An Improved Fireworks Algorithm with Landscape Information for Balancing Exploration and Exploitation

Junfeng Chen†, Qiwen Yang‡†, Jianjun Ni‡, Yingjuan Xie†, Shi Cheng†

*College of IOT Engineering, Hohai University, Changzhou China
†Jiangsu Key Laboratory of Power Transmission and Distribution Equipment Technology, Changzhou China
‡Division of Computer Science, University of Nottingham Ningbo, China

Abstract—Fireworks algorithm is a newly risen and developing swarm intelligence algorithm, the performance of which is determined by the tradeoff between exploration and exploitation. How to develop a satisfactory weight for exploration and exploitation is an interesting and challenging work. In this paper, the landscapes of optimization problem are firstly analyzed, and then a new sparks explosion strategy is designed to represent and mine the landscape information. Moreover, the exploration and exploitation coexist in the improved fireworks algorithm, which can automatically adjust the search strategies according to landscape structure. Finally, numerical experiments are performed for the algorithms investigations, performance analysis, and comparisons. The simulation results indicate that the proposed algorithm has a significant performance on all the test functions and can achieve the global minimum for most test functions.

Index Terms—Fireworks algorithm, landscape information, exploration, exploitation

I. INTRODUCTION

Fireworks Algorithm (FWA) [1], [2] is a newly risen and developing swarm intelligence algorithm, which is inspired by the explosion and illumination of fireworks in the night sky. It is originally attributed to Tan and Zhu, and was modified for solving practical problems, such as nonnegative matrix factorization [3], digital filters [4], spam detection [5], and oil crop fertilization [6]. Similar to most swarm algorithms, FWA makes few or no assumptions about the problem being optimized and takes an advantage of swarm searching with a population (called a swarm) of candidate solutions (called sparks) to seek high quality solutions of complicated optimization problems efficiently and effectively.

Many researchers believe that the behavior of a swarm algorithm is determined by two forces: exploration and exploitation [7]. Excessive exploitation will be high-efficiency, but easy to induce premature convergence. Excessive exploration will increase accuracy, but consume too much computation time. Thus, how to get a tradeoff between the exploration and exploitation is really a key to the global performance of swarm intelligence algorithms. Many strategies for balancing exploration and exploitation were proposed in the past decade. For instance, Blum and Roli pointed out that the mechanisms of metaheuristics to efficiently explore a search space are all based on intensification and diversification, although they are inspired by different philosophies [7]. Liu et al. introduced an entropy-driven exploration and exploitation approach for evolutionary algorithms [8]. Olorunda and Engelbrecht developed a diversity rate for a particle swarm optimization to quantify the rate of change from exploration to exploitation [9]. Zhao et al. designed fitness function, crossover operator and mutation operator for the genetic algorithms to improve the exploitative capabilities and the exploitation effects [10]. Epitropakis et al. proposed a hybrid differential mutation method, which was a linear combination of the explorative and exploitative operators [11]. Lin and Gen introduced fuzzy logic control into genetic algorithm for balancing between exploration and exploitation [12]. Li et al. provided a well-designed density estimator for a dynamic neighborhood evolutionary algorithm, which achieves auto-tuning between proximity and diversity [13]. Pei et al. employed elite individuals sampling, the elite neighborhood sampling and random sampling strategies into the FWA for approximating fitness landscape and enhancing search capability [14]. Liu et al. provided a new parameter for FWA to control the exploration and exploitation dynamically [15]. Crepinsek et al. discussed what components of evolutionary algorithms contribute to exploration and exploitation, when exploration and exploitation are controlled and how balance between exploration and exploitation is achieved [16].

In view of the above-mentioned facts, FWA is not sophisticated enough and can be further improved by developing a satisfactory weight for exploration and exploitation. Moreover, the same search strategies may result in different optimization effects, when dealing with simple and smooth landscapes, or dealing with bumpy and rough landscapes. In our view, the search efficiency of FWA is not only influenced by search strategies, but also related to the fitness landscape of optimization problems to be solved. As a result, the main objective of this paper is to reveal the relationship between the landscape information of optimization problems and the interaction mechanisms of exploration and exploitation of FWA.

The remaining paper is organized as follows. In Section
II. the basic optimization process of FWA is reviewed. In Section III, an improved FWA for auto-tuning exploration and exploitation is proposed and the balance mechanism is analyzed, followed by experimental simulation and result discussion on benchmark functions in Section IV. Finally, our concluding remarks are contained in Section V.

