Introduction to Fireworks Algorithm

Ying Tan, School of Electronics Engineering and Computer Science, Peking University, Beijing, China

Chao Yu, School of Electronics Engineering and Computer Science, Peking University, Beijing, China

Shaoqiu Zheng, School of Electronics Engineering and Computer Science, Peking University, Beijing, China

Ke Ding, School of Electronics Engineering and Computer Science, Peking University, Beijing, China

ABSTRACT

Inspired by fireworks explosion at night, conventional fireworks algorithm (FWA) was developed in 2010. Since then, several improvements and some applications were proposed to improve the efficiency of FWA. In this paper, the conventional fireworks algorithm is first summarized and reviewed and then three improved fireworks algorithms are provided. By changing the ways of calculating numbers and amplitudes of sparks in fireworks' explosion, the improved FWA algorithms become more reasonable and explainable. In addition, the multi-objective fireworks algorithm and the graphic processing unit (GPU) based fireworks algorithm are also presented, particularly the GPU based fireworks algorithm is able to speed up the optimization process considerably. Extensive experiments on 13 benchmark functions demonstrate that the three improved fireworks algorithms significantly increase the accuracy of found solutions, yet decrease the running time dramatically. At last, some applications of fireworks algorithm are briefly described, while its shortcomings and future research directions are identified.

Keywords: *Fireworks Algorithm, Improved Fireworks Algorithm, Multi-objective Fireworks Algorithm, Graphic Processing Unit, Function Optimization*

1 INTRODUCTION

In most engineering fields, many problems can be simplified as numerical optimization problems through mathematical modeling. In some of the problems – not only the optimal solution, but also multiple feasible solutions and viable localized optimal solutions need to be identified to provide enough information for decision makers. Such problems are generally referred to as multi-modal and multi-objective optimization problems. To solve those problems, the maximum or the minimum values of the functions need to be found out.

Traditional methods generally solve a continuous and differentiable function using mathematical techniques based on gradient information. However, when dealing with multi-modal and multi-objective optimization problems, traditional methods cannot always obtain even the reasonable solutions. In order to solve function optimization problems efficiently, many algorithms inspired by biological behavior are suggested recently.

The study of biological phenomena is no longer constrained in the biology discipline alone, but expanded to mathematics, computer science, information science and other research fields. Inspired by the behavior of groups of animals, many swarm intelligence algorithms are

designed in the field of computer science.

Swarm can be described as a number of individuals in adjacent areas and those individuals interact with each other. In nature, a bee, or an ant, or a bird can hardly survive without its kin. A group of organics, therefore, such as the aforementioned bees, ants or birds, has more chances to survive than the lone individual. The survival chance for a group is not a simple composition of each individual's chance, but a more complex summary of social and group dynamics. The character of animal groups can greatly help its individuals adapt to their environment. Each individual obtains information from social interaction and that information gained by an individual in a group is more than the information any single individual can obtain alone. Information is then transferred among the group and each individual processes this transferred information and change its own behavior, including its own behavioral patterns and norms. Therefore, the whole group has some capabilities and characteristics, especially the ability to adapt to their environment that a single individual can hardly gain when working alone. The ability of an individual to change according with environment is known as intelligence and this intelligence is gained by the clustering of individuals.

Inspired by nature, many swarm intelligence algorithms are proposed. Observing the way of ants finding food, ant colony optimization (ACO) algorithm was proposed by Colorni and his partners in 1991 (Colorni, Dorigo, & Maniezzo, 1991). Moreover, particle swarm optimization (PSO) algorithm was put forward by Kennedy and Eberhart in 1995 (Kennedy, & Eberhart, 1995). The algorithm mimics the pattern of birds flying to find food. Yet, differential evolution (DE) algorithm is another swarm intelligence algorithm, which was given by Storn and Price in 1995 (Storn, & Price, 1995). In this algorithm, the differences between individuals are fully utilized. The recently announced artificial bee colony algorithm (ABC) and fish school search algorithm (FSS), were proposed in 2008 and 2009 respectively (Karaboga, & Basturk, 2008; Filho, de Lima Neto, Lins, Nascimento, & Lima, 2009). The most recently proposed fireworks algorithm (FWA) is a swarm intelligence algorithm that was published by Tan and Zhu in 2010 (Tan, & Zhu, 2010). This algorithm is inspired by fireworks explosion at night and is quite effective at finding global optimal value. As a firework explodes, a shower of sparks is shown in the adjacent area. Those sparks will explode again and generate other shows of sparks in a smaller area. Gradually, the sparks will search the whole solution space in a fine structure and focus on a small place to find the optimal solution.

As a practical optimization algorithm, fireworks algorithm can fulfill three user requirements (Storn, & Price, 1997). First of all, FWA can process linear, non-linear and multi-model test functions. Secondly, FWA can be parallelized in order to deal with complex practical problems. Thirdly, FWA has good convergence properties and can always find the global minimization.

This article completely summarizes fireworks algorithm, including the conventional fireworks algorithm, its improvements and its applications. All algorithms are tested on standard datasets. The remainder of this article is organized as follows. Section II presents the conventional fireworks algorithm. Section III to section V describes three improvements by several researchers. Some experiments are designed and the experimental results are shown in section VI. Section VII states multi-objective fireworks algorithm and section VIII describes GPU-based fireworks algorithm. Applications of fireworks algorithm on solving optimization problems are given in section IX. In the last section X, conclusions and further research directions are drawn to enrich the research and enlarge the range of application of fireworks algorithm.

2 Fireworks Algorithm

After a firework exploded, the sparks are appeared around a location. The process of exploding can be treated as searching the neighbor area around a specific location. Inspired by fireworks in real world, fireworks algorithm (FWA) is proposed. Fireworks algorithm utilizes N D-dimensional parameter vectors x_i^G as a basic population in each generation. Parameter i varied from 1 to N and parameter G stands for the index of generations.

Every individual in the population 'explodes' and generates sparks around him/her. The number of sparks and the amplitude of each individual are determined by certain strategies. Furthermore, a Gaussian explosion is used to generate sparks to keep the diversity of the population. Finally, the algorithm keeps the best individual in the population and selects the rest $(N-1)$ individuals based on distance for next generation.

More specific strategies of fireworks algorithm can be described as follows.

2.1 Explosion Sparks Strategy

The explosion sparks strategy mimics the explosion of fireworks and is the core strategy in fireworks algorithm. When a spark blasts, the spark is vanished and many sparks appear around it. The explosion sparks strategy mimicking this phenomenon is used to produce new individuals by explosion.

In this strategy, two parameters need to be determined. The first one is the number of sparks.

$$S_i = \hat{S} \cdot \frac{Y_{max} - f(x_i) + \varepsilon}{\sum_{i=1}^{N}(Y_{max} - f(x_i)) + \varepsilon} \quad (1)$$

In the formula, S_i represents the number of sparks generated by an individual from the population, where i varies from 1 to N. As a controlling parameter of the total number of generated sparks, $\hat{S}$ is set as a constant. Suppose the goal is to find the minimal of a function. Variable Y_{max} stands for the worst fitness value in the current generation, while $f(x_i)$ is the fitness value for an individual x_i. The last parameter expressed as ε is used to prevent the denominator from becoming zero.

The second parameter in this strategy is the amplitude of sparks.

$$A_i = \hat{A} \cdot \frac{f(x_i) - Y_{min} + \varepsilon}{\sum_{i=1}^{N}(f(x_i) - Y_{min}) + \varepsilon} \quad (2)$$

Variable A_i gives the amplitude for an individual x_i to generate the explosion sparks and $\hat{A}$ is a constant to control the amplitudes. The best fitness value Y_{min} is used to calculate amplitudes. In this formula, the last parameter helps to avoid the error of having the denominator being zero. If an individual is close to the boundary, the generated sparks may lie out of the feasible space. Therefore, a mapping method is used to keep sparks inside of the feasible space.

2.2 Mapping Strategy

The mapping strategy ensures all the individuals stay in the feasible space. If there are some outlying sparks from the boundary, they will be mapped to their allowable scopes.

$$x_i = x_{min} + |x_i| \,\%\, (x_{max} - x_{min}), \quad (3)$$

where x_i represents the positions of any sparks that lie out of bounds, while x_{max} and x_{min} stand for the maximum and minimum boundary of a spark position. The symbol % stands for the modular arithmetic operation. Aside from the explosion sparks strategy, another way to generate sparks is proposed as Gaussian sparks strategy.

2.3 Gaussian Sparks Strategy

To keep the diversity of the population, Gaussian sparks strategy is used to generate sparks with Gaussian distribution. Suppose the position of current individual is stated as x_k^j, the Gaussian explosion sparks are calculated as

$$x_k^j = x_k^j \cdot g, \quad (4)$$

where g is a random number in Gaussian distribution.

$$g = Gaussian\,(1, 1) \quad (5)$$

Parameter g obeys the Gaussian distribution

with both mean value and standard deviation are 1. After normal explosions and Gaussian explosions, we consider a proper way to select individuals for next generation. Here, a distance based selection method is suggested.

2.4 Selection Strategy

To select the individuals for next generation, the best individual is always kept at first. Then the next $(N-1)$ individuals are selected based on their distance to other individuals. The individual that is far from other individuals gets more chance to be selected than those individuals with smaller distances to other individuals.

The general distance between two locations is calculated by

$$\mathrm{R}(x_i) = \sum_{j\in K} d(x_i, x_j) = \sum_{j\in K} \|x_i - x_j\|, \quad (6)$$

where location x_i and x_j (i≠j) can be any locations and K is the set of all current locations. For the distance measurements, many methods can be used, including Euclidean distance, Manhattan distance and Angle-based distance. Inspired by the immune density (Lu, Tan, & Zhao, 2002), Euclidean distance is used in the fireworks algorithm (Tan, & Zhu, 2010).

$$\mathrm{d}(x_i, x_j) = |f(x_i) - f(x_j)|, \quad (7)$$

where $\mathrm{f}(x_i)$ is the fitness for location x_i and $\mathrm{d}(x_i, x_j)$ represents the distance between two locations.

As last, a roulette wheel method is used to calculate the possibility of selecting the locations.

$$\mathrm{p}(x_i) = \frac{R(x_i)}{\sum_{j\in K} R(x_i)} \quad (8)$$

The individuals with larger distance from others have more chance to be selected. In this way, the diversity of a population can be ensured.

The flowchart and pseudo code for fireworks algorithm is stated as follows.

