Hindawi
Computational Intelligence and Neuroscience
Volume 2018, Article ID 6094685, 27 pages
https://doi.org/10.1155/2018/6094685

Research Article

PS-FW: A Hybrid Algorithm Based on Particle Swarm and Fireworks for Global Optimization

Shuangqing Chen, Yang Liu, Lixin Wei, and Bing Guan

School of Petroleum Engineering, Northeast Petroleum University, Daqing 163318, China

Correspondence should be addressed to Yang Liu; ly001@nepu.edu.cn

Received 21 September 2017; Accepted 10 January 2018; Published 20 February 2018

Academic Editor: Raşit Köker

Particle swarm optimization (PSO) and fireworks algorithm (FWA) are two recently developed optimization methods which have been applied in various areas due to their simplicity and efficiency. However, when being applied to high-dimensional optimization problems, PSO algorithm may be trapped in the local optima owing to the lack of powerful global exploration capability, and fireworks algorithm is difficult to converge in some cases because of its relatively low local exploitation efficiency for noncore fireworks. In this paper, a hybrid algorithm called PS-FW is presented, in which the modified operators of FWA are embedded into the solving process of PSO. In the iteration process, the abandonment and supplement mechanism is adopted to balance the exploration and exploitation ability of PS-FW, and the modified explosion operator and the novel mutation operator are proposed to speed up the global convergence and to avoid prematurity. To verify the performance of the proposed PS-FW algorithm, 22 high-dimensional benchmark functions have been employed, and it is compared with PSO, FWA, stdPSO, CPSO, CLPSO, FIPS, Frankenstein, and ALWPSO algorithms. Results show that the PS-FW algorithm is an efficient, robust, and fast converging optimization method for solving global optimization problems.

1. Introduction

Global optimization problems are common in engineering and other related fields [1–3], and it is usually difficult to solve the global optimization problems due to many local optima and complex search space, especially in high dimensions. For solving optimization problems, many methods have been reported in the past few years. Recently, the stochastic optimization algorithms have attracted increasing attention because they can get better solutions without any properties of the objective functions. Therefore, many effective meta-heuristic algorithms have been presented, such as simulated annealing (SA) [4], differential evolution (DE) [5], genetic algorithm (GA) [6], particle swarm optimization (PSO) [7], ant colony optimization (ACO) [8], artificial bee colony (ABC) [9], and fireworks algorithm (FWA) [10].

Among these intelligent algorithms, the PSO and FWA have shown pretty outstanding performance in solving global optimization problems in the last several years. PSO algorithm is a population-based algorithm originally proposed by Kennedy and Eberhart [7], which is inspired by the foraging behavior of birds. Fireworks algorithm is a new swarm intelligence algorithm that is motivated by observing fireworks explosion. Owing to the less decision parameters, simple implementation, and good scalability, PSO and FWA have been widely applied since they were proposed, including shunting schedule optimization of electric multiple units depot [11], optimal operation of trunk natural gas pipelines [12], location optimization of logistics distribution center [13], artificial neural networks design [14], warehouse-scheduling [15], fertilization optimization [16], power system reconfiguration [17], and multimodal function optimization [18].

Although PSO and FWA are highly successful in solving some classes of global optimization problems, there are certain problems that need to be addressed when they are extended to handling complex high-dimensional optimization problems. The PSO algorithm has a significant efficiency in unimodal problems, but it can easily be trapped in local optima for multimodal problems. Moreover, the FWA is

difficult to converge for the optimization problems which do not have their optimal solutions at the origin. This is because the two algorithms cannot keep the balance between the exploration and exploitation properly. Due to the optimal particle dominating the solving process, the PSO algorithm has inferior swarm diversity in the later stage of iterations and relatively poor exploration ability [19], while the fireworks and sparks in FWA are not well-informed by the whole swarm [20] and the FWA framework lacks the local search efficiency for noncore fireworks [21]. In order to improve the performance of PSO and FWA, a considerable number of modified algorithms have been proposed. For example, Nickabadi et al. presented AIWPSO algorithm, in which a new adaptive inertia weight approach was adopted [22]. By embedding a reverse predictor and adding a repulsive force into the basic algorithm, the RPPSO was developed [23]. Wang and Liu used three strategies to ameliorate the standard algorithm, including best neighbor replacement, abandoned mechanism, and chaotic searching [24]. Souravlias and Parsopoulos introduced a PSO-based variant, which could dynamically assign different computational budget for each particle based on the quality of its neighbor [25]. Based on self-adaption principle and bimodal Gaussian function, the advanced fireworks algorithm (AFWA) was proposed [26]. Liu et al. presented several methods for computing the explosion amplitude and number of sparks [27]. Pei et al. proposed to use the elite point of approximation landscape in the fireworks swarm and discussed the effectiveness of surrogate-assisted FWA [28]. Zheng et al. improved the new explosion operator, mutation operator, selection strategy, and mapping rules of FWA, which led to the formation of enhanced fireworks algorithm (EFWA) [29, 30] and dynamic search in fireworks algorithm (dynFWA) [31]. Zheng et al. proposed the new cooperative FWA framework (CoFFWA), in which the independent selection method and crowdedness-avoiding cooperative strategy were contained [21]. Li et al. investigated the operators of FWA and introduced a novel guiding spark in FWA [32] and proposed the adaptive fireworks algorithm (AFWA) [33] and bare bones fireworks algorithm (BBFWA) [34].

Hybrid algorithms can utilize various exploration and exploitation strategies for high-dimensional multimodal optimization problems, which have gradually become the new research areas. For example, Valdez et al. combined the advantages of PSO with GA and proposed a modified hybrid method [35]. In the new PS-ABC algorithm introduced by Li et al., the global optimum could be obtained by combining the local search phase in PSO with two global search phases in ABC [19]. Pandit et al. presented the SPSO-DE, in which the domain information of PSO and DE was shared with one another to overcome their respective weaknesses [36]. Through changing the generation and selection strategy of explosive spark, Gao and Diao proposed the CA-FWA [37]. Zhang et al. proposed BBO-FW algorithm which improved the interaction ability between fireworks [38]. By combining the FWA with the operators of DE, a novel hybrid optimization algorithm was proposed [20].

In this paper, by utilizing the exploitation ability of PSO and the exploration ability of FWA, a novel hybrid optimization algorithm called PS-FW is proposed. Based on the solving process of PSO algorithm, the operators of FWA are embedded into the update operation of the particle swarm. In the iteration process, in order to promote the balance of exploitation and exploration ability of PS-FW, we presented three major techniques. Firstly, the abandonment and supplement strategy is used, to abandon a certain number of particles with poor quality and to supplement the particle swarm with new individuals generated by FWA. Meanwhile, considering the information exchanges between the optimal firework and its neighbor in each dimension, the method for obtaining the explosion amplitude is designed as adaptive, and the mode of generating the explosion sparks is modified by combing the greedy algorithm. Furthermore, the conventional Gaussian mutation operator is abandoned, and the novel mutation operator based on the thought of the social cognition and learning is proposed. The performance of PS-FW is compared with several existing optimization algorithms. The experimental results show that the proposed PS-FW is more efficacious in solving the global optimization problems.

The rest of the paper is organized as follows: Section 2 describes the standard PSO and FWA. Section 3 presents the PS-FW algorithm, in which the algorithm details are proposed. Section 4 introduces the simulation results over 22 high-dimensional benchmark functions and the corresponding comparisons between PS-FW and other algorithms are executed. Finally, the conclusion is drawn in Section 5.

2. Related Work

2.1. PSO Algorithm. In PSO algorithm, the particles scatter in search space of the optimization problems and each particle denotes a feasible solution. Each particle contains three aspects of information: the current position x_i, the velocity v_i, and the previous best position $pbest_i$. Assume that the optimization problem is D-dimensional and M represents the size of the swarm population; then the position and velocity of ith ($i = 1, 2, \ldots, M$) particle can be denoted as $x_i = (x_{i,1}, x_{i,2}, \ldots, x_{i,D})$ and $v_i = (v_{i,1}, v_{i,2}, \ldots, v_{i,D})$, respectively, while the previous best position is represented as $pbest_i = (pbest_{i,1}, pbest_{i,2}, \ldots, pbest_{i,D})$. Besides, the best position encountered by the entire particles so far is known as current global best position $gbest_i = (gbest_1, gbest_2, \ldots, gbest_D)$. In each generation, v_i and x_i are updated by the following equations:

$$v_{i,k}(t+1) = w \cdot v_{i,k}(t) + c_1 \cdot r_1 \cdot \left[pbest_{i,k}(t) - x_{i,k}(t)\right] + c_2 \cdot r_2 \cdot \left[gbest(t) - x_{i,k}(t)\right], \quad (1)$$

$$x_{i,k}(t+1) = x_{i,k}(t) + v_{i,k}(t+1), \quad (2)$$

where c_1 and c_2 are two learning factors that indicate the influence of the cognitive and social components, r_1 and r_2 are the random real numbers in interval $[0, 1]$, respectively, and w is the inertia weight which controls the convergence speed of the algorithm.

2.2. Fireworks Algorithm. In FWA, a firework or a spark denotes a potential solution of optimization problems, while

the process of producing sparks from fireworks represents a search in the feasible space. As in other optimization algorithms, the optimal solutions are obtained by successive iterations. In each iteration, the sparks can be produced by two ways: the explosion and the Gaussian mutation. The explosion of fireworks is dominated by the explosion amplitude and the number of explosion sparks. Compared to the fireworks with lower fitness, the fireworks with better fitness will have smaller explosion amplitude and more explosion sparks. Suppose that N denotes the number of fireworks; then the ith ($i = 1, 2, \ldots, N$) firework can be denoted as $\overline{x} = (\overline{x}_{i,1}, \overline{x}_{i,2}, \ldots, \overline{x}_{i,D})$ for D-dimensional optimization problems. Besides, the explosion amplitude can be obtained by (3) and the sparks number can be calculated by (4):

$$A_i = \widehat{A} \cdot \frac{f(\overline{x}_i) - y_{\min} + \varepsilon}{\sum_{i=1}^{N} (f(\overline{x}_i) - y_{\min}) + \varepsilon}, \tag{3}$$

$$s_i = M_e \cdot \frac{y_{\max} - f(\overline{x}_i) + \varepsilon}{\sum_{i=1}^{N} (y_{\max} - f(\overline{x}_i)) + \varepsilon}, \tag{4}$$

where $f(\overline{x})$ denotes the objective function value of the ith firework, $i = 1, 2, \ldots, N$, A_i and s_i are the explosion amplitude and the number of explosion sparks of the ith firework, respectively, $y_{\max} = \max(f(\overline{x}_i))$, $y_{\min} = \min(f(\overline{x}_i))$, $\widehat{A}$ and M_e are two constants that dominate the explosion amplitude and the number of explosion sparks, respectively, and ε is the machine epsilon.

Moreover, the bounds of s_i are defined as follows:

$$s_i = \begin{cases} \text{round}(a \cdot M_e), & s_i < a \cdot M_e \\ \text{round}(b \cdot M_e), & s_i > b \cdot M_e \\ \text{round}(s_i), & \text{otherwise,} \end{cases} \tag{5}$$

where a, b are two constants that control the minimum and maximum of population size, respectively.

In order to generate each explosion spark of ith firework, an offset is added to $\overline{x}_i$ according to the following equation:

$$\widehat{x}_i^j = \overline{x}_i + \Delta h, \tag{6}$$

where $\widehat{x}_i^j$ is the jth explosion spark of ith firework and $\Delta h = A_i \cdot \text{rand}(-1, 1) \cdot \widehat{B}$, where $\widehat{B}$ is a D-dimensional vector which has $\widehat{z}_i^j$ values of 1 and $D - \widehat{z}_i^j$ values of 0, where $\widehat{z}_i^j$ denotes the number of randomly selected dimensions of $\overline{x}$ and $\widehat{z}_i^j = D \cdot \text{rand}()$, $j = 1, 2, \ldots, s_i$, where rand$(-1, 1)$ and rand() are random numbers in the intervals $[-1, 1]$ and $[0, 1]$, respectively.

Another type of sparks known as the Gaussian sparks is generated based on the Gaussian mutation operator. In each generation, a certain number of Gaussian sparks are generated and each Gaussian spark is transformed from a firework which is selected randomly. For the selected firework $\overline{x}_i$, its Gaussian spark is generated based on

$$\widetilde{x}_j = \left(\widetilde{O} - \widetilde{B}_i\right) \cdot \overline{x}_i + \text{Gaussian}(1, 1) \cdot \overline{x}_i \cdot \widetilde{B}_i, \tag{7}$$

where $\widetilde{x}_j$ is the jth Gaussian spark, $\widetilde{O}$ is a D-dimensional vector whose values are 1 in each dimension, $\widetilde{B}$ is a D-dimensional vector which has $\widetilde{z}_i$ values of 1 and $D-\widetilde{z}_i$ values of 0, $\widetilde{z}_i$ represents the number of randomly selected dimensions of $\overline{x}_i$ and $\widetilde{z}_i = D \cdot \text{rand}()$, and Gaussian$(1, -1)$ represents a random number subordinated to the Gaussian distribution with the mean of 1 and the standard deviation of 1.

For the purpose of passing information to the next generation, new fireworks populations are chosen to continue the iteration. All the fireworks, the explosion sparks, and Gaussian sparks have the chance to be selected for the next iteration. The location with best fitness is kept for the next generation, while the other $N - 1$ locations are selected based on the selection operator and the selection operator is denoted as follows:

$$\begin{aligned} R(X_i) &= \sum_{j \in K} d(X_i, X_j) = \sum_{j \in K} \left\| X_i - X_j \right\|, \\ p(X_i) &= \frac{R(X_i)}{\sum_{k \in K} R(X_k)}, \end{aligned} \tag{8}$$

where K denotes the set comprised of all the original fireworks and both types of sparks, X_i, X_j, and X_k are ith, jth, and kth location of K, respectively, $R(X_i)$ is the distance between ith location and the rest of all the locations, and $p(X_i)$ denotes the probability of being selected for the ith location.

3. Hybrid Optimization Algorithm Based on PSO and FWA

The exploitation process focuses on utilizing the existing information to look for better solutions, whereas the exploration process attaches importance to seek the optimal solutions in the entire space. For PSO, under the guidance of their historical best solutions and the current global best solution, the particles can quickly find better solutions, and the excellent exploitation efficiency of algorithm is shown. In FWA, the fireworks can find the global optimal solution in the whole search space by performing explosion and mutation operations while the outstanding exploration capability of FWA is demonstrated. To utilize the advantages of the two algorithms, a hybrid optimization method (PS-FW) based on PSO and FWA is proposed.

