Dynamic Search Fireworks Algorithm with Covariance Mutation for Solving the CEC 2015 Learning Based Competition Problems

Chao Yu[1], Ling Chen Kelley[2,3], and Ying Tan[1,*]
[1] The Key Laboratory of Machine Perception and Intelligence (Ministry of Education), Department of Machine Intelligence, School of Electronics Engineering and Computer Science, Peking University, Beijing, China, 100871
[2] Department of History, School of Humanities, Tsinghua University. Beijing, China, 100084
[3] Department of World Languages and Cultures, College of Liberal Arts and Sciences, Auburn University at Montgomery, Montgomery, AL, United States 36117
Email: chaoyu@pku.edu.cn, lingchenkelley@gmail.com, ytan@pku.edu.cn

***Abstract*—As a revolutionary swarm intelligence algorithm, fireworks algorithm (FWA) is designed to solve optimization problems. In this paper, the dynamic fireworks algorithm with covariance mutation (dynFWACM) is proposed. After applying the explosion operator, the mutation operator is introduced, which calculates the mean value and covariance matrix of the better sparks and produces sparks according with Gaussian distribution. DynFWACM is compared with the most advanced fireworks algorithms to proof its effectiveness. In addition, 15 functions of CEC 2015 competition on learning based real-parameter single objective optimization are used to test the performance of our new proposed algorithm. The experimental results show that dynFWACM outperforms both AFWA and dynFWA, as well as the experimental results of the 15 functions given.**

I. Introduction

Conventional fireworks algorithm (FWA) was first proposed in 2010 by Tan and Zhu to solve function optimization problems [1] and worked effectively on some optimization functions. The performances of twelve evolutionary algorithms are tested and compared by Bureerat in 2011, in which FWA ranked at sixth [2]. Yet, conventional FWA has its own disadvantages.

Many researchers soon discovered the shortcomings of conventional FWA and proposed algorithmic solutions and supplemental improvement variations. Zheng, et al. published an enhanced fireworks algorithm (EFWA) [3], which improved the conventional FWA in the following five aspects: 1) the lower limit of the explosion amplitude was given to help the fireworks produce more useful sparks; 2) the explosion offsets in each dimension were different, whereas they were the same in conventional FWA; 3) a new mapping operator was proposed by randomly mapping the outlying sparks back into the search space; 4) the Gaussian mutation was improved by using the information of the best fireworks; 5) a time-saving selection strategy was utilized to improve conventional FWA effectiveness. Zheng, et al. also introduced a dynamic search FWA, abbreviated as dynFWA, to further improve the performance of EFWA [4]. Other improvement versions of FWA are as follows: 1) Liu, et al. improved EFWA by introducing a new method to calculate the number of sparks and the amplitudes of the explosions [5]; 2) Yu, et al. put forward FWA with differential mutation (FWA-DM) by using differential operators [6], [7]; 3) Li, et al. proposed AFWA and made the explosion amplitudes adaptable [8]; 4) Zheng, et al. proposed a hybrid FWA by using differential operators in FWA [9]; 5) Zhang, et al. researched the biogeography information and put forward a biogeography based FWA [10]. The global convergence ability of FWA was proofed systematically and the time complexity was carefully analyzed by Liu, et al [11]. A summary of other FWA improvements can be found in our article: *Introduction to Fireworks Algorithm* published in 2013 [12].

FWA had been applied to many fields successfully. Zheng, et al. solved the multi-objective optimization problems by multi-objective FWA [13]. Ding, et al. implemented FWA on graphic processing unit (GPU) [14]. Gao and Diao proposed cultural fireworks algorithm to design digital filters [15]. Janecek and Tan used FWA to deal with the non-negative matrix factorization problems [16]–[18]. He, et al. designed FWA to optimize the parameters of an algorithm to filter junk mail [19]. Therefore, FWA is applied to many fields and solves practical problems effectively.

Despite the achievements of FWA in the applications, FWA and its variants are far from perfect. For example, the mutation operator in some FWA variants, e.g. AFWA and FWA-DM, largely depends on the information conveyed by the best firework, which leaves a substantial quantity of unused information from the remaining sparks produced by less fitting fireworks. The mutation operator is eliminated in dynFWA, which generated better sparks. The Covariance mutation was applied to AFWA [20]. In this paper, the covariance mutation is implemented in dynFWA. As such, the information provided from the best firework and the better sparks are all utilized.

The rest of the paper is organized as follows. The details of our new algorithm dynFWACM are first introduced in section II. The mutation operator of dynFWACM is analyzed in section III. The experimental settings and results are shown in section

* Prof. Y. Tan is the corresponding author.

