Hindawi
Scientific Programming
Volume 2021, Article ID 6676449, 14 pages
https://doi.org/10.1155/2021/6676449

Research Article

Adaptive Particle Swarm Optimization with Gaussian Perturbation and Mutation

Binbin Chen,[1] **Rui Zhang**,[2] **Long Chen**,[2] **and Shengjie Long**[3]

[1]*Graduate School, Xi'an International Studies University, Xi'an 710128, China*
[2]*School of Information Engineering, Zunyi Normal College, Zunyi 563002, China*
[3]*School of Traffic & Transportation Engineering, Central South University, Changsha 410004, China*

Correspondence should be addressed to Shengjie Long; longshengjie12@csu.edu.cn

Received 10 October 2020; Revised 18 January 2021; Accepted 23 January 2021; Published 4 February 2021

Academic Editor: Wei Li

The particle swarm optimization (PSO) is a wide used optimization algorithm, which yet suffers from trapping in local optimum and the premature convergence. Many studies have proposed the improvements to address the drawbacks above. Most of them have implemented a single strategy for one problem or a fixed neighborhood structure during the whole search process. To further improve the PSO performance, we introduced a simple but effective method, named adaptive particle swarm optimization with Gaussian perturbation and mutation (AGMPSO), consisting of three strategies. Gaussian perturbation and mutation are incorporated to promote the exploration and exploitation capability, while the adaptive strategy is introduced to ensure dynamic implement of the former two strategies, which guarantee the balance of the searching ability and accuracy. Comparison experiments of proposed AGMPSO and existing PSO variants in solving 29 benchmark functions of CEC 2017 test suites suggest that, despite the simplicity in architecture, the proposed AGMPSO obtains a high convergence accuracy and significant robustness which are proven by conducted Wilcoxon's rank sum test.

1. Introduction

Particle swarm optimization (PSO) is an evolutionary computing technique proposed by Kennedy and Eberhart in 1995 [1], originating from the simulation of predation and other behaviors of bird flocks and fish schools. The solution of each optimization problem in the algorithm is similar to a "particle" in the search space. The particle swarm algorithm randomly generates an initial swarm and gives each particle a random velocity. During the optimization process, the particles adjust the velocity and trajectory according to the experience of themselves and companions, so that the whole swam contains the ability to fly to a better search area. Involving few parameters and with easy implementation, PSO has been widely used in many fields such as function optimization, neural network training, fuzzy system control, pattern recognition, and engineering application. However, the PSO algorithm still has problems such as premature and easily falling into local optimum when tackling complex multimodal problems. In order to improve the solving ability of particle swarm optimization, researchers have proposed methods, such as an adjustment of the inertial parameters of particle swarm algorithm, including dynamic policies and adaptive methods, learning factors, and social factors [2], a neighborhood searching strategy to strengthen the exploration of the neighborhood of the current population [3], an adoption of the information-sharing mechanism to enhance population diversity and avoid premature algorithm convergence [4], and the integrations with other algorithms, such as the combination of particle swarm optimization algorithm and immune algorithm, genetic algorithm, and artificial bee colony algorithm [5].

A variety of improved methods are proposed to solve the existing problems of PSO. The inertia weight and velocity parameters were dynamically adjusted through the particle swarm's convergence state to speed up the convergence speed and balance the global search and local search capabilities [6]. Alatas et al. [7] proposed a strategy to learn from outstanding individuals other than the optimal particles to adapt to the solution of high-dimensional problems. Zhao et al. [8] introduced biology principles to give the particles the ability of multi-crossover and swarm colonization behaviors. Liang et al. [9] implemented a neighborhood development strategy to improve the algorithm's search ability. Chen et al. [10] leveraged the optimal information for all other particles to update the velocity of the particles in different dimensions. A particle swarm algorithm with lifecycle and challenging behavior is proposed to preserve particles' activity during the evolution of particle swarm, which is beneficial to the global range search [11]. Frans and Engelbrecht [12] incorporated chaos into the particle movement process, so that the particle swarm alternately moves between chaos and stability, gradually approaching the best point. Tian [13] initialized the particle swarm in a chaotic manner to ensure that the particle swarm can be evenly distributed in the solution space and achieve better global search capabilities. Du et al. [14] and Munlin and Anantathanavit [15] proposed to use multiple methods to realize the particles' evolution to improve the search capability of the algorithm. Kiran [16] improved the efficiency of the algorithm evolution with a new evolutionary mechanism. It improves the calculating speed of the algorithm and particularly has an advantage in solving multimodal problems.

These methods improve the performance of PSO to certain extent, but there are still many flaws, such as high complexity in architecture and low convergence speed. The solutions, such as adaptive, perturbation, and mutation, were incorporated to address these problems.

Aiming to attain the prominent performance of PSO, the adaptive strategy was introduced to dynamically update the parameters of the algorithm in prior studies, i.e., inertia weight [17–21], velocity and position [22, 23], and ω, c_1, and c_2 of each particle [5, 24]. Wang et al. [25] and Li and Cheng et al. [26] introduced a mixed adaptive strategy to adjust the parameters in order to balance the search and convergence capabilities. Beyond adaptive updates the parameters, more complex adaptive strategies are proposed. The particles are randomized based on the detection of the changes of *gbest* value [27]. In order to adaptively maintain the social attribution of swarm, the inactive particles are taken off based on the diversity of fitness between current particle and the best historical experience [28].

In order to promote particles to jump out of local optimum further improving the global searching ability, multiple perturbation strategies were introduced. A chaotic perturbation was incorporated into the PSO algorithm, which improved particles' diversity [29]. Mahmoodabadi et al. [30] utilized Cauchy perturbation and reverse learning to accelerate the particle swarm's convergence and escape the local optimal solution. Wang et al. [31] proposed nonuniform mutation and multistage perturbation of particles, which perturbs the optimal solution at different stages of evolution, thereby increasing group diversity and increasing the probability of jumping out of local extreme points.

For increasing vitality and diversity of particles, mutation strategy has been implemented in many optimizations of PSO. Pehlivanoglu [32] applied a mutation strategy into global random diversity and local controlled diversity. The undesired particles are replaced following the mutation strategy to accelerate the convergence speed [33]. Large-scale mutation and small-scale mutation are conducted to prevent the premature convergence while guaranteeing the convergence speed [34].

