algorithms

MDPI

Article

Analysis and Improvement of Fireworks Algorithm

Xi-Guang Li [1], Shou-Fei Han [1,*] and Chang-Qing Gong [1]

School of Computer, Shenyang Aerospace University, Shenyang 110136, China;
lixiguang@sau.edu.cn (X.-G.L.); gongchangqing@sau.edu.cn (C.-Q.G.)
* Correspondence: hanshoufei@gmail.com; Tel.: +86-158-0405-7359

Academic Editors: Sergio Rajsbaum and George Karakostas
Received: 12 December 2016; Accepted: 14 February 2017; Published: 17 February 2017

Abstract: The Fireworks Algorithm is a recently developed swarm intelligence algorithm to simulate the explosion process of fireworks. Based on the analysis of each operator of Fireworks Algorithm (FWA), this paper improves the FWA and proves that the improved algorithm converges to the global optimal solution with probability 1. The proposed algorithm improves the goal of further boosting performance and achieving global optimization where mainly include the following strategies. Firstly using the opposition-based learning initialization population. Secondly a new explosion amplitude mechanism for the optimal firework is proposed. In addition, the adaptive *t*-distribution mutation for non-optimal individuals and elite opposition-based learning for the optimal individual are used. Finally, a new selection strategy, namely Disruptive Selection, is proposed to reduce the running time of the algorithm compared with FWA. In our simulation, we apply the CEC2013 standard functions and compare the proposed algorithm (IFWA) with SPSO2011, FWA, EFWA and dynFWA. The results show that the proposed algorithm has better overall performance on the test functions.

Keywords: fireworks algorithm; opposition-based learning; *t*-distribution; disruptive selection; explosion amplitude

1. Introduction

The Fireworks Algorithm (FWA) [1] is a newly developed evolutionary algorithm. Like other evolution algorithms, it also aims to find the vector with the best (usually minimum) fitness in the search space. Inspired by real fireworks, the main idea of the FWA is to use the explosion of the fireworks to search the feasible space of the optimization function, which is a brand new search manner. At present, the fireworks algorithm has been applied to many practical optimization problems [2], the application areas include the factorization of a non-negative matrix [3], the design of digital filters [4], the parameter optimization for the detection of spam [5], the reconfiguration of networks [6], the mass minimization of trusses [7], the parameter estimation of chaotic systems [8], the scheduling of multi-satellite control resources [9], etc.

However, similar to other intelligent optimization algorithms, fireworks algorithms have some disadvantages such as slow convergence speed and low accuracy, thus, many improved algorithms have been proposed. An enhanced fireworks algorithm (EFWA) was put forward by analyzing the explosion operator, mutation operator, selection strategy and mapping rule of the fireworks algorithm [10]. An adaptive firework algorithm (AFWA) is proposed to carry out the self-tuning for the explosion amplitude [11]. That is to say, the explosion amplitude of the fireworks is determined by the distance between the individual with the best population fitness value and the distance between a specific individual. A dynamic search firework algorithm (dynFWA) is proposed, which divides the fireworks population into optimal firework with the optimal fitness value and non-optimal fireworks. By doing so, the two populations maintained a good balance during the evolution and showed good performance [12]. However, the fireworks algorithm is still in the development stage, to be further studied to enhance the performance.

In this paper, firstly we use the opposition-based learning strategy to initialize population to increase the diversity of the population and improve the probability to search the global optimal solution. Secondly a new explosion amplitude mechanism for the optimal firework is proposed from the aspects of population evolution rate and population aggregation degree, to enhance the ability to search the global optimal solution of the optimal firework. In addition, adaptive *t*-distribution for non-optimal fireworks and elite opposition-based learning for optimal firework are applied, in order to improve the global exploration ability and local development ability and make the FWA jump out of the local optimum effectively. Finally, a new selection strategy is proposed to reduce the run-time of the algorithm. Based on this, an improved fireworks optimization algorithm (IFWA) is proposed to improve the convergence speed and precision and reduce the run-time.

The paper is organized as follows. In Section 2, the fireworks algorithm is introduced. The IFWA algorithm is presented in Section 3. The simulation experiments and results analysis are given in details in Section 4. Finally, the conclusion summarizes in final part.

2. Fireworks Algorithm

In FWA, the explosion amplitude of each fireworks and the number of sparks produced by the explosion are calculated based on its relative fireworks fitness values in the fireworks population. Assume that the number of fireworks is N and the number of dimensions is d, then the explosion amplitude A_i (Equation (1)) and the number of explosion sparks S_i (Equation (2)) for each firework X_i are calculated as follows.

$$A_i = A \times \frac{f(X_i) - y_{\min} + \varepsilon}{\sum\limits_{i=1}^{N} (f(X_i) - y_{\min}) + \varepsilon} \tag{1}$$

$$S_i = m \times \frac{y_{\max} - f(X_i) + \varepsilon}{\sum\limits_{i=1}^{N} (y_{\max} - f(X_i)) + \varepsilon} \tag{2}$$

where $y_{\max} = \max(f(X_i))$, $y_{\min} = \min(f(X_i))$, A and m are two constants to control the explosion amplitude and the number of explosion sparks, and ε is the machine epsilon to avoid A_i or S_i is equal to 0.

In order to limit the good fireworks do not produce too many explosive sparks, while the poor fireworks do not produce too few sparks, its scope S_i is defined as.

$$S_i = \begin{cases} round(a \times m), & S_i < a \times m \\ round(b \times m), & S_i > b \times m \\ round(S_i), & otherwise \end{cases} \tag{3}$$

where a and b are fixed constant parameters that confine the range of the population size.

Based on A_i and S_i, the explosion operator is performed (confer Algorithm 1). For each of the Si explosion sparks of each firework X_i, Algorithm 1 is performed once. In Algorithm 1, the operator % refers to the modulo operation, and ${X_{min}}^k$ and ${X_{max}}^k$ refer to the lower and upper bounds of the search space in dimension k.

Algorithm 1: Generating Explosion Sparks

```
Initialize the location of the explosion sparks: $X_j = X_i$
Calculate the number of explosion sparks $S_i$
Calculate the explosion amplitude $A_i$
Set $z$ = rand(1,$d$)
For $k$ = 1:$d$ do
  If $k \in z$ then
    $X_j^k = X_j^k$ + rand(0,$A_i$)
    If $X_j^k$ out of bounds
      $X_j^k = X_{min}^k + |X_j^k| \% (X_{max}^k - X_{min}^k)$
    End if
  End if
End for
```

In order to increase the diversity of the population, the fireworks algorithm introduced mutation operator used to generate mutation sparks, namely Gaussian mutation sparks. Gaussian mutation of the spark produced as follows: First of all in the fireworks population randomly select a fireworks X_i, and then the fireworks randomly select a certain number of dimensions for Gaussian mutation operation. Algorithm 2 shows this process, it is performed N_g times, each time with a randomly selected firework X_i (N_g is a constant to control the number of Gaussian sparks).

Algorithm 2: Generating Gaussian Sparks

```
Initialize the location of the explosion sparks: $X_j = X_i$
Calculate offset displacement: g = Gaussian(1,1)
Set $z$ = rand(1,$d$)
For $k$ = 1:$d$ do
  If $k \in z$ then
    $X_j^k = X_j^k \times g$
    If $X_j^k$ out of bounds
      $X_j^k = X_{min}^k + |X_j^k| \% (X_{max}^k - X_{min}^k)$
    End if
  End if
End for
```

In order to transmit the excellent information in the fireworks population to the next generation population, the algorithm chooses a certain number of individuals (including fireworks, explosion sparks and Gaussian sparks) as the next generation fireworks.

In FWA, the current best location is always kept for the next iterations. In order to keep the diversity, the remaining $N - 1$ locations are selected based on the method of roulette. For location X_i, the selection probability $\mathrm{p_i}$ is calculated as follows:

$$p(X_i) = \frac{R(X_i)}{\sum\limits_{j \in K} R(X_j)} \tag{4}$$

$$R(X_i) = \sum_{j \in K} d(X_i, X_j) = \sum_{j \in K} ||X_i - X_j|| \tag{5}$$

where K is the set of all current locations including original fireworks and both types of sparks (without the best location). As a result, fireworks or sparks in low crowded regions will have a higher probability to be selected for the next iteration than fireworks or sparks in crowded regions.

Algorithm 3 summarizes the framework of FWA. Based on Algorithms 1 and 2, explosion sparks and Gaussian sparks are produced, respectively. For explosion sparks, the number of sparks and the amplitude of the explosion depend on the fitness value of the fireworks. In contrast, Gaussian sparks are generated by using the Gaussian mutation process. After that, a best location and $n-1$ locations are selected for the next explosion.

