# Enhanced Fireworks Algorithm with an Improved Gaussian Sparks Operator

Jinglei Guo and Wei Liu(✉)

School of Computer Science, Central China Normal University,
No. 152 Luoyu Road, Wuhan 430074, Hubei, China
{guojinglei,liuwei}@mail.ccnu.edu.cn

**Abstract.** As a population-based intelligence algorithm, fireworks algorithm simulates the firework's explosion process to solve optimization problem. A comprehensive study on Gaussian spark operator in enhanced fireworks algorithm (EFWA) reveals that the search trajectory is limited by the difference vector and the diversity of swarm is not effectively increased by new sparks adding. An improved version of EFWA (IEFWA) is proposed to overcome these limitations. In IEFWA, a new Gaussian spark operator utilizes the location information of the best firework and randomly selected firework to calculate the center position and explosion amplitude, which enhance the search for potential region. Experiments on 20 well-known benchmark functions are conducted to illustrate the performance of IEFWA. The results turn out IEFWA outperforms EFWA and dynFWA on most testing functions.

**Keywords:** Fireworks algorithm · Gaussian distribution · Explosion

## 1 Introduction

In the last two decades, many swarm intelligence (SI) algorithms, which simulate the behavior of simple nature agents to produce the intelligent ability, have been proposed to solve engineering optimization problems. SI algorithms include such as ant colony algorithm (ACO) [1], particle swarm optimization (PSO) [2], firefly algorithm (FA) [3], cuckoo search [4], wolf colony algorithm (WCA) [5], whale optimization algorithm (WOA) [6], fireworks algorithm (FWA) [7] and so on. FWA is inspired by the explosion process of real fireworks in night sky and firstly proposed by Tan in 2010 [7]. The research work on FWA can be classified into two categories: (1) algorithm improvements: single objective FWA [8–11], multi-objective FWA [12], parallel FWA [13, 14], hybrid FWA with DE [15, 16]; (2) algorithm applications: FWA has been used in digital filter design [17], parameters optimization [18], harmonic elimination [19], multi-satellite control resource scheduling [20], and so on.

The remainder of this paper is organized as follows. Section 2 presents a detailed introduction of FWA. The review of well-known improvement FWA versions is presented in Sect. 3. In Sect. 4, the behavior of Gaussian sparks in EFWA is analyzed and a new Gaussian sparks operator is proposed to enlarge the search region. Experimental results are presented to validate the performance of IEFWA in Sect. 5. The conclusion is drawn in Sect. 6.


H. Peng et al. (Eds.): ISICA 2018, CCIS 986, pp. 38–49, 2019.
https://doi.org/10.1007/978-981-13-6473-0_4

## 2 The Classical FWA

With generality, FWA [7] is for solving global minimal problem,

$$\min \mathrm{f}(\vec{x}),\ \vec{x} = (x_1, \cdots, x_D) \in \Omega = \prod_{i=1}^{D} [a_i, b_i], \tag{1}$$

where $f$: $R^D \rightarrow R$ is a continuous problem, the dimension of $\vec{x}$ is $D$, $a_i$ and $b_i$ are the lower bound and upper bound of the $i$th component in feasible region $\Omega$, and $-\infty < a_i < b_i < +\infty$.

FWA is a population-based heuristic algorithm which consists of three main operators: the explosion operator, the Gaussian sparks operator and the selection operator.

### 2.1 Explosion Operator

The explosion operator generates numerous sparks which explode with the center (firework $x_i$). For each firework $x_i$, its explosion sparks' number $s_i$ is defined as follows,

$$s_i = M \frac{y_{max} - f(x_i) + \xi}{\sum_{i=1}^{n} (y_{max} - f(x_i)) + \xi} \tag{2}$$

where $n$ is the number of fireworks, $M$ is the total number of sparks generated by $n$ fireworks, $y_{max} = \max(f(x_i))\ (i = 1, 2, \ldots, n)$ is the worst value of the objective function of the fireworks, and $\xi$ is smallest constant to ensure the divisor is nonzero.

To avoid the overwhelming effects of the best fireworks, $s_i$ is limited in a range, as follows:

$$s_i = \begin{cases} S_{min} & if\ s_i < S_{min} \\ S_{max} & if\ s_i < S_{max} \\ S_i & otherwise \end{cases}, \tag{3}$$

where $S_{min}$ and $S_{max}$ are the lower bound and upper bound for the spark number.