3.1. Feasibility Analysis. The formation of a hybrid algorithm is mainly due to the effective combination of the operators of its composition algorithms in a certain way. To clarify the performance enhancement caused by combining the PSO algorithm with fireworks algorithm, we draw Figures 1 and 2 to illustrate the optimization mechanism. As shown in Figure 1, for standard PSO algorithm, the ith particle moves from point 1 to point 4 under the common influence of velocity inertia, self-cognition, and social information. When the operators of FWA are added, the particle is transformed into firework and performs explosion and mutation operations and eventually reaches the position of firework or sparks, such as point 5 shown in Figure 1. By performing the

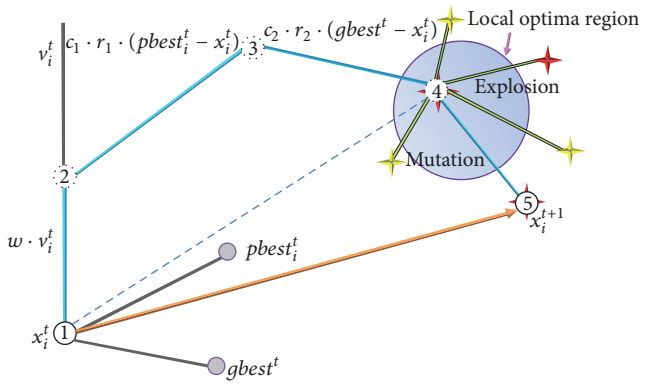


Figure 1: Optimization mechanism of adding operators of FWA to PSO algorithm.

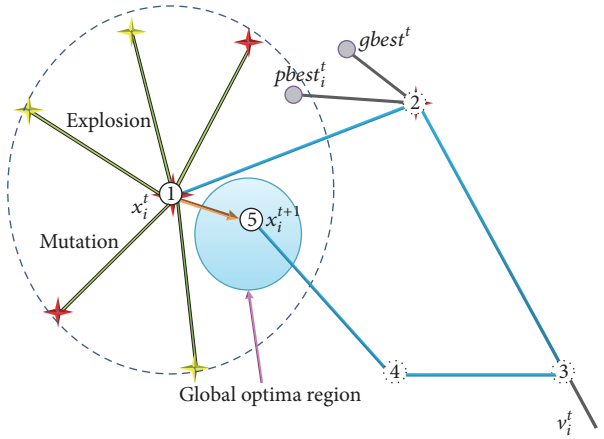


Figure 2: Optimization mechanism of adding operators of PSO to FWA.

operators of FWA, the particle can explore better solutions in multiple directions and jump out of the local optima region as depicted in Figure 1. Thus we can argue that the operators of FWA improve the global search ability of PSO algorithm. As we know, the searching region is determined by the explosion amplitude and fireworks with poor quality have bigger amplitude, which may lead to an uncomprehensive search without considering the cooperation with other fireworks. When the firework with poor quality generates the explosion sparks and mutation sparks, the new selected location may skip over the global optima region without the attraction from the rest of fireworks and arrive at point 2. By adding the operators of PSO after the ith firework updates its location, the information of its own historical best location and current global best location is taken into account; then the new solution is found in point 5, which is shown in Figure 2. Therefore, the operators of PSO could strengthen the local search efficiency of FWA. Based on the above analysis, it is concluded that the combination of PSO and FWA is an effective way to form a superior optimization algorithm.

3.2. The Abandonment and Supplement Mechanism. The particles with their memory ability can be quickly converged to the current optimal solution. However, the aggregation effect of the particle swarm reduces the diversity of the population, which makes the search in the whole feasible space inefficient. In this paper, in order to enhance the balance between exploitation ability and exploration ability of PS-FW, we adopt the abandonment and supplement strategy which includes three main steps. (i) All the particles in the particle swarm $x_1, x_2, \ldots, x_M$ are sorted in ascending order. Then the P_{num} particles with better fitness are retained for the next iteration, and the FW_{num} (satisfying $P_{\text{num}} + \text{FW}_{\text{num}} = M$) particles with lower fitness are abandoned. (ii) The P_{num} excellent individuals denoted as $x_{F1}, x_{F2}, \ldots, x_{FP_{\text{num}}}$ are used to implement the explosion operator, the mutation operator, and the selection operator. (iii) The new individuals obtained by the operators of FWA are added to the original population, to balance the number of particles and to generate the new particle swarm for the next iteration. The abandonment and supplement strategy not only retains the information of the excellent individuals so that they can participate in the subsequent calculation, but also avoids the individuals with poor quality wasting computing resources. However, the problem arises: how to determine P_{num}? For this, through analyzing the process of solving the optimization problems, we should enhance the exploration ability of the algorithm and search the optimal solution in the global scope at early stage of iterations, which means the number of particles executing the operators of FWA should be the majority. In the later stage of iteration, we should focus on searching around the current global optimal solution, so the number of excellent individuals retained in the algorithm should be more. Based on the discussion above, the calculation of FW_{num} in this paper is shown in (9), in which FW_{num} decreases with iteration process.

$$\text{FW}_{\text{num}} = \text{round}\left[(\text{FW}_{\max} - \text{FW}_{\min}) \cdot \left(\frac{I_{\max} - t}{I_{\max}}\right)^r + \text{FW}_{\min}\right], \tag{9}$$

where $\text{FW}_{\max}$ and $\text{FW}_{\min}$ are the upper and lower bounds of number of abandoned particles, respectively, $I_{\max}$ is the maximum number of iterations, t denotes the current number of iterations, round[] indicates that the values in brackets are rounded, and r represents a positive integer.

3.3. Modified Explosion Operator

3.3.1. Adaptive Explosion Amplitude. Based on the analysis above, the definition of the explosion amplitude in standard FWA limits the diversity of the explosion sparks generated by the excellent fireworks, thus decreasing the local search ability of algorithm. In the enhanced fireworks algorithm (EFWA) [29], in order to avoid the weakness of the explosion amplitude generation in FWA, a minimal explosion amplitude check mechanism is proposed, which defines the explosion amplitude less than a certain threshold to obtain the same value as the threshold while the threshold is reducing with the iteration process. Suppose that δ denotes the threshold of

explosion amplitude; then the explosion amplitude less than the threshold is defined as (10) in EFWA.

$$\widehat{A} = \widehat{A}_{\text{init}} - \frac{\widehat{A}_{\text{init}} - \widehat{A}_{\text{final}}}{I_{\max}} \cdot \sqrt{(2I_{\max} - t)\, t}, \tag{10}$$

where $\widehat{A}_{\text{init}}$ and $\widehat{A}_{\text{final}}$ are the upper and lower bounds of the explosion amplitude, respectively.

In this paper, based on the minimal explosion amplitude detection mechanism, the basic explosion amplitude of each firework is calculated according to (3), and the explosion amplitude is adjusted by the following two methods.

(1) For the fireworks whose explosion amplitude is greater than the threshold δ, a control factor λ of the explosion amplitude is added. The control factor makes the explosion sparks generated by the algorithm have larger search scope in the early stage of iterations, which can effectively enhance the exploration ability of the algorithm. In the later stage of iterations, the explosion amplitude is reduced to improve the search efficiency around the current global optimal solution. The adjustment of the explosion amplitude is shown in (11), and the control factor is calculated as shown in (12).

$$\widehat{A}_i = \widehat{A}_i \cdot \lambda, \quad \forall \widehat{A}_i > \delta, \tag{11}$$

$$\lambda = \lambda_{\min} \cdot \left(\frac{\lambda_{\max}}{\lambda_{\min}} \right)^{1/(1+t/I_{\max})}, \tag{12}$$

where $\lambda_{\max}$ and $\lambda_{\min}$ are the lower and upper bounds of the control factor, respectively.

(2) When the explosion amplitude of firework $\overline{x}_i$ is less than the threshold, the optimal firework and its neighbor information are used to determine the explosion amplitude in the hybrid algorithm. Since the PS-FW algorithm is based on the framework of PSO, the position of all individuals will approach the current best position, which leads to the fitness of current optimal individual close to its neighbor individuals. That is to say, if the explosion amplitude of a firework is too small, indicating that the firework may be located near the current best location, therefore, by considering the deviation information of all corresponding dimensions between the current best firework and its neighbor firework, a new explosion amplitude of the firework $\overline{x}_i$ is generated. The explosion amplitude generation method can adaptively optimize the solving process, which can be interpreted from two aspects. When the algorithm is in the early iteration stage, the position of fireworks is scattered, and the deviation in dimensions between the optimal firework and its neighbor firework is larger, which leads to the larger explosion amplitude and the improved probability of finding the global optimal solution. As the algorithm enters the later iterations, the fireworks gather around the current best location, and the offset of each dimension between the current best firework and its neighbor firework is reduced, which results in the decrement of explosion amplitude and the improvement of the local search ability for PS-FW. There are two main steps to obtain the explosion amplitude. (i) Randomly select a firework $\overline{x}_j$ around the current optimal firework according to the fitness. (ii) Update the explosion amplitude of the ith firework according to the following equation:

$$\widehat{A}_i = \frac{\sum_{k=1}^{D} \left(\left| \overline{x}_{\text{best},k} - \overline{x}_{j,k} \right| \right)}{D}, \tag{13}$$

where $\overline{x}_{\text{best},k}$ denotes the value of the kth dimension of current optimal firework.

3.3.2. Modified Explosion Sparks Generation. In FWA, when generating an explosion spark, the offset Δh is only calculated once, which results in the same changes for all the selected dimensions and an ineffective search for different directions. In the PS-FW algorithm proposed in this paper, a new explosion sparks generation method is introduced. Firstly, when generating the explosion sparks, the location offset is performed in all the dimensions of the fireworks instead of randomly selecting part of dimensions. Furthermore, for each dimension of the fireworks, the different offsets are calculated according to (14), thereby increasing the diversity of the explosion sparks and the global search capability of the hybrid algorithm. Meanwhile, suppose that $\overline{x}_{\text{temp}}$ denotes the ith firework without a location offset and $\overline{x}_+$ indicates the ith firework whose kth dimension adds a offset; then $\overline{x}_-$ denotes the ith firework whose kth dimension subtracts an offset. As shown in (15), inspired by greedy algorithm, when the fireworks generate their explosion sparks, the hybrid algorithm determines which offset to be selected based on the value of objective function, which can effectively improve the local search capability of the algorithm and accelerate the convergence.

$$\Delta \widehat{h}_k = \widehat{A} \cdot \text{Gaussian}\,(0, 1)\,, \tag{14}$$

$$\widehat{x}_{i,k}^{j} = \begin{cases} \overline{x}_{i,k} + \Delta \widehat{h}_k, & f\left(\overline{x}_+\right) \le \min\left(f\left(\overline{x}_{\text{temp}}\right), f\left(\overline{x}_-\right)\right) \\ \overline{x}_{i,k} - \Delta \widehat{h}_k, & f\left(\overline{x}_-\right) \le \min\left(f\left(\overline{x}_{\text{temp}}\right), f\left(\overline{x}_+\right)\right) \\ \overline{x}_{i,k}, & f\left(\overline{x}_{\text{temp}}\right) \le \min\left(f\left(\overline{x}_+\right), f\left(\overline{x}_-\right)\right), \end{cases} \tag{15}$$

where $\widehat{x}_{i,k}^{j}$ and $\Delta \widehat{h}_k$ are the value and offset of the kth dimension of the jth explosion spark for the ith firework, respectively, Gaussian(0, 1) represents a random number that follows the standard normal distribution, i and j are integers in the intervals $[1, P_{\text{num}}]$ and $[1, s_i]$, respectively, and min() indicates the minimum values in parentheses.

Assume that num_E denotes the total number of explosion sparks generated by all fireworks, $S_{\min}$ and $S_{\max}$ represent the lower and upper bounds for the search scope, and $S_{\min,k}$ and $S_{\max,k}$ are corresponding to the bounds of kth dimension, respectively. Based on the explosion operator introduced in Sections 3.3.1 and 3.3.2, the detailed codes of explosion operator are represented in Algorithm 1.

3.4. Novel Mutation Operator. As the Gaussian mutation operator effectively increases the diversity of feasible solutions, the performance of traditional FWA has been significantly improved. However, the numerical experiments

(1) Input: P_{num} particles sorted in ascending order according to their fitness.
(2) Initialize the location of fireworks: $\overline{x}_i = x_{Fi}$, $i = 1, 2, \ldots, P_{\text{num}}$.
(3) for $i = 1$ to P_{num} do
(4) Calculate the explosion amplitude A_i of ith firework by using (3).
(5) Calculate the number of explosion sparks s_i of ith firework by using (4).
(6) Update the number of explosion sparks of ith firework by using (5).
(7) if $\widehat{A}_i > \delta$ do
(8) Update the explosion amplitude of ith firework by using (11) and (12).
(9) else do
(10) Randomly select a firework $\overline{x}_j$ around the current optimal firework.
(11) Update the explosion amplitude of ith firework by using (13).
(12) end if
(13) end for
(14) Initialize the total number of explosion sparks $\text{num}_E = 0$.
(15) for $i = 1$ to P_{num} do
(16) for $j = 1$ to s_i do
(17) Initialize the location of the jth explosion spark: $\widehat{x}_i^j = \overline{x}_i$.
(18) for $k = 1$ to D do
(19) Calculate the offset by using (14).
(20) Update the value of kth dimension of jth explosion spark by using (15).
(21) if $\widehat{x}_{i,k}^j > S_{\max,k}$ or $\widehat{x}_{i,k}^j < S_{\min,k}$ do
(22) Update the $\widehat{x}_{i,k}^j$ by using (17).
(23) end if
(24) end for
(25) $\text{num}_E = \text{num}_E + 1$
(26) end for
(27) end for
(28) Output: num_E explosion sparks.

Algorithm 1: Generating explosion sparks by the explosion operator of PS-FW.

show that the combined application of Gaussian operator and mapping operator makes the Gaussian sparks mostly concentrated around the zero point, which is the reason why FWA has the fast convergence speed for the problems with their optimal solutions at zero [31]. In order to improve the adaptability of the algorithm for the nonzero optimization problems and maintain the contribution of the mutation operator to the population diversity, a new mutation operator is proposed in the PS-FW. Compared with the standard FWA, there are two main differences in this paper. (i) In PS-FW, we randomly select a certain number of explosion sparks to generate the mutation sparks instead of using the fireworks. Because the explosion sparks have better quality compared to the fireworks based on (15), the mutation sparks generated by the explosion sparks can effectively enrich the diversity of the population and have better global search ability. (ii) In this paper, the Gaussian random number is no longer used in mutation operator and the interaction mechanism of particles in PSO is used for reference to design the mutation operator. The mutation sparks generated by our mutation operator can not only maintain the better information of the explosion sparks, but also have a proper movement towards the current best location, which leads to promoting the convergence of hybrid algorithm. The proposed mutation operator is shown as follows.

$$\widetilde{x}_{i,k} = \mu_1 \cdot \left(\widehat{x}_{\text{best},k} - \widehat{x}_{j,k}\right) + \mu_2 \cdot \widehat{x}_{j,k}, \tag{16}$$

where $\widetilde{x}_{i,k}$ and $\widehat{x}_{j,k}$ indicate the value of kth dimension of ith mutation spark and jth explosion spark, respectively, $\widehat{x}_{\text{best},k}$ is the current optimal explosion spark, μ_1 and μ_2 are the random number in $[0, 1]$, and j denotes the random integer of the interval $[1, \text{num}_E]$, $i = 1, 2, \ldots, \text{num}_M$, where num_M indicates the total number of mutation sparks.