IV, followed by the discussion and conclusion.

II. Dynamic Fireworks Algorithm with Covariance Mutation

DynFWACM is proposed to deal with optimization problems. dynFWACM mimics the phenomena that fireworks explode in the night sky by producing sparks in their adjacent areas. There are two kinds of sparks generated in dynFWACM, as 'explosion sparks' and 'Gaussian sparks', respectively. The explosion sparks are generated by explosion operator, while the Gaussian sparks are produced by covariance mutation. All the produced sparks are evaluated and some of them are selected and passed down to the next generation. This iteration continues until the termination conditions are met. Typically, the maximum number of function evaluations and the accuracy requirement are the termination conditions.

DynFWACM consists of explosion operator, mutation operator, mapping rules and selection strategy. The details of them are as follows.

A. Explosion Operator

The explosion operator mimics the phenomena of fireworks explosion. The image in Fig. 1(a) represents the fireworks explosion in the night sky, while the Fig. 1(b) is the explosion operator in FWA. The five-pointed star in Fig. 1(b) indicates the position of the firework, whereas its surrounded circles stand for the sparks produced by the explosion operator.

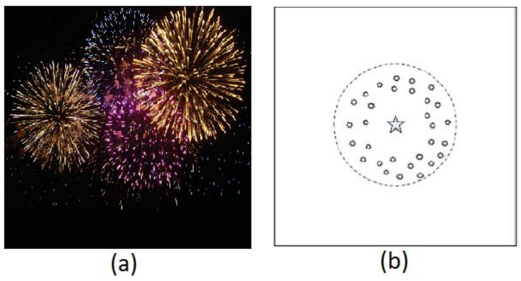

Fig. 1. Fireworks explosion is shown in (a), and an explosion in FWA is represented in (b).

When the fireworks explode, the number of the sparks and the amplitude of the explosion need to be determined. In dynFWACM, the number of sparks is gained the same as the conventional FWA.

Parameter S_i represents the number of sparks for the i^{th} firework.

$$S_i = \hat{S} * \frac{Y_{\max} - f(x_i) + \varepsilon}{\sum\limits_{i=1}^{N} (Y_{\max} - f(x_i)) + \varepsilon}. \tag{1}$$

In Eq. 1, the parameter $\hat{S}$ controls the number of sparks. The parameter Y_{max} is the fitness value of the worst individual, whereas the function $f(x_i)$ represents the fitness value of the individual x_i. The parameter N stands for the number of fireworks and the parameter ε is used to prevent the denominator from becoming zero. To cope with the overwhelming effects of the best firework, the lower and upper boundaries for the parameter S_i are denoted as N_{min} and N_{max}, respectively. The two boundaries are set empirically.

In dynFWACM, the phrase 'core firework' represents the firework with the best fitness value. The amplitude for the core firework is different from the amplitudes of other fireworks. Let parameter A_{CF} denotes the amplitude of the core firework and parameter A_i stands for the amplitude for the i^{th} firework.

$$A_{CF}(g+1) = \begin{cases} A_{CF}(g) * C_a, iff(\hat{X}_b) < f(X_{CF}) \\ A_{CF}(g) * C_r, otherwise \end{cases}, \tag{2}$$

In Eq. 2, parameter g is the number of generations. C_a and C_r are the amplification and reduction parameters, whereas $\hat{X}_b$ and X_{CF} denote the best firework and core firework, respectively. Function $f(x)$ stands for the fitness value of individual x. For a minimization problem, if $f(\hat{X}_b)$ is less than $f(X_{CF})$ in the g generation, the amplification parameter C_a is used to amplify the amplitude of the core firework. Otherwise, the reduction parameter C_r is utilized.

$$A_i = \hat{A} * \frac{f(x_i) - Y_{\min} + \varepsilon}{\sum\limits_{i=1}^{N} (f(x_i) - Y_{\min}) + \varepsilon}. \tag{3}$$

In Eq. 3, the parameter $\hat{A}$ is used to control the amplitude of the explosion. Function $f(x_i)$ means the fitness value of the individual x_i. The parameter Y_{min} stands for the worst fitness value of the individuals. The parameter N denotes the number of fireworks and the parameter ε is the same as aforementioned.

The details of explosion operator are described in Alg. 1.