By contrast with the prior studies, we proposed an adaptive principle according to which the perturbation and mutation are conducted to balance the convergence accuracy and rate. Our main contributions are summarized as follows:

(1) An adaptive adjusting rule is incorporated following the cosine law, in order that the particles are interfered with larger amplitude to improve the particle's global search ability in the early stage and with a smaller amplitude to improve the convergence accuracy.

(2) Following the adaptive strategy, the Gaussian perturbation is incorporated to pump the optimal particle to jump out of the local optimum.

(3) Identically, according to the adaptive strategy, the mutation is implemented to improve the diversity of particles that have stagnated evolution and to balance the ratio of inheritance and mutation to ensure the population's searching ability.

2. Adaptive Mutation Particle Swarm Optimization Algorithm with Gaussian Perturbation

2.1. Basic Particle Swarm Optimization. PSO first initializes a swarm. A particle in the swarm represents a solution of each search space, and each particle has two parameters: position and velocity. Assuming that the size of the current swarm $P(t)$ is N, the position, and the velocity of the i-th particle in the swarm are expressed as $v_i(t)\{v_{i1}(t),\ldots,v_{i\,D}(t)\}$ and $X_i(t)\{X_{i1}(t),\ldots,X_{i\,D}(t)\}$, where D is the dimension of the problem and t is an evolutionary algebra. The particles are evaluated by a previously designed fitness function. The particle i updates its velocity and position through the swarm optimal *gbest* and the individual historical optimal $pbest_i$ in the iterative process. The equations for updating the velocity and position of a particle are as follows:

$$\begin{aligned} v_{id}(t+1) &= w\cdot v_{id}(t) + c_1\cdot r_1\cdot\left(pb\mathrm{set}_{id} - x_{id}(t)\right)\\ &\quad + c_2\cdot r_2\cdot\left(gb\mathrm{set} - x_{id}(t)\right),\\ x_{id}(t+1) &= x_{id}(t) + v_{id}(t+1), \end{aligned} \tag{1}$$

where w is the particle's inertia weight, which determines the degree of influence of the particle's previous velocity on the current velocity, c_1 is the particle self-cognition learning coefficient, c_2 is the social cognitive learning coefficient, and r_1 and r_2 are random numbers between 0 and 1.

2.2. Task Definition. As the number of iterations increases in the standard PSO algorithm, the particles will gradually approach the optimal solution, the evolutionary rate and swarm diversity will gradually decrease. Once the optimal particle falls into the local optimum, it hardly escapes. To address this problem, we incorporate the Gaussian perturbation and mutation strategy where the threshold stop_num is set to define whether the particles are in the evolutionary stagnation state or not. Since fitness of a particle ceased to evolve, record the continuous times as tag, namely, the times of i-th particle as tag(i) and the times of *gbest* as tag(g). If tag$(g) \geq$ stop_num, it means that the evolution of the population has stagnated and the Gaussian perturbation is applied to pump the population to jump out of the local optimum. If tag$(i) \geq$ stop_num, it means that the evolution of this particle has stagnated, and the mutation is conducted to update the particle. Either perturbation or mutation is conducted following the proposed adaptive strategy, guaranteeing the balance of searching ability and accuracy. The process of AGMPSO is shown in Figure 1.

2.2.1. Adaptive Strategy. In many previous studies, the amplitude of interference in PSO parameters remains the same during the iterations, which is not beneficial for the convergence in the later period. Aiming to reach the ideal state of PSO, a larger amplitude of interference is required at the early iterative stage to ensure a better global searching ability, and a smaller one in the late iterative stage to guarantee the convergence. Hence, a dynamically adaptive strategy is necessary. In this study, we introduce the probability Pc adaptively altering following the cosine law as iteration increases, and the equation is as follows:

$$Pc(t) = c_3 \cdot \left(1 + \cos\left(\frac{t \cdot \pi}{\text{Max\_Gen}}\right)\right), \tag{2}$$

where t is current evolutionary iteration, $Pc(t)$ (as shown in Figure 2) is the probability of application of Gaussian perturbation or mutation of t iteration, c_3 is the adaptive coefficient, and Max_Gen is the maximum number of iterations.

2.2.2. Gaussian Perturbation Strategy. Gaussian perturbation is adopted in the particle *gbest*, which is in the evolutionary stagnation state, to improve the ability to jump out of the local optimum. In order to ensure that each dimension of *gbest* has a possibility of escaping the local optimum, each dimension $gbest_d$ of *gbest* is updated by Gaussian perturbation with probability $Pc(t)$, which guides the perturbation to conduct adaptively so as to ensure the better ability of escaping from the local optimum at early stages and also the better convergence ability during the later stages.

In addition to adapting the algorithm stages, via adaptive variance δ, the Gaussian perturbation strategy is capable of adapting the proposed PSO algorithm to different functions according to whose value spaces (as shown in equation (4).

When the evolution of particle *gbest* is stagnant, that is, tag$(g) \geq$ stop_num, the perturbation is implemented as follows:

$$gbest_d = gbest_d \cdot r_3 \cdot \text{guass}_d, \tag{3}$$

$$\text{guass}_d = \text{Guassion}\left(0, \delta_d^2\right), \tag{4}$$

$$\delta_d = \frac{1}{5} \cdot \left(X_{\max} - X_{\min}\right), \tag{5}$$

where r_3 is a random number between 0 and 1, $gbest_d$ is the optimal particle of the d-th dimension, and δ_d is the adaptive variance of the d-th dimension.

2.2.3. Mutation Strategy. The mutation strategy is utilized to improve the particle diversity of the algorithm and balance the ratio of the mutation and the inheritance to ensure the convergence. Similar to the Gaussian perturbation strategy, the mutation strategy can adapt our algorithm to different functions as follows:

$$\begin{aligned} v_{id}(t+1) = w \cdot v_{id}(t) + c_1 \cdot r_1 \cdot \left(pb\text{set}_{id} - x_{id}(t)\right) \\ + c_4 \cdot r_4 \cdot \text{mut}_d, \end{aligned} \tag{6}$$

$$\text{mut}_d = X_{\min} + rand\left(X_{\max} - X_{\min}\right), \tag{7}$$

where c_4 is an adaptive mutation coefficient, r_4 is a random number between 0 and 1, and mut_d is the degree of adaptive mutation of d-th dimension.