Algorithm 3: Construction of FWA

```
    Initialize $n$ positions of fireworks randomly
    While Stop criterion is false
      For each firework $X_i$
        Calculate the explosion amplitude $A_i$ by Equation (1)
        Calculate the number of explosion sparks $S_i$ by Equation (2)
        Generating explosion sparks based on Algorithm 1
      End for
      For $k$ = 1:$N_g$ do
        Select a firework $X_i$ randomly
        Generating Gaussian sparks based on Algorithm 2
      End for
      Save the best position to the next explosion
      Based on the given probability by Equation (4), select $n-1$ position randomly from two sparks and the
current fireworks.
    End while
```

3. Analysis and Improvement of Fireworks Algorithm

3.1. Opposition-Based Learning Population Initialization

Opposition-based learning (OBL) is first proposed by Tizhoosh [13]. OBL simultaneously considers a solution and its opposite solution; the fitter one is then chosen as a candidate solution in order to accelerate convergence and improve solution accuracy. It has been used to enhance various optimization algorithms, such as the differential evolution [14], the particle swarm optimization [15], the firefly algorithm [16], the adaptive fireworks algorithm [17] and the quantum firework algorithm [18]. Inspired by these, OBL was add to FWA to initialize population.

Definition 1. Assume X = (x_1,x_2,...,x_d) is a solution with d dimensions, where x_1,x_2,...,x_d∈R and x_i∈[L_i,U_i], i=1,2,...,d. The opposite solution OX = (ox_1,ox_2,...,ox_d) is defined as follows:

$$ox_i = L_i + U_i - x_i \tag{6}$$

In fact, according to probability theory, 50% of the time an opposite solution is better. Therefore, based on a solution and an opposite solution, OBL has the potential to accelerate convergence and improve solution accuracy.

In the population initialization, both a random solution and an opposite solution OP are considered to obtain fitter starting candidate solutions.

Algorithm 4 is performed for opposition-based population initialization as follows.

Algorithm 4: Opposition-Based Population

```
Initialize fireworks $P$ with a size of $N$ randomly
Calculate an opposite fireworks OP based on Equation (6)
Assess $2 \times N$ fireworks' fitness
Choose the fittest individuals from P and OP as initial fireworks
```

3.2. Analysis and Improvement of Explosion Amplitude

The main purpose of Equation (1) is that the explosion amplitude of the fireworks is inversely proportional to the fitness value of the function. It enhances the local search ability of the fireworks. However, if we apply the optimal fireworks into Equation (1), the result as follows.

$$A_i = A \times \frac{\varepsilon}{\sum\limits_{i=1}^{N} (f(X_i) - y_{\min}) + \varepsilon} \tag{7}$$

Since the numerator is the smallest constant expressed in the computer, the result of the Equation (7) is equal to 0. It is obviously inconsistent with the original design intent of the fireworks algorithm. The fireworks algorithm requires the optimal firework generated the largest number of sparks, i.e., sparks do not create much searches but increase the amount of calculation in vain.

To solve this problem, [10] gives the linear decreasing (Equation (8)) and non-linear decreasing (Equation (9)) explosion amplitude strategies as follows.

$$A_i = A_{init} - (A_{init} - A_{final}) \times \frac{t}{T} \tag{8}$$

$$A_i = A_{init} - \frac{A_{init} - A_{final}}{T} \times \sqrt{(2T - t)t} \tag{9}$$

where T is the maximum number of iterations, t is the number of iterations, A_{init} and A_{final} are the initial and final value of the explosion amplitude respectively.

Although the two strategies effectively avoid the optimal firework explosion amplitude approaching to 0, it relies on the maximum number of iterations heavily which needs to be set manually. Based on this, we propose a new method to calculate the explosion amplitude for the optimal firework. According to the aspects of population evolution rate and population aggregation degree, this method dynamically changes the explosion amplitude.

Definition 2. Assume the t generation global optimal value is denoted as $Y_{min}(t)$, the global optimal value of the $t - 1$ generation is $Y_{min}(t - 1)$. The population evolution rate $a(t)$ is defined as follows.

$$0 < a(t) = \frac{Y_{\min}(t) + \varepsilon}{Y_{\min}(t-1) + \varepsilon} \leq 1 \tag{10}$$

Fireworks algorithm retains the optimal fireworks for each iteration, the current global optimal value is always better than or equal to the global optimal value of the last iteration. From Equation (10), the value of $a(t)$ changes greatly means that the evolution speed is fast. When $a(t)$ is equal to 1 after several iterations, it indicates that the algorithm stagnates or finds the optimal value.

Definition 3. Assume the t generation global optimal value is denoted as $Y_{min}(t)$, the average fitness value of all fireworks in the t generation is denoted as $Y_{avg}(t)$. The population aggregation degree $b(t)$ is defined as follows.

$$0 < b(t) = \frac{Y_{\min}(t) + \varepsilon}{Y_{avg}(t) + \varepsilon} \leq 1 \tag{11}$$

From Equation (11), $b(t)$ is larger, indicating the distribution of fireworks in the population more concentrated.

According to Definitions 2 and 3 can be clearly reflected optimization process in FWA. If we adjust explosion amplitude of the optimal firework with the population evolution rate and population aggregation degree, it means combining the explosion amplitude and optimization process.

When *a(t)* is small, the evolution speed is fast, and the algorithm can search in a large space. That is, the optimal firework can be optimized in a large scope; when $a(t)$ is too large, the search is performed in a small scope to find the optimal solution faster.

When *b(t)* is small, the fireworks are scattered and are less likely to fall into local optima, which is more likely to happen when *b(t)* assumes greater values. At this time, it is necessary to increase the explosion amplitude to increase the search space and improve the global searching ability of FWA.

To sum up, the explosion amplitude should decrease as the population evolution rate increases, and increase with population aggregation degree increases. This paper describes this phenomenon in a simplified way.

$$\begin{cases} A_i = A_i \times up, & a(t) \neq 1 or\, b(t) = 1 \\ A_i = A_i \times low & otherwise \end{cases} \tag{12}$$

where A_i is the explosion amplitude , and the initial value is set as the size of the objective function search space. *up* is the enlargement factor, *low* is the reduction factor. Of course, larger *up* and smaller *low* cannot help to search exactly. Thus, *up* and *low* should be set a fit value.

This improvement is discussed in the following section:

1. When the *a(t)* is not equal to 1, it means the algorithm finds a better solution than the last generation, and the explosion amplitude should be enlarged. We emphasize that increasing the explosion amplitude may speed up the rate of convergence: assume that the current optimal firework is far from the global optimum. Increasing the explosion amplitude is a direct and efficient way to help the algorithm move faster towards global optimization. However, it should also be noted that the probability of finding a better firework will decrease as the search space increases (obviously, this depends on the optimization function to a large extent).
2. When the *b(t)* is equal 1, it means the algorithm may fall into local optima, and the fireworks are concentrated, increasing the explosion amplitude to make the fireworks are scattered, which help the algorithm jump out the local optima effectively.
3. When the *a(t)* is equal 1 and the *b(t)* is not equal 1, it means the algorithm does not find out a better solution than the last generation and the fireworks are scattered. In this case, the optimal firework explosion amplitude will be reduced to narrow the search to a smaller area, thereby enhancing local development capability of the optimal firework. In general, the probability of finding a better firework increases as the explosion amplitude decreases.

In this paper, the explosion amplitude of the optimal firework is calculated by Equation (12). In contrast, the explosion amplitude of non-optimal fireworks is calculated still by Equation (1). Algorithm 5 is performed for updating explosion amplitude as follows.

Algorithm 5: Update Explosion Amplitude

```
Find the optimal firework from all fireworks of t generation : $X_{best}$
Calculate the fitness of the optimal firework of t generation : $Y_{min}(t)$
Calculate the fitness of the optimal firework of last iteration: $Y_{min}(t-1)$
Calculate the average fitness value of all fireworks of t generation : $Y_{avg}(t)$
Calculate the population evolution rate : $a(t)$
Calculate the population aggregation degree : $b(t)$
For the optimal firework:
If $a(t) \neq 1$ or $b(t) = 1$ then
    $A_i = A_i \times$ up
else
    $A_i = A_i \times$ low
End if
For the non-optimal fireworks:
$\mathrm{A_i}$ is calculated by Equation (1)
```

Figure 1 depicts the process of explosion enlargement and reduction during the optimization of the Sphere function. An alternating behavior is noted, with reduction being performed more often, on the one hand because *up* and *low* are set to 1.2 and 0.9, on the other hand because the initial value of the explosion amplitude is set to the size of the search space, which is a considerable initial value.

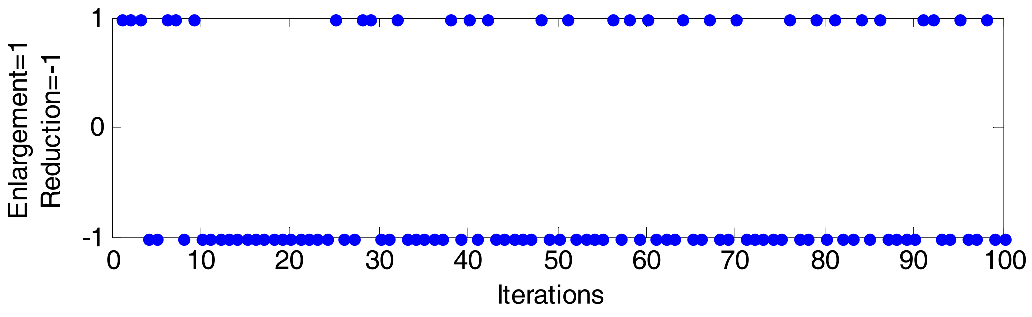


Figure 1. Enlargement and reduction of explosion amplitude (on the Sphere function).

