Orienting Mutation Based Fireworks Algorithm

Junzhi Li and Ying Tan
Key Laboratory of Machine Perception (Ministry of Education), Department of Machine Intelligence,
School of Electronics Engineering and Computer Science, Peking University, Beijing, 100871, P.R. China
Email:{ljz,ytan}@pku.edu.cn

***Abstract*—In this paper, a novel orienting mutation operator is designed to improve the performance of the fireworks algorithm, which is a recently proposed swarm intelligence algorithm for optimization. For each firework, the orienting mutation operator creates a new promising solution by adding to the firework a proper step size towards the local minimal point. By making use of the ready-made information of the optimization function, the orienting mutation operator enhances the local search ability of the algorithm. Its principles are analyzed and its effect is tested experimentally to show that it is a significant improvement.**

I. Introduction

The fireworks algorithm (FWA) [1] is a newly proposed swarm intelligence algorithm for optimization problems. It has proven to be a very competitive algorithm in optimization and other applications. [2]–[20].

There are three main operators in FWA: explosion, mutation and selection. Many research works have been conducted in order to improve these operators. However, with more and more studies on the mutation operator, its contribution is revealed to be not as much as it's designed and expected to be, especially since the performance of the algorithm has been greatly improved [16].

So far, there have been three kinds of mutation operators proposed in FWA and its variants. In the conventional FWA [1], a Gaussian mutation operator was adopted for the first time. Then in the enhanced fireworks algorithm (EFWA) [10], the Gaussian mutation operator in FWA was thoroughly analyzed and modified to conquer its drawbacks. But in a most recent version of FWA—dynamic search fireworks algorithm (dynFWA) [16], the new Gaussian mutation operator was also regarded as not effective and was thus removed. Besides, in [20], the authors adopted another kind of mutation operator called differential mutation in EFWA. However, the framework used in [20] was totally different from that of the conventional FWA, and it's more a hybrid algorithm of differential evolution and FWA than an improvement of FWA itself, so it will not be discussed in this paper.

Despite the fact that previous mutation operators are not satisfying in the state-of-the-art FWA versions, the initial idea of introducing a mutation operator to increase the diversity of the population is believed to be reasonable and necessary. In FWA, the individuals are all sampled within a certain range around the fireworks, so the diversity of the population is very limited without mutation.

The reasons why the previous mutation operators in FWAs gradually lose their effect with the algorithm developing are various, but they have two common drawbacks in their design: 1) they don't make full use of the information provided by other operators in the algorithm and 2) they have no clear purpose, other than improving the diversity. In this paper, a new kind of orienting mutation (OM) operator is designed for further developments of FWA. The OM operator uses the information provided by the explosion operator to learn an accurate direction and a proper step size and adds very promising solutions to the population to improve the diversity and quality. Through analyzing its principles theoretically, it's proved that the direction of the OM operator is close to the direction of the minus gradient and its step size adjusts itself according to the distance from the local minimal point. Experimental results verified that the new operator helps the algorithm to conduct local search and significantly improves the overall performance of dynFWA.

The rest of this paper is organized as follows: The framework of FWA and the operators used in dynFWA are introduced in Section II and III respectively. The analyses of the previous mutation operators in FWA and its variants are given in Section IV. The orienting mutation operator is proposed and analyzed theoretically in Section V. Experimental results are presented in Section VI. Finally, Section VII concludes the paper.

II. General Framework of FWA

Assume without loss of generality that the algorithm is dealing with a minimization problem:

$$\min_{\mathbf{x}} f(\mathbf{x}), \tag{1}$$

where $\mathbf{x}$ is a vector in the d dimensional Euclidean space. That is, a solution with smaller evaluation value (fitness value) is better.

Like other evolutionary algorithms, FWA searches in the feasible space for better solutions by keeping the iteration of generating and selection. It consists of four main steps:

1) Initialization:
   Set the constant parameters and initialize the fireworks randomly in the feasible space.
2) Explosion:
   Each firework "explodes" and generates a certain number of explosion sparks within a certain range (explosion amplitude), which is the key step in the algorithm. The numbers of explosion sparks and the explosion amplitudes are calculated according to the qualities of the fireworks. The principle of the calculation is to make the better fireworks explode in smaller ranges and generate more sparks in order to conduct local search and the worse fireworks explode

978-1-4799-7492-4/15/$31.00

in larger ranges and generate less sparks to conduct global search.

3) Mutation:
Some mutation sparks are generated to increase the diversity of the population. This step was eliminated in dynFWA.