The firework's explosion amplitude is calculated as follows:

$$A_i = \hat{A} \frac{f(x_i) - y_{min} + \xi}{\sum_{i=1}^{n} (f(x_i) - y_{min}) + \xi}, \tag{4}$$

where $\widehat{\mathrm{A}}$ is the maximum explosion amplitude, $y_{min} = \min(f(x_i))\ (i = 1, 2, \ldots, n)$ is the best value of the objective function of the fireworks.

From the formula (2) and (4) above, it can be found out the better the firework's fitness is, the more sparks it generates and the smaller amplitude it produces. Thus, the firework's fitness determines the search behavior, that is, the firework with better fitness conducts exploitation and the firework with worse fitness conducts exploration.

Then using the formula (2) and (4), the explosion sparks are generated by Algorithm 1.

**Algorithm 1**. Generate an explosion spark of firework $x_i$

Initialize the location of the spark $\widetilde{p}_j = x_i$;
Randomly select $z$ dimensions of $\widetilde{p}_j$;
Calculate the amplitude for $\widetilde{p}_j$ by formula (2), $h$=$A_i$*rand(-1,1);
**for** each dimension $\widetilde{p_j^k} \in$ {pre-selected $z$ dimensions of $\widetilde{p}_j$} do
  $\widetilde{p_j^k} = \widetilde{p_j^k} + h$
  **if** $\widetilde{p_j^k} < b_k$ or $\widetilde{p_j^k} > a_k$ then
    $\widetilde{p_j^k} = a_k + \left|\widetilde{p_j^k}\right| mod(b_k - a_k)$
  **end if**
**end for**

## 2.2 Gaussian Sparks Operator

To keep the population diversity, Gaussian sparks operator is used to generate sparks in Gaussian distribution with the center which is the selected firwork's location. The Gaussian sparks operator is computed by

$$\widetilde{p_j^k} = \widetilde{p_j^k} \cdot (1 + \mathrm{N}(0,1)), \tag{5}$$

where N(0,1) is the Gaussian distribution function with the mean value 0 and the standard deviation 1.

Then using the formula (5), the Gaussian sparks are generated by Algorithm 2.

**Algorithm 2**. Generate a Gaussian spark of firework $x_i$

Initialize the location of the spark $\widetilde{p}_j = x_i$;
Randomly select $z$ dimensions of $\widetilde{p}_j$;
**for** each dimension $\widetilde{p_j^k} \in$ {pre-selected $z$ dimensions of $\widetilde{p}_j$} do
  Calculate the amplitude for $\widetilde{p}_j$ by formula (5), $\widetilde{p_j^k} = \widetilde{p_j^k} \cdot (1 + \mathrm{N}(0,1))$;
  **if** $\widetilde{p_j^k} < b_k$ or $\widetilde{p_j^k} > a_k$ then
    $\widetilde{p_j^k} = a_k + \left|\widetilde{p_j^k}\right| mod(b_k - a_k)$
  **end if**
**end for**

## 2.3 Selection of Locations

Like other EA algorithms, certain number sparks and fireworks should be survived for the next generation in FWA. Obviously, the best location ($x^*$) with the minimal objective function $f(x^*)$ is an optimal location in current generation and should be kept

for the next generation. After that, for maintaining the diversity of the generation, the other locations are selected based on their distance to other locations. The total distance $R(x_i)$ between the spark $x_i$ and other locations $x_j$ is defined as follows:

$$R(x_i) = \sum_{j \in K} d(x_i, x_j), \tag{6}$$

where $K$ is the number of fireworks and sparks.

The selection probability $P(x_i)$ of location $x_i$ is calculated by formula (7).

$$P(x_i) = R(x_i) / \sum_{j \in K} R(x_j) \tag{7}$$

From (7), the selection probability is proportional to the distance. The larger total distance $R(x_i)$ is, the higher selection probability location $x_i$ has. Thus, the locations which are far from other sparks and fireworks are prone to be selected. The selection scheme ensures the population diversity to some extent.

### 2.4 Framework of Basic FWA

The framework of basic FWA is described as follows.

