−05**	4.41E−05	4.03E−05	0.1323	1.33E−01	0.112	0.0081	0.0114	3.6519	0.2107	0.3059	0.0037	0.7885	0.7886	0.7697
F10	0.7952	1.70E+00	5.18E−09	656.6867	4643.09	0.0402	0.0234	0.1581	7.13E−07	3536.6438	12616.33	0.0229	0.0028	0.0028	0.0025
F11	**0**	0	0	776.24	5.49E+03	0	**0**	0	0	1898.62	9596.356	0	**0**	0	0
F12	**1.88E−09**	3.04E−09	7.49E−12	0.077	1.00E−01	0.0094	2.10E−05	6.43E−05	1.37E−08	0.5109	5.83E−01	0.0097	1.26E−05	1.27E−05	1.16E−05
F13	**0.8039**	8.11E−01	0.5637	6823.169	7.03E+03	4604.183	0.8132	0.8392	0.4653	14408.155	14676.83	8206.3354	**0.2939**	0.2939	0.2723
F14	−448.739	1.95	−450	343.6726	3.97E+03	−449.971	−449.976	0.1578	−450	61180.984	63232.59	10000	−**449.997**	0.0028	−449.997
F15	−**450**	1.24E−08	−450	−449.791	3.07E−01	−449.944	−449.982	0.04273	−450	273933	287470.6	10000	−447.959	2.0429	−448.257

Table 6
Results based on multimodal functions with $D = 50$.

	MSFOA			CFOA			CMFOA			IFFO			FOA		
	fit	*RMSE*	*minfit*	*fit*	*RMSE*	*minfit*	*fit*	*RMSE*	*minfit*	*fit*	*RMSE*	*minfit*	*fit*	*RMSE*	*minfit*
F16	0.6643	6.68E−01	0.4973	18.3905	1.84E+01	13.1238	0.4458	0.4561	0.2705	18.3673	18.3759	16.3262	**0.05795**	0.0579	0.0557
F17	**0.1002**	1.01E−01	0.0784	68.3847	6.93E+01	34.385	0.4735	0.5149	0.0702	16.9073	17.8582	6.6449	**0.0918**	0.0918	0.0898
F18	**39.3483**	3.94E+01	35.2296	413.1082	4.14E+02	331.7594	259.6809	262.2488	164.805	333.8484	335.0115	264.6297	**7.1659**	7.1741	6.6083
F19	**6.7695**	6.86	4.0319	1705.063	1.81E+03	119.2348	2321.455	2321.712	2241.511	3123.4775	3128.798	2721.8541	**0.0055**	0.0055	0.005
F20	**0.7694**	7.74E−01	0.5546	0.857	8.68E−01	0.548	**0.0005**	0.0009	0.0001	1.5572	1.5595	1.4737	1.5167	1.5167	1.5147
F21	**1.0144**	1.01E+00	0.9533	251.0033	2.58E+02	168.37	38.3089	41.2859	911663	571.729	582.862	306.8883	**7.27E−06**	7.30E−06	5.86E−06
F22	−**48.864**	1.37E−01	−48.9086	−7.7466	4.13E+01	−11.8059	−38.6263	10.5073	−44.1906	−18.7337	30.3893	−23.3547	−32.1087	18.1223	−46.2982
F23	**37.1414**	2.21E+04	27.913	65.325	2.21E+04	27.1035	−**647.13**	21402	−823.366	−914.6715	21139.83	−1794.305	48.8665	22098.87	47.9129
F24	9.6389	9.65	8.4221	4.6271	4.84	1	5.3249	5.3461	4.2536	5.4477	5.4673	4.5098	**0.0017**	0.0017	0.0013
F25	**1.7125**	1.73	1.1719	522.8578	5.25E+02	389.7509	2.2831	2.3771	1.1714	255.6745	259.3706	171.4577	164.2982	173.3458	15.8443
F26	**1.7402**	1.76	1.1827	453.1223	4.58E+02	118.8136	2.0501	2.1484	1.0664	240.4316	244.5045	137.7445	131.1847	153.5999	5.1885
F27	**0.2037**	2.05E−01	0.1998	18.3381	1.85E+01	13.5998	3.7217	3.8141	2.1999	25.9018	26.0572	18.4998	**0.0624**	0.0632	0.0535
F28	**3.7172**	3.72E+00	3.3112	64.903	6.50E+01	53.161	5.5468	5.5752	4.4716	46.2691	46.7507	29.8591	83.1681	83.2089	75.0286
F29	1149.869	1149.869	1149.869	1364.441	1367.508	1083.201	**834.2366**	836.2195	684.892	206719.77	661386.8	434.7899	1160.898	1160.9	1155.175