4) Selection:
Fireworks of the new generation are selected from the candidates including the current fireworks, explosion sparks and mutation sparks.

Besides, if the optimization problem is constrained, there is a mapping operator to map the out-of-bound sparks back into the feasible space. But the mapping operator is not to be discussed in this paper.

III. Dynamic Search FWA

In the following, we will give the explosion and selection operators of dynFWA in detail, as the mutation operator will be discussed in Section IV.

For each firework $\mathbf{X}_i$, its explosion sparks' number is calculated as follows:

$$\lambda_i = \hat{\lambda} \cdot \frac{\max_j(f(\mathbf{X}_j)) - f(\mathbf{X}_i)}{\sum_j (\max_j(f(\mathbf{X}_j)) - f(\mathbf{X}_j))}, \tag{2}$$

where $\hat{\lambda}$ is a constant parameter controlling the general number of sparks, which can be approximately seen as the total number of explosion sparks in each generation. Moreover, in order to avoid the overwhelming effects of good fireworks, the number is bounded by:

$$\lambda_i = \begin{cases} \mathrm{round}(\alpha\hat{\lambda}) & \lambda_i < \alpha\hat{\lambda} \\ \mathrm{round}(\beta\hat{\lambda}) & \lambda_i > \beta\hat{\lambda} \\ \mathrm{round}(\lambda_i) & \text{otherwise} \end{cases}, \tag{3}$$

where $\mathrm{round}(x)$ is the closest integer to x, α and β are two constant parameters controlling the lower and upper bounds of the sparks' number respectively.

It's obvious that, the smaller (better) the firework's fitness value is, the more sparks it can generate, and vice versa.

In each generation, the firework with the best fitness is called the core firework (CF):

$$\mathbf{X}_{\mathrm{CF}} = \arg\min_{\mathbf{X}_i}(f(\mathbf{X}_i)). \tag{4}$$

In dynFWA, the other fireworks' explosion amplitudes are calculated just as in the previous versions of FWA:

$$A_i = \hat{A} \cdot \frac{f(\mathbf{X}_i) - f(\mathbf{X}_{\mathrm{CF}})}{\sum_j (f(\mathbf{X}_j) - f(\mathbf{X}_{\mathrm{CF}}))}, \tag{5}$$

where $\hat{A}$ is a constant parameter controlling the amplitudes generally. That is, the better (smaller) the firework's fitness value is, the smaller its explosion amplitude is.

But for the CF, its explosion amplitude is adjusted according to the search results in the last generation.

$$A_{\mathrm{CF}}(t) = \begin{cases} A_{\mathrm{CF}}(1) & t = 1 \\ C_r A_{\mathrm{CF}}(t-1) & f(\mathbf{X}_{\mathrm{CF}}(t)) = f(\mathbf{X}_{\mathrm{CF}}(t-1)) \\ C_a A_{\mathrm{CF}}(t-1) & f(\mathbf{X}_{\mathrm{CF}}(t)) < f(\mathbf{X}_{\mathrm{CF}}(t-1)) \end{cases}, \tag{6}$$

where $A_{\mathrm{CF}}(t)$ is the explosion amplitude of the CF in generation t. In the first generation, the CF is the best among all the randomly initialized fireworks, and its amplitude is preset to a constant number which is usually the diameter of the feasible space. After that, if in generation $t-1$, the algorithm found a better solution than the best in generation $t-2$, the amplitude of CF will be multiplied by an amplification coefficient $C_a > 1$, otherwise it will be multiplied by a reduction coefficient $C_r < 1$. Readers may be confused with the right hand conditions in (6) as the selection operator has not been introduced yet, so it should be noted here that the best solution in generation $t-1$ is always kept in generation t as the CF. The main idea of this dynamic explosion amplitude is described as follows: if in one generation no better solution is found, that means the amplitude is too big (radical) and thus need to be reduced to increase the probability of finding a better solution, and otherwise it may be too small (conservative) and cannot make the largest progress and thus need to be amplified. By the dynamic control, the algorithm can keep the amplitude within a relatively proper range.

Algorithm 1 shows how the sparks are generated for each firework. For each firework, its sparks are generated with a uniform distribution within a hypercube around the firework. Inspired from the mutation rate in genetic algorithm (GA), only about half of the sparks' dimensions are different from the firework.