**Algorithm 3**. Framework of the classical FWA

Randomly initialize $n$ locations of fireworks
**while** stop condition is not satisfied **do**
  generate explosion sparks for fireworks by algorithm 1
  generate Gaussian sparks for fireworks by algorithm 2
  select the best location in the fireworks and sparks
  randomly select other $n$-1 locations according to the probability (7)
**end while**

## 3 Related Work

In enhanced Fireworks algorithm (EFWA) [8], for the problem that the explosion amplitude of the best fireworks is close to 0, the authors proposed a linear and non-linear decreasing method with the evolution process to set the lower bound $A_{min}$ of explosion amplitude. Because the solutions out of feasible search space are mapped close to the origin by the mapping operator, FWA has worse results with increasing distance between function optimum and the origin of the search space. For this drawback, the mapping operator in EFWA is replaced by uniform random mapping operator in range $[a_i, b_i]$. In FWA, calculation of the distance between locations in selection strategy costs a high computational time. The Elitism-Random selection method whose computational complexity is linear with respect of the number of fireworks is applied in EFWA, therefore the runtime of EFWA is reduced significantly.

In [9], a new mutation operator with the covariance matrix (CM) is proposed. In FWACM, $\mu$ better sparks ($v_i$) are selected from the sparks in each generation and calculated the mean value $\mathrm{m} = \frac{1}{\mu}\sum_{i=1}^{\mu} v_i$. Then the element $cov(v_i,v_j)$ in covariance matrix C is calculated. Finally, Gaussian sparks are produced according with Gaussian distribution $N(m,C)$ by mean value $m$ and covariance matrix $C$. The covariance mutation operator produces Gaussian sparks nearly at the direction of the gradient of the function, which makes the new algorithm useful at finding the local optimum. From the experimental results on CEC 2015 competition problems, FWACM outperforms AFWA [10] on unimodal functions and hybrid functions.

The dynamic search fireworks algorithm (dynFWA) [11] is a state-of-the-art version of the fireworks algorithm. It outperforms FWA and EFWA on the 28 benchmark functions in CEC2013. There are two main improvements in dynFWA:

(1) Dynamic explosion amplitude for the elitist. In FWA and EFWA, the explosion amplitude solely depends on the fitness. But, in dynFWA, the explosion amplitude of the core firework (CF) which is the firework with the best fitness is dynamic in each generation. In the initial stage, CF is the best one among all randomly initialized fireworks. After that, if in generation $g$, the sparks of CF find a better location than the best in generation $g$-1, the amplitude will be enlarged by an amplification coefficient $C_a > 1$, otherwise it will be reduced by a coefficient $C_r < 1$. Thus, the explosion amplitude is dynamically adjusted according the search performance in the last generation,

$$A_{CF,g} = \begin{cases} A_{CF,g-1} * C_a & f\left(x_{CF,g}\right) < f\left(x_{CF,g-1}\right) \\ A_{CF,g-1} * C_r & f\left(x_{CF,g}\right) = f\left(x_{CF,g-1}\right) \end{cases}, \tag{8}$$

where $A_{CF,g}$ is the explosion amplitude of the CF in generation $g$.

(2) Elimination of the Gaussian sparks operator. Based on the analysis of the Gaussian sparks location, the Gaussian mutation operator is removed by dynFWA.

## 4 The IEFWA

### 4.1 Analysis of Gaussian Sparks Operator in EFWA

In EFWA, the Gaussian sparks are generated by the formula:

$$\widetilde{p_j^k} = \widetilde{p_j^k} + \left(x_{best}^k - \widetilde{p_j^k}\right) * e, \tag{9}$$

where $x_{best}$ is the best firework that has been found so far and $e$ is the Gaussian distribution $N(0,1)$.

From the formula (9), the new Gaussian sparks position will be located along the direction of difference vector $x_{best} - \widetilde{p_j}$. For 2D problem, it can be represented as Fig. 1. In Fig. 1, the black dot is the position of $x_{best}$, the white dot is the position of $\widetilde{p_j}$ and the grey dots are the possible locations that may be generated by Gaussian sparks.

According to the Gaussian sparks operator, the position of the sparks may be located at three situations: (1) close to the best firework $x_{best}$; (2) close to the selected firework $\widetilde{p}_j$; (3) some distance to both $x_{best}$ and $\widetilde{p}_j$.