The detailed codes of mutation operator are represented in Algorithm 2.

3.5. Main Process of PS-FW. In PS-FW, the algorithm consists of two main stages, which are initialization stage and iterations stage. In the initialization phase, we need to initialize the position and velocity of the particle swarm, as well as to initialize the control parameters. In the iterative phase, the PS-FW algorithm inherits all the parameters and operators of the PSO algorithm, and all particles are used as the main carrier for storing feasible solutions. Firstly, in each iteration, the particles update their speed and position according to the operators of the PSO algorithm and then perform the abandonment and supplement operation. Besides, in the process of generating the supplement particles by using the operators of FWA, we first generate num_E explosion sparks according to the excellent P_{num} particles and the modified explosion operator; then the fitness of the explosion sparks is given. Secondly, the num_M mutation sparks are generated by the explosion sparks and the novel mutation operator. Finally, the FW_{num} supplement individuals are selected by the

(1) Input: num_E explosion sparks and best explosion spark $\hat{x}_{\text{best}}$
(2) for $i = 1$ to num_M do
(3) Generate a random integer j in the interval $[1, \text{num}_E]$.
(4) Initialize the location of the ith mutation spark: $\tilde{x}_i = \hat{x}_j$.
(5) Calculate the number of dimensions to perform the mutation: $\tilde{z}_i = D \cdot \text{rand}()$.
(6) Randomly select $\tilde{z}_i$ dimensions of $\tilde{x}_i$.
(7) for each dimension $\tilde{x}_{i,k} \in$ pre-selected $\tilde{z}_i$ dimensions of $\tilde{x}_i$ do
(8) Calculate the value of $\tilde{x}_{i,k}$ by using (16).
(9) if $\tilde{x}_{i,k} > S_{\max,k}$ or $\tilde{x}_{i,k} < S_{\min,k}$ do
(10) Update the value of $\tilde{x}_{i,k}$ by using (17).
(11) end if
(12) end for
(13) end for
(14) Output: num_M mutation sparks.

Algorithm 2: Generating mutation sparks by the mutation operator of PS-FW.

combination of elite strategy and roulette strategy. When each iteration is completed, it is judged whether the termination condition is satisfied. If the stopping criterion is matched, the iteration will be stopped and the best solutions are output. Otherwise, the iteration phase will be repeated.

In the procedures above, there are two points to be noted. (i) In the implementation process of the hybrid algorithm, it is necessary to detect whether the position of individuals is within the feasible scope while the individuals consist of particles, fireworks, explosion sparks, and mutation sparks. As shown in (17), if the position of individuals exceeds the feasible scope, it is adjusted by using the mapping criteria in the EFWA algorithm [29].

$$Y_{i,k} = S_{\min,k} + e \cdot \left(S_{\max,k} - S_{\min,k}\right) \quad \forall Y_{i,k} > S_{\max,k} \text{ or } Y_{i,k} < S_{\min,k}, \tag{17}$$

where $Y_{i,k}$ indicates the value of the kth dimension of the individual and e is a random number in $[0, 1]$.

(ii) The selection strategy of FWA based on the density of feasible solutions is abandoned in the PS-FW algorithm. Although it is possible to maintain the diversity of the population by selecting the location which has fewer individuals around with a larger probability, relatively more time is wasted by calculating the spatial distance between the individuals and the efficiency of the algorithm is reduced. Therefore, a selection strategy based on fitness is applied in PS-FW, which means the elite strategy is used to retain the best individual directly into the next iteration and the remaining $\text{FW}_{\text{num}} - 1$ locations are selected by the roulette criterion according to the fitness.

According to the description above, the main codes of the PS-FW algorithm are given in Algorithm 3.

4. Problems, Experiments, and Discussion

4.1. Test Problems. In order to evaluate the efficacy and accuracy of the proposed algorithm, the performance of PS-FW is tested by the 22 high-dimensional benchmark functions. The test problems which consist of multimodal functions and unimodal functions are listed in Table 1, and the corresponding optimal solutions and search scope are presented in Table 1. Compared with solving unimodal problems, it is difficult to find the global optimum of multimodal problems because the local optima will induce the optimization algorithms' fall into their surroundings. Therefore, if the algorithm can efficiently find the optimal solutions of multimodal functions, it can be proved that the algorithm is an excellent optimization algorithm.

4.2. Comparison of PS-FW with PSO and FWA. In this section, we compare the performance of the PS-FW with the PSO and FWA based on the 22 benchmark functions. In order to explore global optimization capability of the three algorithms on solving the high-dimensional optimization problem, three experiments with different dimensions are carried out. The dimensions of experiments are set to $D = 30$, $D = 60$, and $D = 100$, respectively, and each algorithm is used to solve all the benchmark functions 20 times independently. In order to make a fair comparison, the general control parameters of algorithms such as the maximum number of iterations ($I_{\max}$) and the population size (M) are set to be of the same value. $I_{\max}$ is set to 1000 and M is set to 50 for each function. Besides, the algorithms used in the experiment are coded by MATLAB 14.0, and the experiment platform is a personal computer with Core i5, 2.02 GHz CPU, 4 G memory, and Windows 7. For the purpose of eliminating the impact on performance caused by the difference in parameter settings, the main control parameters of PS-FW algorithm are consistent with those of PSO and FWA, and the other detailed control parameters are shown in Table 2.

For all the benchmark functions, the mean and standard deviation of best solutions obtained by PS-FW and other algorithms in 20 independent runs are recorded, and the optimization results are shown in Tables 3–5. Meanwhile, the ranks are also presented in tables, and the three algorithms are ranked mainly based on the mean of best solutions. In addition, the average convergence speed of the proposed PS-FW is compared with other algorithms for functions f_{12}, f_{13}, and f_{20}; therefore the convergence curves are shown in Figure 3.

According to the ranks shown in Tables 3–5, the average values of best solutions for the proposed PS-FW outperform those of the other algorithms. Besides, the performance of PS-FW over standard deviation of best solutions is also better than the rest of the algorithms. For 22 problems with $D = 30$, the PS-FW can obtain the global optimum of f_2, f_3, f_4, f_5, f_6, f_8, f_{12}, f_{15}, f_{17}, f_{18}, f_{20}, and f_{21}, which shows excellent ability for solving optimization problems. As the dimensions of problems increase, the hybrid algorithm maintains outstanding performance and obtains the optimal solutions of the 10 functions, except for functions f_3 and f_6, compared with results in Table 3. When the dimensions of

(1) Input: Objective function $f(x)$ and constraints.
(2) **Initialization**
(3) Parameters initialization: assign values to M, $w_{\max}$, $w_{\min}$, c_1, c_2, $\widehat{A}$, M_e, ε, δ, a, b, r, num_M, $I_{\max}$, $\text{FW}_{\max}$, $\text{FW}_{\min}$, $\lambda_{\min}$, $\lambda_{\max}$
(4) Population initialization: generate the random values for x_i and v_i of each particle in the feasible domain, calculate the *gbest* of initial population.
(5) Set $pbest_i = x_i$, $(i = 1, 2, \ldots, M)$ and $t = 0$.
(6) **Iterations**
(7) while $t \leq I_{\max}$
(8) $t = t + 1$
(9) for $i = 1$ to M
(10) for $j = 1$ to D
(11) Update the velocity of particle x_i by using (1).
(12) Update the position of particle x_i by using (2).
(13) if $x_{i,k} > S_{\max,k}$ or $x_{i,k} < S_{\min,k}$
(14) Update the value of $x_{i,k}$ by using (17).
(15) end if
(16) end for
(17) end for
(18) Calculate FW_{num} by using the (9).
(19) Sort the particle population in ascending order and select the P_{num} particles with better fitness.
(20) Generate num_E explosion sparks by using Algorithm 1.
(21) Calculate the fitness of explosion sparks and storage the best explosion spark $\widehat{x}_{\text{best}}$.
(22) Generate num_M mutation sparks by using Algorithm 2.
(23) Select the FW_{num} individuals from the explosion sparks and mutation sparks by using the selection strategy.
(24) Combine the P_{num} particles with FW_{num} individuals to generate the new population.
(25) Calculate *gbest* and $pbest_i$ of new population.
(26) end while
(27) Output: $gbest = (gbest_1, gbest_2, \ldots, gbest_D)$

Algorithm 3: The main codes of PS-FW algorithm.

problems are 60 and 100, PS-FW can get the global optimum of functions f_3 and f_6, but not each run can succeed. This is because functions f_3 and f_6 are multimodal problems and the number of local optima increases rapidly as the dimensions of the problems increase, which adds the difficulty of avoiding trapping in the local optima. In addition, according to the ranks and values shown in Tables 3–5, the PS-FW can get the highest rank for all the functions. It is also needed to point out that the PS-FW obtains more stable solutions than PSO and FWA for all problems with the increasing of dimensionality. The convergence speed of the three algorithms can be seen in Figure 3, and the descend rate of average best solutions of PS-FW is obviously higher than the other two algorithms. This is because the advantages of PSO and FWA are combined into the PS-FW so that the hybrid algorithm enhances its global and local search ability. Therefore, PS-FW is efficient and robust in dealing with the high-dimensional benchmark functions.

From the above analysis, it is possible to show that the PS-FW algorithm performs well in solving the functions in Table 1. However, because the optimums of these functions are mostly at the origin, we need to further explore the performance of PS-FW algorithm on the nonzero problems. Then the experiment of nonzero problems is carried out to prove the comprehensive performance of PS-FW. In this experiment, the optimums of test functions derived from Table 1 are shifted and the specific values are displayed in Table 6. In addition, in order to achieve a fair comparison between the experiments, the parameters settings of three algorithms are consistent with Table 2 and the dimension is set to $D = 30$. The optimization results of three algorithms are shown in Table 7 and the convergence curves of three algorithms over functions f_{12}, f_{13}, and f_{20} are displayed in Figure 4.

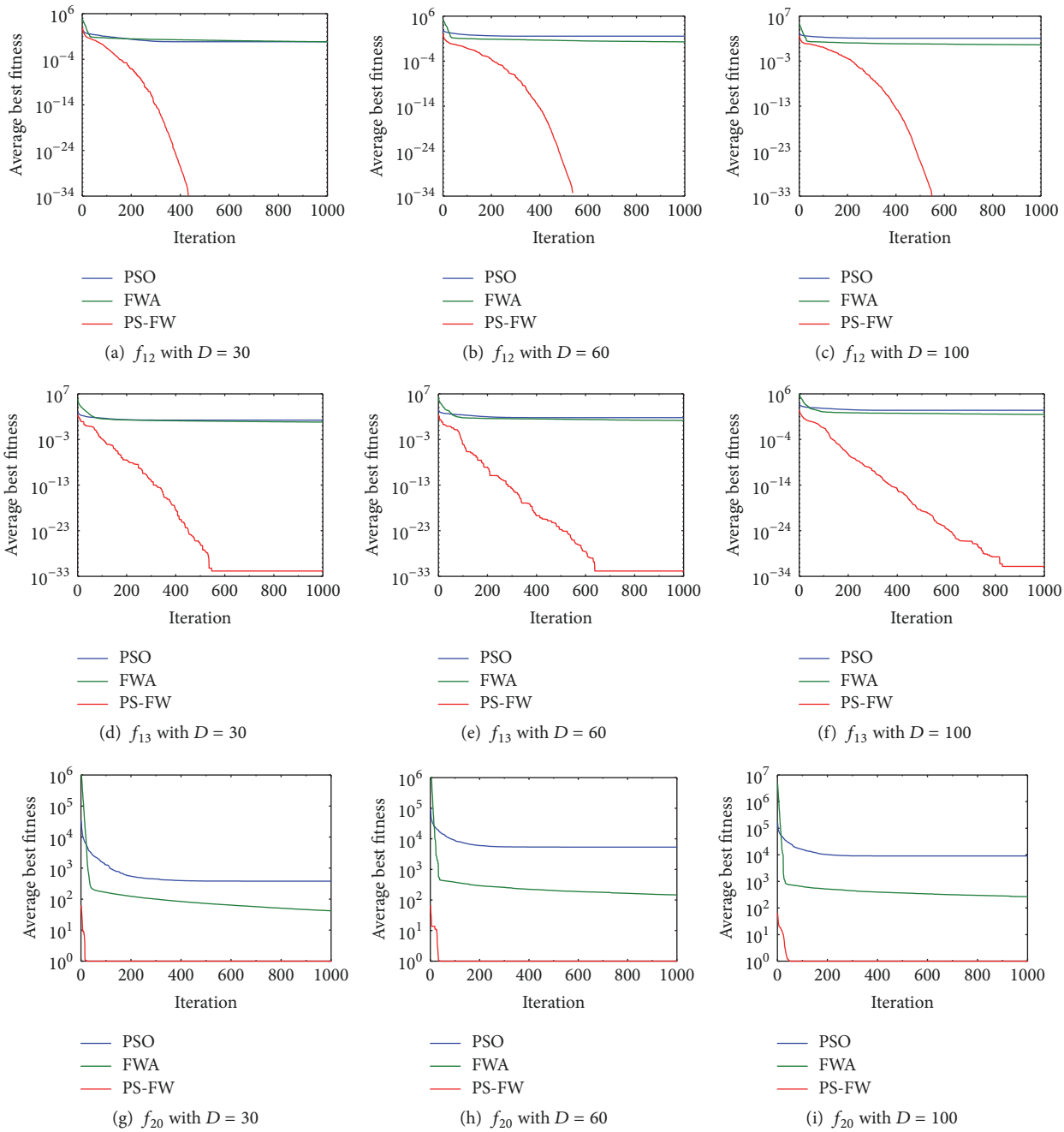
From Table 7, we can know that the PS-FW algorithm keeps high performance and can obtain the optimal solutions of 11 functions in Table 6. Besides, the PS-FW achieves the best rank of three algorithms for all the functions with shift optimums, which present the powerful solving ability over optimization problems with nonzero optimums. By comparing Table 7 with Table 3, it is known that fireworks algorithm is relatively weak in searching for nonzero optimums. However, the PS-FW algorithm that derives from the fireworks algorithm and covers operators of PSO shows better performance, which demonstrates the correctness of the combination of the two algorithms. In addition, the result of PS-FW over function 16 is worse than the previous experiment. This is because f_{16} is a multimodal function and the slight deviations from the optimums can cause the significant increase in the value of the objective function. By observing the convergence curves in Figure 4, we can state that the convergence speed of the PS-FW also remains fast. In order to determine whether the convergence performance of PS-FW algorithm is superior to the other two algorithms

Table 1: The 22 high-dimensional benchmark functions.