Algorithm 1 Generating sparks for $\mathbf{X}_i$

Require: Firework's location $\mathbf{X}_i$, Explosion amplitude A_i and the number of explosion sparks λ_i

```
for j = 1 to λ_i do
    for each dimension k = 1, 2, ...d do
        sample κ from U(0, 1)
        if κ < 0.5 then
            sample η from U(−1, 1)
            s_ij^(k) ← X_i^(k) + η · A_i
        else
            s_ij^(k) ← X_i^(k)
        end if
    end for
end for
return all the s_ij
```

The selection operator used in dynFWA is called Elitism-Random Selection. To select the fireworks for the next generation, the best individual in the population (including the current fireworks, explosion sparks and mutation sparks) is firstly selected (as CF), while the rest of fireworks are selected uniformly randomly from the rest of the candidates.

In summary, Algorithm 2 is the general framework of dynamic search Fireworks Algorithm. Again, the mutation operator is removed.

IV. Analysis of the Mutation Operators in FWAs

There are two kinds of mutation operators proposed in FWA and EFWA. They are both referred to as Gaussian mutation (GM). In the conventional fireworks algorithm (FWA) [1], the Gaussian Mutation was designed to increase the diversity

Algorithm 2 Dynamic Search Fireworks Algorithm

randomly initialize μ fireworks in the potential space
evaluate the fireworks' fitness
repeat
  calculate λ_i according to Eq.(2) and Eq.(3)
  calculate A_i according to Eq.(5) and Eq.(6)
  for each firework, generate λ_i sparks within amplitude A_i according to Algorithm 1
  ~~generate mutation sparks~~
  evaluate all the sparks' fitness
  keep the best individual as a firework
  randomly choose other $\mu - 1$ fireworks among the rest of individuals
until termination criteria is met
return the best individual and its fitness

of the population. Later in the Enhanced Fireworks Algorithm (EFWA) [10], the authors claimed that the Gaussian mutation operator in FWA was unreasonably designed and misleading, so they modified it and proposed a new version of Gaussian mutation in EFWA.

Algorithm 3 shows how the GM sparks are generated in FWA.[1]

Algorithm 3 Generating GM Sparks in FWA

Require: The number of GM sparks ω and fireworks' locations
for $j \leftarrow 1$ **to** ω **do**
  randomly select a firework $\mathbf{X}_i$
  randomly generate a natural number $n < d$
  randomly generate a d dimensional 0-1 vector $\mathbf{v}$ with n ones
  sample η from $\mathcal{N}(1, 1)$
  for each dimension $k = 1, 2, ...d$ **do**
    if $\mathbf{v}_k == 1$ **then**
      $\mathbf{M}_j{}^{(k)} \leftarrow \eta \mathbf{X}_i^{(k)}$
    else
      $\mathbf{M}_j{}^{(k)} \leftarrow \mathbf{X}_i{}^{(k)}$
    end if
  end for
end for
return all the $\mathbf{M}_j$

From Algorithm 3, we can see that the GM operator in FWA generates sparks along the direction from the firework toward the origin, which is not reasonable because in terms of a black-box optimization problem, there is nothing special about the origin. In the original paper of FWA [1], this GM operator works well (maybe too well) because all the benchmark functions' minimal points in the paper are located near the origin.

Algorithm 4 shows how the GM sparks are generated in EFWA.

[1]Readers should note that the dimension selection mechanism in FWA is slightly different from that of EFWA. Actually this is an unintentional error of EFWA for the authors wrongly thought these two ways are equivalent. It is just a detail in the algorithm, but it does have some influence on the performance. For more details, please refer to [21].

Algorithm 4 Generating the GM Sparks in EFWA

Require: The number of GM sparks ω and fireworks' locations
for $j \leftarrow 1$ **to** ω **do**
  randomly select a firework $\mathbf{X}_i$
  sample η from $\mathcal{N}(0, 1)$
  for each dimension $k = 1, 2, ...d$ **do**
    sample κ from $\mathcal{U}(0, 1)$
    if $\kappa < 0.5$ **then**
      $\mathbf{M}_j{}^{(k)} \leftarrow \mathbf{X}_i{}^{(k)} + \eta \cdot (\mathbf{X}_{\mathrm{CF}}^{(k)} - \mathbf{X}_i{}^{(k)})$
    else
      $\mathbf{M}_j{}^{(k)} \leftarrow \mathbf{X}_i{}^{(k)}$
    end if
  end for
end for
return all the $\mathbf{M}_j$

In EFWA, the GM operator is different, where the spark is located on the line between the firework and the CF. It is far more reasonable than that of FWA, because the location of the CF means the best location found by the algorithm so far. However, the authors didn't study how much such an operator contributes to the algorithm. In [16], the authors made an experiment on the CEC2013 benchmark suite [22] consisting of 28 functions, and it turned out that EFWA without GM operator performs better than EFWA on 16 functions. As for dynFWA, there is almost no difference in performance after eliminating the GM operator.