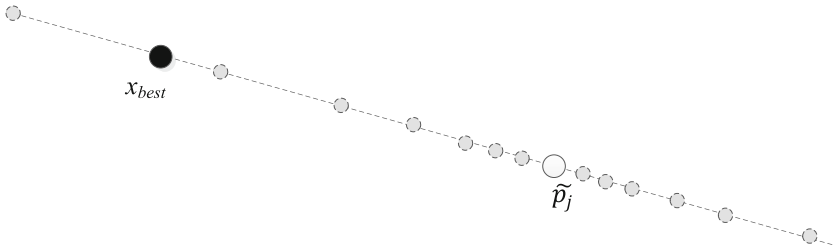



**Fig. 1.** Gaussian sparks' locations in EFWA.

Figure 2 is the bell-shaped curve of Gaussian distribution. From the mathematical statistic, the points have 95.45% possibility located in $[-2, 2]$. That means the first two situations have a high probability to occur. However, for the first two situations, it is similar to the effect of the explosion process made by the best firework $x_{best}$ and the selected firework $\widetilde{p}_j$.

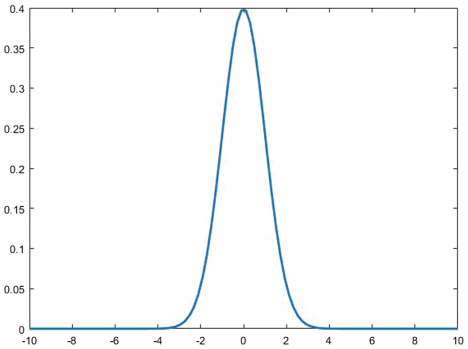



**Fig. 2.** Gaussian distribution N(0,1).

### 4.2 An Improved Gaussian Sparks Operator in EFWA

From the above discussion in Sect. 4.1, Gaussian sparks operator has been removed by dynFWA, because it has little effect in EFWA evolution process. In this section, a new Gaussian Sparks operator is proposed.

Similar to the Gaussian operator in EFWA, the best firework $x_{best}$ found so far and the selected firework $\widetilde{p}_j$ are still selected in the improved Gaussian sparks operator. But in the new Gaussian sparks, the center of $x_{best}$ and $\widetilde{p}_j$ is assigned a new vector, the

Euclidean distance $\left\|x_{best} - \widetilde{p}_j\right\|$ is the amplitude for the Gaussian distribution. The new Gaussian sparks formula is

$$\widetilde{p}_j^k = \frac{x_{best}^k + \widetilde{p}_j^k}{2} \pm \left\|x_{best} - \widetilde{p}_j\right\| * (1 + N(0,1)), \tag{10}$$

where the symbol "±" means the minus and plus operator can be selected randomly, and each has 50% possibility.

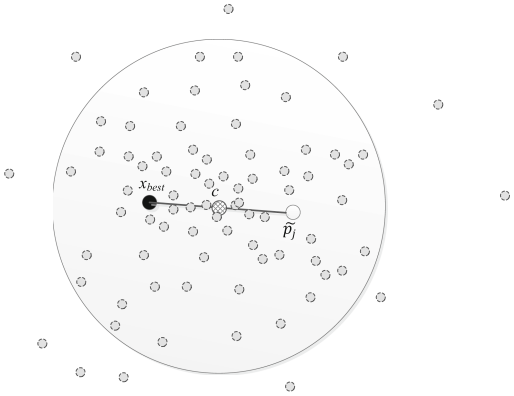



**Fig. 3.** The improved Gaussian sparks' locations.

As shown in Fig. 3, because the distribution character of the Gaussian function, the new mutation operator will have a high chance to scatter the spark in the area with the center c $= \frac{x_{best}^k + \widetilde{p}_j^k}{2}$ and the radius $\left\|x_{best} - \widetilde{p}_j\right\|$. The search trajectory of Gaussian sparks operator in EFWA is along the line between $x_{best}$ and $\widetilde{p}_j$, but the improved Gaussian sparks operator enlarge the search region.

The details of the improved Gaussian operator are given in Algorithm 4.