3.3. *Analysis and Improvement of Gaussian Mutation*

Zheng pointed out the shortcomings of Gaussian mutation in FWA [10], and proposed a new type to generating location of Gaussian sparks, which is calculated as follows.

$$x_i^{\ k} = X_i^{\ k} + (X_b^{\ k} - X_i^{\ k}) \times g \tag{13}$$

where g = Gaussian(0,1), $X_b^{\ k}$ is the position of the optimal firework in the k dimension of the current fireworks population.

Cauchy mutation has a strong global search ability due to larger search range, and Gaussian mutation has a strong local development ability with small search range [19]. Therefore, the advantage of Equation (13) is to improve the local development capability of the algorithm, and does not improve the global search ability in the early stage of algorithm. Zhou pointed out t-distribution mutation combined with the two advantages of the Cauchy and Gaussian mutation [20], which has a strong global search ability in the early stage of algorithm and a good local development ability in the later stage of algorithm.

From Equation (13), when the optimal firework of the current population is selected for Gaussian mutation exactly, apply it into Equation (13).

$$x_i^{\ k} = X_i^{\ k} \tag{14}$$

As we know, the optimal firework is the best information for the current population carrier, but Gaussian mutation does not have any effect on the optimal firework in Equation (14).

To sum up, the adaptive t-distribution mutation is proposed for non-optimal fireworks to effectively keep a better balance between exploration and exploitation. Elite opposition-based learning for optimal firework to make the FWA jump out of the local optimum effectively and accelerate the global search ability.

3.3.1. Adaptive t-Distribution Mutation

T-distribution, also known as the student's t-distribution, includes n degrees of freedom. When $t(n \to \infty)$, it is equal to Gaussian(0,1); when $t(n \to 1)$, it is equal to Cauchy(0,1). That is the Gaussian distribution and the Cauchy distribution are two boundary special cases of t-distribution [20].

Definition 4. Adaptive *t*-distribution mutation for non-optimal fireworks is used to generate location of sparks as follows.

$$x_i{}^k = X_i{}^k + (X_b{}^k - X_i{}^k) \times t(n) \tag{15}$$

where n is the number of iterations, that is the number of iterations is the freedom of *t*-distribution.

Algorithm 6 is performed for Adaptive *t*-distribution mutation for non-optimal fireworks to generate location of sparks as follows.

Algorithm 6: Generating *t*-Distribution Mutation Sparks

```
Initialize the location of the explosion sparks: $X_j = X_i$
Set $z$ = rand(1,$d$)
For $k$ = 1:$d$ do
  If $k \in z$ then
    $X_j{}^k = X_j{}^k + (X_b{}^k - x_j{}^k) \times t(n)$
    If $X_j{}^k$ out of bounds
      $X_j{}^k = X_{min}{}^k + |X_j{}^k| \% (X_{max}{}^k - X_{min}{}^k)$
    End if
  End if
End for
```

In the early stage of the algorithm, the value of n is small and the *t*-distribution mutation is similar to Cauchy distribution mutation, and it has a good global exploratory ability. In the later stage of the algorithm, the value of n is large, and the *t*-distribution mutation is similar to Gaussian distribution mutation, and it has a good local development ability. In the mid-run of the algorithm, the *t*-distribution mutation is between the Cauchy distribution mutation and the Gaussian distribution mutation. Therefore, the *t*-distribution combines the advantages of Gaussian distribution and Cauchy distribution, balancing the exploration and exploitation.

3.3.2. Elite Opposition-Based Learning

The basic idea of opposition-based learning is as follows: for a feasible solution, we evaluate the opposition-based solution simultaneously, and the optimal solution is selected as the next generation in the current feasible solution and opposition-based solution. Opposition-based learning keeps the diversity of population but large, if all the fireworks produce opposition-based solution, it is blind and increasing the amount of calculation. Therefore, here we choose the optimal individual to perform opposition-based learning.

Definition 5. Assume $X_{best} = (x_{best,1}, x_{best,2}, \ldots, x_{best,d})$ is a solution of the optimal firework with d dimensions, where $x_{best,1}, x_{best,2}, \ldots, x_{best,d} \in R$ and $x_i \in [\min_i, \max_i]$, $i = 1,2,\ldots,d$. The opposite solution $OX_{best} = (ox_{best,1}, ox_{best,2}, \ldots, ox_{best,d})$ is defined as follows.

$$ox_{best,i} = rand \times (\min_i + \max_i) - x_{best,i} \tag{16}$$

where *rand* is a uniform distribution on the interval [0, 1], and $\min_i$ and $\max_i$ are the minimum and maximum values of the current search interval on the i dimension.

By Definition 5, *rand* is a uniformly distributed random number on [0, 1]. When rand takes different values, the optimal firework from the current population can produce a number of different optimal opposition-based fireworks, which are effective in increasing the diversity of the population and avoid the algorithm getting into the local optimal solution.

Algorithm 7 is performed for elite opposition-based learning for optimal firework to generate location of sparks. This algorithm is performed N_{op} times (N_{op} is a constant to control the number of elite opposition-based sparks).

Algorithm 7: Generating Elite Opposition-Based Sparks

```
Find the location of optimal firework: $X_{best} = (x_1,x_2,...,x_d)$
For $i$ = 1:$d$ do
    Find $\min_i$ and $\max_i$ of the current search interval on the $i$ dimension
    $ox_{best,I}$ = rand × ($\min_i + \max_i$) − $x_{best,i}$
    If $ox_i$ out of bounds
    $ox_i = X_{min}{}^{i} + |ox_i| \% (X_{max}{}^{i} - X_{min}{}^{i})$
    End if
End for
```

3.4. Analysis and Improvement of Selection Strategy

From Equations (4) and (5), the selection strategy is based on the distance measure in FWA. However, this requires that the euclidean distance matrix between any two points in each generation, which will lead to fireworks algorithm time consuming. Based on this, this paper proposes a new selection strategy: Elitism-Disruptive selection strategy.

The same as FWA, the Elitism-Disruptive selection also requires that the current best location is always kept for the next iterations. In order to keep the diversity, the remaining $N-1$ locations are selected based on disruptive selection operator. For location X_i, the selection probability p_i is calculated as follows [21]:

$$p_i = \frac{Y_i}{\sum\limits_{n=1}^{SN} Y_n} \tag{17}$$

$$Y_i = \left|Y_i - Y_{avg}\right| \tag{18}$$

where Y_i is the fitness value of the objective function, Y_{avg} is the mean of all fitness values of the population in generation t, SN is the set of all fireworks.

The selection probabilities determined by this method can give both good and poor individuals more chances to be selected for the next iteration, while individuals with mid-range fitness values will be eliminated. This method can not only maintain the diversity of the population, reflect the better global searching ability, but also reflect greatly reduce the run-time compared with the FWA.

4. Global Convergence Analysis of IFWA

Tan studied the convergence of the fireworks algorithm, the main conclusions are as follows [2]:

Lemma 1. *The random process of FWA ($\{\varepsilon(t)\}_{t=0}^{\infty}$) is an absorbing Markov random process [2].*

Definition 6. Given an absorption Markov process ($\{\varepsilon(t)\}_{t=0}^{\infty}$) and an optimization state space ($Y^*\subset Y$). $\lambda(t) = P\{\varepsilon(t)\in Y^*\}$ represents the probability of reaching the optimal state at the t time. If $\lim\limits_{t\to\infty}\lambda(t) = 1$, $\{\varepsilon(t)\}_{t=0}^{\infty}$ convergence [2].

Theorem 1. *Given an absorption Markov process ($\{\varepsilon(t)\}_{t=0}^{\infty}$) and an optimization state space ($Y^*\subset Y$), for$\forall t$. If $P\{\varepsilon(t) \in Y^*|\varepsilon(t-1) \notin Y^*\} \geq d \geq 0$ and $P\{\varepsilon(t) \in Y^*|\varepsilon(t-1) \in Y^*\} = 1$, $P\{\varepsilon(t) \in Y^*\} \geq 1-(1-d)^t$ [2].*

Based on this, the global convergence of IFWA is given as follows:

IFWA contains a t-distribution mutation, which is assumed to be a random variation for simplicity.

Theorem 2. *Given an absorption Markov process ($\{\varepsilon(t)\}^{\infty}{}_{t=0}$) of IFWA and an optimization state space ($Y^*\subset Y$). The conclusion as follows:*

$$\lim_{t\to\infty}\lambda(t) = 1 \tag{19}$$

Proof of Theorem 2. Assume P_t is the probability that a firework from the non-optimal region to the optimal region R_{best} in IFWA, under the action of t-distribution mutation:

$$P_t = \frac{\nu(R_{best}) \times N}{\nu(S)} \tag{20}$$

where $\nu(S)$ is the Lebesgue measure of the problem space S; N is the number of fireworks.