When a particle falls into the evolutionary stagnation, the mutation operator is introduced into partial dimensions in the speed updated equation (6), which increases the diversity of the population getting rid of the constraints of *gbest* particles, especially improves the search ability of particles with low speed due to converging near *gbest*, and promotes the particle utilization. Since the dimension d of the particle also uses the probability Pc to mutate, the mutated range of the algorithm's particles is large in the initial period, which is favorable to the global search. In the later period, the mutated range and the inheritance ratio are small, which is beneficial for algorithm convergence.

2.2.4. Algorithm Complexity Analysis. The computational costs of the standard PSO include the initialization O(mn), fitness evaluation O(mn), and velocity and position update O$(2mn)$ (m and n are the swarm size and dimension, respectively). Thus, the time complexity of the PSO is O(mn). Compared with the standard PSO, AMGPSO involves two operators. However, the Gaussian perturbation operator O(n) or the mutation operator O(mn) need to be conducted separately only when the global best position is stagnant, or the personal best position is stagnant within several iterations. The worst-case time complexity of AGMPSO is O(mn), including the initialization O(mn), evaluation O$(mn + n)$, and update O$(2mn + mn)$. The conclusion can be drawn from the above component complexity analyses;

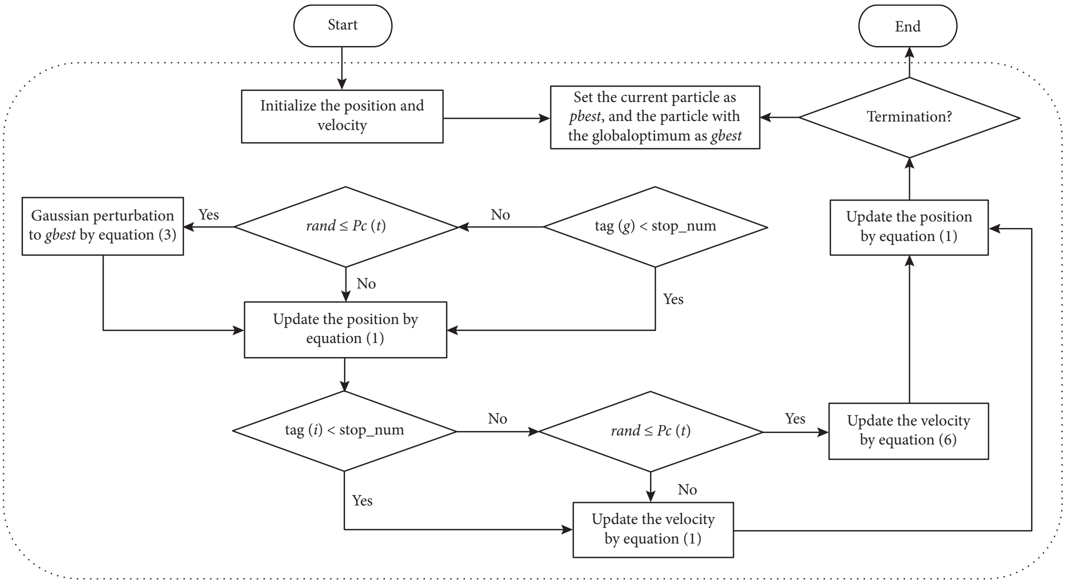

Start
End
Initialize the position and velocity
Set the current particle as *pbest*, and the particle with the globaloptimum as *gbest*
Termination?
Gaussian perturbation to *gbest* by equation (3)
Yes
rand ≤ *Pc* (*t*)
No
tag (*g*) < stop_num
Update the position by equation (1)
No
Yes
Update the position by equation (1)
tag (*i*) < stop_num
No
rand ≤ *Pc* (*t*)
Yes
Update the velocity by equation (6)
Yes
No
Update the velocity by equation (1)

Figure 1: Flowchart of AGMPSO.

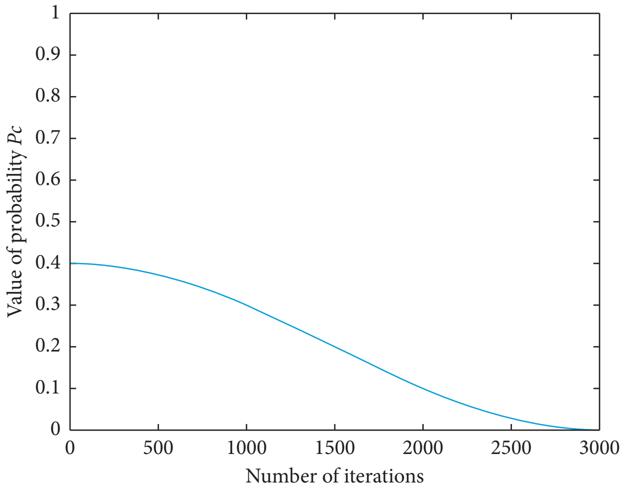


Figure 2: The curve of probability *Pc* during the whole iterations.

AGMPSO contains the same level of time complexity as the standard PSO algorithm.

3. Experiments and Discussions

3.1. Algorithm Aggregation Degree Analysis. The ideal status of PSO is that, in the early stage, the particles can explore the solution space more dispersedly, but in the later stage, they can better aggregate to obtain higher convergence accuracy. We introduce an aggregation degree to analyze the convergence status of the standard PSO and the AGMPSO. When the particle fitness deviation from the group average value is larger, and the particle aggregation degree θ is larger, the particle diversity is better, and the algorithm search ability is stronger. Aggregation degree θ of the t-th generation particle is expressed as follows:

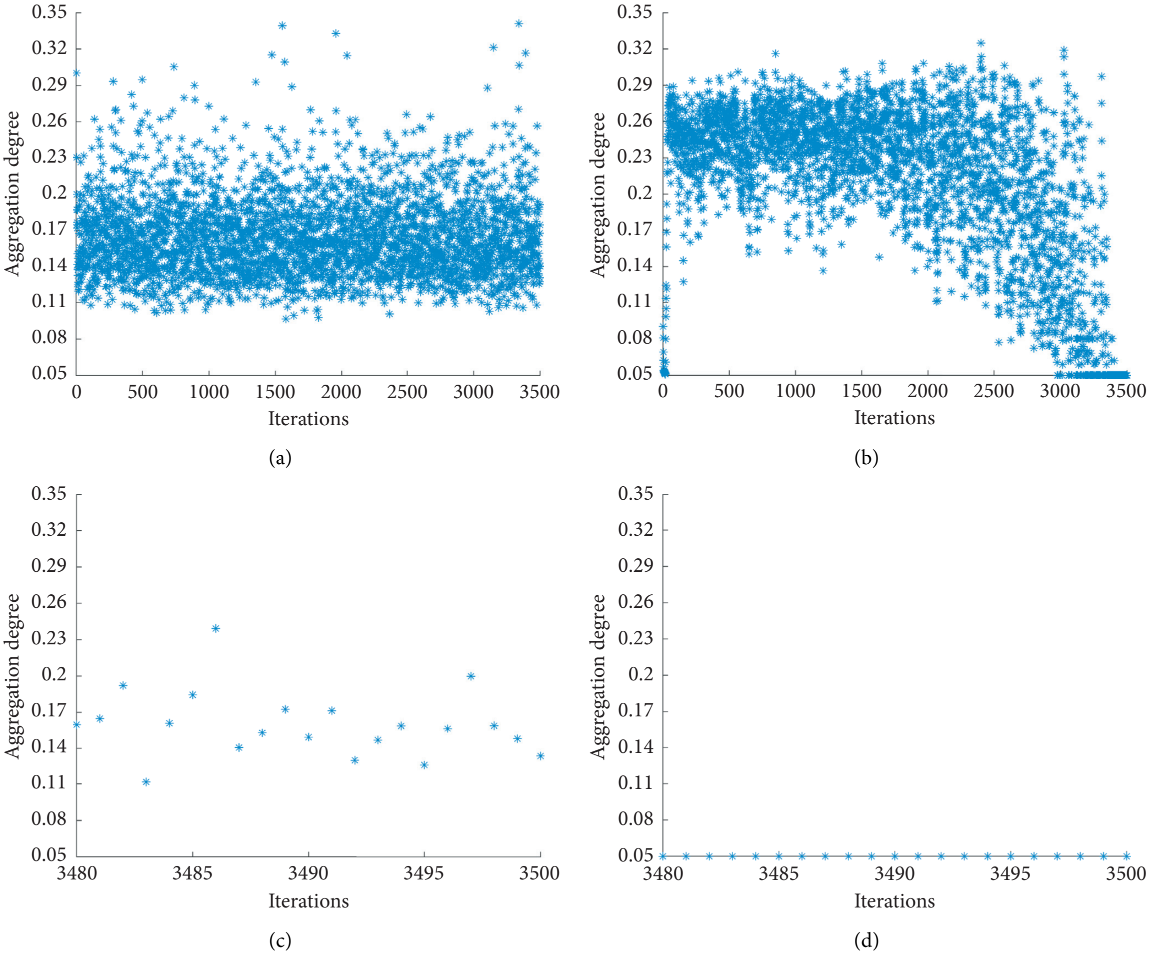
$$\theta = \frac{1}{N}\cdot\sum_{i=1}^{N}\left|\frac{f\left(x_i^t\right) - f_{\text{avg}}^t}{f_{\max}^t - f_{\min}^t}\right|, \tag{8}$$

where f_{avg}^t is the average fitness value of t-th generation particle, $f_{\max}^t$ is the maximum fitness value of t-th generation particle, and $f_{\min}^t$ is the minimum fitness value of t-th generation particle.

Figure 3 shows the comparison of aggregation degree between the standard PSO and AGMPSO while solving the Rastrigin function. (A) and (B) are the aggregation curves of standard PSO and AGMPSO in 3500 iterations, and (C) and (D) are enlarged screenshots of the last 20 iterations. It can be seen that the standard PSO holds the higher particle aggregation even in later iterations, which indicates the high diversity, that is, the poor convergence. It is worth noting that the aggregation degree remains a high level, which means the swarm did not converge to a satisfactory extent till the end. In contrary, AGMPSO can maintain higher diversity throughout the early period and lower diversity in the later period, which ensures both the global search ability and the convergence.

3.2. Comparison with PSO Variants. In order to evaluate the performance of the proposed algorithm, the comparison experiments of AGMPSO with PSO, TSLPSO, HFPSO, and MPEPSO are conducted in this section, and parameters of each algorithm are listed in Table 1. All experiments were performed under Windows 10 system, eight-core processor (Intel (R) Core (TM) i7-10700K CPU @ 3.80 GHz), 16G memory using MATLAB R2018a.

3.2.1. Benchmark Functions. CEC 2017 [38] test suites are introduced in experiments, including the 29 benchmark functions divided into four categories: unimodal functions, simple multimodal functions, hybrid functions, and



(a)

(b)

(c)

(d)

Figure 3: Aggregation degree of standard PSO and AGMPSO in solving Rastrigin function.

Table 1: The parameters of all comparison competitors.

Algorithm	Parameter settings
AGMPSO	$w = 0.729$, $c_1 = c_2 = c_4 = 1.49445$, $c_3 = 0.2$
TSLPSO [35]	$w = 0.9{\sim}0.4$, $c_1 = c_2 = 1.5$, $c_3 = 0.5{\sim}2.5$
HFPSO [36]	$w_i = 0.9$, $w_f = 0.5$, $c_1 = c_2 = 1.49445$, $\alpha = 0.2$, $B_0 = 2$, $\gamma = 1$
MPEPSO [37]	$w = 0.9{\sim}0.4$, $c_1 = c_2 = 0.5{\sim}2.5$, $\text{LP} = 10$, $\lambda_h = 0.1$
PSO [1]	$w = 0.9{\sim}0.4$, $c_1 = c_2 = 2.0$

composition functions. Based on the description of the definitions of CEC 2017 test suits, F2 has been excluded because of its unstable behavior especially for higher dimensions.

Two series of experiments are performed, the dimension of each test function of each series is 10 and 30, respectively, the population size is 30, and each algorithm runs each benchmark function for 30 runs independently. According to the definitions of CEC 2017 test suits, the stop condition of each run is that the maximum number of function evaluations (MaxFES) reaches 10000 * D, that is, MaxFES = 100,000 for 10D, MaxFES = 300,000 for 30D. Table 2 shows the search range and the global optimum of benchmark functions.

3.2.2. Comparison of Simulation Results for Benchmark Functions. The comparison results of mean values (mean), standard deviation (std), and Wilcoxon rank sum test (*h*) produced by all compared PSO variants on 10-dimensional tested functions are represented in Table 3 and 30-dimensional in Table 4, where the optimal results are marked in bold. The comparison results of computation time of each run are in Figures 4 and 5.