**Algorithm 4**. Generate an improved Gaussian spark of firework $x_i$

Initialize the location of the spark $\widetilde{p}_J = x_i$;
Randomly select $z$ dimensions of $\widetilde{p}_J$;
**for** each dimension $\widetilde{p_J^k} \in$ {pre-selected $z$ dimensions of $\widetilde{p}_J$} do
  Calculate the amplitude for $\widetilde{p}_J$ by formula (10), $\widetilde{p_J^k} = \frac{x_{best}^k + \widetilde{p_J^k}}{2} \pm \left\|x_{best} - \widetilde{p}_J\right\| * (1 + N(0,1))$
  **if** $\widetilde{p_J^k} < b_k$ or $\widetilde{p_J^k} > a_k$ then
    $\widetilde{p_J^k} = a_k + \left|\widetilde{p_J^k}\right| mod(b_k - a_k)$
  **end if**
**end for**

### 4.3 Framework of IEFWA

Algorithm 5 summarizes the framework of IEFWA. In each generation, the population goes through two types of explosions that generates explosion sparks and Gaussian sparks. After the evaluation process, the best one will be conserved to next generation. Suppose the algorithm is executed in T generations, the complexity of the IEFWA is O (T•(M + m)).

Algorithm 5. Framework of the IEFWA

Randomly initialize $n$ locations of fireworks;
$FEs$=0;
**While** FEs<$FEs_{max}$ **do**
$A_{min}^{k}(t) = A_{init} - \frac{A_{init}-A_{final}}{evals_{max}} \cdot t$;
**for** $i$=1:$n$ **do**
Calculate the number $s_i$ of sparks that the firework $x_i$ produces according to (2);
Calculate the amplitude $A_i$ of sparks that the firework $x_i$ produces according to (4);
Limit the $A_i$ above the low bound: If $A_i$<$A_{min}$ $A_i$=$A_{min;}$
generate explosion sparks for fireworks by algorithm 1;
**end for**
**for** $k$=1:$m$ **do**
Randomly select a firework $x_j$;
generate Gaussian sparks for fireworks by algorithm 4;
**end for**
$FEs = FEs + M + m$;
select one best location among the fireworks and sparks;
randomly select other $n$-1 locations;
**end while**

## 5 Experiments and Discussion

### 5.1 Benchmark Functions

To investigate the performance of the proposed IEFA, we conducted the experiments on 20 benchmark functions for 30 dimensions. The function name, the feasible bounds and the optimal fitness are listed in Table 1. The functions in Table 1 can be classified into three types, (1) unimodal functions; (2) multimodal functions; (3) shift functions.

**Table 1.** Benchmark functions

| Type | Functions | Name | $f(\overrightarrow{X^*})$ | Search range |
|---|---|---|---|---|
| Unimodal functions | $F_{01}$ | Sphere | 0 | [−100, 100] |
| | $F_{02}$ | Schwefel 2.22 | 0 | [−10, 10] |
| | $F_{03}$ | Schwefel 1.2 | 0 | [−100, 100] |
| | $F_{04}$ | Schwefel 2.21 | 0 | [−100, 100] |
| | $F_{05}$ | Rosenbrock | 0 | [−100, 100] |
| | $F_{06}$ | Step | 0 | [−100, 100] |
| | $F_{07}$ | Quartic with Noise | 0 | [−1.28, 1.28] |
| Multimodal functions | $F_{08}$ | Schwefel 2.26 | −1259.5 | [−500, 500] |
| | $F_{09}$ | Rastrigin | 0 | [−5.12, 5.12] |
| | $F_{10}$ | Ackley | 0 | [−32, 32] |
| | $F_{11}$ | Griewank | 0 | [−600, 600] |
| | $F_{12}$ | Penalized1 | 0 | [−50, 50] |
| | $F_{13}$ | Penalized2 | 0 | [−50, 50] |
| Shift functions | $F_{14}$ | Shift Sphere | 2 | [−100, 100] |
| | $F_{15}$ | Shift Schwefel 1.2 | 2 | [−100, 100] |
| | $F_{16}$ | Shift Schwefel 1.2 with Noise | 2 | [−100, 100] |
| | $F_{17}$ | Shift Griewank | 2 | [−600, 600] |
| | $F_{18}$ | Shift Ackley | 2 | [−32, 32] |
| | $F_{19}$ | Shift Penalized1 | 2 | [−50, 50] |
| | $F_{20}$ | Shift Penalized2 | 2 | [−50, 50] |

### 5.2 Comparison Experiments Among EFWA, dynFWA and IEFWA

In this section, we compare the performance of IEFWA with EFWA and dynFWA in terms of both convergence speed and optimization accuracy. The parameters are set as those used in the original literature.