II. CANONICAL FIREWORKS ALGORITHM

For simplicity, we focus on an unconstrained minimization problem in bounded space with finite dimensions in the form of

\[ \forall \bar{x} \in \mathcal{S} : \min f(\bar{x}) \]  

where \( \bar{x} \) designates a position or candidate solution in the search-space, \( \mathcal{S} \) is a finite dimensional search space, and \( f(\bar{x}) \) is the fitness or cost function to be minimized.

The canonical FWA works on a new search mechanism which describes the evolution of a population of sparks in the search space for problem solving. A set of sparks \( X = [\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_N] \) composed by \( N \) sparks with encoding length \( M \), can be written in the following vector and matrix forms:

\[
X = \begin{bmatrix}
\bar{x}_1 \\
\bar{x}_2 \\
\vdots \\
\bar{x}_N
\end{bmatrix} = \begin{bmatrix}
x_{11} & x_{12} & \cdots & x_{1M} \\
x_{21} & x_{22} & \cdots & x_{2M} \\
\vdots & \vdots & \ddots & \vdots \\
x_{N1} & x_{N2} & \cdots & x_{NM}
\end{bmatrix} = \begin{bmatrix}
\bar{g}_1 \\
\bar{g}_2 \\
\vdots \\
\bar{g}_M
\end{bmatrix}^T
\]  

where \( \bar{x}_i = [x_{i1}, x_{i2}, \ldots, x_{iM}] \in \mathcal{S} \) is the \( i \)-th sparks in \( X \), and \( x_{ij} \) is the \( j \)-th solution component of the \( i \)-th sparks, \( i = 1, 2, \ldots, N, \ j = 1, 2, \ldots, M \). For the sake of convenience, the \( j \)-th vector which consists of the \( j \)-th components of all the sparks in \( X \) is called the \( j \)-th set component, written as \( \bar{g}_j = [x_{1j}, x_{2j}, \ldots, x_{Nj}]^T \).

Generally speaking, FWA has three main operators: calculating, creating and mapping. More specifically, calculating operator consists of computing explosion number and amplitude, creating operator refers to generating explosion sparks and Gaussian sparks, and mapping operator means bounding sparks back to search space. A principle FWA is described as Algorithm 1.

The number of sparks generated by a fireworks \( \bar{x}_i \) is defined as follow.

\[
H_i = K \cdot \frac{f(\bar{x}) - f(\bar{x}_i)}{\sum_{j=1}^{N}(f(\bar{x}) - f(\bar{x}_j))}
\]  

where \( K \) is a parameter controlling the number of explosion sparks, \( \bar{x}_i \) is the \( i \)-th sparks, \( \bar{x} \) is the worst fireworks with the maximum fitness value, \( i.e., f(\bar{x}) = \max(f(\bar{x}_i)) \), \( i = 1, 2, \ldots, N \).

To avoid overwhelming effects of splendid fireworks, bounds are defined for \( H_i \).

\[
H_i = \begin{cases} 
H_{\min} & \text{if } H_i < H_{\min} \\
H_{\max} & \text{if } H_i > H_{\max} \\
H_i & \text{otherwise}
\end{cases}
\]  

where \( H_{\min} \) and \( H_{\max} \) are the lower and upper bounds of sparks numbers.

Amplitude of explosion for each firework is defined as follows

\[
A_i = R \cdot \frac{f(\bar{x}_i) - f(\bar{x})}{\sum_{i=1}^{N}(f(\bar{x}_i) - f(\bar{x}))}
\]  

where \( R \) denotes the maximum explosion amplitude and \( \bar{x} \) is the best fireworks with the minimum fitness value, \( i.e., f(\bar{x}) = \min(f(\bar{x}_i)), \ i = 1, 2, \ldots, N \).

The basic FWA contains a rough thinking of the tradeoff between exploration and exploitation. The exploitation implies the fireworks with better fitness generate a larger population of explosion sparks within a smaller range. While exploration means the fireworks with lower fitness can only generate a smaller population with higher explosion amplitude.