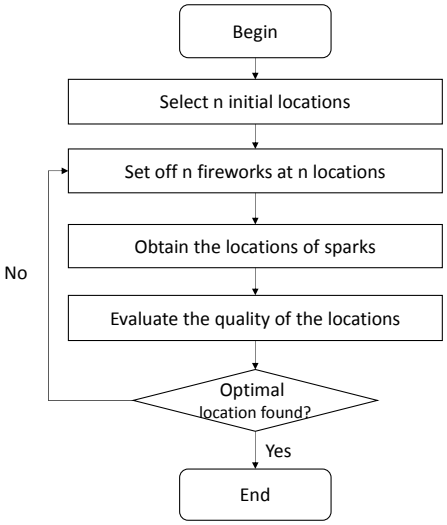


Figure 1. The flowchart of fireworks algorithm

Fireworks algorithm works quite well on following parameters, where n =5, m =50, a =0.04, b =0.8, $\hat{A}$ =40 and $\hat{m}$ =5. Although fireworks algorithm reaches great progress at several problems, there are still some places for improvement. Zheng et al. (Zheng, Janecek, & Tan, 2013) proposed an enhanced fireworks algorithm, which significantly increased the accuracy of result on test functions. Liu et al. (Liu, Zheng, & Tan, 2013) studied the exploration and exploitation abilities of fireworks algorithm and then designed a transfer function to calculate the number and the amplitude for sparks. Pei et al. (Pei, Zheng, Tan, & Takagi, 2012) presented an empirical study on the influence of fitting methods of fireworks algorithm. Other related references including but not limited to (Zheng, & Tan, 2013), (Zhou, &

Tan, 2009), (Zhou, & Tan, 2011), (Bureerat, 2011), (Lou, Li, Jin, & Li, 2012), (Lou, Li, Shi, & Jin, 2013) and (Tan, & Xiao, 2007).

```
for (i = 0; i < N; i++)
randomly generate an individual.
while ( count < gen_max) {
// gen_max is the maximum number of generation
// For each individual, generate Si sparks within
amplitude Ai.
for (i = 0; i < N; i++) {
Set z^k = round(rand(0, 1)), k = 1, 2, …, dimension
For each dimension of x_i^k
if(z^k == 1) x_i^k += A_i * rand(−1, 1);
if(x_i^k is out of scope)      execute mapping operation;
calculate fitness;
evaluation count += Si;
}
// Generate Gaussian Sparks
for (i = 0; i < m̂; i++) {
Set z^k = round(rand(0, 1)), k = 1, 2, …, dimension
For each dimension of x_i^k
if(z^k == 1) x_i^k *= Gaussian(1, 1);
if(x_i^k is out of bounds)     execute mapping operation;
calculate fitness;
evaluation count += m̂;
}
//Selection
Keep the best individual for next generation
Then select (N – 1) individuals
if(evaluation count > evals_max) break;
//evals_max is the maximum number of evaluations
}
```

3 Enhanced Fireworks Algorithm (EFWA)

To overcome the disadvantages of fireworks algorithm, many researchers have attempted in different ways to improve it. Zheng et al. (Zheng, Janecek, & Tan, 2013) proposed an enhanced fireworks algorithm through improvements it in the following five aspects.

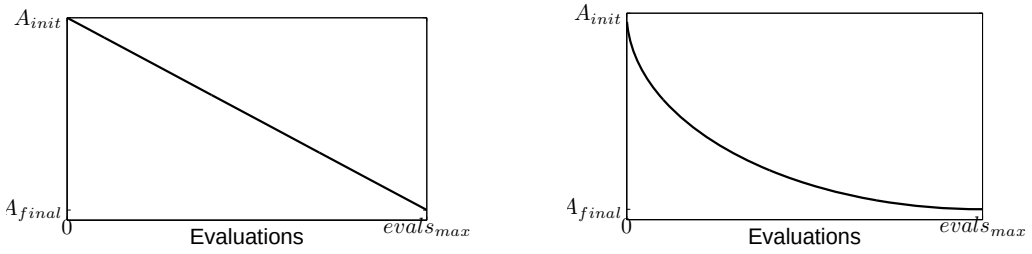
3.1 Minimal Explosion Amplitude Setting

In the evolution process, some explosion amplitude may be close to zero, which is not conducive to find the global best value. Since the explosion amplitude was closely related to fitness values, two ways to limit the minimum amplitude boundary were proposed. One way is based on a linear function and the other is based on a non-linear function.

$$A_{\min}^{k}(t) = A_{init} - \frac{A_{init} - A_{final}}{evals\_max} * t \qquad (9)$$

$$A_{\min}^{k}(t) = A_{init} - \frac{A_{init} - A_{final}}{evals\_max} * \sqrt{(2 * evals\_\max - t) * t} \qquad (10)$$

In both formulae, $A_{min}^{k}(t)$ means the lower boundary for an individual in the k dimension when the function is evaluated t times. The two new parameters A_{init} and A_{final} stands for the initiate and final amplitudes. The last parameter is the maximum evaluation times, which is expressed as $evals\_max$. The schematic diagrams for linear and non-linear minimal explosion amplitudes are drawn below.

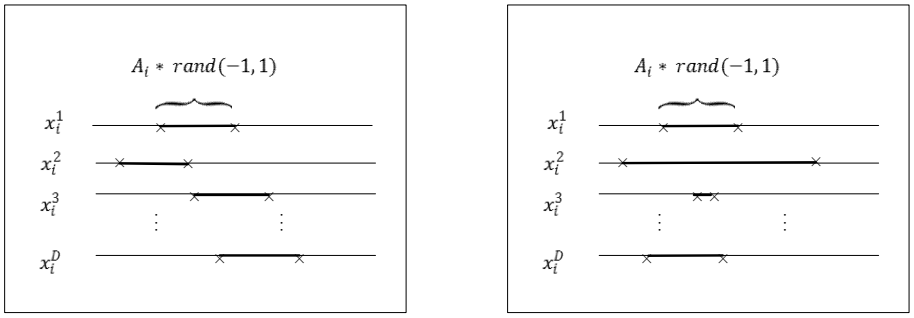


(a) Linear decrease (b) Non-linear decrease

Figure 2. The schematic diagrams of minimal amplitude for linear and non-linear decreases. This figure is obtained from the work of (Zheng, Janecek, & Tan, 2013).

3.2 Explosion Sparks Strategy

In the fireworks algorithm, the same increment will be added to some selected dimensions of an individual.



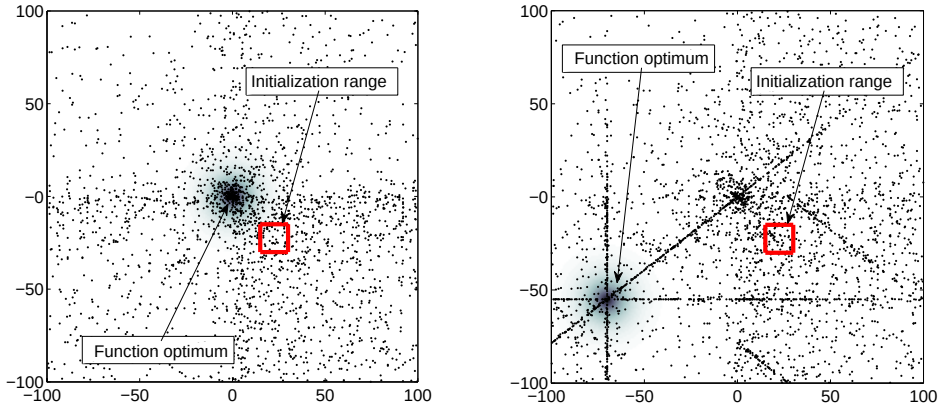
(a) FWA (b) EFWA

Figure 3. Increment of FWA and EFWA in each selected dimension

As it is shown in the following figure, the same increment may cause a loss of diversity to a population. Hence, it is necessary to generate different increments and add the increments to each selected dimension for an individual to obtain the diversity of population. In the above figure, $x_i^j(\mathrm{j} = 1, 2, \ldots, \mathrm{Dimension})$ stands for the value in the j[th] dimension of the i[th] individual. A_i is the amplitude for that individual x_i.

3.3 Gaussian Sparks Strategy

The fireworks algorithm works significantly well on functions that will reach their optimal at the origin of coordinate. For example, the optimal value of a two-dimensional Ackley function lies at the origin of its coordinate. But if the function is shifted, e.g. the optimal value is shifted to [-70, -55], the fireworks algorithm performs badly. The following figure shows the location of the Gaussian sparks in fireworks algorithm. It can be seen that Gaussian sparks can easily find the optimal value at the origin of coordinate when the function is not shifted. But Gaussian sparks work poorly on the shifted function.



(a) Optimal at origin (b) Shifted optimal value

Figure 4. Effect of the Gaussian sparks. This figure comes from the work of (Zheng, Janecek, & Tan, 2013).

To overcome the disadvantage of Gaussian sparks, Zheng et al. (Zheng, Janecek, & Tan, 2013) used another way to generate Gaussian sparks. Referring the position of the current global best individual, the Gaussian sparks are generated by

$$x_i^k = x_i^k + \left(x_{Best}^k - x_i^k\right) * g, \qquad (11)$$

where g is a random number obeyed Gaussian distribution, i.e.,

$$g = Gaussian\,(0, 1)\ . \qquad (12)$$

In the formula, x_i^k stands for the selected individual to generate Gaussian sparks and x_{Best}^k is the best individual the algorithm has find out so far. Parameter g obeys the Gaussian

distribution of mean 0 and standard deviation 1.

3.4 Mapping Strategy

The proposed fireworks algorithm used modular arithmetic operation to map individuals back into scope. However, modular arithmetic operation is time consuming. Besides, some of the individuals are mapped to a place near the origin, straying from the diversity of population. For example, suppose the solution space varies from -20 to 20. If there is an individual who has a value of -21, then it maps to 1 according to the formula suggested in fireworks algorithm. Hence, a new mapping operator is proposed.

$x_i^k = x_{min}^k + rand(0,1) * (x_{max}^k - x_{min}^k)$, (13)

where x_{max}^k and x_{min}^k are the lower and upper boundary of the solution space.

3.5 Selection Strategy

The most time consuming part of conventional fireworks algorithm lies in the selection. In the selection strategy of conventional fireworks algorithm, the distances between individuals need to be calculated. Hence, the computational complexity of selection strategy is much higher than random selection strategy.

The selection operation is called as Elitism Random Selection (ERS). According to the work of Pei et al. (Pei, Zheng, Tan, & Takagi, 2012), the best individual is always preserved for next generation, while the other $(N-1)$ individuals are selected randomly. In this way, the running time for fireworks algorithm is largely decreased and furthermore, the computational complexity is linear.