Obviously, $\nu(R_{best}) > 0$, so $P_t > 0$;

Based on random Markov process of IFWA, the conclusion as follows:

$$\lambda(t) = P\{\varepsilon(t) \in Y^*|\varepsilon(t-1) \notin Y^*\} = P_t + P_e \tag{21}$$

where P_e is the probability that a firework from the non-optimal region to the optimal region R_{best} in IFWA, under the action of explosion.

From Equation (21), the conclusion is as follows:

$$P\{\varepsilon(t) \in Y^*|\varepsilon(t-1) \notin Y^*\} \geq P_t \geq 0 \tag{22}$$

Because the iterative process of the IFWA retains the optimal firework, that is, if the optimal firework is the global solution in the last iterative process, the optimal firework must be the global solution in the current iterative process. The conclusion as follows:

$$P\{\varepsilon(t) \in Y^*|\varepsilon(t-1) \in Y^*\} = 1 \tag{23}$$

And because the Markov process of IFWA is an absorbing Markov process, the condition of theorem 1 is satisfied, so have the follow conclusion:

$$P\{\varepsilon(t) \in Y^*\} = 1 - (1 - P_t)^t \tag{24}$$

that is, $\lim\limits_{t\to\infty} P\{\varepsilon(t) \in Y^*\} = \lim\limits_{t\to\infty}(1 - (1 - P_t)^t) = 1$

Based on Definition 6, the Markov process of IFWA will converge to the optimal state.

5. Simulation Results and Analysis

To assess the performance of IFWA, it is compared with FWA [8], EFWA [10], dynFWA [12] and SPSO2011 [22].

5.1. Simulation Settings

Similar to FWA, the number of fireworks in IFWA is set to 5; and the number of elite opposition-based sparks is also set to 5; but in IFWA, the maximum number of sparks each generation is set to 200. The reduction and amplification factors of IFWA are set to 0.9 and 1.2 based on experience. The explosion amplitude is initialized to the size of the search space to keep the high exploratory ability at the beginning of the algorithm. FWA parameters set in accordance with [1]. EFWA parameters set in accordance with [10]. dynFWA parameters set in accordance with [12]. SPSO2011 parameters set in accordance with [22].

In the experiment, the function of each algorithm is repeated 51 times, and the final results after the 300,000 function evaluations are presented. In order to verify the performance of the algorithm proposed in this paper, we use the CEC2013 test set [23], including 28 different types of test functions, which are listed in Table 1. All experimental test functions dimensions are set to 30, $d = 30$.

Finally, we use Matlab R2014a software on a PC with a 3.2 GHz CPU (Intel Core i5-3470), and 4 GB RAM, and Windows 7(64 bit).

Table 1. CEC2013 test set.

	Function Number	Function Name	Optimal Value
Unimodal Functions	1	Sphere function	−1400
	2	Rotated high conditioned elliptic function	−1300
	3	Rotated bent cigar function	−1200
	4	Rotated discus function	−1100
	5	Different powers function	−1000
Basic Multimodal Functions	6	Rotated rosenbrock's function	−900
	7	Rotated schaffers F7 function	−800
	8	Rotated Ackley's function	−700
	9	Rotated weierstrass function	−600
	10	Rotated griewank's function	−500
	11	Rastrigin's function	−400
	12	Rotated rastrigin's function	−300
	13	Non-continuous rotated rastrigin's function	−200
	14	Schewefel's function	−100
	15	Rotated schewefel's function	100
	16	Rotated katsuura function	200
	17	Lunacek Bi_Rastrigin function	300
	18	Rotated Lunacek Bi_Rastrigin function	400
	19	Expanded griewank's plus rosenbrock's function	500
	20	Expanded scaffer's F6 function	600
Composition Functions	21	Composition function 1 (N = 5)	700
	22	Composition function 2 (N = 3)	800
	23	Composition function 3 (N = 3)	900
	24	Composition function 4 (N = 3)	1000
	25	Composition function 5 (N = 3)	1100
	26	Composition function 6 (N = 5)	1200
	27	Composition function 7 (N = 5)	1300
	28	Composition function 8 (N = 5)	1400

5.2. Simulation Results and Analysis

5.2.1. Verify Each Improvement

This paper proposed the below four improvements:

1. The opposition-based learning used to initialize population.
2. A new mechanism to adjust an explosion amplitude of the optimal firework.
3. *t*-distribution mutation for non-optimal fireworks, and the elite opposition-based learning for optimal firework.
4. A new selection strategy, called disruptive selection, is used to select next generation.

This section verifies each improvement to compare with FWA, the results are shown in Table 2. The FWA is the basic fireworks algorithm, FWA-I is basic fireworks algorithm with improvement 1, FWA-II is basic fireworks algorithm with improvements 1 and 2, FWA-III is basic fireworks algorithm with improvements 1–3, and the IFWA is the FWA with all improvements.

Table 2. Average fitness value and total number of rank 1.

Functions		**FWA**	**FWA-I**	**FWA-II**	**FWA-III**	**IFWA**
f1	Mean	−1396.7	−1396.81	**−1400**	**−1400**	**−1400**
f2	Mean	2.3×10^7	2.21×10^7	$\mathbf{3.62 \times 10^5}$	3.39×10^6	4.03×10^5
f3	Mean	7.2×10^9	7.0×10^9	6.76×10^8	4.92×10^8	$\mathbf{1.21} \times 10^8$
f4	Mean	2.18×10^4	2.13×10^4	1.33×10^4	1.47×10^4	**−1099.89**
f5	Mean	−997.58	−997.71	**−1000**	**−1000**	**−1000**
f6	Mean	−815	−820.71	−856.34	−859	**−872**
f7	Mean	−639	−646.13	−614.85	−528	**−709**
f8	Mean	−679.06	−679.07	−679.07	−679.08	**−679.13**
f9	Mean	−565.52	−566.13	−566.46	−566.78	**−576.12**
f10	Mean	-464.8	−468.12	−499.63	−499.70	**−499.978**
f11	Mean	−384.10	**−385.32**	−368.52	−348.19	−304.89
f12	Mean	114.19	121.19	8.43	6.78	**−158.02**
f13	Mean	191.23	185.04	165.74	159.10	**−1.124**
f14	Mean	647.11	723.65	**609.74**	797.02	2644.91
f15	Mean	5014.04	5012.96	4473.48	4601.22	**3930.46**
f16	Mean	201.73	201.64	201.48	201.41	**200.377**
f17	Mean	**357.08**	357.16	365.53	399.63	410.71
f18	Mean	825.03	816.93	784.66	773.59	**575.27**
f19	Mean	**505.4**	505.85	506.20	507.29	506.6
f20	Mean	614.76	614.57	614.15	614.06	**612.38**
f21	Mean	1082.4	1081.18	1050.15	1027.13	**1008.53**
f22	Mean	1528.44	1506.69	1488.48	1755.95	**1488.47**
f23	Mean	7009.33	6996.97	6270.16	6217.96	**3294.51**
f24	Mean	1307.75	1307.54	1306.38	1279.66	**1266.55**
f25	Mean	1458.45	1436.27	1431.78	1430.13	**1387.58**
f26	Mean	1419.42	1418.34	**1400.11**	1415.18	1409.01
f27	Mean	2582.52	2580.52	2555.85	2534.58	**2224.13**
f28	Mean	4647.6	4645.67	3627.81	3245.98	**1640.48**
total number of rank 1						
		2	1	5	2	22

As Table 2 shows, FWA-I, FWA-II, FWA-III and IFWA compared to the fireworks algorithm have different degrees of performance, and the IFWA shows better performance.

5.2.2. Searching Curves Comparison

Due to limited space, this paper selects eight functions which have great difference in evolution speed in five algorithms. Figure 2 shows searching curves of eight functions for FWA, EFWA, dynFWA, SPSO2011 and IFWA. The Figure A1 shows the searching curves of remaining 20 functions.