III. IMPROVED FWA WITH LANDSCAPE INFORMATION

A. Landscape Information of Optimization Problems

No free lunch theorem [17] indicates that no search strategy can keep optimal when optimizing arbitrary problems. In other words, a general-purpose and universal optimization algorithm is impossible. The only way that one strategy can outperform another is if it is specialized to the structure of the specific problem under consideration. However, we know nothing about the landscape structure when dealing with a black-box problem. As a result, How to get and sustain the domain knowledge of specific optimization problems is a fundamental issue in the swarm intelligence field.

In this paper, we try to design a new sparks explosions strategy which is characterized of representing and mining the landscape information of optimization problem. Specifically, the landscape of optimization problem is approximately described as the distribution structure of candidate solutions during the searching process of FWA.

Here, we take a simple mathematical function \( f(x, y) = \sin(x) \cdot \sin(y) \) to illustrate the basic idea of the proposed method. When \( x, y \in [0, 2\pi] \), the landscape structure is shown as Fig. 1.

From Fig. 1 (a), the three-dimensional surface image has two maxima and two minima, and its corresponding contours describe the curves with the same function values. The color is deeper and cooler and the value is smaller. Fig. 1 (b) is the contour chart with the accumulated candidate solutions of FWA iterating. The black dots represent the candidate solutions, plotted by the variable \( x \) on the horizontal axis and the variable \( y \) on the vertical axis. In general, a swarm intelligence algorithm only makes full use of evaluation function rather than other additional information. In other words, swarm algorithms only consider about whether a candidate solution is superior to another or not, regardless of distribution differences of the candidate solutions. However, there is much useful information in the function landscape. For example, Fig. 1 (c) represents the distribution structure of candidate solutions with small evaluation values. It can be seen that these candidate solutions outline the local landscapes of minima, which are located on the lower-right and the upper-left parts.

In practice, there are lots of difficult optimization problems, which landscapes are relatively complicated and diverse. Some
Algorithm 1: Pseudocode of Fireworks Algorithm

1. Initialize $N$ sparks randomly and evaluate their quality (i.e., fitness);
2. while have not found “good enough” solution or not reached the pre-determined maximum number of iterations do
   for each firework $\vec{x}_i$ do
   3. Calculate the number of sparks $H_i$;
   4. Calculate the amplitude of explosion $A_i$;
   5. for $j=1$ to $H_i$ do
      6. Select $P$ components from $\vec{x}_i$, $P = \text{round}(M \cdot \text{rand}(0,1))$;
      7. for each selected components $k=1$ to $P$ do
         8. Update the displacement of components;
           $\vec{s}_j \leftarrow \vec{x}_i$ with updated $x_{ik}$;
         9. if $s_{jk} < LB$ or $s_{jk} > UB$ then
            10. Mapping $s_{jk}$ to the search space $s_{jk} = LB + \text{rand}(0,1) \cdot (UB - LB)$;
            11. for $h=1$ to $G_i$ do
               12. Randomly select $Q$ components from $\vec{x}_i$;
               13. for each selected components $l=1$ to $Q$ do
                  14. Update the components $x_{il} = \text{rand}(0,1) \cdot (x^*_{il} - x_{il})$;
                  15. $\vec{g}_h \leftarrow \vec{x}_i$ with updated $x_{il}$;
                  16. if $g_{hl} < LB$ or $g_{hl} > UB$ then
                     17. Mapping $g_{hl}$ to the search space $g_{hl} = LB + \text{rand}(0,1) \cdot (UB - LB)$;
   18. Evaluate the fitness values of the sparks $\vec{s}_j$ and $\vec{g}_h$, $j \in [1, H_i], h \in [1, G_i]$;
   19. Keep the best sparks to next generation;
   20. Select the other $N-1$ sparks from the generated sparks based on the selection probability;

are multi-dimensional functions with a great number of peaks and valleys or with strong variables linkage. However, a multi-dimensional rugged landscape may be regarded as a combination of several one-dimensional models. Fig. 2 lists the common formats of one-dimensional curves. These landscapes may be a local part of the one-dimensional landscape of sophisticated problems.

![Fig. 2. The typical one-dimensional landscapes.](image)

It can be seen that Fig. 2 (a) is a simple valley, which is often regarded as an easy landscape for almost all the swarm algorithms. Adding some noise to the function in Fig. 2 (a) will form the rugged landscape in Fig. 2 (b). A multi-funnel landscape is shown in Fig. 2 (c). If the global optimal solution is far away from the local optima to some extent, some swarm algorithms may be trapped in those local optima. According to the curve in Fig. 2 (d), the introducing fluctuations into the landscape contribute to increasing the optimization hardness. Fig. 2 (e) and Fig. 2 (f) is an opposite pulse function without or with noise signal. In contrast to the curve in Fig. 2 (e), the local disturbance helps lessen the optimization hardness, because there is some useful information in the landscape in Fig. 2 (f).