4 Improved Fireworks Algorithm (IFWA) with Two Different Selection Methods

Liu et al. (Liu, Zheng, & Tan, 2013) put forward another effective improvement of fireworks algorithm. The individuals are sorted by their fitness values in increasing order and two new formulae are given concerning the number and the amplitude of sparks.

$$S_i = \hat{S} \cdot \frac{t(i)}{\sum_{i=1}^{N} t(i)}, \quad (14)$$

$$A_i = \hat{A} \cdot \frac{t(N-i+1)}{\sum_{i=1}^{N} t(N-i+1)}, \quad (15)$$

where t(i) is a transfer function with a

$$\mathrm{t(i)} = \frac{1}{1+e^{\frac{(i-1)}{a}}}. \quad (16)$$

Transfer function helps to decrease the number and the amplitude of sparks evenly. Parameter a varies from 20 to 1 with an even number of distributions for each generated value. $\hat{S}$ and $\hat{A}$ are set as constant parameters controlling of the total number and the maximum amplitude of sparks respectively. N stands for the number of individuals, while S_i and A_i are the number and the scope for an individual x_i to generate the explosion sparks.

A random function is proposed to replace the function to generate Gaussian sparks.

$x_i^k = x_{min}^k + \left(x_{max}^k - x_{min}^k\right) * rand(0,1)$ (17)

The function randomly generates individuals and ensures the generated individuals are in the feasible space. x_{min}^k and x_{max}^k are the maximum and minimum boundaries of the k^{th} dimension.

Two different selection methods are proposed, named as the best fitness selection and roulette fitness selection.

A. Best Fitness Selection (BFS)

The best fitness selection is first proposed by Zheng et al. (Zheng, Janecek, & Tan, 2013). The best N individuals are selected for next generation from both the basic individuals and the generated sparks.

B. Roulette Fitness Selection (RFS)

After the algorithm selects the best individual, other individuals are selected by roulette based on their fitness values. The probability for each individual to be selected is calculated as

$$\mathrm{p}(x_i) = \frac{Y_{max} - f(x_i)}{\sum_{i=1}^{K}(Y_{max} - f(x_i))}, \quad (18)$$

where Y_{max} stands for the worst fitness value in the population, while $\mathrm{f}(x_i)$ is the fitness for an individual x_i. Parameter K means the total number of individuals, including basic individuals, explosion sparks and Gaussian

sparks. It can be seen that the individuals with lower fitness value have more chance to be selected.

5 The Influence of Approximation Approaches on Enhanced Fireworks Algorithm

To figure out the influence of sampling methods and fitting methods on enhanced fireworks algorithm, Pei et al. (Pei, Zheng, Tan, & Takagi, 2012) proposed three sampling methods and two fitting methods in enhanced fireworks algorithm. Also, they used a random selection method to choose individual for next generation.

5.1 Sampling Methods

A. Best Sampling.

The best K individuals are selected as sampling data.

B. Distance near the Best Fitness Individual Sampling.

By calculating the Euclidean distance between the best individual and the other individuals, the nearest K individuals are selected as sampling data.

C. Random Sampling.

K individuals are selected randomly as sampling data.

5.2 Fitting Methods

In order to generate a new spark, three sampling methods and two fitting methods are used and compared. The first sampling method is to select the best K individuals (BST), whereas K can be defined as 3, 5 or 10. The second sampling method is to pick up the K individuals, which have the smallest distance from the best individual (DIS). The third sampling method is to choose the K individuals randomly (RAN). The two fitting methods are linear least square approximation (LS1) and non-linear two degree polynomial approximation (LS2).

Three sampling method are tested with two fitting method. Hence, six different methods are proposed. For example, LS1-BST3 means to select the best three individuals among the population and generates a new spark from these selected individuals using linear least square approximation method. In each dimension, a line segment generates and the value of the middle point is taken as the new spark. As for non-linear fitting methods, the value of the extreme point is treated as the new spark. The new spark replaces the worst individual in the population if the new spark is better.

5.3 Selection Methods

In the paper (Pei, Zheng, Tan, & Takagi, 2012), Pei et al. keeps the best individual for next generation and randomly selects the rest $(N-1)$ individuals. The selection method is named as Random Selection with Replacement (RSR).

6 Experiments

6.1 Design of Experiments

Thirteen test functions are chosen to verify the performance of conventional fireworks algorithm, its variants and standard particle swarm optimization (Bratton, & Kennedy, 2007).

Table 1. The details of benchmark functions

No.	Name	Attributes	Optimization Value	Initial Population	Dimension
F1	Sphere	Unimodal	0.0	$[-100, 100]^D$	30
F2	Schwefel's Problem 1.2	Unimodal	0.0	$[-100, 100]^D$	30
F3	Generalized Rosenbrock	Unimodal	0.0	$[-30, 30]^D$	30
F4	Ackley	Unimodal	0.0	$[-32, 32]^D$	30
F5	Generalized Griewank	Unimodal	0.0	$[-600, 600]^D$	30
F6	Penalized Function F8	Multimodal	0.0	$[-50, 50]^D$	30
F7	Penalized Function P16	Multimodal	0.0	$[-50, 50]^D$	30
F8	Six-hump Camel Back	Multimodal	-1.032	$[-5, 5]^D$	2
F9	Goldstein-Price	Multimodal	3.0	$[-2, 2]^D$	2
F10	Schaffer F6	Multimodal	0	$[-100, 100]^D$	2
F11	Axis Parallel Hyper Ellipsoid	Multimodal	0	$[-5.12, 5.12]^D$	30
F12	Rotated Hyper Ellipsoid	Multimodal	0	$[-65.536, 65.536]^D$	30
F13	Generalized Rastrigin	Multimodal	0	$[-5.12, 5.12]^D$	30

According to the work of Zheng et al. (Zheng, Janecek, & Tan, 2013), fireworks algorithm works extremely well on those functions whose optimum is located at original point (0, 0), because the Gaussian sparks can easily find that point. To shift the global optimal value away from point (0, 0), a number of shifted values are added to the functions. Here, the optimum of the functions is shifted to the right corner of the feasible search space. Table 2 shows the shifted values.

Table 2. Shifted index and shifted value (index zero means no shift)

Shifted Index	Shifted Value
0	0
1	0.025 * $(X_{max}–X_{min})$
2	0.050 * $(X_{max}–X_{min})$
3	0.100 * $(X_{max}–X_{min})$
4	0.150 * $(X_{max}–X_{min})$
5	0.250 * $(X_{max}–X_{min})$
6	0.350 * $(X_{max}–X_{min})$

X_{max} and X_{min} means the maximum and minimum boundaries for the individual, respectively. Still, there are two more parameters in EFWA named as A_{init} and A_{final}. The two parameters are set as $(X_{max}–X_{min})$ * 0.02 and $(X_{max}–X_{min})$ * 0.001, respectively.

The experimental platform is Visual Studio 2012 and the program is running on 64-bit Window 8 operation system with an Intel Core i7-3820QM with 2.70GHz and 2GB RAM. Each experiment runs 30 times and during each run, the fitness functions are evaluated just over 300,000 times. The function evaluation cannot be equal to 300,000 because the number of sparks is not fixed in each generation. Therefore, once the number of function evaluations exceeds 300,000 at the end of a generation, there will not be any further generations.

According to reference (Bratton, & Kennedy, 2007), standard particle swarm optimization (SPSO) includes a ring topology when the particles are only communicated with their two neighbors. Moreover, the number of particles is set as 50, while the initialization is non-uniform and the evaluation operations are skipped if the particles are out of the feasible search space.

6.2 Experimental Results

Six swarm intelligence algorithms are compared, including the conventional fireworks algorithm, four improved fireworks algorithms and the SPSO algorithm.

The parameters of SPSO algorithm are the same as in reference Bratton et al (Bratton, & Kennedy, 2007), while the other four improved fireworks algorithms have the same parameters with the conventional algorithm in reference (Tan, & Zhu, 2010). EFWA is proposed by Zheng (Zheng, Janecek, & Tan, 2013), whereas improved fireworks algorithm with fitness value selection (IFWAFS) and improved fireworks algorithm with best selection (IFWABS) can be found in reference (Liu, Zheng, & Tan, 2013). The algorithm named LS2-BST10 is the best algorithms stated in article (Pei, Zheng, Tan, & Takagi, 2012) with extinguished sampling numbers and fitting methods. LS2-BST10 means the sampling method is non-linear and the best ten individuals are selected.

In order to make the figures more easily readable, the experiment results are divided into two figures, namely Figure 5(a) and Figure 5(b). Thus, the horizontal and vertical axis in Figure 5(a) have the same meaning as Figure 5(b). The horizontal axis stands for the six algorithms along with 13 functions and the vertical axis represents the mean values in the form of logarithm. Some bar figures are not shown because the corresponding mean values are below zero and the logarithm operation cannot be performed.

Figure 5. (a) Mean values of conventional fireworks algorithm and its variants on function 1 to 7 (b) Mean values of conventional fireworks algorithm and its variants on function 8 to 13

Figure 6 depicts the running time for each algorithm on the 13 functions. The vertical axes in both Figure 6 (a) and Figure 6 (b) represent the running time in seconds. The higher the bar

graph is, the more time consuming the algorithm for each function.