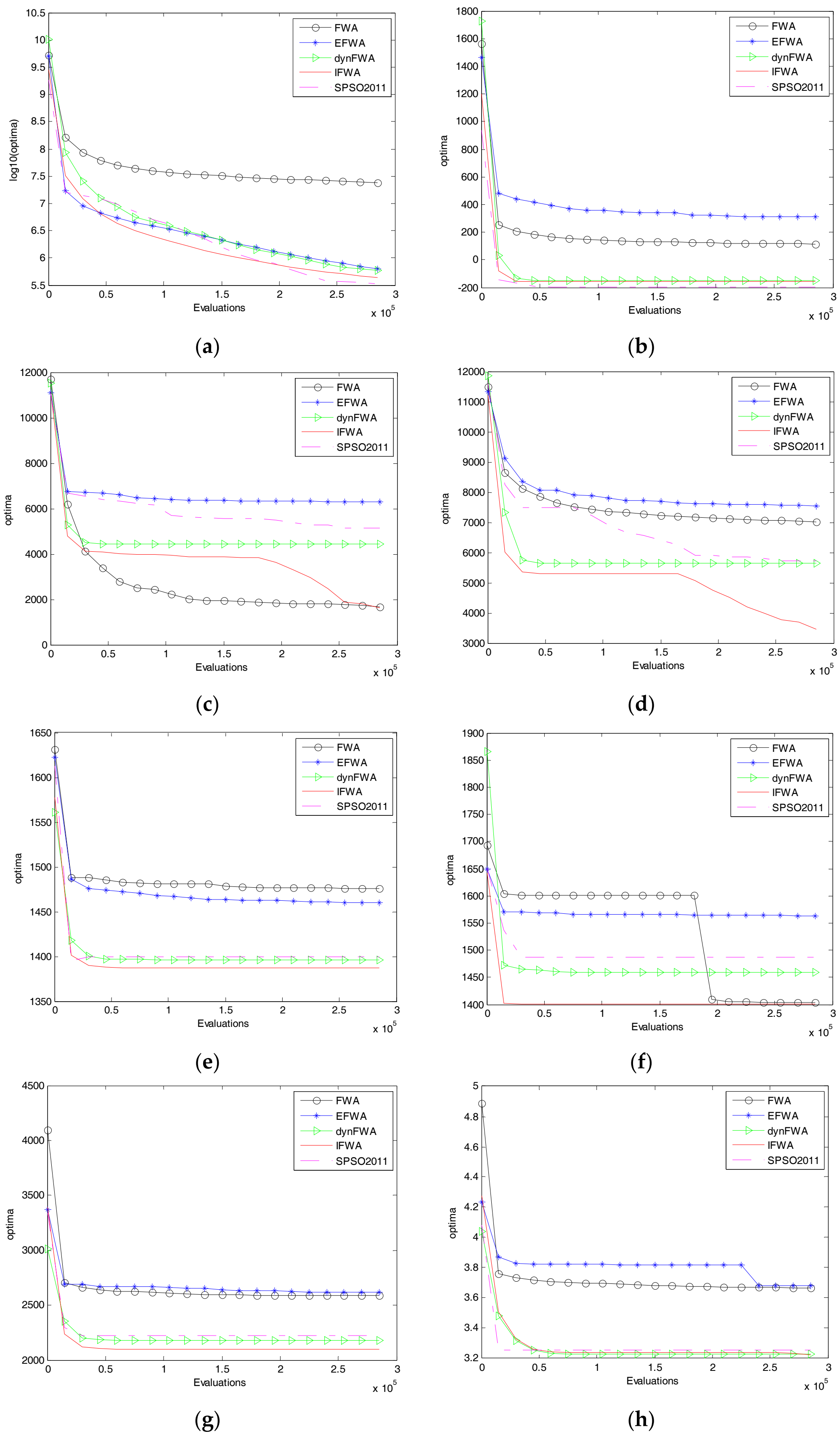


Figure 2. The FWA, EFWA, dynFWA, SPSO2011 and IFWA searching curves. (**a**) f2 function; (**b**) f12 function; (**c**) f22 function; (**d**) f23 function; (**e**) f25 function; (**f**) f26 function; (**g**) f27 function; (**h**) f28 function.

Figure 2 shows that IFWA have faster convergence for eight functions than FWA, EFWA, dynFWA. However, in f2 and f12, the SPSO2011 is better, and in other functions, IFWA still has faster convergence. Thus, IFWA is the best one in terms of solution accuracy on most functions.

5.2.3. Comparison of Average Fitness Value and Average Rank

Table 3 shows comparison of average fitness value and total number of rank 1 for FWA, EFWA, dynFWA, SPSO2011 and IFWA.

Table 3. Average fitness value and total number of rank 1.

Functions		**SPSO2011**	**FWA**	**EFWA**	**dynFWA**	**IFWA**
f1	Mean	**−1400**	−1396.7	−1399	**−1400**	**−1400**
	Rank	1	3	2	1	1
f2	Mean	$\mathbf{3.371} \times 10^5$	2.3×10^7	6.85×10^5	8.69×10^5	4.03×10^5
	Rank	1	5	3	4	2
f3	Mean	2.88×10^8	7.2×10^9	$\mathbf{7.76} \times 10^7$	1.23×10^8	1.21×10^8
	Rank	4	5	1	3	2
f4	Mean	3.75×10^4	2.18×10^4	−1098.9	−1089.6	**−1099.89**
	Rank	5	4	2	3	1
f5	Mean	**−1000**	−997.58	−999.92	**−1000**	**-1000**
	Rank	1	3	2	1	1
f6	Mean	−862	-815	−850	−869	**−872**
	Rank	3	5	4	2	1
f7	Mean	**−712**	-639	−627	−700	−709
	Rank	1	4	5	3	2
f8	Mean	-679.08	−679.06	−679.07	−679.10	**−679.13**
	Rank	3	5	4	2	1
f9	Mean	−571.23	−565.52	−568.46	−575.87	**−576.12**
	Rank	3	5	4	2	1
f10	Mean	−499.66	−464.8	−499.16	−499.95	**−499.978**
	Rank	3	5	4	2	1
f11	Mean	−295.04	**−384.10**	5.8198	−295.89	−304.89
	Rank	4	1	5	3	2
f12	Mean	**−196.04**	114.19	399.44	−142.22	−158.02
	Rank	1	4	5	3	2
f13	Mean	**−6.1406**	191.23	298.57	53.83	−1.124
	Rank	1	4	5	3	2
f14	Mean	3891	**647.11**	2724	2918	2644.91
	Rank	5	1	3	4	2
f15	Mean	**3909.3**	5014.04	4459.5	4022.7	3930.46
	Rank	1	5	4	3	2
f16	Mean	201.31	201.73	200.63	200.58	**200.377**
	Rank	4	5	3	2	1
f17	Mean	416.26	**357.08**	624.61	442.61	410.71
	Rank	3	1	5	4	2
f18	Mean	**520.63**	825.03	576.61	587.82	575.27
	Rank	1	5	3	4	2
f19	Mean	509.51	**505.4**	510.22	507.26	506.6
	Rank	4	1	5	3	2
f20	Mean	613.46	614.76	614.66	613.28	**612.38**
	Rank	3	5	4	2	1
f21	Mean	1008.8	1082.4	1117.8	1010.2	**1008.53**
	Rank	2	4	5	3	1

Table 3. *Cont.*

Functions		SPSO2011	FWA	EFWA	dynFWA	IFWA
f22	Mean	5098.8	1528.44	6318.1	4126.2	**1488.47**
	Rank	4	2	5	3	1
f23	Mean	5731.3	7009.33	7580.9	5652.6	**3294.51**
	Rank	3	4	5	2	1
f24	Mean	1266.7	1307.75	1345.2	1272.9	**1266.55**
	Rank	2	4	5	3	1
f25	Mean	1399.3	1458.45	1442.6	1397	**1387.58**
	Rank	3	5	4	2	1
f26	Mean	1486.1	1419.42	1546.1	1460.7	**1409.01**
	Rank	4	2	5	3	1
f27	Mean	2304.6	2582.52	2621	2280.4	**2224.13**
	Rank	3	4	5	2	1
f28	Mean	1801.3	4647.6	4765.1	1696.1	**1640.48**
	Rank	3	4	5	2	1
total number of rank 1						
		8	4	1	2	17

The results from Table 3 indicate that the total number of rank 1 of IFWA (17) is the best in the five algorithms.

5.2.4. Comparison of Average Run-Time Cost

Figures 3 and 4 show comparison of average run-time cost in 28 functions for FWA, EFWA, dynFWA, SPSO2011 and IFWA.

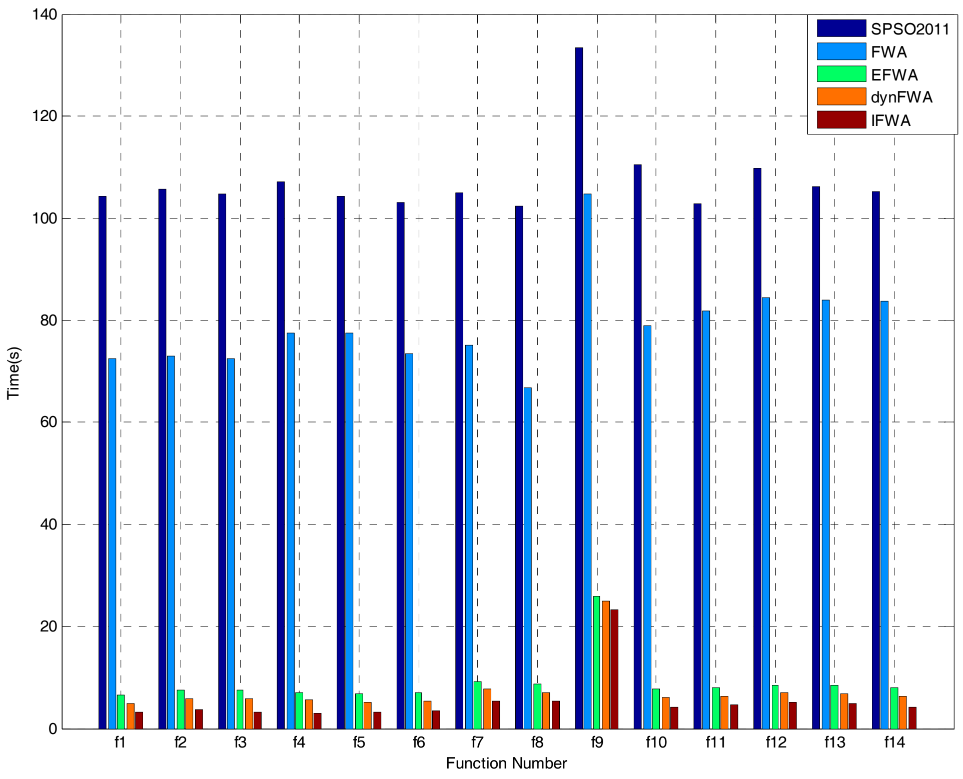


Figure 3. The FWA, EFWA, dynFWA, SPSO2011 and IFWA run-time cost.