B. Creating sparks with Landscape Information

In this paper, an improved fireworks algorithm is developed with exploration and exploitation coexisting. The search strategies can be switched according to landscape structure adaptively. The useful information in the landscape is represented by the dispersing extent and dispersion degree of the candidate solutions. Algorithm 2 describes the proposed operator of creating sparks with landscapes information.

As shown in Algorithm 2, the landscape structure is represented by the distribution information of the selected sparks, denoted as a set of vectors $\{\vec{u}, \vec{v}, \vec{w}\}$, where $\vec{u}$ is the kernel, $\vec{v}$ is the coverage and $\vec{w}$ is the dispersion of the selected sparks. And then this landscape information is used to create
Algorithm 2: Pseudocode of the creating sparks operator

1 Select $L$ sparks from the generated sparks and the fitness values of selected sparks meet the predicate conditions;
2 Determine the best spark as the kernel in the selected sparks $\mathbf{u} = [u_1, u_2, \ldots, u_M] = \mathbf{x}$;
3 Calculate the coverage of the selected sparks $\mathbf{v} = [v_1, v_2, \ldots, v_M] = \frac{1}{L-1} \sum_{i=1}^{L} (\mathbf{x}_i - \mathbf{u})^2$;
4 Mapping the fitness values $f(\mathbf{x})$ to the confidence interval $(0, 1]$;
5 for each couple of sparks and their fitness values $(\mathbf{x}_i, f(\mathbf{x}_i))$ do
6 \hspace{1em} Calculate $\mathbf{\hat{o}}_i = \sqrt{\frac{1}{2 \ln f(\mathbf{x}_i)}}$
7 Calculate the mean of $\mathbf{\hat{o}}_i$ as $\mathbf{\bar{o}}$, i.e., $\mathbf{\bar{o}} = \frac{1}{L} \sum_{i=1}^{L} \mathbf{\hat{o}}_i$;
8 Calculate the dispersion of the selected sparks $\mathbf{\bar{w}} = [w_1, w_2, \ldots, w_M] = \frac{1}{L-1} \sum_{i=1}^{L} (\mathbf{\hat{o}}_i - \mathbf{\bar{o}})$;
9 Create the $N$ new sparks based on the landscape information $\{\mathbf{u}, \mathbf{v}, \mathbf{w}\}$;
10 for $j = 1$ to $N$ do
11 \hspace{1em} Generate a normally distributed random vector $\mathbf{p}$ with the kernel $\mathbf{v}$ and the coverage $\mathbf{w}$, i.e., $\mathbf{p} = \text{NormRand}(\mathbf{v}, \mathbf{w})$;
12 \hspace{1em} Generate a normally distributed random spark $\mathbf{x_j}$ with the kernel $\mathbf{u}$ and the coverage $\mathbf{p}$, i.e., $\mathbf{x_j} = \text{NormRand}(\mathbf{u}, \mathbf{p})$;
new sparks for the next generation. As a matter of fact, we can control exploration and exploitation by selecting different sparks in the creating sparks operator. Fig. 3 and Fig. 4 show two different selecting and creating processes for function \( f(x, y) = \sin(0.5x) \cdot \cos(2y), x \in [0, 2\pi] \) and \( y \in [0, \pi] \).

As shown in Fig. 3 (a), the function values of selected sparks are lesser than average value and the corresponding landscape vectors are calculated as a set of vectors \( \{ \vec{u}_1, \vec{v}_1, \vec{w}_1 \} \), where the kernel \( \vec{u}_i = [3.4207, 1.5953] \), the coverage \( \vec{v}_1 = [1.8441, 0.7417] \), and the dispersion \( \vec{w}_1 = [0.5677, 0.2275] \). It can be seen that the elite of the selected sparks is situated on \([3.4207, 1.5953]\). The coverage of the selected sparks on the horizontal axis is greater than the coverage on the vertical axis. Meanwhile, the sparks in x axis are scattered, while in y axis are concentrated. Fig. 3 (b) shows the new generated sparks on account of these landscape information.