As shown in Table 3, AGMPSO performs well for unimodal functions, especially getting optimal mean fitness value on F3, despite the same result of TSLPSO on F3. In solving the multimodal functions from F4 to F10, AGMPSO has superior advantage, achieving better results on 5 out of 7 functions. However, AGMPSO does not obtain advantage solving the hybrid functions (F11–F20), and HFPSO gets the same achievement as proposed algorithm: 4 out of 10. It is still worth noting that the standard deviations of AGMPSO are smaller than HFPSO, which indicates more stability. For composition functions (F21–F30), AGMPSO algorithm perfectly reflects the better ability to search global optimum on 6 test functions, compared with the other competitors. In general, AGMPSO is top ranked on 14 out of all functions. Although on F3, F28, and F29, the proposed algorithm does not contain the statistically significant advantage, its mean

Table 2: Description of benchmark functions of CEC2017 test suits.

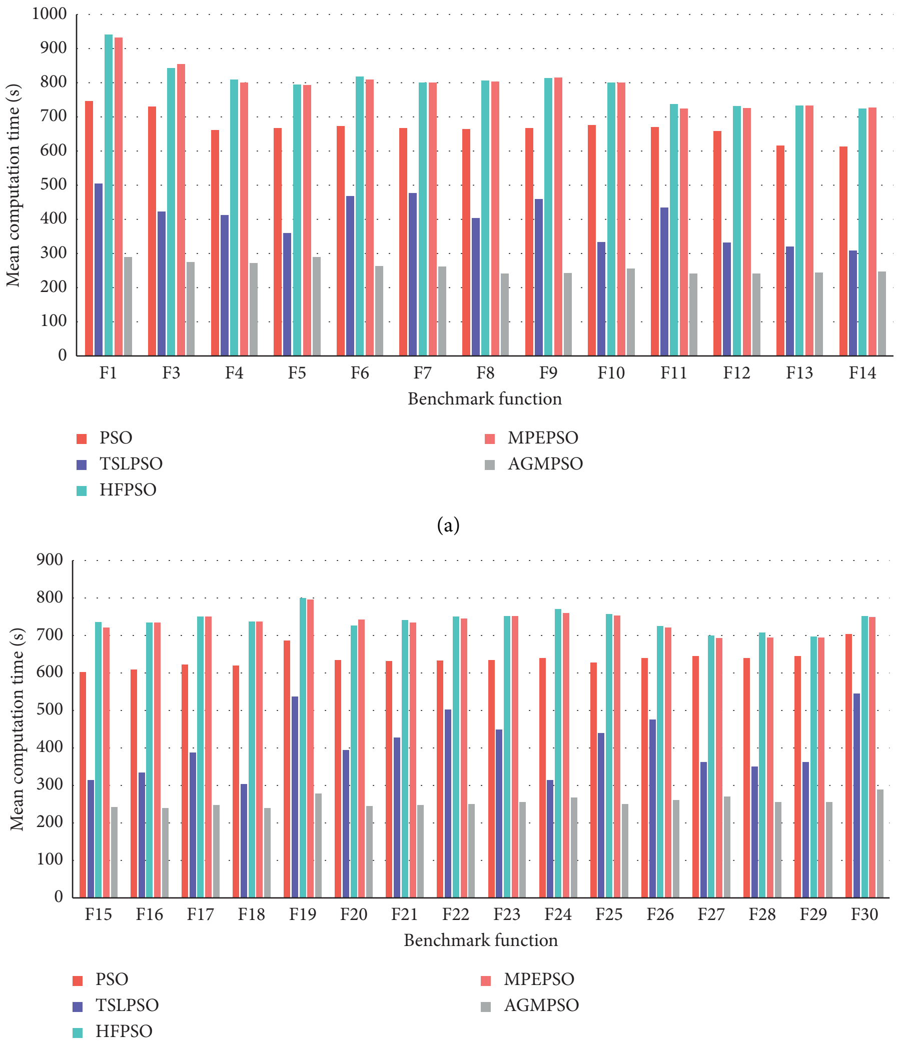
No.	Function name	Search range	Global optimum	Type
F1	Shifted and rotated bent cigar function	(−100, 100)	100	Unimodal
F3	Shifted and rotated Zakharov function	(−100, 100)	300	Unimodal
F4	Shifted and rotated Rosenbrock's function	(−100, 100)	400	Multimodal
F5	Shifted and rotated Rastrigin's function	(−100, 100)	500	Multimodal
F6	Shifted and rotated expanded Scaffer's F6 function	(−100, 100)	600	Multimodal
F7	Shifted and rotated Lunacek Bi_Rastrigin function	(−100, 100)	700	Multimodal
F8	Shifted and rotated noncontinuous Rastrigin's function	(−100, 100)	800	Multimodal
F9	Shifted and rotated Levy function	(−100, 100)	900	Multimodal
F10	Shifted and rotated Schwefel's function	(−100, 100)	1000	Multimodal
F11	Hybrid function 1 ($N=3$)	(−100, 100)	1100	Hybrid
F12	Hybrid function 2 ($N=3$)	(−100, 100)	1200	Hybrid
F13	Hybrid function 3 ($N=3$)	(−100, 100)	1300	Hybrid
F14	Hybrid function 4 ($N=4$)	(−100, 100)	1400	Hybrid
F15	Hybrid function 5 ($N=4$)	(−100, 100)	1500	Hybrid
F16	Hybrid function 6 ($N=4$)	(−100, 100)	1600	Hybrid
F17	Hybrid function 6 ($N=5$)	(−100, 100)	1700	Hybrid
F18	Hybrid function 6 ($N=5$)	(−100, 100)	1800	Hybrid
F19	Hybrid function 6 ($N=5$)	(−100, 100)	1900	Hybrid
F20	Hybrid function 6 ($N=6$)	(−100, 100)	2000	Hybrid
F21	Composition function 1 ($N=3$)	(−100, 100)	2100	Composition
F22	Composition function 2 ($N=3$)	(−100, 100)	2200	Composition
F23	Composition function 3 ($N=4$)	(−100, 100)	2300	Composition
F24	Composition function 4 ($N=4$)	(−100, 100)	2400	Composition
F25	Composition function 5 ($N=5$)	(−100, 100)	2500	Composition
F26	Composition function 6 ($N=5$)	(−100, 100)	2600	Composition
F27	Composition function 7 ($N=6$)	(−100, 100)	2700	Composition
F28	Composition function 8 ($N=6$)	(−100, 100)	2800	Composition
F29	Composition function 9 ($N=3$)	(−100, 100)	2900	Composition
F30	Composition function 10 ($N=3$)	(−100, 100)	3000	Composition

Table 3: Comparison performance between AGMPSO and PSO, TSLPSO, HFPSO and MPEPSO on CEC 2017 benchmark functions (10-D).