In summary, both of the mutation operators in FWA variants lack a reasonable and clear target or purpose. In addition, in both of the mutation operators, the abundant information generated by the algorithm is not made use of.

In FWA, all the explosion sparks are generated in the neighbourhoods of the fireworks, so the diversity of the population is very limited, especially for the fireworks with small explosion amplitudes. A more reasonable and efficient mutation operator is necessary, but so far, neither of the mutation operators in FWA and EFWA is satisfying. Eliminating it is easy, but inventing another is not.

V. Orienting Mutation Operator

When a number of explosion sparks are generated by a firework, their evaluation values are of much information about the local area of the evaluation function. It's regretful that they were not made full use of in the previous versions of FWA. In this section, on the contrary, we are going to propose an orienting mutation (OM) operator for enhancing the local search capability and diversity of FWA by making full use of these information. For each firework, the OM operator generates an orienting mutation spark through shifting the firework by a direction towards the local optimal point. Actually, in the OM operator, two things are to be learned from these sparks: the estimator of the (minus) gradient's direction, which is the main reason why it is called orienting mutation operator, and a proper step size.

A. Principle

The position of the OM spark $\mathbf{M}_i$ for firework $\mathbf{X}_i$ is determined by Algorithm 5.

Algorithm 5 Generating Orienting Mutation Sparks for $\mathbf{X}_i$

Require: $\mathbf{X}_i$, $\mathbf{s}_{ij}$,$f(\mathbf{s}_{ij})$,λ_i and a parameter σ

1: sort the sparks by their fitness values $f(\mathbf{s}_{ij})$ in the ascending order

2: $\mathbf{\Delta}_i \leftarrow \frac{1}{\sigma\lambda_i}(\sum_{j=1}^{\sigma\lambda_i} \mathbf{s}_{ij} - \sum_{j=\lambda_i-\sigma\lambda_i+1}^{\lambda_i} \mathbf{s}_{ij})$

3: $\mathbf{M}_i \leftarrow \mathbf{X}_i + \mathbf{\Delta}_i$

4: **return** $\mathbf{M}_i$

Notes:

1) If $\sigma\lambda_i$ is not an integer, use $\lceil \sigma\lambda_i \rceil$ instead[2].
2) Unlike other mutation operators (in Section IV), there is no dimension selection mechanism in OM, because it will make the direction unreasonable.
3) Actually in line 1, it's not necessary to sort all the explosion sparks. Only the top and bottom $\sigma\lambda$ explosion sparks are needed, especially when λ is very large.
4) For each firework, it generates only one OM spark, which means the total number of mutation sparks is equal to the number of fireworks and is no longer a free parameter.
5) For each firework, the algorithm only requires its own sparks, because only neighbourhood sparks can reflect the property of the evaluation function locally.

In Algorithm 5, $\mathbf{\Delta}_i$ is the difference between the top σ of the sparks' mean location and the bottom σ of the sparks' mean location. It is expected that the direction of $\mathbf{\Delta}_i$ is a direction from a bad location to a good location, and is close to the direction of the minus gradient (at which the evaluation function decreases fastest) at $\mathbf{X}_i$. In the OM operator, the firework not only walks towards the good sparks, but also learn lessons from the bad sparks and walks backwards them.

Why don't we just use the best explosion spark and the worst explosion spark?

Firstly, it's expected that by using the best and worst populations, their unrelated values will be cancelled out. Most of the dimensions of the best explosion spark are good, but the rest of them are not, which means to learn from the sole best individual is to learn both the good and bad parts of it. While, learning from the good population is another thing, as long as the algorithm learns the common qualities of them. Except for their common qualities, the other information of them are not related to their good behaviors, and can be regarded as random noises to be cancelled out. This also holds for the bad explosion sparks. In the following of this section, we will show that by using the information of the population, the learned direction will be more stable and accurate.

The second main concern is the step size. If the local minimal point of the evaluation function is out of the explosion amplitude of the firework, we wish to create a mutation spark which can lead the firework and accelerate the search process. While if the local minimal point is already within the explosion amplitude, we don't want the step size to be too long and can not contribute to the search. So, the step size should be adjustable, according to the distance from the local minimal point. In the following in this section, we will also show that by using the information of the population, the step size does change with the distance automatically.

[2] $\lceil x \rceil$ is the smallest integer larger than x.