Algorithm 2 MSFOA

```
Input:
    benchmarks function and definition domain
Output:
    functional minimum value
 1: $Get(R_{min}, R_{max})$
 2: $Set(iter, sizepop, \omega_0, \alpha, n)$
 3: for each $t \le iter$ do
 4:   $\omega = \omega_0 \times \alpha^t$
 5:   for each $i \le sizepop$ do
 6:     for each $j \le dim$ do
 7:       $x_{i,j}^t = X_j^t + \omega \times rand(R_{min}, R_{max})$
 8:       $S_{i,j}^t = Dist_{i,j}^t = x_{i,j}^t$
 9:     end for
10:     $Calculate(Smell_i^t)$
11:   end for
12:   $Get(Pbest, Pi, Gbest, Gi)$
13:   $X = x_{G_i}$
14:   if $\Delta Gbest = 0$ then
15:     $times++$
16:   end if
17:   if $times \ge dim/2$ then
18:     $X = MSCM()$
19:   end if
20: end for
21: Return $Gbest$
```

Figs. 2 and 4 show that for unimodal benchmark functions, MSFOA steadily and gradually approaches the global optimum, especially for functions F8/9/10/11/12/14. In those cases, MSFOA has the best convergence. For other benchmark functions, it also converges fast. Figs. 3 and 5 demonstrate that for multimodal benchmark functions, MSFOA gradually approaches the optimal solution after taking several turns. This explicitly illustrates MSFOAs ability to escape local optima and gradually approach global optima. For benchmark functions F15/25/26/28 in particular, MSFOA achieves the best solutions and fastest convergence. The experimental results indicate that MSFOA outperforms IFFO, CFOA, CMFOA and the original FOA in global convergence ability.

6. Conclusions

In this paper, by analyzing FOA, we showed that the convergence of FOA depends on the initial location of the swarm, and that the randomness of the initial location of the swarm usually results in a local optimum. To address this issue, we proposed MSFOA, a novel multi-scale cooperative mutation Fruit Fly Optimization Algorithm. When reaching a local optimum, MSFOA triggers the MSCM mechanism that generates a new swarm location to escape the local optimum. The experimental results based on 29 widely used benchmark functions demonstrate that MSFOA significantly outperforms FOA and three most recent improved versions of FOA, i.e., IFFO, CFOA and MCFOA 36 out 54 cases, especially in high-dimensional cases. In the future, we are planning to further improve the performance of MSFOA. Specifically, we are going to extend it to multi-objective optimization and apply it to solve real-world engineering problems.

Acknowledgments

This work was supported by the National Key Technology R&D Program (No. 2015BAK24B01), the General Research for Humanities and Social Sciences Project of Chinese Ministry of Education (No. 15YJAZH112), the Natural Science Foundation of Anhui Province of China (No. 1408085MF132) and the Key Project of Nature Science Research for Universities of Anhui Province of China (No. KJ2016A038).