F*n*	AGMPSO		TSLPSO			HFPSO			MPEPSO			PSO		
	Mean	Std	Mean	Std	*h*	Mean	Std	*h*	Mean	Std	*h*	Mean	Std	*h*
F1	**1.02E+03**	1.94E+03	2.53E+03	2.06E+03	+	6.03E+08	2.83E+08	+	3.82E+10	2.79E+08	+	4.86E+08	1.76E+08	+
F3	**3.00E+02**	5.46E+01	**3.00E+02**	1.16E+01	−	3.78E+03	1.62E+03	+	3.29E+03	3.78E+03	+	8.02E+03	5.18E+03	+
F4	**4.06E+02**	1.37E+01	4.83E+02	5.09E+01	−	4.47E+02	3.70E+01	+	5.44E+03	4.46E+01	+	4.16E+02	1.93E+01	+
F5	**5.44E+02**	1.83E+01	5.94E+02	3.36E+00	+	5.83E+02	1.50E+01	+	6.91E+02	6.36E+00	+	5.79E+02	2.46E+01	+
F6	6.19E+02	9.40E+00	6.56E+02	8.10E+00	+	**6.11E+02**	8.77E+00	−	6.76E+02	7.65E+00	+	6.40E+02	1.32E+01	+
F7	**7.36E+02**	1.13E+01	8.15E+02	3.21E+00	+	7.40E+02	7.74E+00	+	8.29E+02	4.04E+00	+	7.77E+02	2.14E+01	+
F8	**8.24E+02**	9.43E+00	1.00E+03	2.54E+00	+	8.25E+02	4.87E+00	+	9.05E+02	4.87E+00	+	8.60E+02	9.96E+00	+
F9	**9.56E+02**	1.07E+02	9.98E+02	3.19E+02	+	1.12E+03	1.13E+02	+	1.73E+03	2.95E+01	+	1.33E+03	3.41E+02	+
F10	2.09E+03	3.46E+02	**1.64E+03**	1.10E+02	−	2.11E+03	2.35E+02	−	4.56E+03	7.17E+02		2.85E+03	2.15E+02	+
F11	1.13E+03	1.85E+01	1.30E+03	1.97E+00	−	**1.11E+03**	9.71E+00	−	2.51E+03	1.05E+02	+	1.57E+03	9.37E+02	+
F12	**5.33E+03**	9.64E+03	1.41E+04	1.25E+04	+	8.13E+05	5.31E+05	+	4.75E+09	9.05E+08	+	7.22E+07	2.03E+08	+
F13	**1.64E+03**	2.42E+01	1.85E+03	1.51E+02	+	6.17E+03	2.34E+03	+	5.26E+08	9.70E+08	+	3.11E+06	4.30E+06	+
F14	**1.46E+03**	2.75E+00	1.60E+03	4.40E+00	+	4.01E+03	1.57E+03	+	1.84E+04	1.51E+04	+	6.01E+03	4.10E+03	+
F15	**1.58E+03**	6.96E+01	1.59E+03	1.86E+00	−	1.71E+03	1.94E+02	+	8.53E+03	3.32E+03	+	1.71E+04	6.12E+03	+
F16	1.86E+03	1.05E+02	**1.60E+03**	2.19E+01	−	1.96E+03	2.85E+01	+	3.10E+03	1.09E+02	+	2.12E+03	1.87E+02	+
F17	1.77E+03	3.23E+01	1.89E+03	4.12E+00	+	**1.72E+03**	1.12E+01	−	2.79E+03	2.59E+02	+	1.83E+03	4.76E+01	+
F18	4.96E+03	8.96E+03	3.59E+04	6.25E+02	+	**1.87E+03**	6.90E+01	−	1.19E+10	3.29E+09	+	1.85E+07	2.77E+07	+
F19	4.00E+03	7.86E+03	5.90E+03	7.03E+01	+	**1.94E+03**	6.64E+01	−	1.11E+10	5.21E+08	+	2.04E+05	5.53E+05	
F20	2.16E+03	3.22E+01	**2.00E+03**	3.77E+01	−	2.10E+03	5.33E+01	−	2.56E+03	1.02E+02	+	2.23E+03	8.20E+01	+
F21	2.30E+03	5.84E+01	2.70E+03	3.91E+01	+	**2.24E+03**	5.69E+01	−	2.76E+03	6.96E+00	+	2.36E+03	5.40E+01	+
F22	2.33E+03	1.45E+01	2.64E+03	2.98E+01	+	**2.40E+03**	5.86E+01	−	4.91E+03	2.81E+01	+	2.37E+03	2.10E+02	+
F23	**2.67E+03**	3.40E+01	2.70E+03	3.48E+00	−	2.94E+03	1.65E+02	+	3.92E+03	8.02E+01	+	2.73E+03	8.73E+01	+
F24	2.70E+03	1.39E+02	2.74E+03	8.46E+01	−	**2.64E+03**	3.51E+01	−	3.36E+03	5.65E+00	+	2.78E+03	1.25E+02	+
F25	**2.92E+03**	2.17E+00	2.98E+03	5.37E+01	−	2.96E+03	1.13E+01	+	4.67E+03	1.90E+01	+	2.99E+03	1.47E+02	+
F26	**3.13E+03**	4.72E+02	3.80E+03	1.43E+02	+	3.56E+03	3.12E+02	+	5.51E+03	5.00E+01	+	3.53E+03	6.80E+02	+
F27	**3.13E+03**	3.73E+01	3.18E+03	1.09E+00	−	3.38E+03	1.34E+02	+	4.66E+03	9.29E+01	+	3.19E+03	7.89E+01	+
F28	**3.32E+03**	9.61E+00	3.78E+03	6.29E+01	+	3.49E+03	2.48E+02	+	4.40E+03	2.51E+01	+	**3.32E+03**	1.20E+02	−
F29	**3.24E+03**	6.01E+00	3.34E+03	1.03E+01	−	**3.24E+03**	2.95E+01	+	3.27E+03	3.86E+01	+	3.46E+03	1.29E+02	+
F30	3.45E+05	7.06E+05	7.61E+05	3.15E+03	+	**1.23E+05**	5.21E+04	+	4.32E+08	1.00E+07	+	1.23E+07	1.65E+07	+
Scores	14		4			10			0			1		
Rank	1		3			2			5			4		

Table 4: Comparison performance between AGMPSO and PSO, TSLPSO, HFPSO and MPEPSO on CEC 2017 benchmark functions (30-D).