Name	Function	Search space	Optimum
Sphere	$f_1(x) = \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	0
Griewank	$f_2(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{D} \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$[-600, 600]^D$	0
Rosenbrock	$f_3(x) = \sum_{i=1}^{D-1} \left[100\left(x_{i+1} - x_i^2\right)^2 + \left(x_i - 1\right)^2\right]$	$[-5, 10]^D$	0
Rastrigin	$f_4(x) = 10D + \sum_{i=1}^{D} \left[x_i^2 - 10\cos\left(2\pi x_i\right)\right]$	$[-5.12, 5.12]^D$	0
Noncontinuous Rastrigin	$f_5(x) = \sum_{i=1}^{D} y_i^2 - 10\cos(2\pi y_i) + 10$ $y_i = \begin{cases} x_i & \lvert x_i\rvert < 0.5 \\ \frac{\text{round}\left(2x_i\right)}{2} & \lvert x_i\rvert \geq 0.5 \end{cases}$	$[-5, 10]^D$	0
Ackley	$f_6(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D} x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos\left(2\pi x_i\right)\right) + 20 + e$	$[-30, 30]^D$	0
Rotated Hyper-Ellipsoid	$f_7(x) = \sum_{i=1}^{D}\sum_{j=1}^{i} x_j^2$	$[-65536, 65536]^D$	0
Noisy Quadric	$f_8(x) = \sum_{i=1}^{D} i x^4 + \text{rand}$	$[-1.28, 1.28]^D$	0
Schwefel's problem 2.21	$f_9(x) = \max_{1\leq i\leq D} \lvert x_i\rvert$	$[-100, 100]^D$	0
Schwefel's problem 2.22	$f_{10}(x) = \sum_{i=1}^{D} \lvert x_i\rvert + \prod_{i=1}^{D} \lvert x_i\rvert$	$[-100, 100]^D$	0
Schwefel's problem 2.26	$f_{11}(x) = \sum_{i=1}^{D} -x_i \sin\left(\sqrt{\lvert x_i\rvert}\right)$	$[-500, 500]^d$	$-418.9829D$
Step	$f_{12}(x) = \sum_{i=1}^{D} \left(\left[x_i + 0.5\right]\right)^2$	$[-10, 10]^D$	0
Levy	$f_{13}(x) = \sin^2\left(\pi y_1\right) + \sum_{i=1}^{D-1} \left(y_i - 1\right)^2 \left[1 + 10\sin^2\left(\pi y_i + 1\right)\right] + \left(y_D - 1\right)^2 \left[1 + \sin^2\left(2\pi y_D\right)\right]$ $y_i = 1 + \frac{x_i - 1}{4}$	$[-10, 10]^D$	0
Powell Sum	$f_{14}(x) = \sum_{i=1}^{D} \lvert x_i\rvert^{i+1}$	$[-1, 1]^D$	0
Sum squares	$f_{15}(x) = \sum_{i=1}^{D} i x_i^2$	$[-10, 10]^D$	0
Zakharov	$f_{16}(x) = \sum_{i=1}^{D} x_i^2 + \left(\sum_{i=1}^{D} 0.5 i x_i\right)^2 + \left(\sum_{i=1}^{D} 0.5 i x_i\right)^4$	$[-5, 10]^D$	0
Mishra 7	$f_{17}(x) = \left(\prod_{i=1}^{D} x_i - D!\right)^2$	$[-D, D]^D$	0
Weierstrass	$f_{18}(x) = \sum_{i=1}^{D} \left[\sum_{k=0}^{k_{\max}} \left(a^k \cos\left(2\pi b^k \left(x_i + 0.5\right)\right)\right) - D\sum_{k=0}^{k_{\max}} a^k \cos\left(\pi b^k\right)\right]$ $a = 0.5, b = 3, k_{\max} = 20$	$[-0.5, 0.5]^D$	0
Bent-Cigar	$f_{19}(x) = x_1^2 + 10^6 \sum_{i=1}^{D} x_i^2$	$[-100, 100]^D$	0

Table 1: Continued.

Name	Function	Search space	Optimum
Trigonometric 2	$f_{20}(x) = 1 + \sum_{i=1}^{D} 8\sin^2\left[7(x_i - 0.9)^2\right] + 6\sin^2\left[14(x_i - 0.9)^2\right] + (x - 0.9)^2$	$[-500, 500]^D$	1
Quintic	$f_{21}(x) = \sum_{i=1}^{D} \left\lvert x_i^5 - 3x_i^4 + 4x_i^3 + 2x_i^2 - 10x_i - 4 \right\rvert$	$[-10, 10]^D$	0
Mishra 11	$f_{22}(x) = \left[\frac{1}{D}\sum_{i=1}^{D} \lvert x_i \rvert + \left(\prod_{i=1}^{D} \lvert x_i \rvert \right)^{1/D} \right]^2$	$[-10, 10]^D$	0

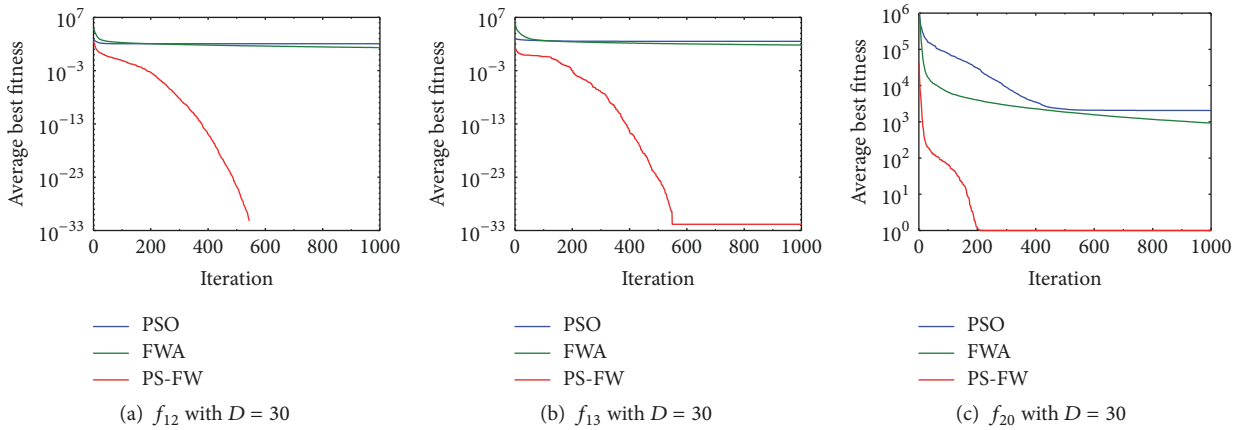


(a) f_{12} with $D = 30$ (b) f_{12} with $D = 60$ (c) f_{12} with $D = 100$

(d) f_{13} with $D = 30$ (e) f_{13} with $D = 60$ (f) f_{13} with $D = 100$

(g) f_{20} with $D = 30$ (h) f_{20} with $D = 60$ (i) f_{20} with $D = 100$

Figure 3: Convergence curves of PSO, FWA, and PS-FW for functions f_{12}, f_{13}, and f_{20}.



(a) f_{12} with $D = 30$ (b) f_{13} with $D = 30$ (c) f_{20} with $D = 30$

Figure 4: Convergence curves of PSO, FWA, and PS-FW for functions f_{12}, f_{13}, and f_{20}.

Table 2: The parameter setting of the algorithms.

Algorithm	Parameter settings
PSO	$w(t) = w_{\max} - t\frac{w_{\max} - w_{\min}}{I_{\max}}$, $w_{\max} = 0.95$, $w_{\min} = 0.4$, $c_1 = c_2 = 1.45$
FWA	$\widehat{A} = 40$, $Me = 50$, $a = 0.04$, $b = 0.8$, $\text{num}_M = 30$, $\varepsilon = 1E - 100$
PS-FW	$w(t) = w_{\max} - t\frac{w_{\max} - w_{\min}}{I_{\max}}$, $w_{\max} = 0.95$, $w_{\min} = 0.4$, $c_1 = c_2 = 1.45$, $\widehat{A} = 40$, $Me = 50$, $a = 0.04$, $b = 0.8$, $\text{num}_M = 30$, $\varepsilon = 1E - 100$, $\delta = 1E - 6$, $\lambda_{\min} = 1E - 25$, $\lambda_{\max} = 1$, $\text{FW}_{\max} = 30$, $\text{FW}_{\min} = 20$, $r = 2$

more clearly, we compute the number of successful runs (success rate) and the average number of iterations in successful runs for each function in Table 6. The optimal solutions obtained by different algorithms are various, so we define the convergence criterion for each function. The convergence criterion can be introduced as that if the best solutions f_{find} found by each of algorithms are satisfying (18) in a run [39], the run is considered to be successful, and the minimum number of iterations satisfying the convergence criterion is counted to calculate the average number of iterations.

$$\left|f_{\text{find}} - f_{\text{opti}}\right| < \tau, \tag{18}$$

where f_{opti} is the optimum of function and τ denotes the error of algorithm.

Suppose that ST denotes the number of successful runs, AI indicates the average number of iterations in successful runs, and U denotes the iterations number when there are no successful runs after 20 runs and its value is set to greater than $I_{\max}$; then Table 8 is shown as follows.

According to the statistical results and ranks presented in Table 8, the success rate and the average iterations number of PS-FW in 20 runs are both superior to other algorithms. For all the benchmark functions in Table 6, the proposed PS-FW can satisfy the convergence criterion for all the 20 runs, whereas the other algorithms can only converge to the criterion for several functions. In addition, the PS-FW obtains the highest ranks for the average number of iterations in successful runs and can converge to the criterion by a relatively small number of iterations. In summary, the PS-FW outperforms the other algorithms in terms of stability and convergence speed and is an efficacious algorithm for optimization problems whose optimums are at origin or are shifted.

4.3. Comparison of PS-FW with PSO Variants. In this section, we compare the performance of the proposed PS-FW with several existing variants of PSO which are introduced in a published paper. The comparison is based on the 12 benchmark functions introduced in the paper of Nickabadi et al. [22] and the orders of functions are consistent with that in this paper. In order to make a fair comparison, the run times and maximum iterations of PS-FW are set to 30 and 200,000, respectively, and the other parameters are set to be the same as those in Section 4.2. The dimension of test problems is set to $D = 30$, and the mean and standard deviation of best solutions obtained by algorithms are calculated. The contrast results are presented in Table 9, and the rank of each algorithm is counted and shown.

According to the results of Table 9, the PS-FW outperforms the other six PSO variants on both the average values and standard deviation of best solutions after 200,000 iterations. Among the 12 benchmark functions, the PS-FW can obtain the optimum of 10 functions, which manifests the highly powerful ability to find the global optimal solution. In addition, the PS-FW acquires the highest rank over almost all the test problems except the function f_{11}, which indicates the PS-FW has significant improvement than other algorithms. Besides the analysis of numerical results obtained by PS-FW and other algorithms, we applied the nonparametric statistical tests to prove the superiority of the PS-FW. The Friedman test and Bonferroni-Dunn test are adopted to compare the performance of PS-FW with the other algorithms.

The Friedman test is a multiple comparison test, to detect the significant differences among algorithms based on the

Table 3: Comparison of the optimization results obtained by PS-FW, PSO, and FWA with $D = 30$ for functions f_1 to f_{22} (the best ranks are marked in bold).

f	D		PSO	FWA	PS-FW
f_1	30	Mean	$8.8371E+01$	$1.3360E-151$	$5.8928E-264$
		Std	$4.3475E+01$	$5.8057E-151$	0
		Rank	3	2	**1**
f_2	30	Mean	$7.1542E-02$	0	0
		Std	$1.2385E-01$	0	0
		Rank	2	**1**	**1**
f_3	30	Mean	$5.5766E+02$	$2.6882E+01$	0
		Std	$7.4828E+02$	$8.3997E-01$	0
		Rank	3	2	**1**
f_4	30	Mean	$6.6547E+01$	0	0
		Std	$3.6430E+01$	0	0
		Rank	2	**1**	**1**
f_5	30	Mean	$6.5810E+01$	0	0
		Std	$4.0117E+01$	0	0
		Rank	2	**1**	**1**
f_6	30	Mean	0	0	0
		Std	0	0	0
		Rank	**1**	**1**	**1**
f_7	30	Mean	$1.4156E+04$	$7.6585E-83$	$4.5128E-122$
		Std	$1.0006E+04$	$3.3383E-82$	$1.8821E-121$
		Rank	3	2	**1**
f_8	30	Mean	$1.0419E-03$	$9.6596E-304$	0
		Std	$1.0584E-03$	0	0
		Rank	3	2	**1**
f_9	30	Mean	$6.3165E-01$	$7.4698E-54$	$3.1588E-97$
		Std	$6.0679E-01$	$2.3638E-53$	$1.2719E-96$
		Rank	3	2	**1**
f_{10}	30	Mean	$1.5661E+01$	$3.2521E-78$	$1.8666E-137$
		Std	$5.0924E+00$	$1.1460E-77$	$8.0013E-137$
		Rank	3	2	**1**
f_{11}	30	Mean	$-7.2662E+03$	$-1.0511E+04$	$-1.2483E+04$
		Std	$6.7867E+02$	$1.9893E+02$	$1.2661E+02$
		Rank	3	2	**1**
f_{12}	30	Mean	$6.9734E-01$	$6.6542E-01$	0
		Std	$2.8586E-01$	$5.0080E-01$	0
		Rank	3	2	**1**
f_{13}	30	Mean	$1.7831E+01$	$6.5460E+00$	$1.4998E-32$
		Std	$8.6204E+00$	$8.6700E-01$	0
		Rank	3	2	**1**
f_{14}	30	Mean	$6.6576E-08$	$4.5613E-191$	$2.1563E-291$
		Std	$5.4575E-08$	0	0
		Rank	3	2	**1**
f_{15}	30	Mean	0	0	0
		Std	0	0	0
		Rank	**1**	**1**	**1**
f_{16}	30	Mean	$2.8937E+02$	$1.5997E-45$	$1.5471E-111$
		Std	$1.5937E+02$	$3.5711E-45$	$6.0668E-111$
		Rank	3	2	**1**

Table 3: Continued.

f	D		PSO	FWA	PS-FW
f_{17}	30	Mean	0	$9.8737E+44$	0
		Std	0	$4.3038E+45$	0
		Rank	**1**	2	**1**
f_{18}	30	Mean	$1.5069E+01$	0	0
		Std	$4.0495E+00$	0	0
		Rank	2	**1**	**1**
f_{19}	30	Mean	$2.8450E+07$	$1.0123E-145$	$1.8302E-252$
		Std	$1.2385E+08$	$3.1288E-145$	0
		Rank	3	2	**1**
f_{20}	30	Mean	$3.8005E+02$	$4.2079E+01$	1
		Std	$8.5739E+01$	$4.6125E+00$	0
		Rank	3	2	**1**
f_{21}	30	Mean	$4.5577E+01$	$1.71130E+01$	0
		Std	$2.3091E+01$	$2.1499E+00$	0
		Rank	3	2	**1**
f_{22}	30	Mean	$7.0166E-01$	$1.1989E-149$	$3.5102E-292$
		Std	$5.9846E-01$	$5.2258E-149$	0
		Rank	3	2	**1**
Average rank			2.5455	1.7273	**1**
Overall rank			3	2	**1**

sets of data [40]. The algorithms are ranked in Friedman test, which means the algorithm with the best performance is ranked minimum, the worst gets the maximum rank, and so on. In this section, the mean and standard deviation of best solutions based on Table 9 are conducted with the Friedman test; therefore the results are given in Table 10. Through observing the results of Friedman test in Table 10, all the p value are lower than the level of significance considered $\alpha = 0.01$, which indicates that the significant differences among the seven algorithms do exist. According to the ranks obtained by the Friedman test in Table 10, the PS-FW has the best performance on the mean and standard deviation of best solutions followed by ALWPSO, CLPSO, and the other four algorithms. Therefore, we can conclude that the accuracy of solutions obtained by PS-FW is better than other algorithms. However, the Friedman test can only detect whether there are significant differences among all the algorithms, but is unable to conduct the proper comparisons between PS-FW and each of the other algorithms. Hence the Bonferroni-Dunn test is executed to check the superiority of PS-FW.