As seen in Fig. 4 (a), we select the superior 15 sparks and calculate landscape vectors, denoted as \( \{ \vec{u}_2, \vec{v}_2, \vec{w}_2 \} \). The kernel \( \vec{u}_i \) is equivalent to \( \vec{u}_1 \), i.e., \( \vec{u}_1 = \vec{u}_2 = [3.4207, 1.5953] \). The coverage \( \vec{v}_2 = [0.9301, 0.1903] \), which is less than \( \vec{v}_1 \). The dispersion \( \vec{w}_2 = [0.3433, 0.0718] \), which is less than \( \vec{w}_1 \). The new creating sparks based on landscape information are shown in Fig. 4 (b). Moreover, we can see that the new generated sparks have distinctive features, by comparing Fig. 3 (b) with Fig. 4 (b). Therefore, the proposed creating operator mingles exploration with exploitation to identify the regions with high quality solutions in the search space quickly by mean of landscapes information.

C. Improved FWA for Auto-tuning Exploration and Exploitation

As we know, optimization problems come in a variety of shapes. It is a good choice to select the explosion operator to generate new sparks for simple unimodal landscapes. And the creating operator with landscapes information contributes to discover the promising region of rugged landscapes in the search space. As for astonishingly complex problem with little useful information in its landscape, random search without preference for any region is an excellent strategy. In this section, the improved FWA integrates the advantages of three creating operators, which not only improves the performance greatly, but also reduce the calculation cost. The improved FWA for auto-tuning exploration and exploitation can be described as Algorithm 3.

As shown in Algorithm 3, the proposed algorithm doesn’t simply run the three creating sparks methods in order or in parallel, but select the proper creating strategy for each component of sparks by using conditional rules. The reason is that the landscapes in each dimension are diverse and variational, especially for multi-dimension and complicated optimum problem. The sole creating algorithm will depress diversity in some components of sparks and then induce premature convergence.

IV. NUMERICAL EXPERIMENTS AND PERFORMANCE ANALYSIS

For better understanding the search behavior of the proposed FWA, a set of numerical experiments are performed for the algorithm investigations, performance analysis, and comparisons. First, we define several benchmark functions which have been widely used in the swarm intelligence community. Then, the experimental results of three versions of FWAs are provided and the empirical results are discussed in this section.

A. Test Functions

Eight well-known benchmark functions are selected as a test suite. Table I illustrates the names, mathematical expressions, dimensions, search space and optimal solution. Based on their properties, these functions can be divided into two groups. Functions \( f_1 - f_4 \) are high dimensional unimodal functions, and functions \( f_5 - f_8 \) are high dimensional multimodal functions with many local minima. Some of them possess the highly non-linear characteristics. In addition, shift values are added to functions \( f_5 \) and \( f_6 \) to form new functions \( f_7 \) and \( f_8 \), which means the global optimum can be converted to somewhere else. To be specific, the optimal positions of the two functions are shifted and their minimal fitness values remain the same. In addition, we transform all the optimal fitness values to zero for ease of comparison of experiment results.

B. Simulation Results and Performance Analysis

Three versions of FWAs are selected to perform the simulation experiments. They are conventional fireworks algorithm (CFWA) [1], enhanced fireworks algorithm (EFWA), and improved fireworks algorithm (IFWA) proposed in this paper, respectively. In all the following experiments, the population size is set to 50, namely \( N = 50 \). As FWAs are nondeterministic algorithms, we set maximum iteration 2000 and a small positive value \( 1.0 \times 10^{-6} \) for each test to keep the algorithms from falling into an infinite loop. The parameters of both the CFWA and the EFWA are set in accord with Ref. [2], which is applied in all the comparison experiments. For IFWA, \( R = 100, K = 200, Q = 5, H_{\text{min}} = 5 \) and \( H_{\text{max}} = 100 \).

All the FWAs are implemented in Matlab 8.0 and the simulations are run on the same PC with Intel Core i3 2350M 2.3GHz, 2 GB memory capacity and the Windows 7 operating system. Table II lists the comparison results of benchmark functions obtained by CFWA, EFWA and IFWA. Mean and Std indicate the average evaluation values and standard deviation for each benchmark function, respectively. Fig. 5 shows the stereograms of Sumcan and Sumcan with logarithmic transformation.