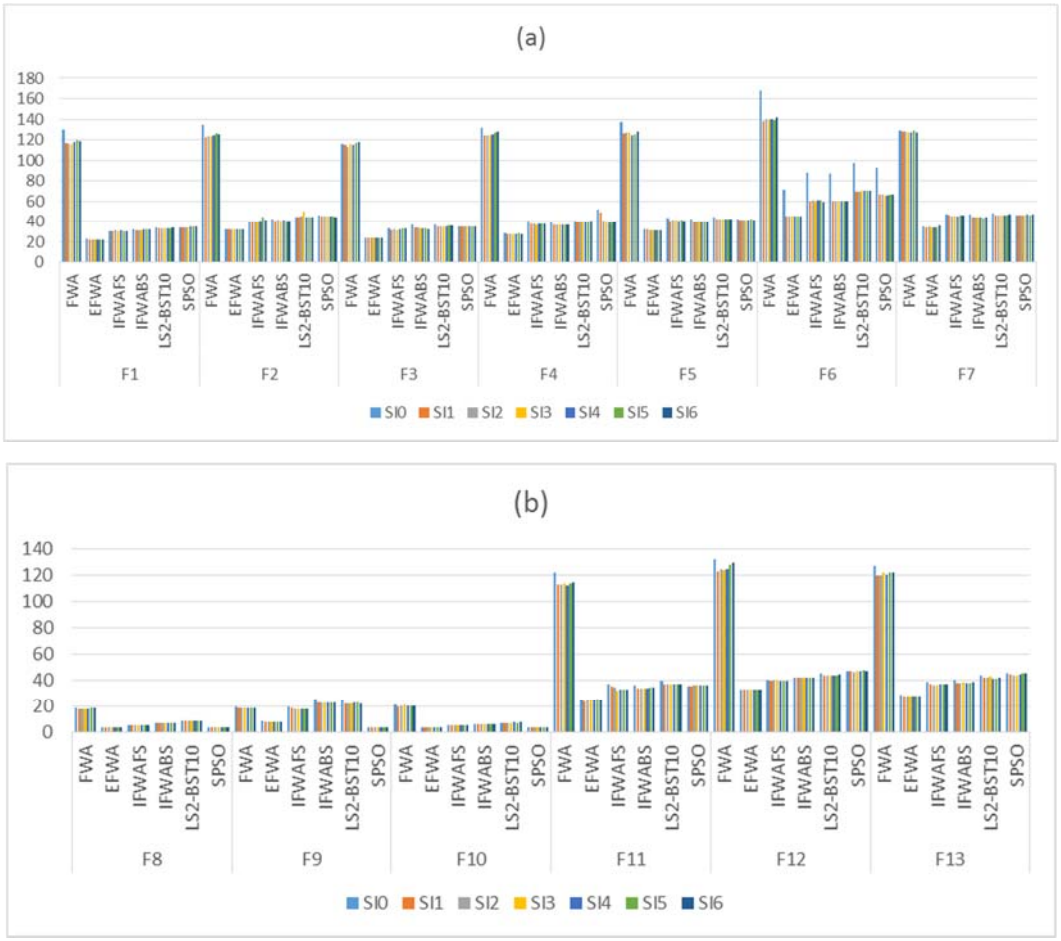


Figure 6. (a) Running time of conventional fireworks algorithm and its variants on function 1 to 7 (b) Running time of the same algorithms on function 8 to 13

To save space, all the experimental results are presented in supplementary files, which can be downloaded from the website of FWA at http://www.cil.pku.edu.cn/research/FWA/.

Table 3 (see supplementary files) represents the mean values and the standard deviations of each algorithm running on all the functions with shifted indexes from 0 to 6. Table 4 (see supplementary files) shows the t-test results of fireworks algorithm against every other algorithms. The running time of each algorithm is shown in Table 5 (see supplementary files).

6.3 Discussions

The following observations can be concluded from the experimental results above.

1) EFWA, IFWAFS, IFWABS and LS2-BST10 are superior to conventional FWA on most functions.

2) With increasing shifted values, EFWA achieves much better results than conventional FWA.

3) EFWA performs steadily even the optimum is shifted to the edge of the feasible space.

4) SPSO achieves better results on large shifted indexes.

5) Improved fireworks algorithms, including EFWA, IFWAFS, IFWABS and LS2-BST10, are worse than SPSO on most functions.

6) EFWA is extremely fast on 11 functions, while SPSO is quicker than other algorithms on 2 other functions.

7) Conventional FWA consumes much more time than all the other algorithms.

7 Multi-objective FWA

Conventional FWA and its variants can solve problems with single objective. When dealing with multi-objective optimization problems, those algorithms are no longer useful and efficient.

Zheng et al. (Zheng, Song, & Chen, 2013) firstly studied multi-objective fireworks algorithm (MOFOA) and applied it to optimize variable-rate fertilization in oil crop production. Three objectives were chosen and the data from three oil crops were used. The distribution of solutions by MOFOA was given and it was also compared with multi-objective random search (MORS).

Fertilizing Oil Crops can be described as a multi-objective problem with three objectives. They are crops quality, fertilizer cost and energy consumption. Compare with conventional FWA, two new strategies were proposed.

7.1 New Fitness Value Assigning Strategy

The new fitness value is represented as the strength of an individual combined with the density value of the individual, whereas the individual strength is the number of other individuals that the individual dominates. The fitness value is evaluated by the formula below.

$$f(x_i) = \sum_{(x_j \in P \cup NP) \wedge (x_j \succ x_i)} \left| \{ x_j \in P \cup NP | x_i \succ x_j \} \right| + \frac{1}{\sigma_k(x_i)} \quad (19)$$

The sign $\succ$ represents the Pareto dominance relationship. $\sigma_k(x_i)$ is the distance of x_i to its kth nearest individual and k is set as the square root of the sample size $|P \cup UP|$.

7.2 New Mutation Strategy

MOFOA algorithm randomly selects three individuals and generates a new individual according to the following formula.

$$v_i = x_{r_1} + F \cdot (x_{r_2} - x_{r_3}) \quad (20)$$

In the formula, r_1, r_2 and r_3 are random indexes of individuals. Parameter F is the scale factor.

The new individual will replace the old individual by a possibility of CR, as described in the following formula.

$$u_i^j = \begin{cases} v_i^j, if\ rand(0,1) < CR\ or\ j = i \\ x_i^j, otherwise \end{cases} \quad (21)$$

Whenever a new individual is generated, it is compared with the old individual to find the better fitness value. For each generation, several individuals will be generated while the best fitness value will be selected.

The solutions of two algorithms are listed in the table below and the distribution is drawn as well.

Table 6. Compared MOFOA with MORS on a variable-rate fertilization problem. This table is from the work of (Zheng, Song, & Chen, 2013).

Algorithms	Solution		
MORS	# 1(56.2, 86.5, 32.9)	# 2(54.8, 84.4, 33.2)	# 3(54.4, 85.7, 31.0)
	# 4(54.1, 85.2, 32.1)	# 5(53.8, 80.3, 34.0)	# 6(53.5, 84.0, 33.8)
	# 7(52.8, 84.9, 30.7)	# 8(52.6, 83.5, 31.9)	# 9(52.5, 82.7, 32.1)
MOFOA	# 1(57.3, 84.9, 31.4)	# 2(56.7, 82.9, 31.2)	# 3(56.4, 82.6, 31.5)
	# 4(56.1, 81.3, 33.4)	# 5(55.6, 83.6, 30.8)	# 6(54.8, 85.1, 30.5)

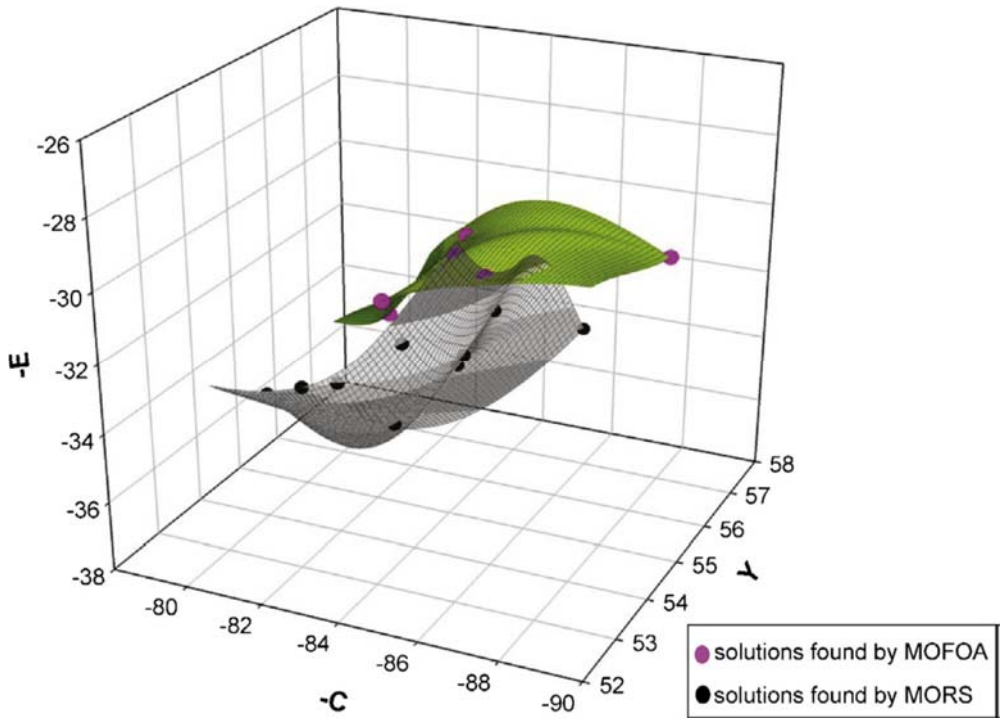


Figure 7. The distribution of solutions in the objective space for MOFOA and MORS. This figure is from the work of (Zheng, Song, & Chen, 2013).

Yet, FWA can work on 0/1 knapsack problems. The 0/1 knapsack problem is an NP hard problem and fireworks algorithm can solve this problem. Zhang J. Q. (Zhang, 2011) first used FWA on knapsack problem and obtained satisfactory solutions.

8 A GPU-based Parallel Fireworks Algorithm

Conventional swarm intelligence algorithms are not designed for GPU architecture. Therefore, they cannot make use of the tremendous computational ability of GPUs. As a result, they are difficult to deal with scaled problems. To overcome the shortcomings, many parallelized swarm intelligence algorithms are proposed to speedup conventional algorithms. Ding et al. (Ding, Zheng, & Tan, 2013) proposed an algorithm named GPU-based parallel fireworks algorithm (GPU-FWA).

GPU-FWA modifies conventional fireworks algorithm so that the algorithm is more suitable for GPU architecture. The implementation of GPU-FWA is based on the CUDA platform and the flowchart of GPU-FWA implementation on CUDA is given in Figure 8.