The Bonferroni-Dunn test can be very intuitive to detect the significant difference between the two or more algorithms. For Bonferroni-Dunn test, the judgment condition for the existence of significant difference between the two algorithms is that their mean ranks differ by at least the critical difference (CD), and the equation of calculating the critical difference is as follows [41]:

$$\mathrm{CD}_{\alpha} = q_{\alpha}\sqrt{\frac{N_i\left(N_i+1\right)}{6N_f}}, \tag{19}$$

where N_i and N_f are the number of algorithms and benchmark functions and the critical values q_{α} at the probability level a are presented as follows:

$$\begin{aligned} q_{0.05} &= 2.77, \\ q_{0.1} &= 2.54. \end{aligned} \tag{20}$$

By utilizing (19) and (20), the critical difference is shown as follows:

$$\begin{aligned} \mathrm{CD}_{0.05} &= 2.44, \\ \mathrm{CD}_{0.1} &= 2.24. \end{aligned} \tag{21}$$

Here we carry out the Bonferroni-Dunn test for the mean of best solutions, success rate, and average number of iterations of successful runs on the basis of the ranks obtained by the Friedman test. In order to provide a more intuitive display of the results obtained by Bonferroni-Dunn test, we illustrate the critical differences among the seven algorithms in Figure 5. For the purpose of comparing the algorithms clearly, a horizontal line which indicates the threshold for the best performing algorithm (the one with pink color) is drawn in the graphs. In addition, another two lines which represent each level of significance considered in the paper are also drawn, and their heights are equal to the sum of minimum rank and the corresponding CD. Then if the bars exceed the lines of significant level, the corresponding algorithms are proved to have worse performance than the best performing algorithm. By observing the results of Bonferroni-Dunn test in Figure 5(a), the bar of the PS-FW has the lowest height among all the algorithms, and the heights of bars corresponding to the

Table 4: Comparison of the optimization results obtained by PS-FW, PSO, and FWA with $D = 60$ for functions f_1 to f_{22} (the best ranks are marked in bold).

f	D		PSO	FWA	PS-FW
f_1	60	Mean	$4.1677E+03$	$2.1235E-146$	$2.4481E-248$
		Std	$4.4284E+03$	$6.3705E-146$	0
		Rank	3	2	**1**
f_2	60	Mean	$3.2482E+00$	0	0
		Std	$9.6094E-01$	0	0
		Rank	2	**1**	**1**
f_3	60	Mean	$7.1638E+04$	$4.5073E+01$	$9.2568E-30$
		Std	$5.5811E+04$	$1.8390E+01$	$1.9330E-29$
		Rank	3	2	**1**
f_4	60	Mean	$3.2219E+02$	0	0
		Std	$4.1863E+01$	0	0
		Rank	2	**1**	**1**
f_5	60	Mean	$3.7498E+02$	0	0
		Std	$5.3191E+01$	0	0
		Rank	2	**1**	**1**
f_6	60	Mean	$1.3162E+01$	0	$7.1054E-16$
		Std	$1.1773E+00$	0	$1.4211E-15$
		Rank	3	**1**	2
f_7	60	Mean	$3.2017E+04$	$4.9633E-68$	$1.2294E-93$
		Std	$1.4529E+04$	$1.48899E-67$	$4.9341E-93$
		Rank	3	2	**1**
f_8	60	Mean	$1.1343E+00$	$1.2096E-288$	0
		Std	$3.2234E+00$	0	0
		Rank	3	2	**1**
f_9	60	Mean	$2.6902E+01$	$4.4049E-51$	$1.5914E-92$
		Std	$5.4555E+00$	$1.3214E-50$	$4.8189E-92$
		Rank	3	2	**1**
f_{10}	60	Mean	$5.5140E+01$	$1.35612E-73$	$3.9617E-130$
		Std	$2.1038E+01$	$4.06287E-73$	$1.7268E-129$
		Rank	3	2	**1**
f_{11}	60	Mean	$-1.1892E+04$	$-1.8005E+04$	$-2.4998E+04$
		Std	$1.1022E+03$	$1.4727E+03$	$1.7201E+02$
		Rank	3	2	**1**
f_{12}	60	Mean	$3.4856E+01$	$1.9695E+00$	0
		Std	$5.9316E+01$	$7.7525E-01$	0
		Rank	3	2	**1**
f_{13}	60	Mean	$6.2329E+01$	$1.5355E+01$	$1.4998E-32$
		Std	$2.0956E+01$	$5.4415E+00$	0
		Rank	3	2	**1**
f_{14}	60	Mean	$2.2365E-07$	$1.6432E-187$	$1.5707E-278$
		Std	$2.3968E-07$	0	0
		Rank	3	2	**1**
f_{15}	60	Mean	0	0	0
		Std	0	0	0
		Rank	**1**	**1**	**1**
f_{16}	60	Mean	$8.0994E+02$	$1.7189E-38$	$6.8924E-104$
		Std	$3.0726E+02$	$5.15482E-38$	$2.9641E-103$
		Rank	3	2	**1**

Table 4: Continued.

f	D		PSO	FWA	PS-FW
f_{17}	60	Mean	0	$2.4945E+145$	0
		Std	0	$5.7208E+145$	0
		Rank	**1**	2	**1**
f_{18}	60	Mean	$3.9564E+01$	0	0
		Std	$5.3138E+00$	0	0
		Rank	2	**1**	**1**
f_{19}	60	Mean	$5.7753E+08$	$6.6011E-137$	$4.5120E-251$
		Std	$2.7159E+08$	$1.9631E-136$	0
		Rank	3	2	**1**
f_{20}	60	Mean	$5.3645E+03$	$1.4665E+02$	1
		Std	$6.2256E+03$	$2.8947E+01$	0
		Rank	3	2	**1**
f_{21}	60	Mean	$1.9709E+02$	$4.8085E+01$	0
		Std	$2.8605E+01$	$7.7355E+00$	0
		Rank	3	2	**1**
f_{22}	60	Mean	$1.5314E+00$	$1.5711E-142$	$1.3216E-280$
		Std	$5.9245E-01$	$4.7133E-142$	0
		Rank	3	2	**1**
Average rank			2.6364	1.7273	**1.0455**
Overall rank			3	2	**1**

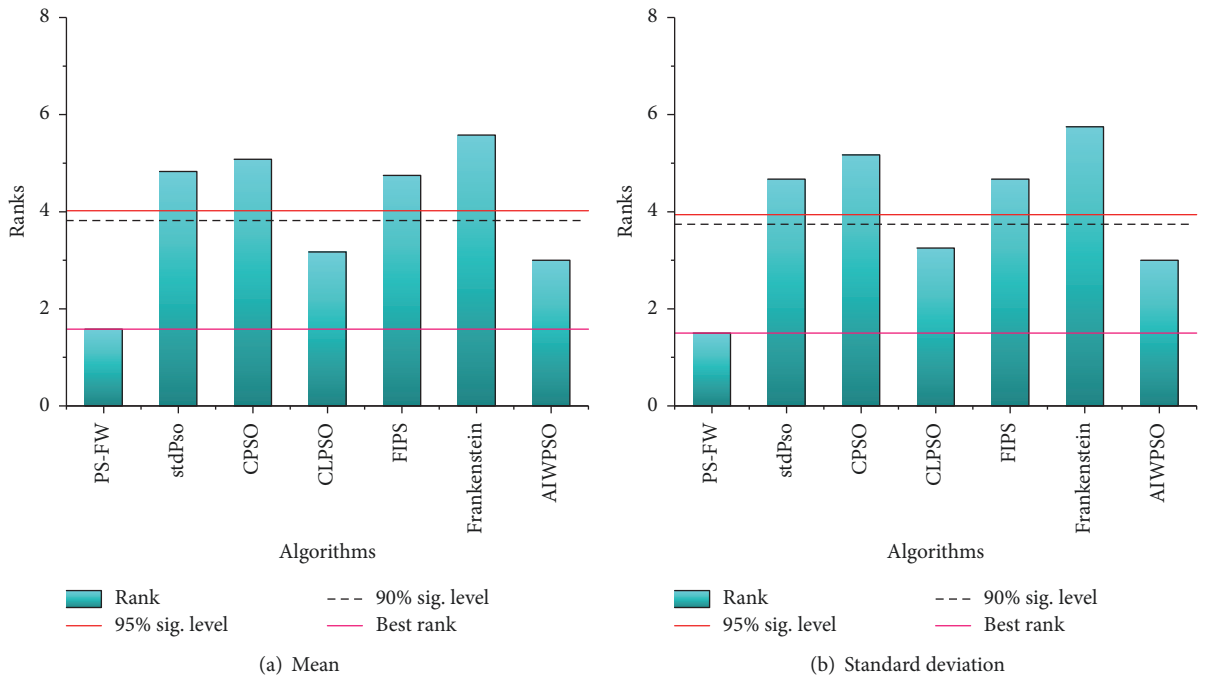


Figure 5: The bar chart of Bonferroni-Dunn test for PS-FW and other PSO variants over mean and standard deviation of best solutions based on Table 10.

stdPSO, CPSO, FIPS, and Frankenstein exceed the lines of significant level, which indicates that the PS-FW performs significantly better than these four algorithms over the solutions accuracy. In addition, the PS-FW acquires the best rank over the standard deviation according to Figure 5(b), and the PS-FW has the obvious advantage compared to the stdPSO, CPSO, FIPS, and Frankenstein. Therefore, we can conclude that the PS-FW is the best performing algorithm followed by ALWPSO, CLPSO, and other four algorithms, and the advantages of PS-FW on the efficiency and solutions accuracy compared with other algorithms are definitely proved.

Table 5: Comparison of the optimization results obtained by PS-FW, PSO, and FWA with $D = 100$ for functions f_1 to f_{22} (the best ranks are marked in bold).

f	D		PSO	FWA	PS-FW
f_1	100	Mean	$6.3501E+03$	$1.7672E-142$	$9.7833E-245$
		Std	$2.9204E+03$	$4.3844E-142$	0
		Rank	3	2	**1**
f_2	100	Mean	$1.1830E+02$	0	0
		Std	$5.1822E+01$	0	0
		Rank	2	**1**	**1**
f_3	100	Mean	$1.7018E+05$	$8.3094E+01$	$1.0341E-26$
		Std	$6.6940E+04$	$2.2198E+01$	$3.8500E-26$
		Rank	3	2	**1**
f_4	100	Mean	$4.7288E+02$	0	0
		Std	$1.0713E+02$	0	0
		Rank	2	**1**	**1**
f_5	100	Mean	$5.1626E+02$	0	0
		Std	$1.4819E+02$	0	0
		Rank	2	**1**	**1**
f_6	100	Mean	$1.3582E+01$	0	$1.0659E-15$
		Std	$2.3679E+00$	0	$1.6281E-15$
		Rank	3	**1**	2
f_7	100	Mean	$2.7218E+06$	$2.70634E-58$	$2.1860E-71$
		Std	$8.2328E+05$	$8.11903E-58$	$4.7535E-71$
		Rank	3	2	**1**
f_8	100	Mean	$1.4283E+01$	$1.5868E-280$	0
		Std	$3.8266E+01$	0	0
		Rank	3	2	**1**
f_9	100	Mean	$2.7189E+01$	$4.2938E-46$	$1.1555E-90$
		Std	$5.0564E+00$	$1.1238E-45$	$2.7315E-90$
		Rank	3	2	**1**
f_{10}	100	Mean	$1.2486E+02$	$2.64613E-69$	$2.2792E-128$
		Std	$2.3963E+01$	$7.93838E-69$	$9.7764E-128$
		Rank	3	2	**1**
f_{11}	100	Mean	$-1.5770E+04$	$-2.4526E+04$	$-4.1743E+04$
		Std	$1.2531E+03$	$1.6861E+03$	$4.3502E+02$
		Rank	3	2	**1**
f_{12}	100	Mean	$1.2670E+02$	$4.2335E+00$	0
		Std	$4.8966E+01$	1.40825853	0
		Rank	3	2	**1**
f_{13}	100	Mean	$2.4848E+02$	$3.1912E+01$	$1.4998E-32$
		Std	$6.1955E+01$	$7.6762E+00$	0
		Rank	3	2	**1**
f_{14}	100	Mean	$4.7875E-07$	$6.5204E-175$	$6.4751E-275$
		Std	$6.7428E-07$	0	0
		Rank	3	2	**1**
f_{15}	100	Mean	0	0	0
		Std	0	0	0
		Rank	**1**	**1**	**1**
f_{16}	100	Mean	$1.4995E+03$	$1.9628E-14$	$2.4731E-93$
		Std	$5.8180E+02$	$5.86607E-14$	$8.4009E-93$
		Rank	3	2	**1**
f_{17}	100	Mean	0	$2.0047E+232$	0
		Std	0	$6.7205E+232$	0
		Rank	**1**	2	**1**

Table 5: Continued.

f	D		PSO	FWA	PS-FW
f_{18}	100	Mean	$6.8687E+01$	0	0
		Std	$1.3221E+01$	0	0
		Rank	2	**1**	**1**
f_{19}	100	Mean	$1.4528E+10$	$3.3916E-130$	$9.0096E-250$
		Std	$1.2994E+10$	$9.8384E-130$	0
		Rank	3	2	**1**
f_{20}	100	Mean	$9.0245E+03$	$2.6557E+02$	1
		Std	$3.8036E+03$	$4.7674E+01$	0
		Rank	3	2	**1**
f_{21}	100	Mean	$4.0256E+03$	$9.1975E+01$	0
		Std	$1.6131E+04$	$1.7966E+01$	0
		Rank	3	2	**1**
f_{22}	100	Mean	$1.6273E+00$	$4.0925E-137$	$4.9253E-273$
		Std	$4.1513E-01$	$3.2175E-137$	0
		Rank	3	2	**1**
Average rank			2.6364	1.7273	**1.0455**
Overall rank			3	2	**1**

Besides the above analysis, we count the number of successful runs and the average number of iterations in successful runs for the PS-FW over 12 benchmark functions, and the statistical results are presented in Table 11. In this section, a successful run means the algorithm can obtain the optimum within the 200,000 iterations. As shown in Table 11, the PS-FW can converge to the optimal solution in each of runs over the vast majority functions, which manifests the robustness of PS-FW in solving the optimization problems. In order to compare the convergence speed of PS-FW with other algorithms fairly, the average numbers of iterations in successful runs are compared over the six functions f_1, f_4, f_6, f_7, f_{10}, and f_{11} introduced in Nickabadi et al.'s paper. According to the numerical results in Table 11, the PS-FW can converge to the optimal solution for all the six functions within 12,000 iterations, whereas the other algorithms have difficulty in obtaining the optimum for functions f_1, f_6, f_7, and f_{10} after 200,000 iterations or can converge to the optimum for functions f_4, f_{11} with a lot more iterations based on the convergence curves in the paper by Nickabadi et al. Therefore, we can argue that the robustness and convergence speed of PS-FW are superior to the other algorithms.