(1) EFWA: $n = 5$, $M = 50$, $\hat{A} = 40$, $A_{init} = 0.02(\text{b} - \text{a})$, $A_{final} = 0.001(b - a)$, $m = 5$;
(2) dynFWA: $M_e$(the maximum number of explosions sparks) = 150, $\hat{A} = 40$, $C_r = 0.9$, $C_a = 1.2$;
(3) IEFWA: $n = 5$, $M = 50$, $\hat{A} = 40$, $A_{init} = 0.02(\text{b} - \text{a})$, $A_{final} = 0.001(b - a)$, $m = 5$;

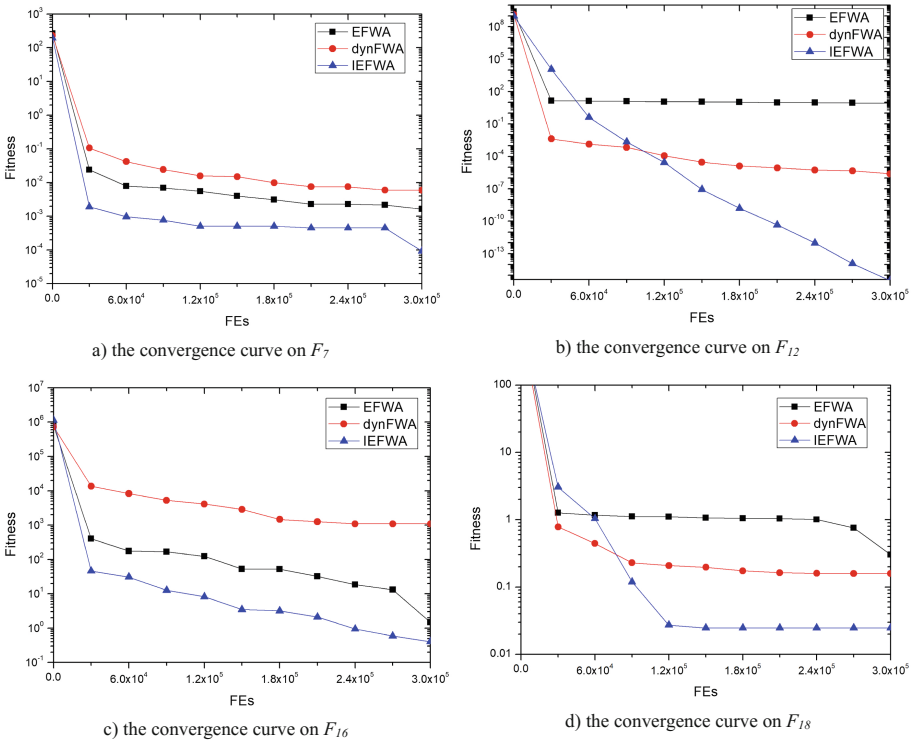
Each algorithm is executed independently 50 times, the mean solution error and standard deviation are reported in Table 2 for D = 30. The wilcoxon's rank-sum test at the 0.05 significance level is employed to judge the significant difference between IEFWA and other algorithms, "+", "−" and "=" represent our proposed algorithm IEFWA is, respectively, better than, worse than and similar to the compared one in the Wilcoxon's rank-sum test.