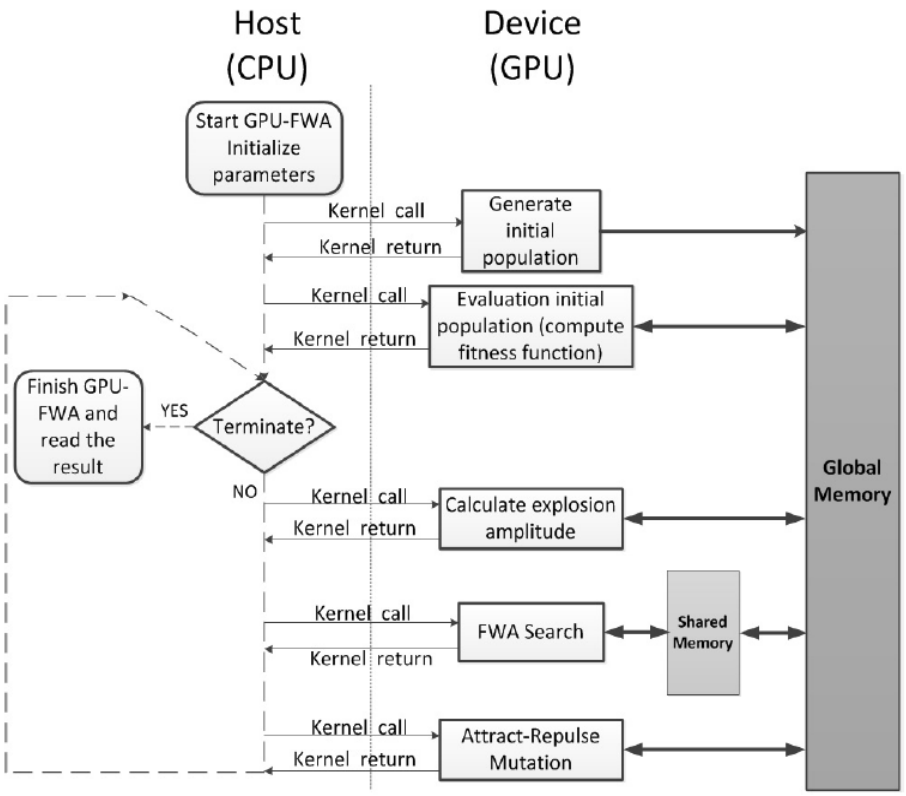


Figure 8. The flowchart of GPU-FWA implementation on CUDA. This figure is according with the work of (Ding, Zheng, & Tan, 2013).

8.1 Two New Strategies

Two new strategies are introduced based on conventional FWA and GPU-FWA, namely FWA search and Attract Repulse Mutation. FWA search mimics the fireworks in the sky and generates a shower of sparks to explore the neighbor area. Attract repulse mutation is used to keep the diversity of the fireworks swarm as it is vital to keep the diversity of a swarm in the optimization procedure. After the mutation operation, some sparks are close to the best spark, while some other sparks are distance from the best spark.

A. FWA Search

```
for i = 1 to L do
    generate m sparks.
    evaluate the fitness value of each sparks.
    find the current best spark with best fitness value.
    update the global best spark.
end for
```

In FWA Search, each firework generates a fixed number of sparks independently. It takes a greedy strategy to decide which spark is selected as the new firework. It guarantees a strong capability of local search. In order to enhance the local search efficiency and minimize the overhead of communication, the greedy search is executed L times before the next strategy is triggered.

B. Attract Repulse Mutation

In order to increase the diversity of fireworks, which is also to improve the global search

capability, attract repulse mutation is applied after all fireworks have finished a round of FWA search. In attract repulse mutation, the firework with the best fitness value is selected as the center. Other fireworks can be either be attracted to the center or repulsed away from it. This process is driven by picking a scaling factor randomly from a uniform distribution lying in $(1-\delta, 1+\delta)$ and the parameter δ varying from 0.1 to 0.9. After that, all dimensions are multiplied by the factor and fireworks are expected to fill the search space uniformly.

Figure 9 shows a general view of attract repulse mutation. The firework with the best fitness value is stationary (bottom left), while a position that is attracted will move towards it (top left) and another position that repels it moves away (bottom right), thus, creating two new positions.

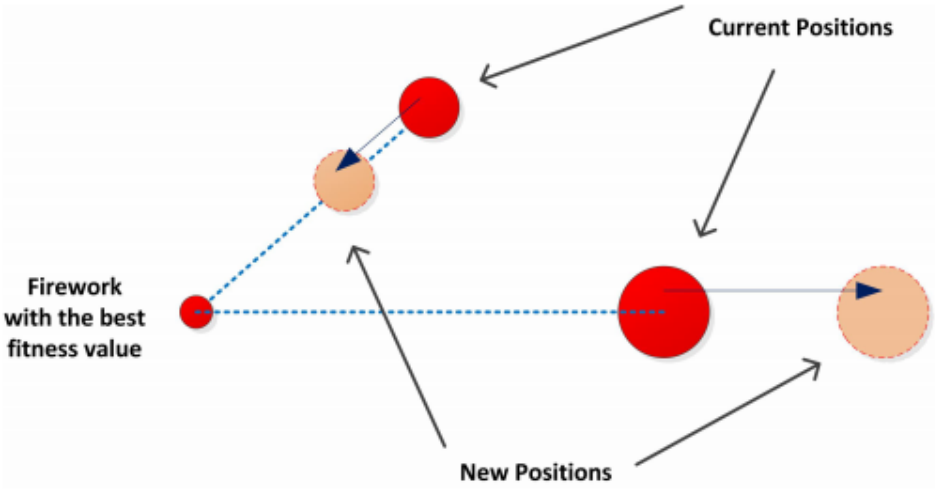


Figure 9. Attract repulse mutation. This figure comes from the work of (Ding, Zheng, & Tan, 2013).

The pseudo code of attract repulse mutation.

Initialize the new location: $\widehat{x_i} = x_i$;
s = $U(1-\delta, 1+\delta)$;
for d = 1 to D do
    r = rand(0, 1) ;
    if r $< \frac{1}{2}$ then
    $\widehat{x_{i,d}} = \widehat{x_{i,d}} + (\widehat{x_{i,d}} - x_{best,d}) \cdot s$;
    end if
    if $\widehat{x_{j,d}} > ub_d$ or $\widehat{x_{j,d}} < lb_d$ then
        $\widehat{x_{j,d}} = lb_d + |\widehat{x_{j,d}} - lb_d| \bmod (ub_d - lb_d)$;
    end if
end for

Thanks to the two new strategies and the massively parallel computing hardware GPU, GPU-FWA is much more effective than FWA and PSO. Compared with FWA and PSO which based on CPU, GPU-based GPU-FWA can reach a speedup as high as 250 times.

8.2 Implementation of GPU-FWA

Here are the three steps for the implementation of GPU-FWA.

A. Thread Assignment

In the GPU-based parallel FWA, each firework is assigned to a group of threads (i.e. 32 continual threads). However, not all of the threads will necessarily be used in the computation. For instance, if the number of sparks is 16 and the number of a group of threads is 32, only half of the threads are used. By using thread assignment, three advantages are revealed. First of all, the threads in the same group can easily interchange information. Secondly, each group processes in the same space and the memory can be shared. As accessing the shared memory costs less time than accessing global memory, computational time can be greatly reduced. Thirdly, any proposed algorithm can be extended with problem scale since GPUs will automatically

dispatch block while running.

B. Data Organization

For each firework, the position and fitness value are stored in the global memory. However, the data of sparks are stored in shared memory. Distinguished from interleaving configuration, both the data of fireworks and sparks are stored in a continuous way. This kind of organization is easy to extend with problem scale.

C. Random Number Generation

Since generating high quality random number is time consuming, the efficient CURAND library (NVIDIA, 2012) is used for generating the random numbers in the implementation.

8.3 Experiments of GPU-FWA

Based on a state-of-the-art commodity Fermi GPU, extensive tests are taken on a suite of well-known benchmark function. GPU-FWA is compared with FWA and PSO on both running time and solution quality. Experimental results demonstrate that GPU-FWA generally outperforms both FWA and PSO, and enjoys a significant speedup as high as 200x, compared to the sequential version of FWA and PSO running on an up-to-date CPU. GPU-FWA also enjoys the advantages of being easy to implement and scalable.

Aside from running on CPU, conventional fireworks algorithm is also available to run on GPU. Comparing conventional FWA and GPU-based FWA, the experimental results are shown in the following table. Note that the functions below are different from the functions listed in the experiments design section.

For GPU-based parallel fireworks algorithm, the experimental results are tested on Windows 7 Professional x64 with 4G DDR3 Memory (1333 MHz) and Intel core i5-2310 (2.9 GHz, 3.1 GHz). The GPU used in the experiments is NVIDIA GeForce GTX 560 Ti with 384 CUDA cores, while the CUDA runtime version is 5.0. For more specific details of parameters setting, please see reference (Ding, Zheng, & Tan, 2013).

Table 7. Comparison of GPU-FWA, FWA and PSO on mean and standard deviation

Functions No.	GPU-FWA		FWA		PSO	
	Average Value	Standard Deviation	Average Value	Standard Deviation	Average Value	Standard Deviation
F1	**1.31e-09**	1.85e-09	7.41e+00	1.98e+01	3.81e-08	7.42e-07
F2	1.49e-07	6.04e-07	9.91e+01	2.01e+02	**3.52e-11**	1.15e-10
F3	**3.46e+00**	6.75e+01	3.63e+02	7.98e+02	2.34e+04	1.84e+04
F4	**1.92e+01**	3.03e+00	4.01e+02	5.80e+02	1.31e+02	8.68e+02
F5	**7.02e+00**	1.36e+01	2.93e+01	2.92e+00	3.16e+02	1.11e+02
F6	-8.09e+03	2.89e+03	**-1.03e+04**	3.77e+03	-6.49e+03	9.96e+03
F7	1.33e+00	1.78e+01	**7.29e-01**	1.24e+00	1.10e+00	1.18e+00
F8	**3.63e-02**	7.06e-01	7.48e+00	7.12e+00	1.83e+00	1.26e+01

Both Table 7 and Table 8 are from previously published experiment results (Ding, Zheng, & Tan, 2013). The better results are shown in bold font. It can be seen from Table 7 and Table 8 that GPU-FWA defeated both FWA and PSO on the 8 functions, yet the GPU-FWA is also the quickest algorithm for calculation.

Table 8. Comparison of GPU-FWA, FWA and PSO on running time and speedup

Number of Sparks	FWA (s)	PSO (s)	GPU-FWA (s)	Speedup (FWA)	Speedup (PSO)
48	36.420	84.615	0.615	59.2	137.6
72	55.260	78.225	0.624	88.6	125.4
96	65.595	103.485	0.722	90.8	143.3
144	100.005	155.400	0.831	120.3	187.0

According to the table above, it is obvious that GPU-FWA greatly reduced the running time compare with FWA and PSO.

For more details, please refer to the reference Ding et al (Ding, Zheng, & Tan, 2013).

9 Applications

Conventional fireworks algorithm and its variants are capable of dealing with optimization problems. Many researches used these algorithms in a variety of applications.