Algorithm 1 Explosion Strategy in dynFWACM

```
1:  calculate $S_i$, $A_{CF}$ and $A_i$.
2:  $p = 0$
3:  for $i = 1 \to N$ do
4:    for $k = 1 \to S_i$ do
5:      $p = p + 1$
6:      for $j = 1 \to D$ do
7:        if $rand(0,1) \leq 0.5$ then
8:          if $X_i$ is the core firework then
9:            $s_{pj} = X_{ij} + rand(-1,1) \cdot A_{CF}$
10:         else
11:           $s_{pj} = X_{ij} + rand(-1,1) \cdot A_i$
12:         end if
13:       end if
14:     end for
15:     if $s_{pj} < LB$ or $s_{pj} > UB$ then
16:       $s_{pj} = LB + rand(0,1) \cdot (UB - LB)$
17:     end if
18:   end for
19: end for
```

In the explosion strategy, the parameter N denotes the number of fireworks and the parameter D stands for the number of dimensions. The function $rand(a, b)$ stands for generating random number according with uniform distribution between a and b. The symbol s_{pj} represents the position of spark s_p in j^{th} dimension and X_{ij} is the position of the selected firework X_i in j^{th} dimension. A_{CF} is the amplitude for the core firework and A_i is the amplitude for the other fireworks. Parameter UB and LB are upper and lower bounds, respectively.

B. Mutation Operator

In dynFWACM, the covariance mutation is used and can be called as Gaussian mutation as well. The covariance mutation takes advantage of the information of better sparks in each generation, preventing to concentrate on using the information of the best spark. The sparks with Gaussian distribution are produced by using the mean value and the covariance matrix of the better sparks. These sparks are abbreviated as Gaussian sparks. The mean value and the covariance matrix are obtained as follows.

First of all, the mutation operator calculates the mean value of the better explosion sparks. The number of better sparks is defined as μ, whereas the number of all the sparks is denoted as λ. The mean value of better sparks is represented as m and calculated by the following equation.

$$m = \sum_{i=1}^{\mu} w_i x_i, \tag{4}$$

where w_i is the weight of the i^{th} spark and individual x_i represents the i^{th} better spark selected. The parameter w_i is calculated in Eq. 5.

$$w_i = 1/\mu \quad (i = 1, 2, ..., \mu). \tag{5}$$

It is clear that m is the mean value of the μ individuals, not of all individuals.

Secondly, the mutation operator calculates the covariance matrix C. Equation 6 shows the details of the matrix C.

$$C = \begin{pmatrix} cov(d_1, d_1) & cov(d_1, d_2) & \cdots & cov(d_1, d_D) \\ cov(d_2, d_1) & cov(d_2, d_2) & & \vdots \\ \vdots & & \ddots & \vdots \\ \text{cov}(d_D, d_1) & \cdots & \cdots & cov(d_D, d_D) \end{pmatrix}, \tag{6}$$

where $cov(d_i, d_j)$ stands for the covariance of the sparks in i^{th} and j^{th} dimension. Vector d_i represents the sparks in the i^{th} dimension and D denotes the number of dimensions.

$$\text{cov}(a, b) = \frac{\sum_{i=1}^{\mu} (a_i - \bar{A})(b_i - \bar{B})}{\mu}. \tag{7}$$

In Eq. 7, $\bar{A}$ and $\bar{B}$ denote the mean values of λ sparks in different dimensions. Moreover, a_i is the i^{th} data in the former dimension and b_i is the i^{th} data in the latter dimension. Note that $\mu - 1$ is taken as a denominator when calculating the covariance, but the denominator is μ in Eq. 7. Therefore, the way to calculate the covariance is different from the usual way.

Thirdly, the mutation operator utilizes the mean value m and covariance matrix C to produce mutation sparks according with Gaussian distribution. The produced sparks have varied names, as mutation sparks, Gaussian sparks or Gaussian mutation sparks. The name Gaussian sparks is used afterwards. Figure 2 illustrates the distribution of Gaussian sparks. The curves are contour lines, whereas the ellipsoid area of dotted lines indicates the boundary of the better sparks. Moreover, the black dot denotes the center of the sparks and the solid line ellipsoid shows the boundary of Gaussian sparks.

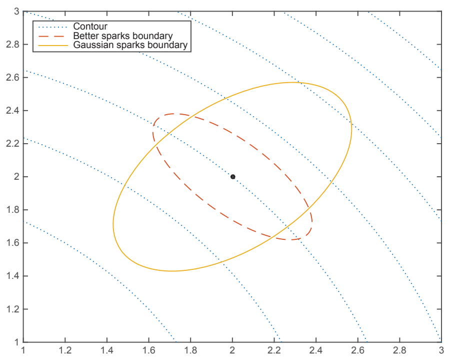

Fig. 2. The Gaussian sparks distribution with $N(m, C)$.

It can be seen from Fig. 2 that the possibility of finding useful sparks increased, as the Gaussian sparks produced by covariance mutation can now better navigate towards the optima direction.

The algorithm of the covariance mutation is given in Alg. 2.