4.4. Experiments to Analyze the PS-FW Control Parameters. In this section, we investigate the impact of the control parameters on the performance of PS-FW. From the previous introduction, the PS-FW has several control parameters including the parameters adopted from PSO and FWA. Here we only analyze the three main control parameters which are the control factors of explosion amplitudes $\lambda_{\min}$, $\lambda_{\max}$ and the number of mutation sparks num_M. In order to test the impact of changes in control parameters on performance exhaustively, six different combinations of parameters were selected and experimented on. Each set of parameters corresponds to 20 runs based on 22 functions introduced in Table 1, and the dimensions of problems are set to 100. Moreover, the other parameters settings of PS-FW except $\lambda_{\min}$, $\lambda_{\max}$, and num_M are the same as those in Section 4.2. In addition, the six combinations of control parameters are represented as six optimization strategies, and their detailed parameters settings are shown in Table 12, and the control parameters of Section 4.2 are marked as Strategy-1 and are presented. As shown in Table 12, we take a contrasting method that changes a parameter and keeps the other parameters unchanged.

Table 6: The benchmark functions with shift optima.

Name	Original optima	Shift optima
Sphere	$[0, 0, \ldots, 0]$	$[70, 70, \ldots, 70]$
Griewank	$[0, 0, \ldots, 0]$	$[70, 70, \ldots, 70]$
Rastrigin	$[0, 0, \ldots, 0]$	$[3, 3, \ldots, 3]$
Noncontinuous Rastrigin	$[0, 0, \ldots, 0]$	$[5, 5, \ldots, 5]$
Ackley	$[0, 0, \ldots, 0]$	$[20, 20, \ldots, 20]$
Rotated Hyper-Ellipsoid	$[0, 0, \ldots, 0]$	$[70, 70, \ldots, 70]$
Schwefel's problem 2.21	$[0, 0, \ldots, 0]$	$[70, 70, \ldots, 70]$
Schwefel's problem 2.22	$[0, 0, \ldots, 0]$	$[70, 70, \ldots, 70]$
Step	$[-0.5, -0.5, \ldots, -0.5]$	$[5, 5, \ldots, 5]$
Levy	$[1, 1, \ldots, 1]$	$[5, 5, \ldots, 5]$
Sum squares	$[0, 0, \ldots, 0]$	$[5, 5, \ldots, 5]$
Zakharov	$[0, 0, \ldots, 0]$	$[5, 5, \ldots, 5]$
Bent-Cigar	$[0, 0, \ldots, 0]$	$[70, 70, \ldots, 70]$
Trigonometric 2	$[0.9, 0.9, \ldots, 0.9]$	$[70, 70, \ldots, 70]$
Mishra 11	$[0, 0, \ldots, 0]$	$[5, 5, \ldots, 5]$

Table 7: Comparison of the optimization results obtained by PS-FW, PSO, and FWA for functions in Table 6 (the best ranks are marked in bold).

f	D		PSO	FWA	PS-FW
f_1	30	Mean	$1.0851E+03$	$2.2555E+00$	0
		Std	$1.1893E+03$	$3.8190E-01$	0
		Rank	3	2	**1**
f_2	30	Mean	$4.7829E+00$	$6.2867E-01$	0
		Std	$1.5089E+00$	$5.3523E-02$	0
		Rank	3	2	**1**
f_4	30	Mean	$1.2559E+02$	$9.8052E+00$	0
		Std	$4.7596E+01$	$1.6323E+00$	0
		Rank	3	2	**1**
f_5	30	Mean	$1.6140E+02$	$2.2289E+01$	0
		Std	$3.7649E+01$	$2.7981E+00$	0
		Rank	3	2	**1**
f_6	30	Mean	$1.0739E+03$	$7.0977E+00$	0
		Std	$1.1986E+03$	$4.3511E-01$	0
		Rank	3	2	**1**
f_7	30	Mean	$1.5716E+04$	$2.2295E+03$	$4.45263E-65$
		Std	$8.7224E+03$	$2.4129E+02$	$2.87935E-65$
		Rank	3	2	**1**
f_9	30	Mean	$4.7379E+01$	$2.1052E+01$	$8.96847E-72$
		Std	$1.5948E+01$	$1.4289E+00$	$1.31198E-71$
		Rank	3	2	**1**
f_{10}	30	Mean	$1.6846E+03$	$2.2370E+02$	0
		Std	$2.6627E+02$	$7.4690E+01$	0
		Rank	3	2	**1**
f_{12}	30	Mean	$1.1359E+02$	$2.1375E+01$	0
		Std	$4.1907E+01$	$2.9107E+00$	0
		Rank	3	2	**1**
f_{13}	30	Mean	$3.2776E+02$	$6.4154E+01$	$1.4998E-32$
		Std	$8.5157E+01$	$1.0092E+01$	0
		Rank	3	2	**1**
f_{15}	30	Mean	0	$2.9887E-04$	0
		Std	0	$1.3027E-03$	0
		Rank	**1**	2	**1**
f_{16}	30	Mean	$8.0214E+00$	$3.1159E+02$	$1.53313E-06$
		Std	$8.1866E+00$	$2.0373E+02$	$1.06687E-06$
		Rank	2	3	**1**
f_{19}	30	Mean	$2.4875E+09$	$2.2700E+08$	0
		Std	$1.3163E+09$	$2.7319E+07$	0
		Rank	3	2	**1**
f_{20}	30	Mean	$2.0564E+03$	$9.2562E+02$	1
		Std	$7.9311E+02$	$7.6748E+01$	0
		Rank	3	2	**1**
f_{22}	30	Mean	$1.7217E+00$	$1.4009E+00$	0
		Std	$1.1645E+00$	$4.6093E-01$	0
		Rank	3	2	**1**
	Average rank		2.8000	2.0667	**1**
	Overall rank		3	2	**1**

Table 8: Comparison of successful rates and average number of iterations for PS-FW, PSO, and FWA with $\tau = 10^{-4}$ for function f_{15} and $\tau = 10^{1}$ for other functions (the best ranks are marked in bold).

f	PSO	FWA	PS-FW
f_1			
ST	0	20	20
Rank	2	**1**	**1**
AI	U	201.7	28.4
Rank	3	2	**1**
f_2			
ST	19	20	20
Rank	2	**1**	**1**
AI	9.6	4.6	2.8
Rank	3	2	**1**
f_4			
ST	0	11	20
Rank	3	2	**1**
AI	U	584.8	228.8
Rank	3	2	**1**
f_5			
ST	0	0	20
Rank	2	2	**1**
AI	U	U	104.9
Rank	2	2	**1**
f_6			
ST	0	20	20
Rank	2	**1**	**1**
AI	U	343	9.8
Rank	3	2	**1**
f_7			
ST	0	0	20
Rank	2	2	**1**
AI	U	U	93.8
Rank	2	2	**1**
f_9			
ST	0	0	20
Rank	2	2	**1**
AI	U	U	26.7
Rank	2	2	**1**
f_{10}			
ST	0	0	20
Rank	2	2	**1**
AI	U	U	41.1
Rank	2	2	**1**
f_{12}			
ST	0	0	20
Rank	2	2	**1**
AI	U	U	11.8
Rank	2	2	**1**
f_{13}			
ST	0	0	20
Rank	2	2	**1**
AI	U	U	3.5
Rank	2	2	**1**
f_{15}			
ST	20	19	20
Rank	**1**	2	**1**
AI	505.3	679.6	13.1
Rank	2	3	**1**
f_{16}			
ST	16	0	20
Rank	2	3	**1**
AI	224	U	208.7
Rank	2	3	**1**
f_{19}			
ST	0	0	20
Rank	2	2	**1**
AI	U	U	208.9
Rank	2	2	**1**
f_{20}			
ST	0	0	20
Rank	2	2	**1**
AI	U	U	160.8
Rank	2	2	**1**
f_{22}			
ST	20	20	20
Rank	**1**	**1**	**1**
AI	94.2	123.2	9.3
Rank	2	3	**1**
Average rank of ST	1.9	1.8	**1**
Overall rank of AI	2.3	2.2	**1**

Then the optimization results and the corresponding ranks of different strategies are shown in Tables 13 and 14, and the results focus on mean and standard deviation of best solutions obtained by different strategies. From the results of Tables 13 and 14, the PS-FW with Strategy-6 and Strategy-7 has the best performance for almost all the benchmark functions and can obtain the highest ranks over both the mean and standard deviation of best solutions. By adopting Strategy-6 and Strategy-7, the PS-FW can get the optimum of 16 functions for the whole 20 runs, especially including the functions f_1, f_3, f_6, f_{14}, f_{19}, and f_{22} which cannot find the global best solutions by other optimization strategies of PS-FW. Therefore, the excellent performance of PS-FW with Strategy-6 and Strategy-7 proves the correctness of proposed mutation operator and indicates that increasing the number of mutation sparks can enhance the global search capability of the algorithm. However, according to the "no free lunch theorem" [42], there is no algorithm that can perform better than others on all the problems; hence the PS-FW with Strategy-6 and Strategy-7 has poor performance for function f_7. It is because function f_7 has a wide search scope so that the solutions have little changes in the later iterations if $\lambda_{\min}$ is small, which results in a relatively slow convergence speed for PS-FW despite the increase in the number of mutation sparks. For other strategies of PS-FW, the different strategies have their own advantages for various test functions, the PS-FW with Strategy-1 performs well for functions f_1, f_3, f_6, f_9, and f_{19}, and the good solutions can be obtained by PS-FW over functions f_7, f_{16} under Strategy-2 and Strategy-3. Meanwhile, the PS-FW with Strategy-4 and Strategy-5 works well in solving the functions f_{10} and f_{22}. In addition, the PS-FW can obtain the optimum of functions f_2, f_4, f_5, f_8, f_{12}, f_{15}, f_{17}, f_{18}, f_{20}, and f_{21} and keep outstanding

Table 9: Comparison of the optimization results obtained by PS-FW and six PSO variants (the best ranks are marked in bold).

$f(x)$	PS-FW	stdPSO	CPSO	CLPSO	FIPS	Frankenstein	AIWPSO
f_1							
Mean	0	$5.198E-40$	$5.146E-13$	$4.894E-39$	$4.588E-27$	$2.409E-16$	$3.370E-134$
Rank	**1**	3	7	4	5	6	2
Std	0	$1.1301E-78$	$7.7588E-25$	$6.7814E-78$	$1.9577E-53$	$2.0047E-31$	$5.1722E-267$
Rank	**1**	3	7	4	5	6	2
f_2							
Mean	0	$2.1625E-02$	$2.1245E-02$	0	$2.4776E-04$	$1.4736E-03$	$2.8524E-02$
Rank	**1**	5	4	**1**	2	3	6
Std	0	$4.5019E-04$	$6.3144E-04$	0	$1.8266E-06$	$1.2846E-05$	$7.6640E-04$
Rank	**1**	4	5	**1**	2	3	6
f_3							
Mean	0	$2.5404E+01$	$8.2648E-01$	$1.3217E+01$	$2.6714E+01$	$2.8156E+01$	$2.5003E+00$
Rank	**1**	5	2	4	6	7	3
Std	0	$5.9031E+02$	$2.3449E+00$	$2.1480E+02$	$2.0025E+02$	$2.3132E+02$	$1.5978E+01$
Rank	**1**	7	2	5	4	6	3
f_4							
Mean	0	$3.4757E+01$	$3.6007E-13$	0	$5.8502E+01$	$7.3836E+01$	$1.6583E-01$
Rank	**1**	4	2	**1**	5	6	3
Std	0	$1.0636E+02$	$1.5035E-24$	0	$1.9185E+02$	$3.7055E+02$	$2.1051E-01$
Rank	**1**	4	2	**1**	5	6	3
f_5							
Mean	0	$2.0956E+01$	$5.3717E-13$	$1.3333E-01$	$6.1883E+01$	$7.0347E+01$	$1.1842E-16$
Rank	**1**	5	3	4	6	7	2
Std	0	$1.8327E+02$	$5.9437E-24$	$1.1954E-01$	$1.4013E+02$	$2.9600E+02$	$4.2073E-31$
Rank	**1**	6	3	4	5	7	2
f_6							
Mean	0	$1.4921E-14$	$1.6091E-07$	$9.2371E-15$	$1.3856E-14$	$2.1792E-09$	$6.9870E-15$
Rank	**1**	5	7	3	4	6	2
Std	0	$1.8628E-29$	$7.8608E-14$	$6.6156E-30$	$2.3227E-29$	$1.7187E-18$	$4.2073E-31$
Rank	**1**	4	7	3	5	6	2
f_7							
Mean	0	$1.4582E+00$	$1.8889E+03$	$1.9217E+02$	$9.4634E+00$	$1.7315E+02$	$1.9570E-10$
Rank	**1**	3	7	6	4	5	2
Std	0	$1.1783E+00$	$9.9106E+06$	$3.8433E+03$	$2.5976E+01$	$9.1577E+03$	$1.2012E-19$
Rank	**1**	3	7	5	4	6	2
f_8							
Mean	0	$1.2375E-02$	$1.0764E-02$	$4.0642E-03$	$3.3047E-03$	$4.1690E-03$	$5.5241E-03$
Rank	**1**	7	6	3	2	4	5
Std	0	$2.3107E-05$	$2.7698E-05$	$9.6184E-07$	$8.6680E-07$	$2.4012E-06$	$1.5358E-05$
Rank	**1**	6	7	3	2	4	5
f_{10}							
Mean	0	$3.4621E-26$	$5.4282E-14$	$9.9748E-39$	$2.6033E+02$	$5.1953E+04$	$1.8317E-137$
Rank	**1**	4	5	3	6	7	2
Std	0	$4.0873E-51$	$8.2868E-27$	$3.7661E-84$	$2.1785E+04$	$1.1136E+09$	$3.4534E-273$
Rank	**1**	4	5	3	6	7	2
f_{11}							
Mean	$-1.2542E+04$	$-1.0995E+04$	$-1.2127E+04$	$-1.2546E+04$	$-1.1052E+04$	$-1.1221E+04$	$-1.2569E+04$
Rank	3	7	5	2	6	4	**1**
Std	$1.4900E+02$	$1.3753E+05$	$3.3795E+04$	$4.2567E+03$	$9.4421E+05$	$2.7708E+05$	$1.1409E-25$
Rank	2	5	4	3	7	6	**1**

Table 9: Continued.