Janecek et al. (Janecek, & Tan, 2011) applied fireworks algorithm to non-negative matrix factorization (NMF). In their paper, a new iterative update strategy for multiplicative update algorithm based on fireworks algorithm is proposed. Experimental results have proved that the new iterative update strategy approach the same approximation error as the standard version in significantly fewer iterations. Besides, the new strategy consumes less time.

Gao et al. (Gao, & Diao, 2011) applied fireworks algorithm to digital filters design. After transforming the design of digital filters to a constrained optimization problem, fireworks algorithm was able to find the global optimum. Computer simulations shown the filters using fireworks algorithm were better than using PSO and improved PSO algorithms.

He et al. (He, Mi, & Tan, 2013) used fireworks algorithm for spam detection. In their article, a new framework to optimize the anti-spam model using swarm intelligence optimization algorithm was proposed and experimental results show a good performance demonstrated on corpora PU1, PU2, PU3 and PUA.

Du (Du, 2013) solved nonlinear equations with fireworks algorithm and compared it with artificial bee colony (ABC) algorithm. From the four equations listed in his paper, fireworks algorithm was better than ABC algorithm on three equations. Therefore, fireworks algorithm worked very well on nonlinear equations.

9.1 FWA for NMF computing

The Non-negative Matrix Factorization (NMF) refers to as low-rank approximation and has been utilized in several different areas such as content based retrieval and data mining applications, et al. NMF can reduce storage and runtime requirements, and also reduce redundancy and noise in the data representation while capturing the essential associations. The NMF method requires all entries in A, W and H to be zero or positive (Lee, & Seung 1999) which makes the interpretation of the NMF factors much easier. The NMF consists of reduced rank nonnegative factors $W \in \mathbb{R}^{m \times k}$ and $H \in \mathbb{R}^{k \times n}$ with $k \ll \min\{m, n\}$ that approximate a matrix $A \in \mathbb{R}^{m \times n}$ by WH. The nonlinear optimization problem underlying NMF can generally be stated as

$$\min_{W,H} f(W,H) = \min_{W,H} \frac{1}{2} || A - WH ||_F^2 . \quad (22)$$

The error between the original data A and the approximation WH are stored in a distance matrix D = A – WH. The schematic diagram of coarse NMF approximation with extremely low rank k is shown in Figure 10.

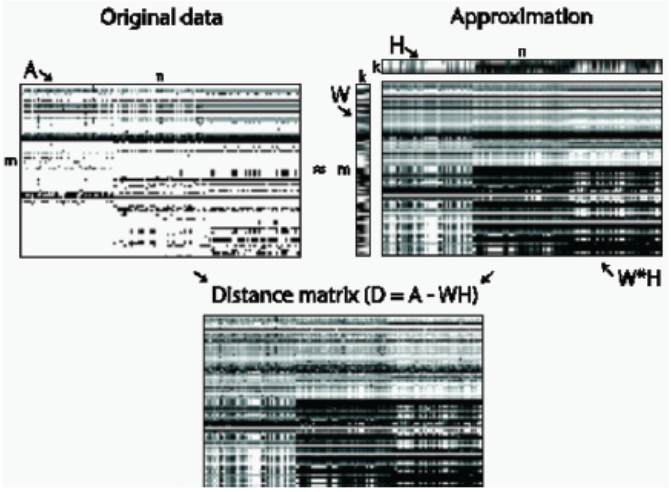


Figure 10. Scheme of coarse NMF approximation with extremely low rank k. This figure is firstly published in the work of (Janecek, & Tan, 2011).

To solve this problem, the nature-inspired optimization heuristics algorithms, genetic algorithms (Goldberg, 1988), particle swarm optimization (Kennedy, & Eberhart, 1995), differential evolution (Storn, & Price 1995), fish school search (Filho, de Lima Neto, Lins, Nascimento, & Lima, 2009), fireworks algorithm (Tan, & Zhu, 2010) are all used.

The parameters in the algorithms are set as following:

GA: mutation rate = 0.5; selection rate = 0.65.

PSO: following (Bratton, & Kennedy, 2007).

DE: crossover probability set to upper limit 1.

FSS: $\text{step}_{\text{ind\_initial}} = 1$, $\text{step}_{\text{ind\_final}} = 0.001$, $W_{\text{scale}} = 10$.

FWA: the number of first selected location is set as 10.

The experiment results of FWA on NMF computing are shown below. Figure 11 shows the convergence curves of accuracy while Figure 12 provides the running time for the six algorithms. It can be seen from the two figures that FWA works well on NMF computing.

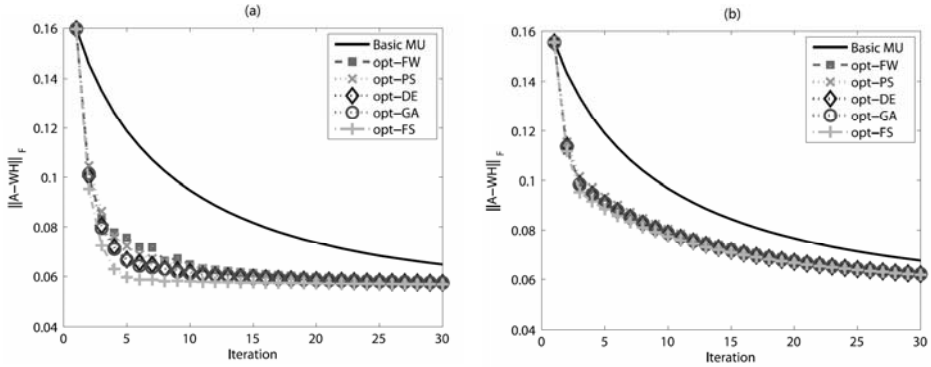


Figure 11. (a) Convergence Curves of the Accuracy when updating only the row of W, m=2, c=20, k=2 (b) Convergence Curves of the Accuracy when updating only the row of W, m=2, c=20, k=5. This figure is according to the work of (Janecek, & Tan, 2011).

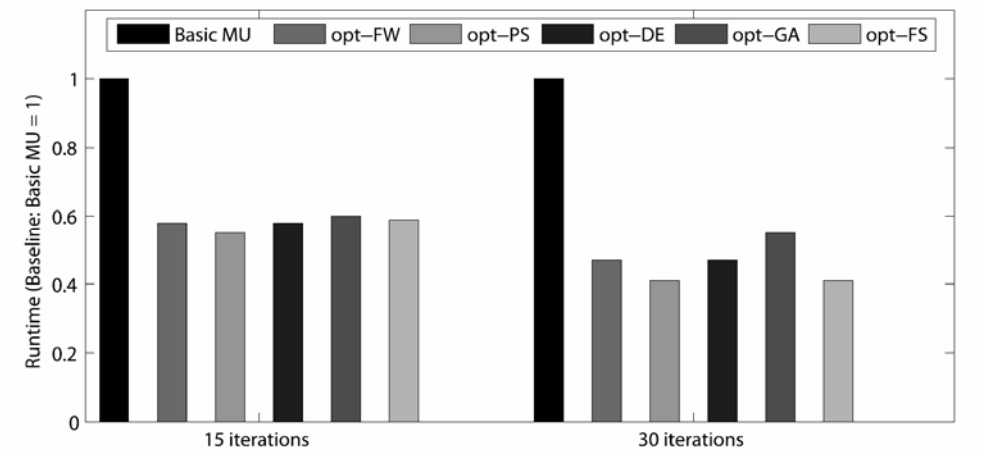


Figure 12. The proportion of running time to achieve the same accuracy. Set the running time of Basic MU as 1 and updating only the row of W, m=2, c=20, k=2. This figure comes from the work of (Janecek, & Tan, 2011).

Pseudo code for the iterative optimization for the Multiplicative Update (MU) algorithm is listed below. The methods used in this algorithm are explained below. Here m is 2 while c is set as 20 which denotes the number of rows and/or columns that are optimized in the current iteration. $\Delta\mathbf{c}$: The value of c is decreased by $\Delta\mathbf{c}$ in each iteration.

For T = 1 to maxIter dimensions do
  $W = W \cdot (AH^T)./(WHH^T + \varepsilon)$
  $H = H \cdot (W^T A)./(W^T WH + \varepsilon)$
  if t < m then
    d_i^r is the i^{th} row vector of $D = A - WH$
    $[Val, IX\_W] = sort(norm(d_i^r), 'descend')$
    $IX\_W = IX\_W(1:c)$
    ∀i ∈ IX_W , use SIO to find w_i^r that minimized $\|a_i^r - w_i^r H_0\|$
    $W = [w_1^r; \cdots w_m^r]$
    d_i^r is the i^{th} row vector of $D = A - WH$
    $[Val,\ IX\_H] = sort(norm(d_i^c), 'descend')$
    $IX\_H = IX\_H(1:c)$
    ∀j ∈ IX_H , use SIO to find h_j^c that minimized $\left\|a_j^c - Wh_j^c\right\|$
    $H = [h_1^c, \cdots h_n^c]$
  end if
end for

FWA performs just the same as the other heuristics algorithms while all of them get better results compared with Basic MU method, the smaller the parameter k is, the more advantages the heuristics algorithms gain.

9.2 FWA on design of digital filters

To design a digital filter, a multi-parameter optimization problem must be solved. However, the existing methods, such as particle swarm optimization (PSO), quantum-behaved particle swarm optimization (QPSO) and adaptive quantum-behaved particle swarm optimization (AQPSO) cannot find the optimal solution effectively. A cultural fireworks algorithm is proposed for digital filter design.

In the finite impulse response (FIR) and infinite impulse response (IIR) digital filters, cultural fireworks algorithm is used to design a joint objective function. The goal for cultural fireworks algorithm is to find the minimal value of the following function.

$$f(x) = \begin{cases} \alpha E_F + \beta E_I, x \in s.t \\ \delta[\alpha E_F + \beta E_I], x \notin s.t \end{cases} \quad (23)$$

f(x) is the objective function while α and β are Boolean parameters. For FIR digital filters, α equals to 1 and β means to 0, while in IIR digital filters, verse vice. δ is set to be larger than 1, whereas E_F and E_I stands for FIR and IIR filters separately. The constraint condition of vector x is represented as s.t. The following table shows the comparison of four algorithms in FIR filter design.

Table 9. Comparison of four algorithms on FIR filter. This figure is taken from the work of (Gao, & Diao, 2011).