References

[1] M. Hewitt, A. Chacosky, S. Grasman, et al., Integer programming techniques for solving non-linear workforce planning models with learning, Eur. J. Oper. Res. 242 (3) (2015) 942–950.
[2] Y. Tang, H. He, J. Wen, et al., Power system stability control for a wind farm based on adaptive dynamic programming, Smart Grid, IEEE Trans. 6 (1) (2015) 166–177.
[3] J. Higle, S. Sen, Stochastic Decomposition: A Statistical Method for Large Scale Stochastic Linear Programming, Springer Science & Business Media, 2013.
[4] M. Chen, Y. Yan, QoS-aware service composition over graphplan through graph reachability, in: IEEE International Conference on Services Computing, Anchorage, Alaska, USA, 2014, pp. 544–551.
[5] D. Liu, S. Trajanovski, P.V. Mieghem, ILIGRA: an efficient inverse line graph algorithm, J. Math. Model. Algorithms Oper. Res. 14 (1) (2015) 13–33.
[6] L. Wu, C. Zou, H. Zhang, A cloud model based fruit fly optimization algorithm, Knowl.- Based Syst. 89 (2015) 603–617.
[7] E. Goodman, Introduction to genetic algorithms, in: Proceedings of the 2014 Conference Companion on Genetic and Evolutionary Computation Sompanion. ACM, 2014, pp. 205–226.
[8] R. Eberhart, Y. Shi, Particle swarm optimization: developments, application and resources, in: Proceedings of IEEE Congress of Evolutionary Computation, Seoul, Korea, 2001, pp. 81–86.
[9] M. Dorigo, V. Maniezzo, A. Colorni, The ant system: optimization by a colony of cooperating agents, IEEE Trans. Syst. Man Cybern. 26 (1996) 29–41.
[10] E. Rashedi, H. Nezamabadi-pour, S. Saryazdi, GSA: a gravitational search algorithm, Inf. Sci. 179 (2009) 2232–2248.
[11] S. Moein, R. Logeswaran, KGMO: a swarm optimization algorithm based on the kinetic energy of gas molecules, Inf. Sci. 275 (2014) 127–144.
[12] W. Pan, A new fruit fly optimization algorithm: taking the financial distress model as an example, Knowl. Based Syst. 26 (2) (2012) 69–74.
[13] J. Li, Q. Pan, K. Mao, A hybrid fruit fly optimization algorithm for the realistic hybrid flowshop rescheduling problem in steelmaking systems, IEEE Transactions on Automation Science & Engineering 9 (9) (2015) 1–18.
[14] Y. Zhang, G. Cui, Y. Wang, et al., An optimization algorithm for service composition based on an improved FOA, Tsinghua Sci. Technol. 20 (1) (2015) 90–99.
[15] Y. Zhang, G. Cui, S. Zhao, et al., IFOA4WSC: a quick and effective algorithm for QoS-aware service composition, Int. J. Web Grid Serv. 12 (1) (2016) 81–108.
[16] H. Li, S. Guo, C. Li, et al., A hybrid annual power load forecasting model based on generalized regression neural network with fruit fly optimization algorithm, Knowl. Based Syst. 37 (2013) 378–387.
[17] S. Lin, Analysis of service satisfaction in web auction logistics service using a combination of fruit fly optimization algorithm and general regression neural network, Neural Comput. Appl. 22 (3–4) (2013) 783–791.
[18] M. Mitić, N. Vuković, M. Petrović, et al., Chaotic fruit fly optimization algorithm, Knowl. Based Syst. 89 (2015) 446–458.
[19] Q. Pan, H. Sang, J. Duan, et al., An improved fruit fly optimization algorithm for continuous function optimization problems, Knowl. Based Syst. 62 (2014) 69–83.
[20] Z. Zhan, J. Zhang, Y. Li, H. Chung, Adaptive particle swarm optimization, IEEE Trans. Syst. Man Cybern. Part B 39 (2009) 1362–1381.
[21] D. Shan, G. Cao, H. Dong, LGMS-FOA: an improved fruit fly optimization algorithm for solving optimization problems, Math. Probl. Eng. (2013) 1–9.
[22] R. Rao, V. Savsani, D. Vakharia, Teachingclearning-based optimization: an optimization method for continuous non-linear large scale problems, Inf. Sci. 183 (2012) 1–15.
[23] X. Zheng, L. Wang, S. Wang, A novel fruit fly optimization algorithm for the semiconductor final testing scheduling problem, Knowl. Based Syst. 57 (2014) 95–103.
[24] J. Niu, W. Zhong, Y. Liang, et al., Fruit fly optimization algorithm based on differential evolution and its application on gasification process operation optimization, Knowl. Based Syst. 88 (2015) 253–263.
[25] X. Yuan, X. Dai, J. Zhao, et al., On a novel multi-swarm fruit fly optimization algorithm and its application, Appl. Math. Comput. 233 (2014) 260–271.