**Table 2.** The results of EFWA, dynFWA and IEFWA

| | EFWA Mean ± Std | | dynFWA Mean ± Std | | IEFWA Mean ± Std |
|---|---|---|---|---|---|
| $F_1$ | 1.53E−01 ± 2.43E−02 | + | 1.74E−15 ± 2.26E−15 | + | **0.00E+00** ± 0.00E+00 |
| $F_2$ | 1.67E−01 ± 1.76E−02 | + | 8.91E−10 ± 6.77E−10 | + | **0.00E+00** ± 0.00E+00 |
| $F_3$ | 1.09E+00 ± 3.22E−01 | + | 5.32E−04 ± 6.41E−04 | + | **0.00E+00** ± 0.00E+00 |
| $F_4$ | 1.87E−01 ± 2.02E−02 | + | 2.72E−05 ± 2.84E−05 | + | **0.00E+00** ± 0.00E+00 |
| $F_5$ | 9.91E+01 ± 1.22E+02 | + | 9.02E+01 ± 9.28E+01 | + | **2.74E+01** ± 2.60E−01 |
| $F_6$ | 7.00E−01 ± 7.50E−01 | + | 4.63E+00 ± 1.96E+00 | + | **0.00E+00** ± 0.00E+00 |
| $F_7$ | 1.98E−03 ± 9.36E−04 | + | 6.99E−03 ± 2.58E−03 | + | **1.11E−04** ± 1.20E−04 |
| $F_8$ | **−7.16E+03** ± 6.06E+02 | = | −7.38E+03 ± 7.77E+02 | = | −7.19E+03 ± 7.74E+02 |
| $F_9$ | 1.52E+02 ± 2.62E+01 | + | 2.36E+01 ± 9.39E+00 | + | **0.00E+00** ± 0.00E+00 |
| $F_{10}$ | 4.69E+00 ± 8.22E+00 | + | 1.56E+00 ± 4.68E+00 | + | **0.00E+00** ± 0.00E+00 |
| $F_{11}$ | 2.38E−01 ± 3.71E−02 | + | 3.21E−02 ± 2.60E−02 | + | **0.00E+00** ± 0.00E+00 |
| $F_{12}$ | 6.78E+00 ± 2.14E+00 | + | 3.46E−03 ± 1.89E−02 | + | **4.80E−06** ± 2.81E−06 |
| $F_{13}$ | 6.48E−03 ± 3.83E−03 | + | **7.91E−12** ± 4.33E−11 | − | 6.61E−05 ± 3.99E−05 |
| $F_{14}$ | 1.56E−01 ± 2.75E−02 | + | **9.80E−15** ± 2.79E−14 | − | 1.47E−03 ± 6.41E−04 |
| $F_{15}$ | 9.76E−01 ± 2.43E−01 | + | **4.76E−04** ± 5.25E−04 | − | 2.36E−01 ± 6.41E−02 |
| $F_{16}$ | 2.08E+00 ± 9.36E−01 | + | 5.45E+02 ± 4.11E+02 | − | **2.22E−01** ± 6.71E−02 |
| $F_{17}$ | 2.47E−01 ± 4.69E−02 | + | **3.60E−02** ± 2.49E−02 | = | 9.97E−02 ± 5.44E−02 |
| $F_{18}$ | 9.27E+00 ± 9.61E+00 | + | 6.02E+00 ± 8.41E+00 | + | **8.89E−03** ± 1.58E−03 |
| $F_{19}$ | 6.58E+00 ± 2.39E+00 | + | 2.37E−04 ± 1.30E−03 | + | **5.61E−06** ± 3.45E−06 |
| $F_{20}$ | 5.68E−03 ± 1.12E−03 | + | **2.28E−13** ± 1.23E−12 | − | 7.19E−05 ± 4.20E−05 |

From Table 2, IEFWA obtains better results than EFWA and dynFWA on 14 out of 20 functions, respectively. For unimodal functions $F_1$–$F_7$, it is clear that IEFWA is the best one, it beats EFWA and dynFWA on each function. That may be the reason that IEFWA uses the globe best firework information in the Gaussian sparks operator and expand the search direction. For multimodal functions $F_8$–$F_{13}$, IEFWA outperforms EFWA on 5 out of 6 functions (except for function $F_8$ where they are even) and IEFWA performs better than dynFWA on 4 functions (except for function $F_8$ where they are even and function $F_{13}$ where dynFWA is better). For shift functions $F_{14}$–$F_{20}$, IEFWA is better than EFWA, but worse than dynFWA on 4 functions. It might be because dynFWA's dynamic explosion amplitude method and removal of Gaussian explosion operator are suitable for shift type functions.

The convergence curves of EFWA, dynFWA and IEFWA on function $F_7$, $F_{12}$, $F_{16}$, $F_{18}$ are shown in Fig. 4. The convergence curves show IEFWA has a fast convergence speed without loss of accuracy.




a) the convergence curve on $F_7$

b) the convergence curve on $F_{12}$

c) the convergence curve on $F_{16}$

d) the convergence curve on $F_{18}$

**Fig. 4.** The convergence curves of EFWA, dynFWA and IEFWA.

## 6 Conclusion

Based on the comprehensive study on EFWA, we propose an improved version of enhanced fireworks algorithm (IEFWA) in this paper. In IEFWA, to fix the problem of monotonous search trajectory in EFWA, a new Gaussian explosion operator which generates the sparks with Gaussian distribution character by the locations and the distance between the best firework and candidate firework is introduced.