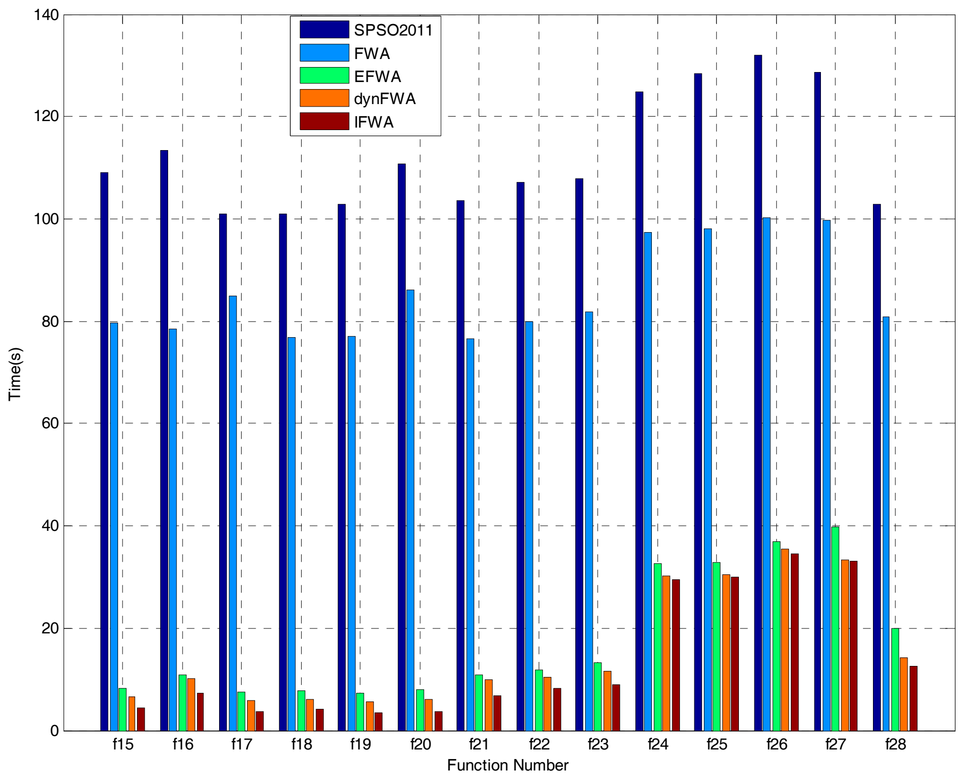


Figure 4. The FWA, EFWA, dynFWA, SPSO2011 and IFWA run-time cost (Continued).

The results from Figures 3 and 4 indicate that the average run-time cost of SPSO2011 is the most expensive among the five algorithm. The time cost of IFWA is the least.

5.2.5. Comparison of Statistical Test

To evaluate whether the IFWA results were significantly different from those of the FWA, EFWA, dynFWA and SPSO2011, the IFWA mean results during iteration for each test function were compared with those of the FWA, EFWA, dynFWA and SPSO2011. The *T* test [24], which is a safe and robust, was utilized at the 5% level to detect significant differences between these pairwise samples for each test function.

The ttest2 function in Matlab R2014a was used to run the T test, as shown in Table 4. The null hypothesis is that the results of IFWA, SPSO2011, FWA, EFWA and dynFWA are derived from distributions of equal mean, and in order to avoid to increase type I error, we correct the *p*-values using the Holm's method, and order the *p*-value for the four hypotheses being tested from smallest to largest, and here we have four t tests. Thus, the *p*-value 0.05 is changed to 0.0125, 0.0167, 0.025, and 0.05, and then the corrected *p*-values were used to compare with p-values respectively.

Here *p*-value is the result of the T test. The "+" indicates the rejection of the null hypothesis at the 5% significance level, and the "−" indicates accept the null hypothesis at the 5% significance level.

Table 4 indicates that IFWA showed a large improvement over FWA, EFWA, dynFWA and SPSO2011 in most functions.

Table 4. T test results of IFWA compare with SPSO2011, FWA, EFWA, dynFWA.

Functions		**SPSO2011**	**FWA**	**EFWA**	**dynFWA**
f1	*p*-value	NaN	0	0	NaN
	significance	−	+	+	−
f2	*p*-value	0.0045	1.102×10^{-201}	5.186×10^{-22}	1.0556×10^{-37}
	significance	+	+	+	+
f3	*p*-value	7.9152×10^{-9}	1.352×10^{-145}	0.0666	0.9222
	significance	+	+	−	−
f4	*p*-value	0	0	4.251×10^{-102}	3.192×10^{-202}
	significance	+	+	+	+
f5	*p*-value	NaN	0	0	NaN
	significance	−	+	+	−
f6	*p*-value	0.0141	4.73×10^{-29}	4.7756×10^{-8}	0.6126
	significance	+	+	+	−
f7	*p*-value	0.4621	6.677×10^{-35}	1.9078×10^{-40}	0.0140
	significance	−	+	+	+
f8	*p*-value	0.4081	0.2895	0.9070	0.0076
	significance	−	−	−	+
f9	*p*-value	3.325×10^{-13}	1.9057×10^{-33}	1.7148×10^{-23}	0.6621
	significance	+	+	+	−
f10	*p*-value	2.006×10^{-109}	3.799×10^{-314}	1.247×10^{-150}	5.2876×10^{-15}
	significance	+	+	+	+
f11	*p*-value	0.0108	3.1582×10^{-38}	1.836×10^{-93}	0.0196
	significance	+	+	+	+
f12	*p*-value	1.5512×10^{-7}	9.2258×10^{-64}	6.3131×10^{-94}	0.0209
	significance	+	+	+	+
f13	*p*-value	0.3041	6.1849×10^{-63}	2.018×10^{-81}	0.1833
	significance	−	+	+	−
f14	*p*-value	1.5171×10^{-33}	9.1071×10^{-51}	0.2502	1.2445×10^{-4}
	significance	+	+	−	+
f15	*p*-value	0.8234	6.6992×10^{-20}	1.9623×10^{-7}	0.3320
	significance	−	+	+	−
f16	*p*-value	9.3987×10^{-51}	1.0955×10^{-65}	3.6577×10^{-12}	6.7638×10^{-9}
	significance	+	+	+	+
f17	*p*-value	1.033×10^{-7}	6.7524×10^{-35}	3.4902×10^{-64}	0.8783
	significance	+	+	+	−
f18	*p*-value	2.8098×10^{-16}	2.0435×10^{-59}	0.1924	0.7002
	significance	+	+	-	-
f19	*p*-value	9.2738×10^{-14}	1.5168×10^{-4}	1.6272×10^{-18}	0.0824
	significance	+	+	+	−
f20	*p*-value	4.9539×10^{-16}	3.9176×10^{-39}	1.3166×10^{-37}	1.7149×10^{-12}
	significance	+	+	+	+
f21	*p*-value	0.9815	2.2951×10^{-9}	4.1069×10^{-16}	0.8828
	significance	−	+	+	−
f22	*p*-value	4.1687×10^{-24}	0.8822	4.648×10^{-33}	2.6117×10^{-16}
	significance	+	−	+	+
f23	*p*-value	1.7789×10^{-9}	6.0406×10^{-17}	2.4782×10^{-20}	4.8364×10^{-9}
	significance	+	+	+	−
f24	*p*-value	0.8823	1.6749×10^{-42}	3.0488×10^{-68}	9.0041×10^{-4}
	significance	−	+	+	+
f25	*p*-value	1.2705×10^{-14}	2.6922×10^{-76}	9.3371×10^{-66}	8.3714×10^{-11}
	significance	+	+	+	+
f26	*p*-value	1.1275×10^{-27}	0.0433	1.1038×10^{-47}	4.2343×10^{-17}
	significance	+	+	+	+
f27	*p*-value	1.0172×10^{-6}	4.8226×10^{-42}	7.6683×10^{-46}	4.2938×10^{-4}
	significance	+	+	+	+
f28	*p*-value	0.0191	9.4714×10^{-68}	2.4439×10^{-69}	0.4121
	significance	+	+	+	−
		total number of significance			
		20	26	24	15

6. Conclusions

Based on the analysis of the FWA, an improved fireworks algorithm (IFWA) is proposed in this paper. IFWA firstly puts opposition-based learning into FWA to initialize the population. Moreover, aiming at the shortage of explosion amplitude in FWA, a new explosion amplitude mechanism is proposed. Then, the adaptive *t*-distribution mutation is proposed for non-optimal fireworks, and elite opposition-based learning for optimal firework. At last, a new selection mechanism is proposed, which reduces the run-time of algorithm.