B. Accuracy of the Direction

For the convenience of our discussion, the dimension selection mechanism (line 3,4,7,8 and 9 in Algorithm 1) is not considered in the following of this section. The dimension selection mechanism has influences on the performance of FWAs, but it won't change the following essential propositions.

Assume[3] $f(\mathbf{x}) = x_1^2$, and the algorithm only adopts one firework $\mathbf{X}$ (thus the subscript i is omitted) with $\mathbf{X}^{(1)} > 0$. That is, at the firework's location, the evaluation function is most sensitive on direction $[-1, 0, 0, \ldots]^{\mathrm{T}}$ (i.e., the minus gradient), and is completely unsensitive on direction $[0, \ldots]^{\mathrm{T}}$. While on other directions, for example $[-1, -1, \ldots]^{\mathrm{T}}$, it's partially sensitive.

As shown in Fig.1, the OM sparks are always located near the direction of $[-1, 0, 0, \ldots]^{\mathrm{T}}$, which is the direction of gradient. The noises are cancelled out, but the useful direction is reserved. The noises on these dimensions cancel each other out and make the $\mathbf{\Delta}$ more close to the real gradient. Actually this is an application and example of the law of large numbers [23]. Only with the information of the population can the algorithm get an accurate direction.

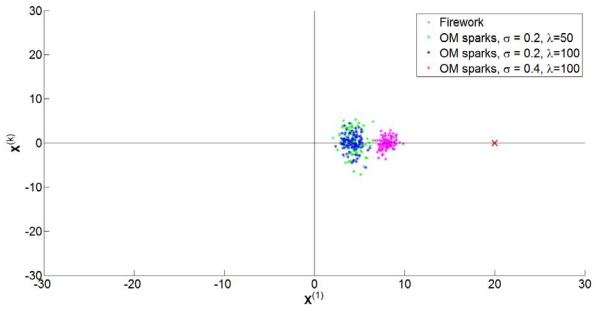

Fig. 1: $\mathbf{X}^{(1)} = 20, A = 10$, repeated 100 times for each set of parameters, the OM sparks are always located near the $\mathbf{X}^{(1)}$-axis. The larger $\sigma\lambda$, the more concentrated.

Now we take a closer look at how long $\mathbf{\Delta}$ will be at the most sensitive direction. There are two cases: $\mathbf{X}^{(1)} > A > 0$ or $A > \mathbf{X}^{(1)} > 0$.

C. Moving Stage

When the local minimum is out of the explosion amplitude of the firework, the search stage is called the moving stage. At this stage, we want the OM operator to explore at the promising direction and accelerate the search process.

[3] Here x_1 means the first dimension of $\mathbf{x}$. While for the notations in the algorithms, such as $\mathbf{X}$ and $\mathbf{\Delta}$, the subscript is the index of the firework, and the subscript with parentheses is the index of dimension.

Actually, the good and bad sparks are separate, so their distance, namely the step size of the orienting mutation, is long. So the OM spark does help to accelerate the moving process by exploring the promising direction. When the firework is still walking towards the local minimum point, $\mathbf{\Delta}$ will be long and the OM spark will be most probably located in the front of the explosion sparks and lead the population towards the local minimum and thus accelerate the moving process, as shown in Fig. 2.

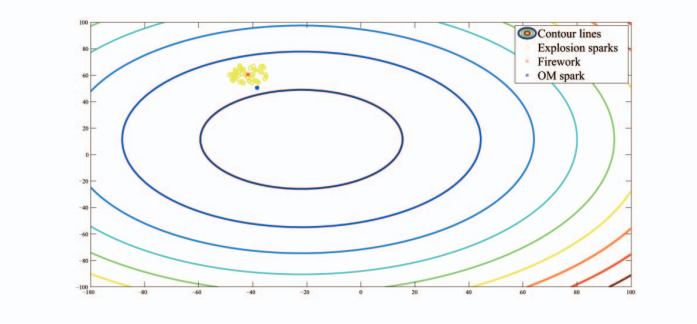

Fig. 2: OM at Moving Stage

D. Fine-tuning Stage

When the local minimum is already within the explosion amplitude of the firework, the search stage is called the fine-tuning stage. At this stage, it's ideal if the step size of the mutation spark gradually decrease and meanwhile the direction is still kept accurate to search more precisely together with other explosion sparks. And this is exactly what it does at this stage.