Algorithm 2 Covariance Mutation

```
1: for $i = 1 \to N$ do
2:     obtain the mean value $m$ according to Eq. 4
3:     calculate the covariance matrix $C$ according to Eq. 6
4:     generate Gaussian sparks with $N(m, C)$
5: end for
```

C. Mapping Rules

Mapping rules are used to deal with outlying sparks. In both explosion and mutation operator, it is possible for a firework to generate sparks that are out of the feasible space. To cope with this problem, a mapping operator is designed. In dynFWACM, the outlying sparks are mapped to locations within the search space.

$$X_i^k = X_{\text{min}}^k + rand(0, 1) * (X_{\text{max}}^k - X_{\text{min}}^k). \tag{8}$$

In Eq. 8, X_i^k is the location of individual X_i in the k^{th} dimension. X_{max}^k and X_{min}^k are the upper and lower bounds for the individual X in its k^{th} dimension, respectively. Function $rand(0, 1)$ produces a random number with uniform distribution.

D. Selection Strategy

N individuals are selected for the next generation. Suppose the N fireworks produced N groups of sparks, the best individual in each group was adopted for the next generation. Therefore, N individuals are chosen. The selection operator keeps the diversity of the population.

To give an overall look of the new algorithm, the algorithm of dynFWACM is presented in Alg. 3.

Algorithm 3 The algorithm of dynFWACM

```
1:  generate $N$ fireworks with uniform distribution randomly.
2:  evaluate the fitness values of the $N$ fireworks
3:  while terminate condition is not met do
4:      calculate the number of sparks $S_i$
5:      calculate the amplitude of explosion $A_{CF}$ and $A_i$
6:      generate $S_i$ explosion sparks within the amplitudes
7:      calculate $m$ and covariance matrix $C$
8:      generate Gaussian sparks according with $N(m, C)$
9:      evaluate all the fitness values of explosion and Gaussian
        sparks
10:     keep $N$ individuals for the next generation
11: end while
12: return  the best individual and its fitness value
```

In the algorithm of dynFWACM, the mean value vector m includes D elements, where D indicates the dimension of the optimization functions. The matrix C is a D-by-D symmetric positive semi-definite matrix, stands for a calculated covariance matrix of better sparks. Both m and C are used to produce sparks. However, spark selection and generation methods differ from dynFWACM and dynFWA, where $N-1$ individuals were selected at random with dynFWA and individuals were chosen from each group of sparks with dynFWACM.

III. Analysis of the Mutation in dynFWACM

Traditional dynFWA does not contain any mutation operators. A covariant mutation was added to dynFWA to create dynFWACM, and the covariance mutation is further analyzed. Using the uniform distribution, N number of fireworks were generated. The dynamic amplitude was calculated for the core firework, whereas for all other fireworks, their amplitudes were calculated using the information of the fitness values from each firework. The explosion sparks were then generated and N groups of sparks were produced by the explosion operator. The mean value and the covariance matrix of better sparks were obtained for the group of sparks with the best firework in last generation. Sparks generated by the covariance mutation follow the Gaussian distribution when a covariance mutation operator was applied. Thus, the sparks can also be referred to as Gaussian sparks. All outlying sparks were mapped back into feasible space. The individual spark (or firework) with the best fitness value from each group when applied to a function was kept for the next generation. The algorithm continued cycling through explosions, mutations, mapping and selection operators until the preset termination conditions were met.

A two-dimensional function $f(x) = \sum_{i=1}^{2} x_i^2$ is taken as an example to visualize the covariance mutation.

In Fig. 3(a), the contour lines of function $f(x)$ are drawn, where the individuals lying on the same contour line have the same fitness values. Then, 50 sparks are produced randomly within scope $1 < x_i < 3 (i = 1, 2)$ with uniform distribution, as shown in Fig. 3(b). The sparks with better fitness values are retained and marked with black 'x' in Fig. 3(c). The produced Gaussian sparks are shown in Fig. 3(d), which are marked as '+'. Note that a few Gaussian sparks beyond the boundary of Fig.3(d) are omitted from this graph.

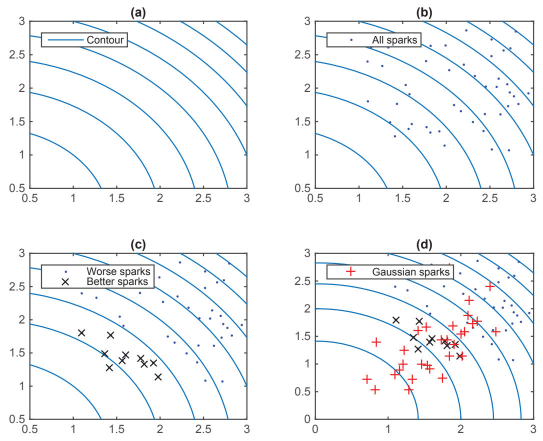

Fig. 3. The process of generating Gaussian sparks by covariance mutation.