From Table II, it can be seen that IFWA outperforms CFWA and EFWA for all the test functions. And it shows significant performance on the shifted functions, which is increasingly difficult to be optimized. Meanwhile, IFWA achieves the global minima of most test functions, except the Sumcan function. This is because there is much useful landscape information in these functions, from which IFWA can benefit.
Algorithm 3: Improved FWA for Auto-tuning Exploration and Exploitation

1. Initialize $N$ sparks randomly and evaluate their quality (i.e., fitness);
2. while a termination criterion is unmet do
   3. Calculate the coverage of all the sparks;
   4. for each set component $\tilde{g}_j$, $j = 1, 2, \ldots, M$ do
      5. if coverage $> \text{half of search space of the component}$ then
         6. Generate a certain number of new components of sparks with a random position;
      7. else if one-fiftieth of search space $< \text{coverage} < \text{half of search space}$ then
         8. Call Algorithm 2 to create the new components;
      9. else if coverage $< \text{one-fiftieth of search space}$ then
         10. Call Algorithm 1 to generate explosion components of sparks;

(a) Calculating landscape vectors $\{\vec{u}_1, \vec{v}_1, \vec{w}_1\}$ based on selected sparks
(b) Creating sparks based on landscape information $\{\tilde{u}_1, \tilde{v}_1, \tilde{w}_1\}$

Fig. 3. Creating processes with landscape information $\{\tilde{u}_1, \tilde{v}_1, \tilde{w}_1\}$

(a) Calculating landscape vectors $\{\vec{u}_2, \vec{v}_2, \vec{w}_2\}$ based on selected sparks
(b) Creating sparks based on landscape information $\{\tilde{u}_2, \tilde{v}_2, \tilde{w}_2\}$

Fig. 4. Creating processes with landscape information $\{\tilde{u}_2, \tilde{v}_2, \tilde{w}_2\}$
greatly. While the Sumcan function is a difficult optimization problem like fishing for a needle in a haystack. Fig. 5 (a) is the visualization of two dimensional Sumcan function, which is hard to locate the global minimum for FWAs. But it gets much easier when we invert the fitness function by use of logarithmic transformation. Fig. 5 (b) shows visualization of two dimensional Sumcan function with logarithmic transformation. We can see that the landscape has experienced the procedure from gradually development to mature.

It is worth pointing out that the FWAs are meta-heuristic algorithms, which performance is not only related to search strategies, but also to initial distribution of solutions, parameter values and maximum iteration involved in algorithms. For example, CFWA can get better optimization results for the Rosenbrock function, if we extend the maximum iteration.

V. CONCLUSIONS

Fireworks algorithm which is inspired by the explosion of fireworks, aims to seek high quality solutions of complicated optimization problems efficiently and effectively. However, how to balance exploration and exploitation is really a key to the global performance of FWAs. In this paper, we hold that the search efficiency of FWA is not only influenced by search strategies, but also related to the fitness landscape of optimization problems to be solved. The landscape information of optimization problem is firstly analyzed and described as a set of vectors: the kernel, the coverage and the dispersion of the selected sparks. And then an improved FWA is proposed to tune exploration and exploitation adaptively. Finally, the