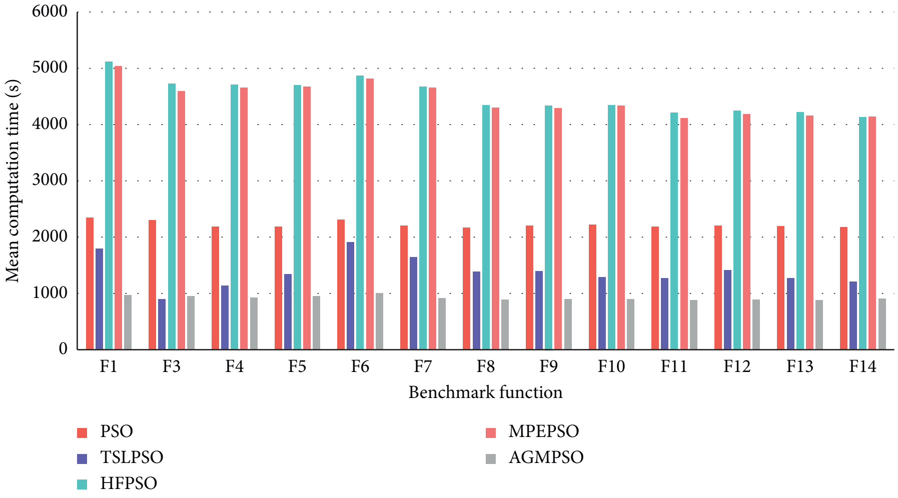
F*n*	AGMPSO		TSLPSO			HFPSO			MPEPSO			PSO		
	Mean	Std	Mean	Std	*h*	Mean	Std	*h*	Mean	Std	*h*	Mean	Std	*h*
F1	**1.03E+03**	1.64E+03	2.41E+03	1.86E+03	+	6.27E+08	2.33E+08	+	2.62E+10	2.49E+08	+	5.20E+08	1.48E+08	+
F3	**3.00E+02**	4.81E+04	**3.00E+02**	9.91E+03	–	4.12E+03	1.44E+03	+	3.33E+03	3.41E+03	+	7.71E+03	4.25E+03	+
F4	**4.39E+02**	1.15E+01	4.79E+02	4.53E+01	–	4.57E+02	2.97E+01	+	5.67E+03	3.57E+01	+	4.41E+02	1.59e+01	+
F5	**5.66E+02**	1.49E+01	6.00E+02	2.76E+00	+	6.01E+02	1.36E+01	+	7.61E+02	5.47E+00	+	6.08E+02	2.14e+01	+
F6	6.50E+02	7.61E+00	6.32E+02	9.91E+00		6.29E+02	7.63E+00	–	**6.16E+02**	6.59E+00	–	6.60E+02	1.09e+01	+
F7	**7.46E+02**	9.29E+00	7.67E+02	2.73E+00	+	8.15E+02	6.89E+00	+	8.87E+02	3.52E+00	+	7.08E+02	1.80e+01	+
F8	**8.03E+02**	1.55E+00	1.06E+03	2.24E+00	+	8.66E+02	3.58E+00	+	8.24E+02	4.09E+00	+	8.91E+02	8.75e+00	+
F9	**1.04E+03**	9.31E+01	N/A	N/A		1.06E+03	1.02E+02	+	1.86E+03	2.63E+01	+	1.42E+03	2.97e+02	+
F10	2.14E+03	2.84E+02	**1.76E+03**	9.87E+01	–	2.05E+03	2.12E+02	+	4.89E+03	6.32E+02	+	2.97E+03	1.92e+02	+
F11	**1.11E+03**	1.47E+00	1.41E+03	1.62E+00	–	1.12E+03	8.45E+00	–	2.47E+03	9.17E+02	+	1.51E+03	7.59e+02	+
F12	**5.19E+03**	7.91E+03	1.30E+04	1.00E+04	+	7.65E+05	4.30E+05	+	4.71E+09	7.51E+08	+	7.80E+07	1.71e+08	+
F13	**1.56E+03**	2.09E+02	1.67E+03	1.26E+02	+	6.17E+03	2.04E+03	+	5.59E+08	8.64E+08	+	2.81E+06	3.58e+06	+
F14	**1.43E+03**	2.31E+01	1.57E+03	3.96E+00	+	4.37E+03	1.42E+03	+	1.75E+04	1.31E+04	+	5.82E+03	3.61e+03	+
F15	**1.54E+03**	5.92E+01	1.58E+03	1.56E+00	–	1.60E+03	1.71E+02	+	9.22E+03	2.70E+03	+	1.64E+04	5.21e+03	+
F16	**1.63E+03**	1.16E+01	1.67E+03	1.82E+01	–	1.97E+03	2.37E+01	+	3.11E+03	9.40E+01	+	2.06E+03	1.52e+02	+
F17	1.73E+03	2.72E+00	1.71E+03	3.30E+00	+	**1.70E+03**	9.58E+00	–	2.80E+03	2.26E+02	+	1.91E+03	4.14e+01	+
F18	4.47E+03	7.71E+03	3.33E+04	5.07E+02	+	**1.85E+03**	5.66E+01	–	1.12E+10	2.96E+09	+	1.78E+07	2.49e+07	+
F19	4.28E+03	6.69E+03	6.37E+03	5.63E+01	+	**1.90E+03**	5.35E+01	–	1.15E+10	4.38E+08	+	1.96E+05	4.59e+05	+
F20	**2.00E+03**	2.48E+01	2.01E+03	3.32E+01	+	2.09E+03	4.53E+01	–	2.49E+03	9.05E+01	+	2.42E+03	7.22e+01	+
F21	2.26E+03	4.67E+01	2.79E+03	3.53E+01	+	**2.33E+03**	4.67E+01	–	3.01E+03	5.92E+00	+	2.29E+03	4.38e+01	+
F22	**2.24E+03**	1.31E+02	2.73E+03	2.48E+01	+	2.29E+03	4.99E+01	–	4.42E+03	2.47E+01	+	2.85E+03	2.10e+02	+
F23	**2.58E+03**	2.76E+00	2.61E+03	3.14E+00	–	2.89E+03	1.46E+02	+	4.08E+03	6.82E+01	+	2.67E+03	7.16e+01	+
F24	2.63E+03	1.20E+02	3.70E+03	7.53E+01	–	2.60E+03	3.02E+01	+	**2.59E+03**	4.70E+00	+	2.73E+03	1.03e+02	+
F25	**2.85E+03**	1.94E+01	2.87E+03	4.35E+01	–	2.91E+03	9.73E+00	+	4.40E+03	1.62E+01	+	2.95E+03	1.21e+02	+
F26	**3.10E+03**	3.97E+01	3.72E+03	1.22E+02	+	3.35E+03	2.53E+02	+	5.57E+03	4.20E+01	+	3.32E+03	5.71e+02	+
F27	**3.09E+03**	3.33E+01	3.13E+03	9.10E+01	–	3.29E+03	1.07E+02	+	4.48E+03	8.27E+01	+	3.13E+03	6.87e+01	+
F28	3.26E+03	8.27E+01	3.70E+03	5.54E+01	+	3.46E+03	2.06E+02	+	4.28E+03	2.16E+01	**+**	**3.25E+03**	1.02e+02	–
F29	**2.96E+03**	4.99E+01	3.25E+03	8.25E+00	+	3.08E+03	2.54E+01	+	3.60E+03	3.09E+01	+	3.25E+03	1.10e+02	+
F30	3.25E+05	5.65E+05	7.00E+05	2.84E+03	+	**8.96E+04**	4.53E+04	+	4.21E+08	8.41E+06	+	1.15E+07	1.32e+07	+
Scores	19		2			5			2			1		
Rank	1		4			2			3			5		