$f(x)$	PS-FW	stdPSO	CPSO	CLPSO	FIPS	Frankenstein	AIWPSO
f_{12}							
Mean	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Std	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
f_{13}							
Mean	$1.4998E-32$	$1.1422E-29$	$2.0913E-15$	$1.4998E-32$	$1.0273E-28$	$5.5136E-18$	$1.4998E-32$
Rank	**1**	2	5	**1**	3	4	**1**
Std	0	$3.2335E-57$	$1.2954E-29$	$1.2398E-94$	$1.0052E-56$	$1.4501E-34$	$1.2398E-94$
Rank	**1**	3	6	2	4	5	2

Table 10: The results of Friedman test for the PS-FW and other PSO variants over the mean and standard deviation of best solutions based on Table 9 (the best ranks are marked in bold).

	Mean	Std
Results		
N	12	12
Chi-square	35.33	37.18
p value	$3.72E-06$	$1.62E-06$
Friedman ranks of Algorithms		
PS-FW	**1.58**	**1.5**
stdPso	4.83	4.67
CPSO	5.08	5.17
CLPSO	3.17	3.25
FIPS	4.75	4.67
Frankenstein	5.58	5.75
AIWPSO	3	3

Table 11: The statistical results of PS-FW in terms of success rate and average number of iterations in successful runs for 12 benchmark functions.

Functions	ST	AT
f_1	30	3828.0
f_2	30	882.6
f_3	30	11266.5
f_4	30	1853.8
f_5	30	2134.7
f_6	30	755.1
f_7	30	5910.4
f_8	30	2281.1
f_{10}	30	6304.7
f_{11}	29	1100.5
f_{12}	30	7516.0
f_{13}	0	U

Table 12: The detailed parameters settings of the different optimization strategies for PS-FW (the square brackets represent the rounding operations).

Strategies	$\lambda_{\max}$	$\lambda_{\min}$	num_M
Strategy-1	1	$1E-25$	30
Strategy-2	1	$1E-10$	30
Strategy-3	1	0.1	30
Strategy-4	0.8	$1E-25$	30
Strategy-5	0.6	$1E-25$	30
Strategy-6	1	$1E-25$	$[0.5 \cdot \text{num}_E]$
Strategy-7	1	$1E-25$	$[0.7 \cdot \text{num}_E]$

performance in other functions under the whole seven strategies. Therefore, the robustness of the proposed algorithm is strongly proved. To compare the convergence speeds for different strategies of PS-FW, the convergence curves over several functions are shown in Figure 6. By observing the curves in Figure 6, the superiority of Strategy-6 and Strategy-7 in terms of convergence speed has been demonstrated, and the PS-FW with all strategies can converge to solutions that are very close to the optimums. Then we conduct the Friedman test and the Bonferroni-Dunn test for the mean and standard deviation of best solutions obtained by different optimization strategies, so as to determine the impact degree of each control parameter on the performance of PS-FW. The results of Friedman test for different strategies of PS-FW are shown in Table 15, and the results of Bonferroni-Dunn test in terms of mean and standard deviation based on Table 15 are presented in Figures 7 and 8.

According to the results of Friedman test in Table 15, the p value is lower than the level of significance considered $\alpha = 0.05$ for both the mean and standard deviation of bets solutions, which indicates that the performance of seven strategies of PS-FW has the significant difference. By observing the ranks obtained by the Friedman test in Table 15, the PS-FW with Strategy-7 has the best performance followed by Strategy-6, Strategy-1, and so on, and the PS-FW with Strategy-2 performs the worst relative to other strategies over the average values of best solutions. In Bonferroni-Dunn test, the values of critical difference are the same as those in Section 4.2, and the lines of best rank and significant level are also drawn in Figures 7 and 8. Through checking the bars corresponding to the different strategies of PS-FW in Figure 7(a), the heights of bars for Strategy-1 to Strategy-5 exceed the lines of significant level. Hence Strategy-7 represents the best combination of control parameters among all the seven strategies,

Table 13: The mean, standard deviation, and corresponding ranks of best solutions obtained by different optimization strategies of PS-FW for functions f_1 to f_{13} (the best ranks are marked in bold).

$f(x)$	Strategy-1	Strategy-2	Strategy-3	Strategy-4	Strategy-5	Strategy-6	Strategy-7
f_1							
Mean	$9.7833E-245$	$6.6617E-217$	$8.1065E-224$	$1.4930E-224$	$6.8133E-231$	0	0
Rank	2	6	5	4	3	**1**	**1**
Std	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
f_2							
Mean	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Std	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
f_3							
Mean	$1.0341E-26$	$7.1483E-16$	$2.5737E-13$	$1.3156E-09$	$2.2836E-09$	0	0
Rank	2	3	4	5	6	**1**	**1**
Std	$3.8500E-26$	$1.3157E-15$	$7.1641E-13$	$4.2629E-09$	$4.5987E-09$	0	0
Rank	2	3	4	5	6	**1**	**1**
f_4							
Mean	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Std	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
f_5							
Mean	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Std	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
f_6							
Mean	$7.1054E-16$	$2.3093E-15$	$1.4211E-15$	$2.3093E-15$	$2.4869E-15$	0	0
Rank	2	4	3	4	5	**1**	**1**
Std	$1.4211E-15$	$1.6945E-15$	$1.7405E-15$	$1.6945E-15$	$1.6281E-15$	0	0
Rank	2	4	5	4	3	**1**	**1**
f_7							
Mean	$2.1860E-71$	$7.0151E-123$	$3.5034E-126$	$2.7732E-62$	$2.0900E-65$	$5.7053E-83$	$2.3724E-87$
Rank	5	2	**1**	7	6	4	3
Std	$4.7535E-71$	$1.8052E-122$	$1.2502E-125$	$1.2084E-61$	$9.0599E-65$	$5.7716E-83$	$9.9762E-87$
Rank	5	2	**1**	7	6	4	3
f_8							
Mean	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Std	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
f_9							
Mean	$1.1555E-90$	$2.5372E-78$	$1.6308E-76$	$2.6199E-86$	$1.4655E-89$	$1.3155E-117$	$6.1364E-130$
Rank	3	6	7	5	4	2	**1**
Std	$2.7315E-90$	$1.1059E-77$	$4.7755E-76$	$7.7290E-86$	$6.2719E-89$	$5.7340E-117$	$2.6737E-129$
Rank	3	6	7	5	4	2	**1**
f_{10}							
Mean	$2.2792E-128$	$5.5926E-118$	$9.1955E-124$	$3.0530E-130$	$2.8788E-130$	$6.7603E-161$	$1.6779E-167$
Rank	5	7	6	4	3	2	**1**
Std	$9.7764E-128$	$2.4326E-117$	$3.4455E-123$	$9.2801E-130$	$1.1346E-129$	$2.9329E-160$	0
Rank	5	7	6	3	4	2	**1**

Table 13: Continued.

$f(x)$	Strategy-1	Strategy-2	Strategy-3	Strategy-4	Strategy-5	Strategy-6	Strategy-7
f_{11}							
Mean	$-4.1743E+04$	$-4.1279E+04$	$-4.1366E+04$	$-4.1366E+04$	$-4.1345E+04$	$-4.1757E+04$	$-4.1790E+04$
Rank	3	6	4	4	5	2	**1**
Std	$4.3502E+02$	$4.1356E+02$	$3.5331E+02$	$4.1470E+02$	$3.4657E+02$	$2.6837E+02$	$1.4566E+02$
Rank	7	5	4	6	3	2	**1**
f_{12}							
Mean	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Std	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
f_{13}							
Mean	$1.4998E-32$	$1.4998E-32$	$1.4998E-32$	$1.4998E-32$	$1.4998E-32$	$1.4998E-32$	$1.4998E-32$
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Std	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**

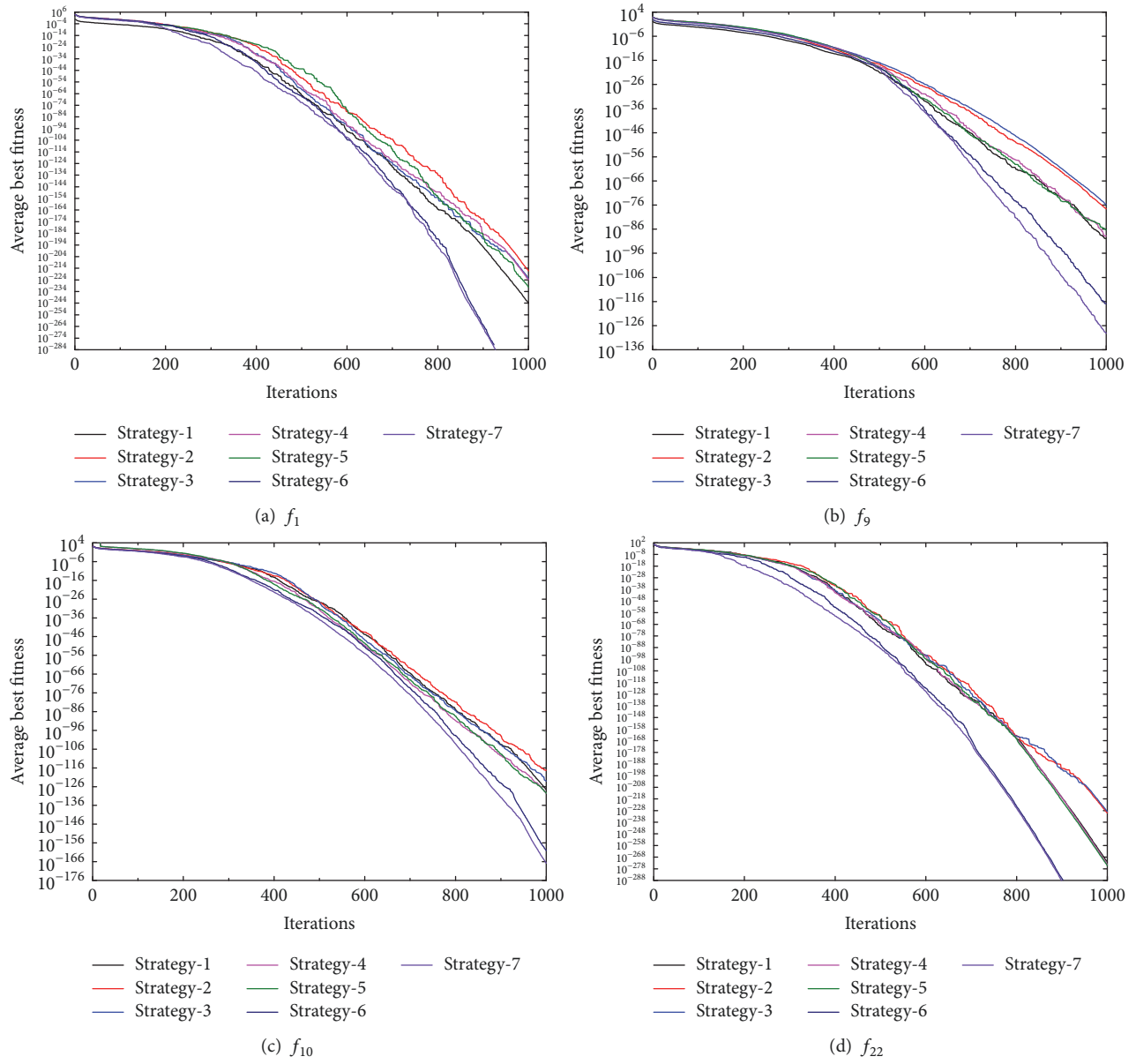
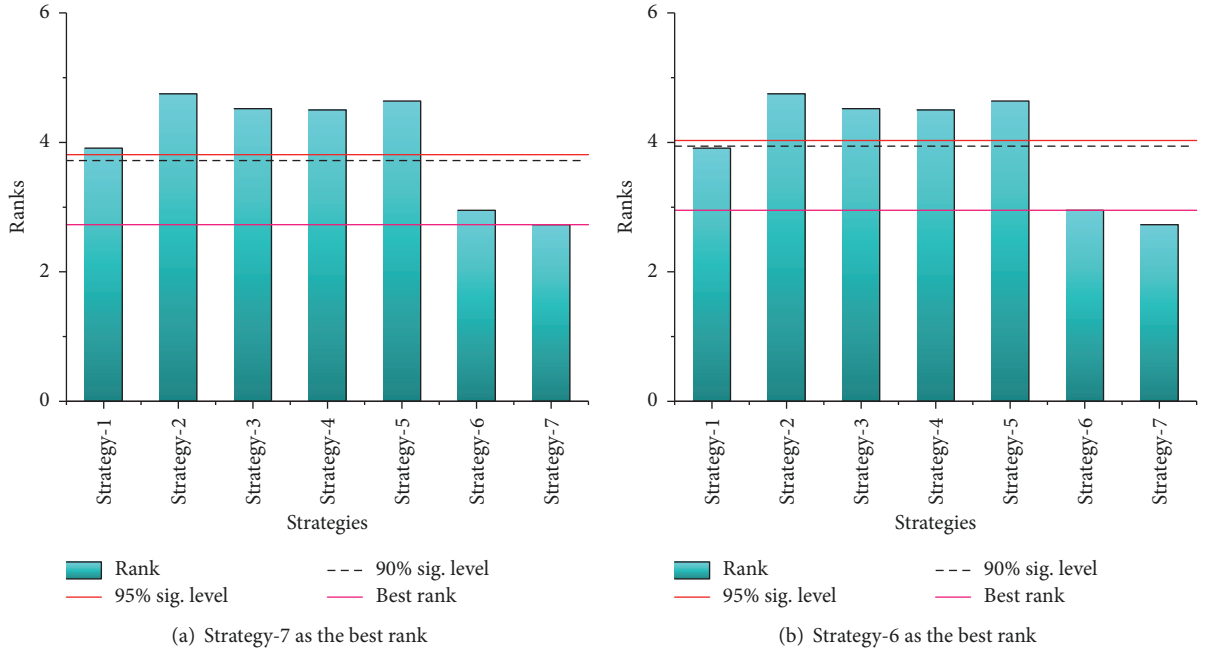


Figure 6: Convergence curves of PS-FW with different strategies for functions f_1, f_9, f_{10}, and f_{22}.

Table 14: The mean, standard deviation, and corresponding ranks of best solutions obtained by different optimization strategies of PS-FW for functions f_{14} to f_{22} (the best ranks are marked in bold).