<table>
<thead>
<tr>
<th>Function Name</th>
<th>Mathematical Expressions</th>
<th>Dim. (D)</th>
<th>Search Space (S)</th>
<th>Optimal fitness (f_{min})</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sphere</td>
<td>( f_1(\vec{x}) = \sum_{i=1}^{D} x_i^2 )</td>
<td>30</td>
<td>(-100, 100)^D</td>
<td>0</td>
</tr>
<tr>
<td>Rosenbrock</td>
<td>( f_2(\vec{x}) = \sum_{i=1}^{D-1} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2] )</td>
<td>30</td>
<td>(-30, 30)^D</td>
<td>0</td>
</tr>
<tr>
<td>Sumcan</td>
<td>( f_3(\vec{x}) = -10^5 + \sum_{i=1}^{D}</td>
<td>\vec{y}_i</td>
<td>) + 1.0 \times 10^5</td>
<td>30</td>
</tr>
<tr>
<td>Sumcan with log</td>
<td>( f_4(\vec{x}) = - \log \left(10^5 + \sum_{i=1}^{D}</td>
<td>\vec{y}_i</td>
<td>\right) + 5 )</td>
<td>30</td>
</tr>
<tr>
<td>Ackley</td>
<td>( f_5(\vec{x}) = -20 \exp \left(-0.2 \sqrt{\frac{1}{D} \left( \sum_{i=1}^{D} x_i^2 \right)} \right) )</td>
<td>30</td>
<td>(-32, 32)^D</td>
<td>0</td>
</tr>
<tr>
<td>Rastrigin</td>
<td>( f_6(\vec{x}) = \sum_{i=1}^{D} [x_i^2 - 10 \cos (2\pi x_i) + 10] )</td>
<td>30</td>
<td>(-5.12, 5.12)^D</td>
<td>0</td>
</tr>
<tr>
<td>Shifted Ackley</td>
<td>( f_7(\vec{x}) = f_5(\vec{x} - \vec{\theta}<em>{new} + \vec{\theta}</em>{old}) )</td>
<td>30</td>
<td>(-32, 32)^D</td>
<td>0</td>
</tr>
<tr>
<td>Shifted Rastrigin</td>
<td>( f_8(\vec{x}) = f_6(\vec{x} - \vec{\theta}<em>{new} + \vec{\theta}</em>{old}) )</td>
<td>30</td>
<td>(-5.12, 5.12)^D</td>
<td>0</td>
</tr>
</tbody>
</table>
The proposed algorithm is applied to numerical optimization problems in comparison with the other two FWAs. The statistical results show the proposed method can identify the regions with high quality solutions in the search space quickly by using the landscape information. Besides that, we explain the reasons why IFWA is hard to locate the global minimum of the Sumcan function.

ACKNOWLEDGMENT

The research work reported in this paper is supported by the National Natural Science Foundation of China under Grant Number 61403121, 61273367, the Fundamental Research Funds for the Central Universities under Grant Number 2013B18614 and 2013B09014, and the Ningbo Science & Technology Bureau with the Science and Technology Project Number 2012B10055.

REFERENCES


<table>
<thead>
<tr>
<th></th>
<th>CFWA</th>
<th>EFWA</th>
<th>IFWA</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_1$</td>
<td>Mean</td>
<td>0.000e+0</td>
<td>9.704e-4</td>
</tr>
<tr>
<td></td>
<td>(Std)</td>
<td>(0.0e+0)</td>
<td>(3.0e-4)</td>
</tr>
<tr>
<td>$f_2$</td>
<td>Mean</td>
<td>1.807e+1</td>
<td>7.933e+1</td>
</tr>
<tr>
<td></td>
<td>(Std)</td>
<td>(1.1e+1)</td>
<td>(8.7e+1)</td>
</tr>
<tr>
<td>$f_3$</td>
<td>Mean</td>
<td>9.886e+4</td>
<td>9.567e+4</td>
</tr>
<tr>
<td></td>
<td>(Std)</td>
<td>(4.334e+4)</td>
<td>(5.991e+4)</td>
</tr>
<tr>
<td>$f_4$</td>
<td>Mean</td>
<td>0.000e+0</td>
<td>0.000e+0</td>
</tr>
<tr>
<td></td>
<td>(Std)</td>
<td>(0.0e+0)</td>
<td>(0.0e+0)</td>
</tr>
<tr>
<td>$f_5$</td>
<td>Mean</td>
<td>0.000e+0</td>
<td>6.430e-2</td>
</tr>
<tr>
<td></td>
<td>(Std)</td>
<td>(0.0e+0)</td>
<td>(3.0e-1)</td>
</tr>
<tr>
<td>$f_6$</td>
<td>Mean</td>
<td>0.000e+0</td>
<td>1.597e+1</td>
</tr>
<tr>
<td></td>
<td>(Std)</td>
<td>(0.0e+0)</td>
<td>(3.6e+0)</td>
</tr>
<tr>
<td>$f_7$</td>
<td>Mean</td>
<td>7.262e-1</td>
<td>8.745e+0</td>
</tr>
<tr>
<td></td>
<td>(Std)</td>
<td>(4.1e-1)</td>
<td>(9.6e+0)</td>
</tr>
<tr>
<td>$f_8$</td>
<td>Mean</td>
<td>9.403e+4</td>
<td>9.567e+4</td>
</tr>
<tr>
<td></td>
<td>(Std)</td>
<td>(2.3e+0)</td>
<td>(3.8e+0)</td>
</tr>
</tbody>
</table>

TABLE II
COMPARISON RESULTS OF CFWA, EFWA AND IFWA.