It can be seen from Fig. 3 that the generated Gaussian sparks by covariance mutation match the directional gradient of the two-dimensional function $f(x)$. Therefore, the covariance mutation is good at finding local optimums, which can also be extended to higher dimensions.

To further analyze the performance of covariance mutation, four experiments were carried out. To make the experimental results impartial, four types of functions from the CEC 2015 Competition were used, including unimodal function 1, multimodal function 3, hybrid function 6, and composition function 9. dynFWACM ran for 51 times and during each run, the numbers of better sparks generated by both operators were recorded. The average better sparks produced by mutation operator and explosion operator were shown in Table I, respectively.

TABLE I. The comparison of explosion and mutation operators in dynFWACM

Function No.	Explosion Sparks	Gaussian Sparks
1	123.65	101.53
3	92.92	122.55
6	110.25	110.04
9	118.25	56.67

To further clarify, the number of sparks produced by each operator during a generation needs to be considered. When 150 explosion sparks are generated, 30 Gaussian sparks are produced. Using this ratio, if the better sparks produced by the mutation operator are one fifth of the number of sparks produced by explosion operator, the mutation operator appears more effective than the explosion operator. The results from Table I reveal that the mutation operator is effective than the explosion operator among all the four functions.

IV. Experiments

The benchmark functions are first introduced, followed by the parameter settings, the experiment results are given, including the comparison between dynFWA, AFWA and dynFWACM. A value of 100 was added to function 1, 200 to function 2, ..., and a value of 1500 was added to function 15, all of which were subtracted from the calculated mean errors so that the best value for each function remains zero.

A. Benchmark Functions

dynFWACM is used to find the global optimum values of 15 benchmark functions from the CEC'15 competition. The details of the 15 functions can be seen from [21]. The names and attributes of the functions are given in Table II.

TABLE II. The attributes and function names

Attributes	Function Names
Unimodal Functions	Rotated High Conditioned Elliptic Function
	Rotated Bent Cigar Function
Multimodal Functions	Shifted and Rotated Ackley's Function
	Shifted and Rotated Rastrigin's Function
	Shifted and Rotated Schwefel's Function
Hybrid Functions	Hybrid Function 1, 3 and 5
Composition Functions	Composition Function 1 to 7

B. Parameters Settings

Each experiment ran 51 times and during each run, the function was evaluated for 10000*D times. The parameter D denoted the dimension of the function. The actual used parameter values were set as follows. Parameters $\hat{A}$, N_{min} and N_{max} were set as 40, 6 and 120, respectively. The number of the better μ sparks was set as 50% of the number of sparks generated from a firework. If μ is not a whole number, then a lower whole number would be selected. The population size N was set as 5 and the lower and upper bounds of the number of explosion sparks were set as 4 and 100, respectively. The number of Gaussian sparks equals to the dimension number. The other parameters are setting as in [4].

The parameters were adjusted for competition as in Table III.

TABLE III. Parameters Tuning

Function No.	Ca	Cr
1	1.11	0.80
2	1.18	0.85
3	1.18	0.85
4	1.25	0.90
5	1.25	0.90
6	1.11	0.80
7	1.25	0.90
8	1.11	0.90
9	1.25	0.90
10	1.25	0.85
11	1.18	0.90
12	1.18	0.85
13	1.25	0.90
14	1.18	0.90
15	1.11	0.80

C. Experimental Results

The experimental platform is Matlab 2014a and the program is run on a Windows 8 operating system.

The mean errors of 30-dimension functions are presented in Table IV. Note that the best values for all functions are zero, as the increments for each function have been trimmed off for better understanding the experimental results. The numbers in bold are the best solutions of all three algorithms. The best result for function 3 is obtained by dynFWA, though the remaining two algorithms have identical values within two decimal points. To make a fair comparison, the parameters used in dynFWACM are not tuned in Table IV.