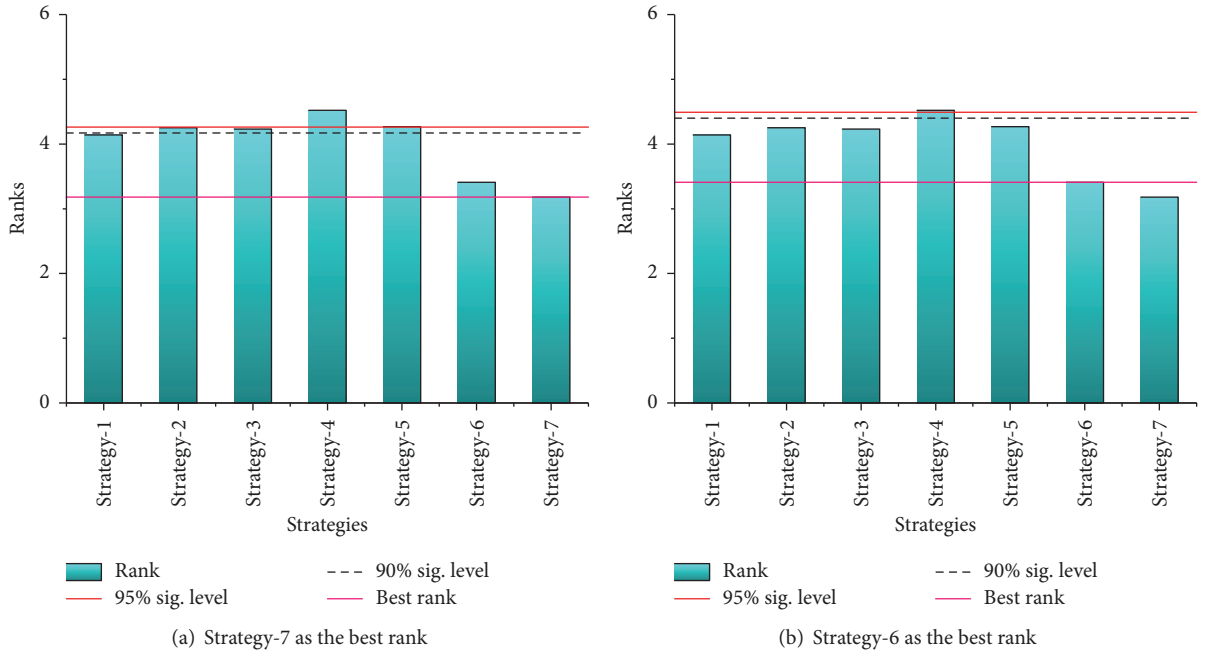
$f(x)$	Strategy-1	Strategy-2	Strategy-3	Strategy-4	Strategy-5	Strategy-6	Strategy-7
f_{14}							
Mean	$6.4751E-275$	$4.6790E-268$	$5.0050E-272$	$1.2035E-283$	$9.7967E-265$	0	0
Rank	3	5	4	2	6	**1**	**1**
Std	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
f_{15}							
Mean	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Std	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
f_{16}							
Mean	$2.4731E-93$	$2.5574E-102$	$1.0668E-102$	$9.2122E-91$	$7.8026E-91$	$2.5290E-114$	$1.7103E-116$
Rank	5	4	3	7	6	2	**1**
Std	$8.4009E-93$	$1.0215E-101$	$3.2290E-102$	$3.7019E-90$	$3.0225E-90$	$4.6404E-114$	$6.2900E-116$
Rank	5	4	3	7	6	2	**1**
f_{17}							
Mean	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Std	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
f_{18}							
Mean	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Std	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
f_{19}							
Mean	$9.0096E-250$	$2.3878E-201$	$1.5857E-189$	$5.9464E-249$	$1.5925E-244$	0	0
Rank	2	5	6	3	4	**1**	**1**
Std	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
f_{20}							
Mean	1	1	1	1	1	1	1
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Std	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
f_{21}							
Mean	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Std	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**
f_{22}							
Mean	$4.9253E-273$	$8.5544E-231$	$1.4963E-229$	$3.8782E-275$	$4.3846E-276$	0	0
Rank	4	5	6	3	2	**1**	**1**
Std	0	0	0	0	0	0	0
Rank	**1**	**1**	**1**	**1**	**1**	**1**	**1**



(a) Strategy-7 as the best rank

(b) Strategy-6 as the best rank

Figure 7: The bar chart of Bonferroni-Dunn test for different strategies over the mean of best solutions based on Table 15.



(a) Strategy-7 as the best rank

(b) Strategy-6 as the best rank

Figure 8: The bar chart of Bonferroni-Dunn test for different strategies over the standard deviation of best solutions based on Table 15.

and the PS-FW with Strategy-7 performs significantly better than the other strategies except Strategy-6. In addition, the PS-FW with Strategy-6 has significant superiority compared with Strategy-2 to Strategy-5 over the average values of best solutions based on Figure 7(b). Besides, as shown in Figure 8, the hybrid algorithm with different strategies has relatively small gaps in standard deviation, Strategy-7 emerges as the best performer over the standard deviation of best solutions followed by Strategy-6, Strategy-1, and other strategies, and Strategy-4 has the worst performance.

Therefore, based on the analysis above, the solutions accuracy and convergence speed of PS-FW are determined by the control parameters $\lambda_{\min}$, $\lambda_{\max}$, and num_M. Compared with $\lambda_{\min}$ and $\lambda_{\max}$, the number of mutation sparks has a greater impact on the performance of PS-FW. Hence we can appropriately increase the number of mutation sparks

Table 15: The results of Friedman test for the different strategies of PS-FW over the mean and standard deviation of optimal solutions based on Tables 13 and 14 (the best ranks are marked in bold).

	Mean	Std
Results		
N	22	22
Chi-square	40.23	22.38
p value	$4.10E-07$	$1.03E-03$
Friedman ranks of algorithms		
Strategy-1	3.91	4.14
Strategy-2	4.75	4.25
Strategy-3	4.52	4.23
Strategy-4	4.5	4.52
Strategy-5	4.64	4.27
Strategy-6	2.95	3.41
Strategy-7	**2.73**	**3.18**

when solving the difficult multimodal global optimization problems. In addition, the value of $\lambda_{\min}$ can be increased properly for solving the optimization problems with large range such as function f_7. Considering that the increase in the number of mutation sparks will make the computing time longer, to improve the computational efficiency, Strategy-1 which ranks third in seven strategies is used to conduct the experiments in Sections 4.2 and 4.3 in this paper. As expected, we should choose the suitable control parameters for various problems by taking all the aspects into consideration.

5. Conclusion

In this paper, a hybrid algorithm named PS-FW is proposed to solve the global optimization problems. In PS-FW, the exploitation capability is applied to find the optimal solution and make the hybrid algorithm converge quickly whereas the exploration ability of FWA is used to search for the better solutions in the entire feasible space. Moreover, the abandonment and supplement mechanism, the modified explosion operator, and the novel mutation operator are proposed to enhance both the global and local search ability of algorithm. Then the validity of PS-FW is confirmed by the 22 well-known high-dimensional benchmark functions. The results show that PS-FW is an efficacious, fast converging, and robust optimization algorithm by comparing with the PSO, FWA, stdPSO, CPSO, CLPSO, FIPS, Frankenstein, and ALWPSO over solving global optimization problems.

The future work is to refine the PS-FW by testing more complex high-dimensional optimization problems. Furthermore, we will try to apply the algorithm to multiobjective optimization problems and real-world problems such as spatial layout optimization, route optimization, and structural parameter optimization.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

This study was funded by National Natural Science Foundation of China (nos. 51674086 and 51534004) and Northeast Petroleum University Innovation Foundation for Postgraduate (no. YJSCX2015-012NEPU).

References

[1] Y. Tan, *Firework Algorithm: A Novel Swarm Intelligence Optimization Method*, Springer, Berlin, Heidelberg, Germany, 2015.

[2] N. Islam, S. Rana, R. Ahsan, and S. Ghani, "An Optimized Design of Network Arch Bridge using Global Optimization Algorithm," *Advances in Structural Engineering*, vol. 17, no. 2, pp. 197–210, 2014.

[3] E. Vinot, V. Reinbold, and R. Trigui, "Global Optimized Design of an Electric Variable Transmission for HEVs," *IEEE Transactions on Vehicular Technology*, vol. 65, no. 8, pp. 6794–6798, 2016.

[4] N. Gabere, *Simulated Annealing Driven Pattern Search Algorithms for Global Optimization*, University of the Witwatersrand, Johannesburg, South Africa, 2007.

[5] R. Storn and K. Price, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[6] P. Kaelo and M. M. Ali, "Integrated crossover rules in real coded genetic algorithms," *European Journal of Operational Research*, vol. 176, no. 1, pp. 60–76, 2007.

[7] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks (ICNN '95)*, vol. 4, pp. 1942–1948, Perth, Western Australia, November-December 1995.

[8] M. Dorigo, M. Birattari, and T. Stützle, "Ant colony optimization," *IEEE Computational Intelligence Magazine*, vol. 1, no. 4, pp. 28–39, 2006.

[9] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Tech. Rep., Erciyes University, Kayseri, Turkey, 2005.

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

[11] J. Wang, B. Lin, and J. Jin, "Optimizing the shunting schedule of electric multiple units depot using an enhanced particle swarm optimization algorithm," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 5804626, 2016.

[12] X. Wu, C. Li, W. Jia, and Y. He, "Optimal operation of trunk natural gas pipelines via an inertia-adaptive particle swarm optimization algorithm," *Journal of Natural Gas Science and Engineering*, vol. 21, pp. 10–18, 2014.

[13] X. Hua, X. Hu, and W. Yuan, "Research optimization on logistics distribution center location based on adaptive particle swarm algorithm," *Optik - International Journal for Light and Electron Optics*, vol. 127, no. 20, pp. 8443–8450, 2016.

[14] B. A. Garroa and R. A. Vázquez, "Designing artificial neural networks using particle swarm optimization algorithms," *Computational Intelligence and Neuroscience*, vol. 2015, Article ID 369298, 20 pages, 2015.

[15] S. Ye, H. Ma, S. Xu, W. Yang, and M. Fei, "An effective fireworks algorithm for warehouse-scheduling problem," *Transactions of the Institute of Measurement and Control*, vol. 39, no. 1, pp. 75–85, 2017.

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

[17] A. Mohamed Imran, M. Kowsalya, and D. P. Kothari, "A novel integration technique for optimal network reconfiguration and distributed generation placement in power distribution networks," *International Journal of Electrical Power & Energy Systems*, vol. 63, pp. 461–472, 2014.

[18] J. Li and Y. Tan, "Loser-out tournament based fireworks algorithm for multi-modal function optimization," *IEEE Transactions on Evolutionary Computation*, 2017.

[19] Z. Li, W. Wang, Y. Yan, and Z. Li, "PS-ABC: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems," *Expert Systems with Applications*, vol. 42, no. 22, pp. 8881–8895, 2015.

[20] Y.-J. Zheng, X.-L. Xu, H.-F. Ling, and S.-Y. Chen, "A hybrid fireworks optimization method with differential evolution operators," *Neurocomputing*, vol. 148, pp. 75–82, 2015.

[21] S. Zheng, J. Li, A. Janecek, and Y. Tan, "A cooperative framework for fireworks algorithm," *IEEE Transactions on Computational Biology and Bioinformatics*, vol. 14, no. 1, pp. 27–41, 2017.

[22] A. Nickabadi, M. M. Ebadzadeh, and R. Safabakhsh, "A novel particle swarm optimization algorithm with adaptive inertia weight," *Applied Soft Computing*, vol. 11, no. 4, pp. 3658–3670, 2011.

[23] L. Li, F. Liu, G. Long, P. Guo, and X. Bie, "Modified particle swarm optimization for BMDS interceptor resource planning," *Applied Intelligence*, vol. 44, no. 3, pp. 471–488, 2016.

[24] C.-F. Wang and K. Liu, "A novel particle swarm optimization algorithm for global optimization," *Computational Intelligence and Neuroscience*, vol. 2016, Article ID 9482073, pp. 1–9, 2016.

[25] D. Souravlias and K. E. Parsopoulos, "Particle swarm optimization with neighborhood-based budget allocation," *International Journal of Machine Learning and Cybernetics*, vol. 7, no. 3, pp. 451–477, 2016.

[26] J.-J. Xue, Y. Wang, H. Li, X.-F. Meng, and J.-Y. Xiao, "Advanced fireworks algorithm and its application research in PID parameters tuning," *Mathematical Problems in Engineering*, vol. 2016, Article ID 2534632, pp. 1–9, 2016.

[27] J. Liu, S. Zheng, and Y. Tan, "The improvement on controlling exploration and exploitation of firework algorithm," in *Proceedings of the International Conference in Swarm Intelligence*, pp. 11–23, Springer, Berlin, Heidelberg, Germany, 2013.

[28] Y. Pei, S. Zheng, Y. Tan, and H. Takagi, "Effectiveness of approximation strategy in surrogate-assisted fireworks algorithm," *International Journal of Machine Learning and Cybernetics*, vol. 6, no. 5, pp. 795–810, 2015.

[29] S. Zheng, A. Janecek, and Y. Tan, "Enhanced fireworks algorithm," in *Proceedings of the IEEE Congress on Evolutionary Computation*, vol. 62, pp. 2069–2077, Cancun, Mexico, June 2013.

[30] S. Zheng, C. Yu, J. Li, and Y. Tan, "Exponentially decreased dimension number strategy based dynamic search fireworks algorithm for solving CEC2015 competition problems," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '15)*, pp. 1–8, Sendai, Japan, 2015.

[31] S. Zheng, A. Janecek, J. Li, and Y. Tan, "Dynamic search in fireworks algorithm," in *Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC '14)*, pp. 3222–3229, China, July 2014.

[32] J. Li, S. Zheng, and Y. Tan, "The Effect of Information Utilization: Introducing a Novel Guiding Spark in the Fireworks Algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 1, pp. 153–166, 2017.

[33] J. Li, S. Zheng, and Y. Tan, "Adaptive fireworks algorithm," in *Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC '14)*, pp. 3214–3221, Springer, Berlin, Heidelberg, China, July 2014.

[34] J. Li and Y. Tan, "The bare bones fireworks algorithm: A minimalist global optimizer," *Applied Soft Computing*, vol. 62, pp. 454–462, 2018.

[35] F. Valdez, P. Melin, and O. Castillo, "Modular Neural Networks architecture optimization with a new nature inspired method using a fuzzy combination of Particle Swarm Optimization and Genetic Algorithms," *Information Sciences*, vol. 270, pp. 143–153, 2014.

[36] M. Pandit, V. Chaudhary, H. M. Dubey, and B. K. Panigrahi, "Multi-period wind integrated optimal dispatch using series PSO-DE with time-varying Gaussian membership function based fuzzy selection," *International Journal of Electrical Power & Energy Systems*, vol. 73, pp. 259–272, 2015.

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

[38] B. Zhang, M.-X. Zhang, and Y.-J. Zheng, "A hybrid biogeography-based optimization and fireworks algorithm," in *Proceedings of the 2014 IEEE Congress on Evolutionary Computation (CEC '14)*, pp. 3200–3206, Beijing, China, July 2014.

[39] M. J. Amoshahy, M. Shamsi, and M. H. Sedaaghi, "A novel flexible inertia weight particle swarm optimization algorithm," *PLoS ONE*, vol. 11, no. 8, Article ID e0161558, pp. 1–42, 2016.

[40] M. Friedman, "A comparison of alternative tests of significance for the problem of m rankings," *The Annals of Mathematical Statistics*, vol. 11, no. 1, pp. 86–92, 1940.

[41] O. J. Dunn, "Multiple comparisons among means," *Journal of the American Statistical Association*, vol. 56, pp. 52–64, 1961.

[42] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.