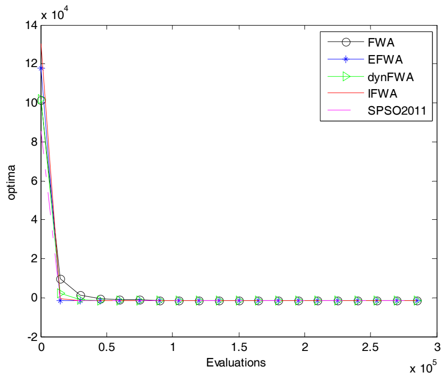
We apply the CEC2013 standard functions to examine and compare the proposed algorithm IFWA with SPSO2011, FWA, EFWA and dynFWA. The results clearly indicate that IFWA can perform significantly better than FWA, EFWA, dynFWA and SPSO2011 in terms of solution accuracy. Overall, the research demonstrates that IFWA performed the best for both solution accuracy and run-time cost.

Acknowledgments: The authors are thankful to the anonymous reviewers for their valuable comments to improve the technical content and the presentation of the paper. This paper is supported by the Liaoning Provincial Department of Education Science Foundation (Grant No. L2013064) and AVIC Technology Innovation Fund (basic research) (Grant No. 2013S60109R).

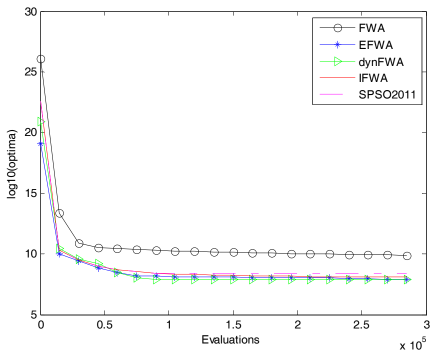
Author Contributions: Xi-Guang Li participated in the draft writing. Shou-Fei Han participated in the concept, design and perform the experiments and commented on the manuscript. Chang-Qing Gong participated in the data collection, and analyze the data.

Conflicts of Interest: The authors declare no conflict of interest.

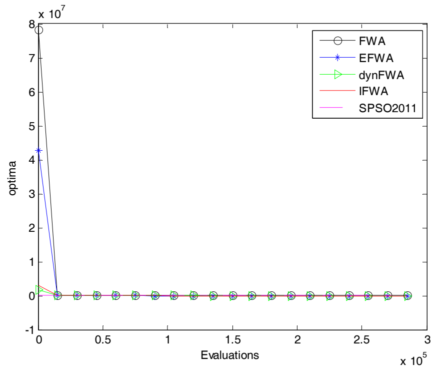
Appendix A



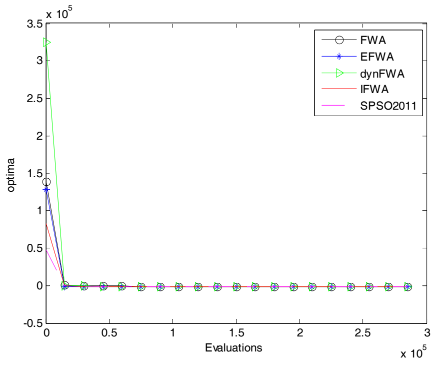
(**a**)



(**b**)

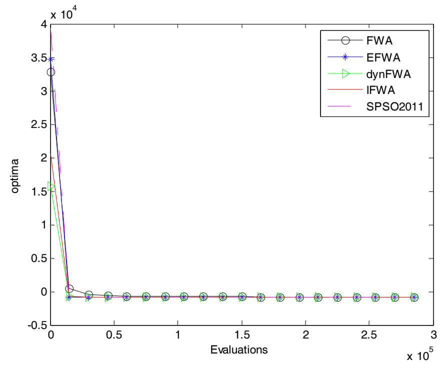


(**c**)

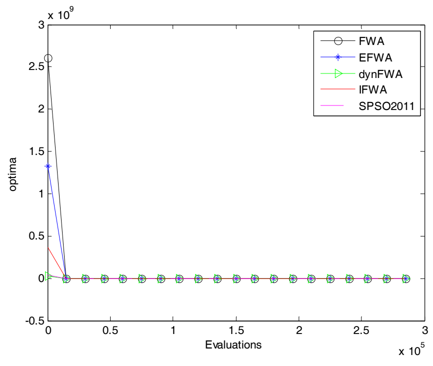


(**d**)

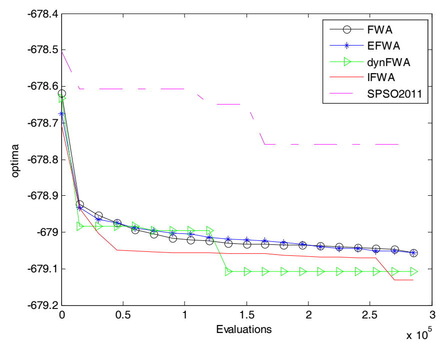
Figure A1. *Cont.*



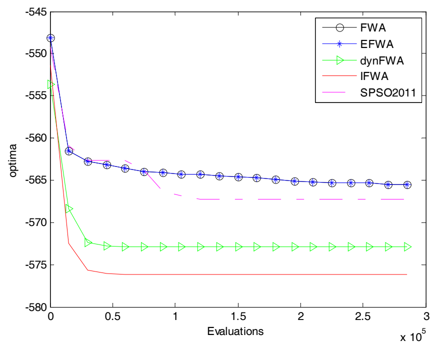
(e)



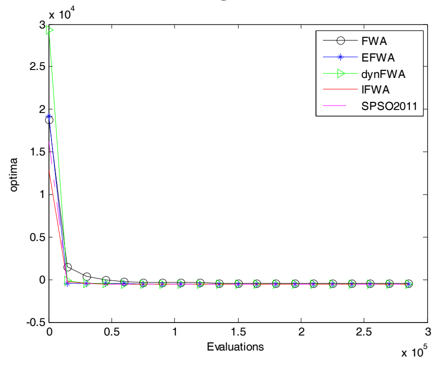
(f)



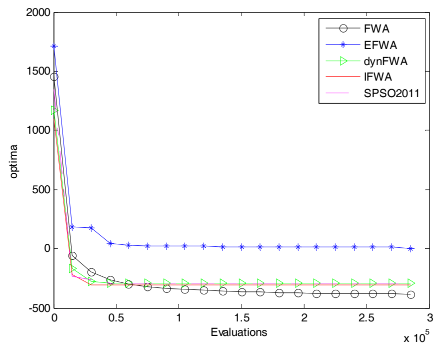
(g)



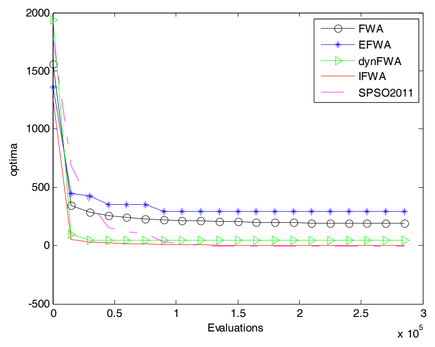
(h)



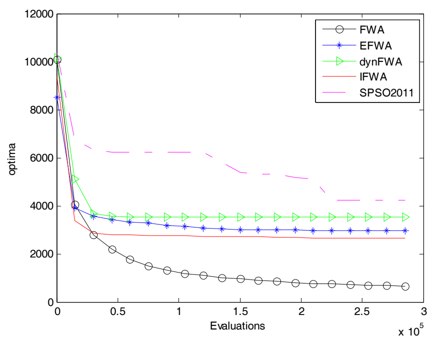
(i)



(j)

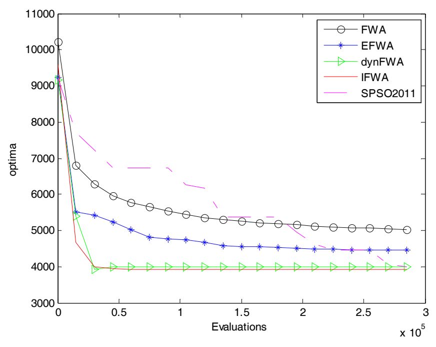


(k)

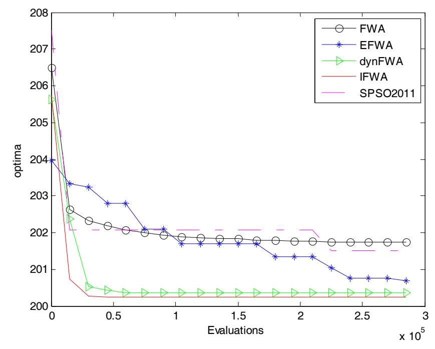


(l)