Experimental results show that the IEFWA outperforms EFWA and dynFWA on most competition functions. However, some research work should be done on IEFWA to enhance the performance on shift functions.

**Acknowledgment.** This work is supported by the self-determined research funds of CCNU from the colleges basic research and operation of MOE (No. CCNU18QN018).

## References


1. Dorigo, M., Gambardella, L.: Ant colony system: a cooperative learning approach to the traveling salesman problem. IEEE Trans. Evol. Comput. **1**(1), 53–66 (1997)
2. Kennedy, J.: Particle swarm optimization. In: IEEE International Conference on Neural Networks, Australia, pp. 1942–1948. IEEE (1995)
3. Yang, X.: Nature-Inspired Metaheuristic Algorithms. Luniver Press, London (2008)
4. Yang, X., Deb, S.: Cuckoo search via levy flights. In: IEEE World Congress on Nature & Biologically Inspired Computing, India, pp. 210–214. IEEE (2009)
5. Liu, C., Yan, X.: The wolf colony algorithm and its application. Chin. J. Electron. **20**(2), 212–216 (2011)
6. Mirjalili, S., Lewis, A.: The whale optimization algorithm. Adv. Eng. Softw. **95**, 51–67 (2016)
7. Tan, Y., Zhu, Y.: Fireworks algorithm for optimization. In: Tan, Y., Shi, Y., Tan, K.C. (eds.) ICSI 2010. LNCS, vol. 6145, pp. 355–364. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-13495-1_44
8. Zheng, S., Janecek, A., Tan, Y.: Enhanced fireworks algorithm. In: IEEE Congress on Evolutionary Computation, Mexico, pp. 2069–2077. IEEE (2013)
9. Yu, C., Tan, Y.: Fireworks algorithm with covariance mutation. In: IEEE Congress on Evolutionary Computation, Japan, pp. 1250–1256. IEEE (2015)
10. Li, J., Zheng, S., Tan, Y.: Adaptive fireworks algorithm. In: IEEE Congress on Evolutionary Computation, China, pp. 3214–3221. IEEE (2014)
11. Zheng, S., Janecek, A., Li, J., Tan, Y.: Dynamic search in fireworks algorithm. In: IEEE Congress on Evolutionary Computation, China, pp. 3222–3229. IEEE (2014)
12. Liu, L., Zheng, S., Tan, Y.: S-metric based multi-objective fireworks algorithm. In: IEEE Congress on Evolutionary Computation, Japan, pp. 1250–1256. IEEE (2015)
13. Ludwing, S., Dawar, D.: Parallelization of enhanced firework algorithm using MapReduce. Int. J. Swarm Intell. Res. **6**(2), 32–51 (2015)
14. Ding, K., Tan, Y.: Attract-repulse fireworks algorithm and its CUDA implementation using dynamic parallelism. Int. J. Swarm Intell. Res. **6**(2), 1–31 (2015)
15. Yu, C., Kelley, L., Zheng, S., et al.: Fireworks algorithm with differential mutation for solving the CEC 2014 competition problems, China, pp. 3238–3245. IEEE (2014)
16. Zheng, Y., Xu, X., Ling, H., et al.: A hybrid fireworks optimization method with differential evolution operators. Neurocomputing **148**, 75–82 (2015)
17. Gao, H., Diao, M.: Cultural firework algorithm and its application for digital filters design. Int. J. Model. Ident. Control **14**(4), 324–331 (2011)
18. Zheng, S., Tan, Y.: A unified distance measure scheme for orientation coding in identification. In: International Conference on Information Science and Technology, China, pp. 979–985. IEEE (2013)
19. Rajaram, R., Palanisamy, K., Ramasamy, S., et al.: Selective harmonic elimination in PWM inverter using fire fly and fireworks algorithm. Int. J. Innov. Res. Adv. Eng. **1**, 55–62 (2014)
20. Liu, Z., Feng, Z., Ke, L.: Fireworks algorithm for the multi-satellite control resource scheduling problem. In: IEEE Congress on Evolutionary Computation, Japan, pp. 1280–1286. IEEE (2015)