TABLE IV. Mean errors of dynFWA, AFWA and dynFWACM

Function No.	dynFWA	AFWA	dynFWACM
1	8.87E+05	1.18E+06	**7.91E+05**
2	4.89E+03	4.02E+03	**3.80E+03**
3	**2.00E+01**	2.00E+01	2.00E+01
4	1.29E+02	1.46E+02	**1.17E+02**
5	3.55E+03	3.30E+03	**3.23E+03**
6	5.41E+04	5.65E+04	**3.00E+04**
7	1.44E+01	1.62E+01	**1.42E+01**
8	4.66E+04	4.30E+04	**2.64E+04**
9	1.36E+02	1.39E+02	**1.08E+02**
10	5.21E+04	5.94E+04	**3.63E+04**
11	7.60E+02	7.87E+02	**6.65E+02**
12	**1.21E+02**	1.31E+02	1.27E+02
13	2.80E-02	2.77E-02	**2.51E-02**
14	4.50E+04	**4.46E+04**	**4.46E+04**
15	**1.00E+02**	**1.00E+02**	**1.00E+02**

The experimental results on 10, 30, 50 and 100-dimension functions of CEC 2015 single objective optimization are given in Table V, VI, VII and VIII, respectively.

TABLE V. Results for 10D

Function No.	Best	Worst	Median	Mean	Std.
1	7.05E+03	4.90E+05	9.23E+04	1.11E+05	9.88E+04
2	1.75E+01	3.30E+04	6.26E+03	8.86E+03	8.90E+03
3	2.00E+01	2.00E+01	2.00E+01	2.00E+01	2.33E-04
4	6.96E+00	3.48E+01	1.59E+01	1.67E+01	6.65E+00
5	1.30E+02	1.09E+03	4.78E+02	5.18E+02	2.40E+02
6	3.59E+01	7.31E+03	8.30E+02	1.74E+03	1.97E+03
7	9.03E-01	2.12E+00	1.45E+00	1.45E+00	2.31E-01
8	1.23E+01	8.03E+03	1.24E+03	1.96E+03	2.13E+03
9	1.00E+02	1.02E+02	1.00E+02	1.00E+02	4.59E-01
10	1.54E+02	1.22E+03	4.75E+02	5.47E+02	2.37E+02
11	1.85E+00	3.03E+02	3.01E+02	1.85E+02	1.47E+02
12	1.08E+02	1.17E+02	1.13E+02	1.13E+02	1.52E+00
13	9.60E-02	1.96E-01	1.20E-01	1.21E-01	1.66E-02
14	1.00E+02	8.74E+03	6.83E+03	6.30E+03	2.12E+03
15	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.44E-13

TABLE VI. Results for 30D

Function No.	Best	Worst	Median	Mean	Std.
1	2.55E+05	1.34E+06	5.57E+05	6.17E+05	2.49E+05
2	2.93E+01	1.46E+04	2.15E+03	3.31E+03	3.59E+03
3	2.00E+01	2.00E+01	2.00E+01	2.00E+01	5.05E-06
4	7.26E+01	2.55E+02	1.21E+02	1.30E+02	3.80E+01
5	1.96E+03	4.88E+03	3.32E+03	3.38E+03	6.98E+02
6	2.91E+03	8.35E+04	2.06E+04	2.69E+04	1.90E+04
7	8.58E+00	2.00E+01	1.49E+01	1.46E+01	2.57E+00
8	3.86E+03	5.00E+04	2.25E+04	2.40E+04	1.32E+04
9	1.06E+02	1.10E+02	1.08E+02	1.08E+02	9.01E-01
10	7.47E+03	7.83E+04	2.57E+04	3.15E+04	2.01E+04
11	3.01E+02	9.75E+02	6.77E+02	6.72E+02	1.54E+02
12	1.10E+02	2.02E+02	1.15E+02	1.17E+02	1.23E+01
13	1.73E-02	4.58E-02	2.48E-02	2.62E-02	7.46E-03
14	4.15E+04	4.61E+04	4.52E+04	4.49E+04	1.02E+03
15	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.75E-13

The computational complexity of dynFWACM is given in Table IX.

V. Discussion

Table IV showed that dynFWACM outperformed other comparison algorithms. The dynFWACM only lagged behind

TABLE VII. RESULTS FOR 50D

Function No.	Best	Worst	Median	Mean	Std.
1	8.16E+05	4.06E+06	1.67E+06	1.80E+06	7.14E+05
2	7.00E-01	3.52E+04	1.30E+03	5.17E+03	8.01E+03
3	2.00E+01	2.00E+01	2.00E+01	2.00E+01	1.30E-06
4	1.27E+02	4.13E+02	2.33E+02	2.44E+02	5.85E+01
5	4.54E+03	7.58E+03	5.64E+03	5.78E+03	7.00E+02
6	2.83E+04	1.90E+05	8.33E+04	8.81E+04	3.82E+04
7	1.78E+01	1.06E+02	5.89E+01	5.70E+01	2.61E+01
8	1.30E+04	1.53E+05	6.09E+04	6.88E+04	4.00E+04
9	1.03E+02	4.79E+02	1.03E+02	1.11E+02	5.26E+01
10	1.35E+04	9.26E+04	3.84E+04	4.21E+04	1.72E+04
11	3.06E+02	1.60E+03	1.11E+03	1.08E+03	2.44E+02
12	1.21E+02	2.05E+02	2.03E+02	1.75E+02	3.80E+01
13	6.43E-02	2.58E-01	1.08E-01	1.17E-01	4.51E-02
14	5.29E+04	8.39E+04	5.33E+04	5.49E+04	6.34E+03
15	1.00E+02	1.00E+02	1.00E+02	1.00E+02	2.78E-13