If $A > \mathbf{X}^{(1)} > 0$, it's obvious that $\mathbf{\Delta}^{(1)}$ is not as long as in the former case, because 1) the best sparks are not located near an end of the amplitude and 2) the worst sparks may be located near both ends and thus cancel each other out. Actually it can also be inferred from the limiting case: when $\mathbf{X}^{(1)} \to 0$, $\mathbf{\Delta}^{(1)}$ also concentrates to 0.

As shown in Fig.3, if $\mathbf{X}^{(1)} > A$, the step size $\mathbf{\Delta}^{(1)}$ is not influenced by the distance and is comparatively large, but when $\mathbf{X}^{(1)} < A$ the step size gradually converges to zero.[4] The λ itself has nearly no influence on the expectation of the step size, but the stability increases with λ. While larger σ causes shorter step size and better stability. As for the direction, there is little chance for the algorithm to make a wrong way (i.e., $\mathbf{\Delta}^{(1)} > 0$), not until the distance $\frac{|\mathbf{X}^{(1)}|}{A}$ is very close to zero.

When the local optimal point is within the explosion amplitude of the firework, it means the firework has already approached the local minimum point, though the algorithm doesn't know where exactly it is. In this case, the good and bad sparks are mixed and the step size of the orienting mutation operator is short. The OM spark will be most probably located inside the explosion amplitude of the firework, and point out roughly the direction of the local minimum point, and meanwhile the length of $\mathbf{\Delta}$ will be small, as shown in Fig. 4. In this case, the OM spark works like an explosion spark, but the expected contribution of the OM spark may be slightly larger than the average of explosion sparks, because at least it is most probably located on the correct direction due to the fact that $\mathbf{\Delta}$ converges faster on irrelevant or noisy directions.

[4]It may seem ideal if $\mathbf{\Delta}^{(1)}$ also increases with $\mathbf{X}^{(1)}$ even when $\mathbf{X}^{(1)} > A$, so that the algorithm can approach the local minimal point as fast as possible, but it's not. Even though the direction of $\mathbf{\Delta}$ is very accurate as we proved, it's accurate only in a small neighbourhood, because nearly no evaluation function has the same gradient everywhere. Still less, if the step size is too long, the little error of the direction will be amplified enormously. Remember that the OM operator at the moving stage can be considered as a kind of extrapolation, whose accuracy is always affected by the step size.

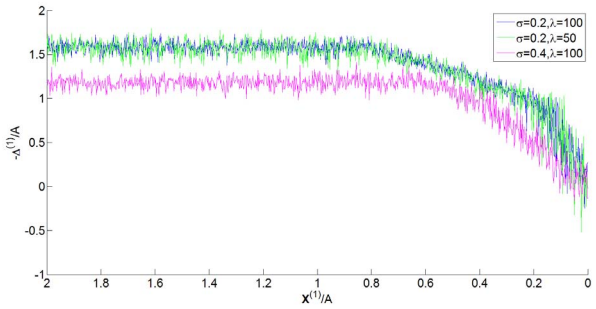

Fig. 3: The relationship between the distance to the local optimal point and the step size

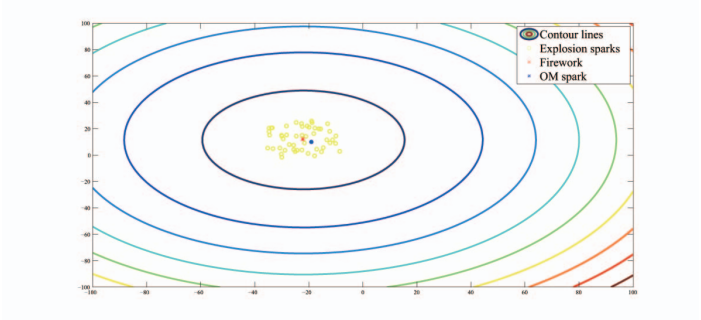

Fig. 4: OM at Fine-tuning Stage

E. How to Choose σ

Generally speaking, any $0 < \sigma \leqslant 0.5$ is possible, but $\sigma > 0.5$ is meaningless as some of the top sparks and the bottom sparks will cancel out.

Larger σ is good at learning the direction of the gradient, while smaller σ may result in an inaccurate direction, comparatively speaking. However, smaller σ makes the firework moves faster because the algorithm doesn't take into consideration theses not-so-good (bad) explosion sparks. In fact, when $\sigma \to 0.5$, $|E(\mathbf{\Delta}^{(1)})| \to A$ at moving stage, and the OM spark has little chance to be out of the explosion amplitude. In addition, smaller σ makes Algorithm 5 faster, especially when the dimensionality d is large. Note that the computational complexity of line 1 is $\Theta(\lambda_i \log \lambda_i)$ (or $\Theta(\lambda_i \log \sigma\lambda_i)$, only to find the top and bottom explosion sparks), but the computational complexity of line 2 is $\Theta(\sigma\lambda_i d)$.