Objective Value	Low-pass filter			
	PSO	QPSO	AQPSO	CFWA
Max	1.8505e-3	8.8845e-9	7.6096e-9	**8.8818e-16**
Min	7.5908e-5	6.6311e-12	1.1427e-10	**0**
Mean	3.9566e-4	5.8634e-10	1.2362e-9	**2.5535e-16**
Variance value	8.6053e-8	1.3445e-18	1.1087e-18	**1.1330e-32**

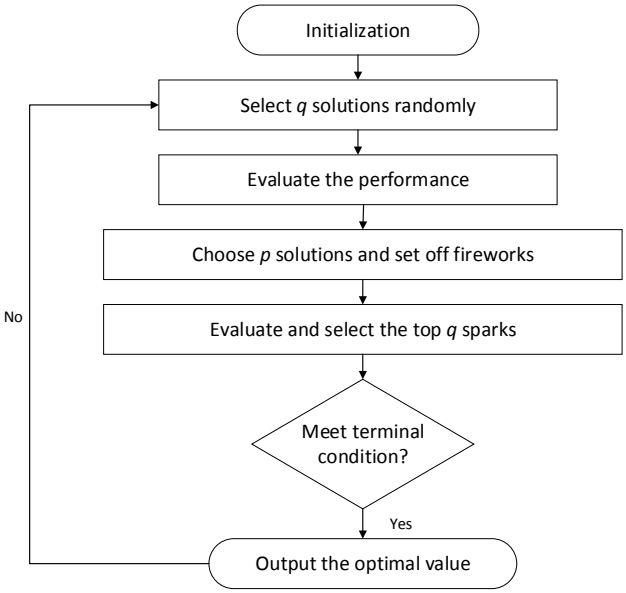


Figure 13. The flow chart of design digital filters by cultural fireworks algorithm. This figure is first published in the work of (Gao, & Diao, 2011).

Figure 13 shows the flow chart of digital filter based on cultural fireworks algorithm (CFWA).

Experimental results show that the proposed cultural fireworks algorithm has a faster convergence and better optimization accuracy than particle swarm optimization, quantum-behaved particle swarm optimization and adaptive quantum-behaved particle swarm optimization algorithms. Therefore, cultural fireworks algorithm is effective and useful on digital filter design. For more experimental results, please refer to reference (Gao, & Diao, 2011).

9.3 FWA on spam detection

In previous research, it is simple to set the parameters manually in the anti-spam process. However, the manually settings may cause several problems. First of all, when setting the parameters without prior knowledge, people have to test many groups of parameters to find

the best one. Secondly, the parameters of different datasets are varied. There are no universal parameters.

To solve the problem of setting parameters, a new framework to find the proper parameters in anti-spam model with fireworks algorithm is proposed. In the anti-spam model, the error rate represents the quality of the model. To make the error rate lower, an optimal vector $P^* = < F_1^*, F_2^*, \cdots, F_n^*, C_1^*, C_2^*, \cdots, C_m^* >$ is suggested, which contains two parts. The first part $F_1^*$to F_n^* means the feature calculation relevant parameters and the second part C_1^* to C_m^* stands for the classifier relevant parameters. $CF(P)$ represents the cost function and it is calculated as

$$\mathrm{CF(P)} = \mathrm{Err(P)}, \tag{24}$$

where $Err(p)$ is the classification error of 10-fold cross-validation on the training set. Different feature extraction methods may need different parameters. In local-concentration model, the selection rate m helps to select the top m percent terms in a term set with descending importance. The proclivity threshold θ equals to the minimal difference of a term's frequency in non-spam emails minus a term's frequency in spam emails. The parameter N is the number of sliding windows.

The flowchart of using fireworks algorithm to optimize parameters in local-concentration model for spam detection is given.

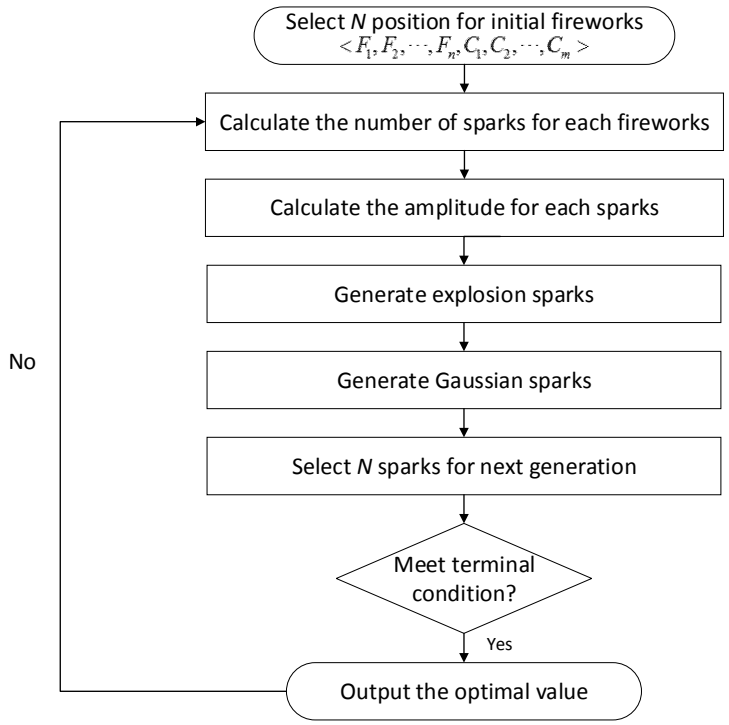


Figure 14. The flowchart of parameter optimization in a local-concentration model for spam detection using fireworks algorithm. This figure is first published in the work of (He, Mi, & Tan, 2013).

Fireworks algorithm is used to optimize the parameters in the model and there are two strategies to build the model. The first strategy is to derive a small dataset from the training set. The small dataset is used as a validation set and do not participate in the training process. After building a model on the training set, the model is validated on the small dataset. The best model is chosen before apply to the test set. The second strategy is to divide the training set into ten even parts and each part is used only once as the validation set. Therefore, ten models are built and the best model is applied to the test set.

The following table shows the experimental results of the comparison between fireworks

algorithm with two strategies each with local concentration (LC) method.

Table 10. Comparison of fireworks algorithm with the first strategy and local concentration method

Corpus	Methods	Precision (%)	Recall (%)	Accuracy (%)	F_1-measure
PU1	LC	94.85	**95.63**	95.87	95.21
	FWA	**96.55**	95.21	**96.33**	**95.81**
PU2	LC	**95.74**	77.86	94.79	85.16
	FWA	95.15	**80.71**	**95.35**	**86.65**
PU3	LC	**96.68**	94.34	96.03	95.45
	FWA	95.81	**95.71**	**96.18**	**95.69**
PU4	LC	95.60	**94.56**	94.91	94.94
	FWA	**96.63**	**94.56**	**95.53**	**95.49**

Table 11. Comparison of fireworks algorithm with the second strategy and local concentration method

Corpus	Methods	Precision (%)	Recall (%)	Accuracy (%)	F_1-measure
PU1	LC	**100.00**	92.36	96.67	95.88
	FWA	**100.00**	**96.64**	**98.57**	**98.22**
PU2	LC	**100.00**	64.00	90.71	74.62
	FWA	**100.00**	**94.17**	**98.57**	**96.57**
PU3	LC	97.84	91.30	95.37	94.34
	FWA	**98.25**	**95.91**	**97.56**	**97.02**
PU4	LC	95.78	90.72	93.64	92.68
	FWA	**98.75**	**96.44**	**97.73**	**97.42**

The two tables above are from the work of He, Mi, & Tan, 2013. The details of evaluation criteria can be found in (He, Mi, & Tan, 2013).

Experimental results show that the fireworks algorithm is better than local concentrate method on corpora PU1, PU2, PU3 and PUA.

9.4 FWA on non-linear equations

In the engineering and scientific fields, many problems can be transferred to non-linear equations. Traditional methods use derivative of the object function to solve non-linear equations. However, traditional methods are sensitive to initial values and convergent in local area. Thus, swarm intelligent algorithms are used to solve non-linear equations. Since artificial bee colony algorithm cannot achieve the best optimal result, fireworks algorithm is used to deal with non-linear equations.

Four non-linear equations from article (Du, 2013) are listed below.

Equation 1:

$$\mathrm{f(x)} = x^3 - 2\mathrm{x} - 5 = 0, \mathrm{x} \in [-4,4]$$

Equation 2:

$$\mathrm{f(x)} = x^3 - 3x^2 - 6x + 8, x \in [0,2]$$

Equation 3:

$$\begin{cases} f_1(x)=x_1+\frac{1}{4}x_2^2x_4x_6+0.75=0 \\ f_2(x)=x_2+0.405e^{(1+x_1x_2)}-1.405=0 \\ f_3(x)=x_3-\frac{1}{4}x_4x_6+1.25=0 \\ f_4(x)=x_4-0.605e^{(1-x_3^2)}-0.395=0 \\ f_5(x)=x_5-\frac{1}{2}x_2x_6+1.5=0 \\ f_6(x)=x_6-x_1x_5+1.5=0 \end{cases}$$

where x_i varies from -2 to 2 and the optimal lies in (-1, 1, -1, 1, -1, 1)T.

Equation 4:

$$\begin{cases} x_1^{x_2}+x_2^{x_1}-5x_1x_2x_3-85=0 \\ x_1^3-x_2^{x_3}-x_3^{x_2}-60=0 \\ x_1^{x_3}+x_3^{x_1}-x_2-2=0 \end{cases}$$

where the ranges of x_i is from 0 to 10 and the best solution is (4, 3, 1)T.

Note that the square of each equation is the objective function. The steps of fireworks algorithm is as follows.

Step 1: Randomly generates n individuals at

initial.

Step 2: Generate common sparks and Gaussian sparks the same as fireworks algorithm.

Step 3: Choose the best individual for next generation and the next $(N-1)$ individuals are choose the same like fireworks algorithm.

Step 4: If the terminal condition is met, stop the procedure. If not, go back to step 2.

The experimental result is listed as follows. The best results are in bold font.

Table 12. Comparison of ABC and FWA on four equations. The table comes from the work of (Du, 2013).

Equation NO.	Variables	ABC	FWA
Equation 1	X	**2.09455448**	2.09465148
Equation 2	X	1.14324234	**0.98973242**
Equation 3	X1	-1.000343	**-1**
Equation 3	X2	0.844234	**1**
Equation 3	X3	-1.64535	**-1**
Equation 3	X4	1.031231	**1**
Equation 3	X5	-0.98232	**-1**
Equation 3	X6	0.9932432	**1**
Equation 4	X1	4.000013	**4**
Equation 4	X2	3.000029	**3**
Equation 4	X3	0.962344	**1**

The result of artificial bee colony (ABC) algorithm is from the work of Zhang J. L. (Zhang, 2012).