TABLE VIII. RESULTS FOR 100D

Function No.	Best	Worst	Median	Mean	Std.
1	1.95E+06	7.50E+06	3.92E+06	3.88E+06	1.24E+06
2	1.30E+01	2.00E+04	1.16E+03	3.66E+03	4.63E+03
3	2.00E+01	2.00E+01	2.00E+01	2.00E+01	2.58E-06
4	4.55E+02	9.29E+02	6.09E+02	6.17E+02	1.07E+02
5	1.02E+04	1.65E+04	1.28E+04	1.29E+04	1.23E+03
6	1.90E+05	1.43E+06	4.43E+05	5.40E+05	2.78E+05
7	3.47E+01	1.87E+02	1.30E+02	1.33E+02	3.85E+01
8	7.72E+04	4.32E+05	1.63E+05	1.83E+05	7.67E+04
9	1.11E+02	1.15E+02	1.13E+02	1.13E+02	9.32E-01
10	1.35E+05	4.72E+05	2.43E+05	2.58E+05	7.35E+04
11	3.05E+02	2.90E+03	2.46E+03	2.27E+03	6.86E+02
12	1.20E+02	2.05E+02	1.26E+02	1.54E+02	3.84E+01
13	5.71E-01	2.90E+00	1.19E+00	1.36E+00	5.95E-01
14	1.24E+05	1.66E+05	1.40E+05	1.41E+05	8.72E+03
15	1.02E+02	1.11E+02	1.04E+02	1.04E+02	1.63E+00

dynFWA on function 3 and 12. Thus, dynFWACM performed well on unimodal functions and hybrid functions. In fact, dynFWACM surpassed AFWA on thirteen out of fifteen functions and tied with AFWA on the remaining two. The experimental results further proved that dynFWACM worked better on unimodal functions, as the covariance mutation increased its local search ability. Practically speaking, CM operator can be used to further enhance AFWA's performance so that a more accurate result can be pinpointed with less margin of error - as in reducing junk mail with greater accuracy or clarifying images quicker with digital filters.

From the experimental results of 10-dimensional functions, the best mean error was in function 13, whereas the fitness value of function 13 was relatively small than other functions. For functions 3, 9, 13 and 15, the standard deviations were small. This meant dynFWACM was steady on these functions.

For the functions with 30 dimensions, experimental results revealed that the promising value was from function 13. However, This didn't mean that dynFWACM was better on function 13 than on other functions, but implied the fitness values for function 13 was smaller than other functions. The experimental results on functions 3, 9, 13 and 15 were stable, as their standard deviations were lower than 1.

Judging from the experimental results of the 50-dimensional functions, the mean error from function 13 was the best. Aside from this best fitness value, dynFWACM failed to find any global optimum with error lower than 1. If the criterion standard was eased, the mean errors for functions 3 and 7 were acceptable, as they were lower than 10. Moreover, functions 3, 9, 13 and 15 were stable as they were in 10 and 30-dimensional functions.

TABLE IX. COMPUTATIONAL COMPLEXITY OF DYNFWACM GIVEN FOR 10, 30, 50 AND 100 DIMENSIONAL FUNCTION 1

	T0	T1	T2	(T2-T1)/T0
D=10	0.136943	1.781273	3.244838	10.6874028
D=30	0.136943	2.142620	4.076168	14.1193635
D=50	0.136943	2.702535	5.253674	18.6292034
D=100	0.136943	4.769288	8.278836	25.6278013

The worst results were found in functions with 100 dimensions without question. Yet, the experimental results of functions 3 and 13 were promising. The two functions were stable, as only their standard deviations were lower than 1.

VI. CONCLUSION

To diverse the population in dynFWA, covariance mutation is introduced. The mutation operator utilizes the information of the better individuals in the current generation and has been proven to be more effective than the explosion operator. Therefore, using the mutation operator, the new algorithm dynFWACM outperforms AFWA and dynFWA on most competition functions. Even if dynFWACM is an improvement of dynFWA, its performance requires proof by comparison with other participated algorithms from the CEC 2015 competition.