Trading off among these influences, $\sigma = 0.2$ is recommended for most cases empirically.

F. Summary and Remarks

The OM operator is a population based algorithm. Generally speaking, the more explosion sparks it adopts, the better it works. It has the following properties:

1) The noises on the unsensitive dimensions are cancelled out and will concentrate to 0 with the increasing of σ and λ, and thus the direction of $\mathbf{\Delta}$ is very close to the minus gradient.
2) If the local minimum is out of the explosion amplitude of the firework, $\mathbf{\Delta}$ will be long at the sensitive direction and thus accelerate the moving.
3) If the local minimum is within the range of the explosion amplitude, $\mathbf{\Delta}$ will be short at the sensitive direction and thus contribute to the fine-tuning.

Compared with previous mutation operators, the principles of the OM operator are more theoretically sound and with clear purpose. OM sparks aim to the local minimal point rather than to the origin point or to a random direction. The step size they take change with the distance toward the minimal point rather than just taking any random number.

The proposed OM operator can fill in the blank of the dynFWA (Line 7, Alg 2). In the following, the dynFWA with OM operator is abbreviated as dynFWA-OM.

VI. Experiments

Experiments on the benchmark suite of CEC 2014 single objective competition [24] are conducted in order to measure how much the OM operator improves the dynFWA generally, especially on complex evaluation functions, because most of the functions in [24] are complex problems. In the experiments, the OM operator is directly added to the complete version of dynFWA and the parameters are all identical to [16], except the new parameter σ.

Experimental Setup: Dimensionality $d = 100$, run 51 times for each function and the maximum evaluation number is $10000d$. Platform: MATLAB R2014a. Parameters are set as follows: $\mu = 5, \hat{\lambda} = 150, \alpha = 0.8, \beta = 0.04, \hat{A} = 40, C_r = 0.9, C_a = 1.2, \sigma = 0.2$.

The mean errors of dynFWA and dynFWA-OM are shown in Table I. Besides, a set of t-tests is also conducted to show if their differences are significant or not. The significantly better results are highlighted. The column "Improved" indicates whether the mean error of dynFWA-OM is better.

Among these 30 functions, dynFWA-OM outperforms dynFWA on 19 of them, and 11 mean errors are significantly better while only 5 are significantly worse.

It may seem that the improvement is only significant on a few of the benchmark functions, but please note that in each iteration, the OM operator only add 5 more evaluations to about 150 evaluations in dynFWA. Hence it's quite difficult to achieve so many significantly better results unless the OM operator is truly efficient.

Table II shows the comparison of time complexities of dynFWA and dynFWA-OM on 100 dimensional functions, following the format in [24]. It can be seen that the OM operator doesn't add much time complexity to the dynFWA.

TABLE I: Mean Error Comparison between dynFWA and dynFWA-OM for 100 Dimensional Functions

Function	dynFWA	dynFWA-OM	Improved	p
1	6.7221E+06	6.5315E+06	T	6.1570E-01
2	2.5967E+04	2.2654E+04	T	9.2535E-01
3	**1.5114E-01**	2.0663E-01	F	5.1369E-04
4	2.1098E+02	2.1870E+02	F	4.3360E-01
5	2.0000E+01	2.0000E+01	F	1.4185E-01
6	9.7967E+01	**6.9997E+01**	T	7.9976E-13
7	3.1403E-03	**8.2090E-04**	T	4.5834E-03
8	5.4771E+02	5.3538E+02	T	3.2192E-01
9	7.0348E+02	**5.3362E+02**	T	3.7180E-09
10	1.2172E+04	1.2073E+04	T	8.4087E-01
11	1.3973E+04	1.4132E+04	F	6.0632E-01
12	9.2972E-01	**1.2496E-01**	T	2.7189E-16
13	7.4624E-01	**5.6599E-01**	T	3.8257E-13
14	**3.4682E-01**	3.8795E-01	F	1.1554E-10
15	5.9834E+01	**3.0183E+01**	T	5.3257E-17
16	4.5016E+01	4.4870E+01	T	4.4948E-01
17	**4.7575E+05**	5.4743E+05	F	3.0638E-02
18	4.2228E+03	**2.9372E+03**	T	4.9883E-02
19	**1.1743E+02**	1.2736E+02	F	1.4304E-02
20	1.1535E+03	1.1460E+03	T	1.0000E+00
21	2.3067E+05	2.7015E+05	F	2.9123E-02
22	2.5137E+03	2.4381E+03	T	4.7806E-01
23	3.5592E+02	**3.5412E+02**	T	2.7176E-04
24	4.1923E+02	**3.9471E+02**	T	6.3037E-18
25	**2.8266E+02**	2.9529E+02	F	5.5922E-03
26	2.0664E+02	**2.0043E+02**	T	1.1411E-06
27	2.8302E+03	**2.2912E+03**	T	2.2288E-10
28	1.0203E+04	9.8145E+03	T	1.3383E-01
29	3.3970E+04	1.8469E+07	F	7.1780E-01
30	6.8505E+04	7.0428E+04	F	7.3286E-01