10 Conclusion and Future Directions

Fireworks algorithm provides a brand new way to solve complex problems. The current fireworks algorithm and its applications prove that it can solve many optimization problems effectively. Furthermore, fireworks algorithm can be parallelized and thus suitable to deal with big data problems. No matter for theoretical or applied researches, fireworks algorithm is worth researching and can bring great scientific and economic benefits.

However, there are still some disadvantages in fireworks algorithm. Firstly, fireworks algorithm simulates behaviors of biomes and lack of necessary mathematical foundation. For example, there is no proof of convergence in fireworks algorithm. Secondly, most of the parameters in fireworks algorithm are set by experience and the parameters largely depend on specific problems. Thirdly, not many applications of fireworks algorithm are currently in use. Furthermore, it is crucial to observe each algorithm in real world problems, rather than strictly theoretical situations in order to fully appreciate its benefits.

Fireworks algorithm has been greatly developed, but still it is not perfect. The direction of its future development can be described as follows. First of all, fireworks algorithm needs its mathematical foundation and theoretical analysis. Secondly, the selection of the control parameters of fireworks algorithm often relies on experience. So how to choose the mostly appropriate parameters needs a theoretical guidance. Thirdly, the prospects of fireworks algorithm applications are still at infancy and require further exploration. Fourthly, as an open source algorithm, fireworks algorithm can learn from other algorithms. How to improve fireworks algorithm is also a useful research direction. Last but certainly not least, the study of GPU to accelerate fireworks algorithm is at its initial stage and will attract more and more researchers who are devoted to apply the FWA to real world problems.

Acknowledgement

This work was supported by the Natural Science Foundation of China with grant no. 61375119, 61170057 and 60875080.

REFERENCES

Banerjee, S., & Caballe, S. (2011, November). Exploring Fish School Algorithm for Improving Turnaround Time: An Experience of Content Retrieval. In *Intelligent Networking and Collaborative Systems (INCoS), 2011 Third International Conference on* (pp. 842-847). IEEE.

Bastos Filho, C. J., de Lima Neto, F. B., Lins, A. J., Nascimento, A. I., & Lima, M. P. (2009). Fish school search. In *Nature-Inspired Algorithms for Optimisation* (pp. 261-277). Springer Berlin Heidelberg.

Bratton, D., & Kennedy, J. (2007, April). Defining a standard for particle swarm optimization. *In Swarm Intelligence Symposium, 2007. SIS 2007* (pp. 120-127). IEEE.

Bureerat, S. (2011). Hybrid population-based incremental learning using real codes. In *Learning and Intelligent Optimization* (pp. 379-391). Springer Berlin Heidelberg.

Bureerat, S. (2011). Improved population-based incremental learning in continuous spaces. In *Soft Computing in Industrial Applications* (pp. 77-86). Springer Berlin Heidelberg.

Colorni, A., Dorigo, M., & Maniezzo, V. (1991, December). Distributed optimization by ant colonies. *In Proceedings of the first European conference on artificial life* (Vol. 142, pp. 134-142).

Ding, K., Zheng, S., & Tan, Y. (2013). A GPU-based parallel fireworks algorithm for optimization. *Genetic and Evolutionary Computation Conference* (pp. 9-16).

Du, Z. (2013) Fireworks algorithm for solving nonlinear equation and system. *Modern Computer* (pp. 18-21). (In Chinese)

Gao, H., & Diao, M. (2011). Cultural firework algorithm and its application for digital filters design. *International Journal of Modelling, Identification and Control*, 14(4), 324-331.

Goldberg, D. E., & Holland, J. H. (1988). Genetic algorithms and machine learning. *Machine learning*, 3(2), 95-99.

He, W., Mi, G., & Tan, Y. (2013). Parameter Optimization of Local-Concentration Model for Spam Detection by Using Fireworks Algorithm. In *Advances in Swarm Intelligence* (pp. 439-450). Springer Berlin Heidelberg.

Janecek, A., & Tan, Y. (2011). Feeding the fish–weight update strategies for the fish school search algorithm. In *Advances in Swarm Intelligence* (pp. 553-562). Springer Berlin Heidelberg.

Janecek, A., & Tan, Y. (2011). Swarm Intelligence for Non-Negative Matrix Factorization. *International Journal of Swarm Intelligence Research (IJSIR)*, 2(4), 12-34.

Janecek, A., & Tan, Y. (2011). Using population based algorithms for initializing nonnegative matrix factorization. In *Advances in Swarm Intelligence* (pp. 307-316). Springer Berlin Heidelberg.

Janecek, A., & Tan, Y. (2011, July). 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.

Karaboga, D., & Basturk, B. (2008). On the performance of artificial bee colony (ABC) algorithm. *Applied soft computing*, *8*(1), 687-697.

Kennedy, J., & Eberhart, R. (1995, November). Particle swarm optimization. In *Neural Networks, 1995. Proceedings. IEEE International Conference on* (Vol. 4, pp. 1942-1948). IEEE.

Lee, D. D., & Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. Nature, 401(6755), 788-791.

Liu, J., Zheng, S., & Tan, Y. (2013). The Improvement on Controlling Exploration and Exploitation of Firework Algorithm. *In Advances in Swarm Intelligence* (Vol. 7928, pp. 11-23). Springer Berlin Heidelberg.

Lou, Y., Li, J., Jin, L., & Li, G. (2012). A Co-Evolutionary Algorithm Based on Elitism and Gravitational Evolution Strategies. *Journal of Computational Information Systems*, *8*(7), 2741-2750.

Lou, Y., Li, J., Shi, Y., & Jin, L. (2013). Gravitational Co-evolution and Opposition-based Optimization Algorithm. *International Journal of Computational Intelligence Systems*, *6*(5), 849-861.

Lu, G., Tan, D. J., & Zhao, H. (2002, November). Improvement on regulating definition of antibody density of immune algorithm. In *Neural Information Processing, 2002. ICONIP'02. Proceedings of the 9th International Conference on* (Vol. 5, pp. 2669-2672). IEEE.

NVIDIA. Toolkit 5.0 CURAND Guide, September, 2012.

Pei, Y., Zheng, S., Tan, Y., & Takagi, H. (2012, October). An empirical study on influence of approximation approaches on enhancing fireworks algorithm. *In Systems, Man, and Cybernetics (SMC), 2012 IEEE International Conference on,* (pp. 1322-1327). IEEE.

Storn, R., & Price, K. (1995). *Differential evolution-a simple and efficient adaptive scheme for global optimization over continuous spaces, Berkeley*. CA, Tech. Rep. TR-95-012.

Storn, R., & Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, *11*(4), 341-359.

Tan, Y., & Zhu, Y. (2010). Fireworks algorithm for optimization. *In Advances in Swarm Intelligence,* (Vol. 6145, pp. 355-364). Springer Berlin Heidelberg.

Zhang J. (2012). Artificial bee colony algorithm for solving nonlinear equation and system. *Computer Engineering and Applications*, 48(22), 45-47. (In Chinese)

Zhang J. (2011). Fireworks algorithm for solving 0/1 knapsack problem. *Journal of Wuhan Engineering Institute,* 23(3), 64-66. (In Chinese)

Zheng, S., Janecek, A., & Tan, Y. (2013). Enhanced fireworks algorithm. *IEEE Congress on Evolutionary Computation* (pp. 2069-2077). IEEE.

Zheng, Y., Song, Q., & Chen, S. (2013). Multiobjective fireworks optimization for variable-rate fertilization in oil crop production. *Applied Soft Computing*, 13(11), 4253-4263.

Zheng, Z., & Tan, Y. (2013). Group explosion strategy for searching multiple targets using swarm robotic. *IEEE Congress on Evolutionary Computation* (pp. 821-828). IEEE.

Zhou, Y., & Tan, Y. (2009, May). GPU-based parallel particle swarm optimization. In *Evolutionary Computation, 2009. CEC'09. IEEE Congress on* (pp. 1493-1500). IEEE.

Zhou, Y., & Tan, Y. (2011). GPU-based parallel multi-objective particle swarm optimization. *International Journal of Artificial Intelligence*, 7(A11), 125-141.

Ying Tan is a professor at Key Laboratory of Machine Perception (MOE), Peking University, and Department of Machine Intelligence, School of Electronics Engineering and Computer Science, Peking University, China, and director of the Computational Intelligence Laboratory of Peking University (CIL@PKU). He received his BSc, MSc, and PhD from Southeast University, in 1985, 1987 and 1997, respectively. He served as Editor-in-Chief of International Journal of Computational Intelligence and Pattern Recognition, Associate Editor of IEEE Transactions on Cybernetics, International Journal of Artificial Intelligence, International Journal of Swarm Intelligence Research (IJSIR), International Journal of Intelligent Information Processing, etc. He is a member of Emergent Technologies Technical Committee (ETTC), Computational Intelligence Society of IEEE. He is or was the general chair of the International Conference on Swarm Intelligence (ICSI 2010-13) and one of the joint general chairs of BRICS CCI'2013, Program committee co-chair of IEEE WCCI 2014, and several international conferences. His primary research interests include computational intelligence, artificial immune system, swarm intelligence, intelligent information processing, machine learning algorithms and their applications in computer security. He has published more than 200 papers in refereed journals and conferences in these areas, published several books and chapters in book, and received 3 patents.

Chao Yu is a student at Key Laboratory of Machine Perception (MOE), Peking University. He is currently working towards his Ph.D. degree in Department of Machine Intelligence, School of Electronics Engineering and Computer Science, Peking University, China. He received his BSc in network engineering and MSc in computer science in School of Computer Science from China University of Geosciences (Wuhan), China. His primary research interests are computational intelligence, evolutionary computation, machine learning and data mining algorithms.

Shaoqiu Zheng received B.S. degree in Department of Acoustic, School of Marine Technology from Northwestern Polytechnic University, Xi'an, China, in 2010. He is currently majoring in Computer Science and Technology, and working towards the Ph.D. degree at Key Laboratory of Machine Perception (Ministry of Education) and Department of Machine Intelligence, School of Electronic Engineering and Computer Science, Peking University, China. His research interests include machine learning, swarm intelligence, biometrics and pattern recognition.

Ke Ding received B.S. degree in Computer Science from China Agricultural University, China, in 2010. He is currently majoring in Computer Science and working towards the Ph.D. degree at Key Laboratory of Machine Perception (Ministry of Education) and Department of Machine Intelligence, EECS, Peking University, China. His research interests are related to swarm intelligence, GPU computing, parallel programming, machine learning and network security.