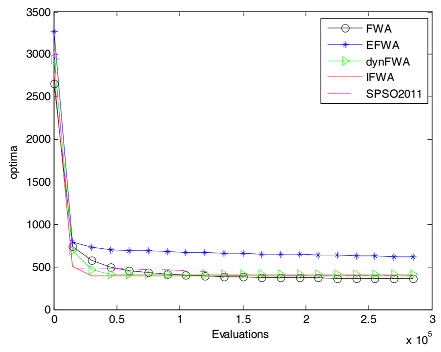
Figure A1. *Cont.*



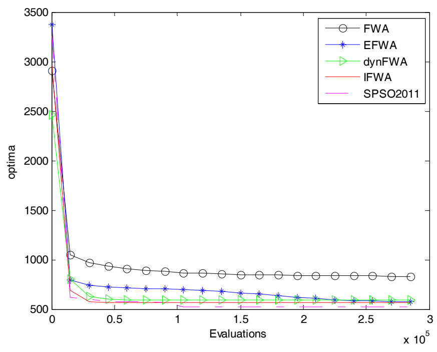
(**m**)



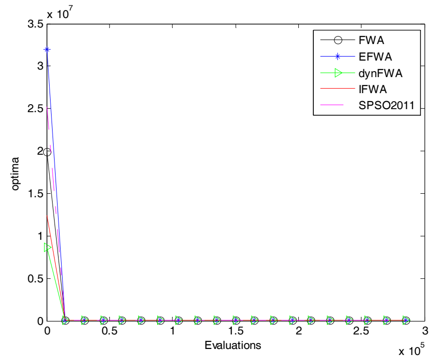
(**n**)



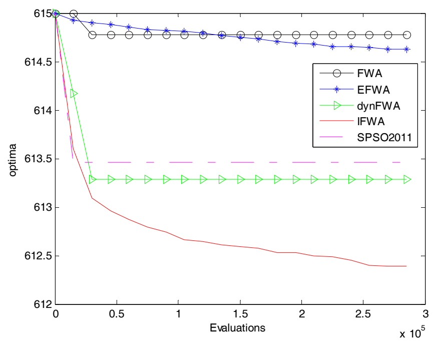
(**o**)



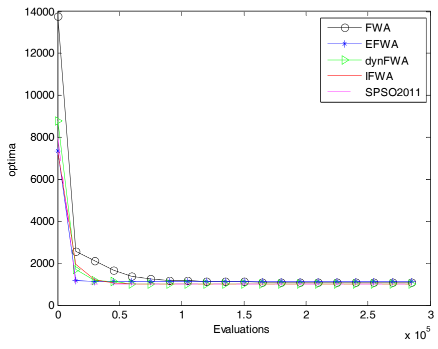
(**p**)



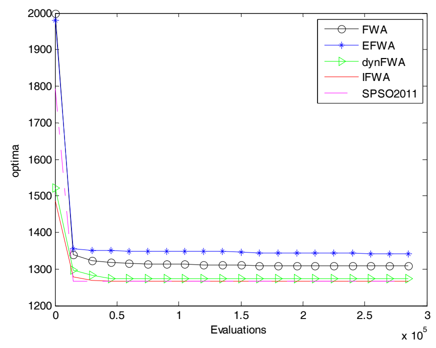
(**q**)



(**r**)



(**s**)



(**t**)

Figure A1. The FWA, EFWA, dynFWA, SPSO2011 and IFWA searching curves. (**a**) f1 function; (**b**) f3 function; (**c**) f4 function; (**d**) f5 function; (**e**) f6 function; (**f**) f7 function; (**g**) f8 function; (**h**) f9 function; (**i**) f10 function; (**j**) f11 function; (**k**) f13 function; (**l**) f14 function; (**m**) f15 function; (**n**) f16 function; (**o**) f17 function; (**p**) f18 function; (**q**) f19 function; (**r**) f20 function; (**s**) f21 function; (**t**) f24 function.

References

1. Tan, Y.; Zhu, Y. Fireworks Algorithm for Optimization. In *Advances in Swarm Intelligence*; Springer: New York, NY, USA, 2010.
2. Tan, Y. *Fireworks Algorithm Introduction*, 1st ed.; Science Press: Beijing, China, 2015; pp. 13–136. (In Chinese)
3. Andreas, J.; Tan, Y. Using population based algorithms for initializing nonnegative matrix factorization. In *Advances in Swarm Intelligence*; Springer: Berlin, Germany, 2011; pp. 307–316.
4. Gao, H.Y.; Diao, M. Cultural firework algorithm and its application for digital filters design. *Int. J. Model. Identif. Control* **2011**, *4*, 324–331. [CrossRef]
5. Wen, R.; Mi, G.Y.; Tan, Y. Parameter optimization of local-concentration model for spam detection by using fireworks algorithm. In Proceedings of the 4th International Conference on Swarm Intelligence, Harbin, China, 12–15 June 2013; pp. 439–450.
6. Imran, A.M.; Kowsalya, M.; Kothari, D.P. A novel integration technique for optimal network reconfiguration and distributed generation placement in power distribution networks. *Int. J. Electr. Power* **2014**, *63*, 461–472. [CrossRef]
7. Nantiwat, P.; Bureerat, S. Comparative performance of meta-heuristic algorithms for mass minimisation of trusses with dynamic constraints. *Adv. Eng. Softw.* **2014**, *75*, 1–13.
8. Li, H.; Bai, P.; Xue, J.; Zhu, J.; Zhang, H. Parameter estimation of chaotic systems using fireworks algorithm. In *Advances in Swarm Intelligence*; Springer: Berlin, Germany, 2015; pp. 457–467.
9. Liu, Z.B.; Feng, Z.R.; Ke, L.J. Fireworks algorithm for the multi-satellite control. In Proceedings of the 2015 IEEE Congress on Evolutionary Computation, Sendai, Japan, 25–28 May 2015; pp. 1280–1286.
10. Zheng, S.; Janecek, A.; Tan, Y. Enhanced fireworks algorithm. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; pp. 2069–2077.
11. Zheng, S.; Li, J.; Tan, Y. Adaptive fireworks algorithm. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation, Beijing, China, 6–11 July 2014; pp. 3214–3221.
12. Zheng, S.; Tan, Y. Dynamic search in fireworks algorithm. In Proceedings of the 2014 IEEE Congress on Evolutionary Computation, Beijing, China, 6–11 July 2014; pp. 3222–3229.
13. Tizhoosh, H.R. Opposition-Based Learning: A New Scheme for Machine Intelligence. In Proceedings of the 2005 International Conference on Computational Intelligence for Modeling, Control and Automation, Vienna, Austria, 28–30 November 2005; pp. 695–701.
14. Rahnamayan, S.; Tizhoosh, H.R.; Salama, M.M. Opposition-based differential evolution algorithms. In Proceedings of the 2006 IEEE Congress on Evolutionary Computation, Vancouver, BC, Canada, 16–21 July 2006; pp. 7363–7370.
15. Wang, H.; Liu, Y.; Zeng, S.Y.; Li, H.; Li, C.H. Opposition-based particle swarm algorithm with Cauchy mutation. In Proceedings of the 2007 IEEE Congress on Evolutionary Computation, Singapore, 25–28 September 2007; pp. 4750–4756.
16. Yu, S.H.; Zhu, S.L.; Ma, Y. Enhancing firefly algorithm using generalized opposition-based learning. *Computing* **2015**, *97*, 741–754. [CrossRef]
17. Chibing, G. Opposition-Based Adaptive Fireworks Algorithm. *Algorithms* **2016**, *9*, 43.
18. Gao, H.; Li, C. Opposition-based quantum firework algorithm for continuous optimisation problems. *Int. J. Comput. Sci. Math.* **2015**, *6*, 256–265. [CrossRef]
19. Lan, K.-T.; Lan, C.-H. Notes on the distinction of Gaussian and Cauchy mutations. In Proceedings of the Eighth International Conference on Intelligent Systems Design and Applications, Kaohsiung, Taiwan, 26–28 November 2008; pp. 272–277.
20. Zhou, F.J.; Wang, X.J.; Zhang, M. Evolutionary Programming Using Mutations Based on the t Probability Distribution. *Acta Electron. Sin.* **2008**, *36*, 121–123. (in Chinese).
21. Kuo, T.; Hwang, S.Y. A genetic algorithm with disruptive selection. *IEEE Trans. Sys. Man Cybern. Part B Cybern.* **1996**, *26*, 299–307.
22. Zambrano-Bigiarini, M.; Clerc, M.; Rojas, R. Standard particle swarm optimization 2011 at CEC2013: A baseline for future PSO improvements. In Proceedings of the 2013 IEEE Congress on Evolutionary Computation, Cancun, Mexico, 20–23 June 2013; pp. 2337–2344.

23. Liang, J.; Qu, B.; Suganthan, P.; Hernandez-Diaz, A.G. *Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization*; Technical Report 201212; Zhengzhou University: Zhengzhou, China, January 2013.
24. Teng, S.Z.; Feng, J.H. *Mathematical Statistics*, 4st ed.; Dalian University of Technology Press: Dalian, China, 2005; pp. 34–35. (In Chinese)