ACKNOWLEDGMENT

This work was supported by the Natural Science Foundation of China (NSFC) under grant no. 61375119, 61170057 and 60875080, and partially supported by National Key Basic Research Development Plan (973 Plan) Project of China with grant no. 2015CB352300.

REFERENCES

[1] Y. Tan and Y. Zhu, "Fireworks algorithm for optimization," in *Advances in Swarm Intelligence*, pp. 355–364, Springer, 2010.

[2] S. Bureerat, "Hybrid population-based incremental learning using real codes," in *Learning and Intelligent Optimization*, pp. 379–391, Springer, 2011.

[3] S. Zheng, A. Janecek, and Y. Tan, "Enhanced fireworks algorithm," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*, pp. 2069–2077, IEEE, 2013.

[4] S. Zheng, A. Janecek, J. Li, and Y. Tan, "Dynamic search in fireworks algorithm," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pp. 3222–3229, IEEE, 2014.

[5] J. Liu, S. Zheng, and Y. Tan, "The improvement on controlling exploration and exploitation of firework algorithm," in *Advances in swarm intelligence*, pp. 11–23, Springer, 2013.

[6] C. Yu, L. Kelley, S. Zheng, and Y. Tan, "Fireworks algorithm with differential mutation for solving the cec 2014 competition problems," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pp. 3238–3245, IEEE, 2014.

[7] C. Yu, J. Li, and Y. Tan, "Improve enhanced fireworks algorithm with differential mutation," in *Systems, Man and Cybernetics (SMC), 2014 IEEE International Conference on*, pp. 264–269, IEEE, 2014.

[8] J. Li, S. Zheng, and Y. Tan, "Adaptive fireworks algorithm," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pp. 3214–3221, IEEE, 2014.

[9] Y. Zheng, X. Xu, H. Ling, and S. Chen, "A hybrid fireworks optimization method with differential evolution operators," *Neurocomputing*, 2012.

[10] B. Zhang, M.-X. Zhang, and Y.-J. Zheng, "A hybrid biogeography-based optimization and fireworks algorithm," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pp. 3200–3206, IEEE, 2014.

[11] J. Liu, S. Zheng, and Y. Tan, "Analysis on global convergence and time complexity of fireworks algorithm," in *Evolutionary Computation (CEC), 2014 IEEE Congress on*, pp. 3207–3213, IEEE, 2014.

[12] Y. Tan, C. Yu, S. Zheng, and K. Ding, "Introduction to fireworks algorithm," *International Journal of Swarm Intelligence Research (I-JSIR)*, vol. 4, no. 4, pp. 39–70, 2013.

[13] Y. J. Zheng, Q. Song, and S. Y. Chen, "Multiobjective fireworks optimization for variable-rate fertilization in oil crop production," *Applied Soft Computing*, vol. 13, no. 11, pp. 4253–4263, 2013.

[14] K. Ding, S. Zheng, and Y. Tan, "A gpu-based parallel fireworks algorithm for optimization," in *Proceeding of the fifteenth annual conference on Genetic and evolutionary computation conference*, pp. 9–16, ACM, 2013.

[15] H. Gao and M. Diao, "Cultural firework algorithm and its application for digital filters design," *International Journal of Modelling, Identification and Control*, vol. 14, no. 4, pp. 324–331, 2011.

[16] A. Janecek and Y. Tan, "Iterative improvement of the multiplicative update nmf algorithm using nature-inspired optimization," in *Natural Computation (ICNC), 2011 Seventh International Conference on*, vol. 3, pp. 1668–1672, IEEE, 2011.

[17] A. Janecek and Y. Tan, "Swarm intelligence for non-negative matrix factorization," *International Journal of Swarm Intelligence Research (IJSIR)*, vol. 2, no. 4, pp. 12–34, 2011.

[18] A. Janecek and Y. Tan, "Using population based algorithms for initializing nonnegative matrix factorization," in *Advances in Swarm Intelligence*, pp. 307–316, Springer, 2011.

[19] W. He, G. Mi, and Y. Tan, "Parameter optimization of local-concentration model for spam detection by using fireworks algorithm," in *Advances in Swarm Intelligence*, pp. 439–450, Springer, 2013.

[20] C. Yu and Y. Tan, "Fireworks algorithm with covariance mutation," in *Evolutionary Computation (CEC), 2015 IEEE Congress on*, IEEE, 2015. (Under review).

[21] J. Liang, B. Qu, P. Suganthan, and Q. Chen, "Problem definitions and evaluation criteria for the CEC 2015 competition on learning-based real-parameter single objective optimization," 2014.