TABLE II: Time Complexity

		dynFWA	dynFWA	dynFWA-OM	dynFWA-OM
T0	T1	T2	(T2-T1)/T0	T2	(T2-T1)/T0
0.1596	3.0633	4.1298	6.6836	4.6519	9.9559

VII. Conclusion

In this paper, we analyzed the drawbacks of the pervious mutation operators in FWA and EFWA, and proposed a new orienting mutation (OM) operator. The orienting mutation operator adds some new promising solutions to the population and increases the diversity. By analyses, it's revealed that the operator can 1) learn a promising direction towards the local minimal point of the evaluation function by making full use of the explosion sparks' information and 2) adjust the step size according to the distance from the local minimal point. Experimental results verified that by adding the operator to dynFWA, the overall performance is significantly improved.

Acknowledgment

Prof. Ying Tan is the corresponding author. This work was supported by the Natural Science Foundation of China (NSFC) under grant no. 61375119 and 61170057, 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, ser. Lecture Notes in Computer Science, Y. Tan, Y. Shi, and K. Tan, Eds. Springer Berlin Heidelberg, 2010, vol. 6145, pp. 355–364. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-13495-1_44

[2] 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, July 2014, pp. 3200–3206.

[3] 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. IEEE, 2011, pp. 1668–1672.

[4] ——, "Using population based algorithms for initializing nonnegative matrix factorization," in Proceedings of the second international conference on Advances in swarm intelligence, ser. ICSI'11. Springer-Verlag, 2011, pp. 307–316.

[5] ——, "Swarm intelligence for non-negative matrix factorization," International Journal of Swarm Intelligence Research (IJSIR), vol. 2, no. 4, pp. 12–34, 2011.

[6] Y. Zheng, X. Xu, and H. Ling, "A hybrid fireworks optimization method with differential evolution," Neurocomputing, 2012.

[7] Y. Pei, S. Zheng, Y. Tan, and T. Hideyuki, "An empirical study on influence of approximation approaches on enhancing fireworks algorithm," in Proceedings of the 2012 IEEE Congress on System, Man and Cybernetics. IEEE, 2012, pp. 1322–1327.

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

[9] 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.

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

[11] S. Zheng and Y. Tan, "A unified distance measure scheme for orientation coding in identification." in Information Science and Technology, 2013 IEEE Congress on. IEEE, 2013, pp. 979–985.

[12] 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, ser. GECCO 2013. New York, NY, USA: ACM, 2013, pp. 9–16. [Online]. Available: http://doi.acm.org/10.1145/2463372.2463377

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

[15] 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. Springer, 2013, pp. 439–450.

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

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

[18] N. Pholdee and S. Bureerat, "Comparative performance of meta-heuristic algorithms for mass minimisation of trusses with dynamic constraints," Advances in Engineering Software, vol. 75, pp. 1–13, 2014.

[19] A. Mohamed Imran and M. Kowsalya, "A new power system reconfiguration scheme for power loss minimization and voltage profile enhancement using fireworks algorithm," International Journal of Electrical Power & Energy Systems, vol. 62, pp. 312–322, 2014.

[20] C. Yu, J. Li, and Y. Tan, "Improve enhanced fireworks algorithm with differential mutation," 2014.

[21] http://www.cil.pku.edu.cn/research/fwa/activities/2014/announcements-of-efwa/index.html.

[22] J. Liang, B. Qu, P. Suganthan, and A. G. Hernández-Díaz, "Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization," 2013.

[23] http://en.wikipedia.org/wiki/Law_of_large_numbers.

[24] J. Liang, B. Qu, and P. Suganthan, "Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization," Technical Report 201311, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore, Tech. Rep., 2013.