(a)

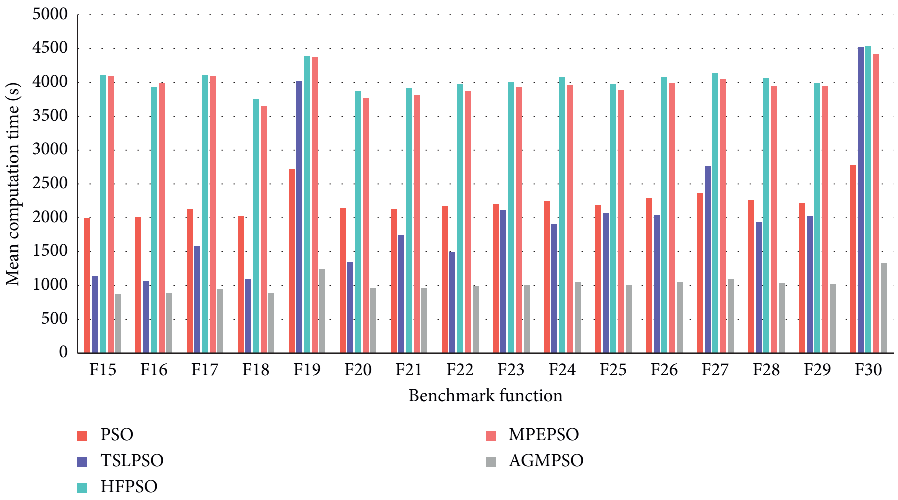
(b)

Figure 4: The average computation time of comparison algorithms on benchmark functions (10-D).



(a)

Figure 5: Continued.



(b)

Figure 5: The average computation time of comparison algorithms on benchmark functions (30-D).

fitness values equal those of winning algorithms, which makes AGMPSO take the second rank on the above-mentioned function.

For 30-dimensional experiments in Table 4, since the iterations greatly increase, the results of all algorithms have improved in varying degrees. AGMPSO achieves optimal mean fitness value on F3 and F20. It gets the same performance as in 10-dimensional experiments in unimodal and multimodal functions, gets 7 out of 10 best results in hybrid functions, and 6 out of 10 best results in composition functions. In general, AGMPSO outperforms peers for the 19 benchmark functions.

In short, AGMPSO is top ranked in both 10-dimensional and 30-dimensional experiments.

The standard deviations of AGMPSO on different benchmark functions are generally smaller than those of comparison PSO variants, which indicates better robustness of our algorithm.

Aiming to analyze the computational efficiency of compared peers, the average computation time for each algorithm to run all benchmark functions is depicted in Figures 4 and 5. It can be concluded that AGMPSO consumes lower computational overhead than its peers and the advantage grows obvious when iteration increases as shown in Figure 5. It is worth noting that, despite shared the same time complexity of standard PSO, the time consumption of AGMPSO is significantly lower indicating higher computational efficiency.

In summary, the outstanding results of AGMPSO in terms of average computational time and fitness values demonstrate that our proposed algorithm obtains higher search accuracy and convergence rate than its peers, meanwhile, with the significant robust.

3.2.3. Wilcoxon's Rank Sum Test Results. The Wilcoxon rank sum test at a significance level of $\alpha = 0.05$ is performed on the ranking between AGMPSO and other PSO variants to analyze their statistical significance. Tables 3 and 4 list the results of Wilcoxon rank sum test on fitness values of all 29 functions. In both tables, h value (+/−/~) indicates that the AGMPSO performs significantly better, significantly worse, or not statistically significant than its competitor.

It can be observed from the results that AGMPSO is significantly better than compared PSO variants in most of test functions. AGMPSO gets the same mean values with TSLPSO, standard PSO, and HFPSO on 10-dimensional F3, F28, and F29, while with TSLPSO on 30-dimensional F3. However, from the rank sum test results, AGMPSO performs significantly worse than the peer on above functions. Although the proposed algorithm does not gain the superior rank on these functions, it still takes the second rank in mean fitness value on each function, and the difference is slight.

3.2.4. Convergence Progresses. In order to observe the convergence speed of all the peer algorithms, the convergence processes in random runs of the comparison algorithms on benchmark functions of 10 dimensions are depicted in Figures 6 and 7.

In Figure 6, AGMPSO does not show the characteristics of rapid convergence on F1 and F3 in the initial period because of the strong particle diversity yielded by proposed mutation strategy; meanwhile, the high solution accuracy is achieved by our algorithm at the later convergence stage.

From F4 to F10, we can observe that the other comparison algorithms fall into the local minima to different extent, while AGMPSO attains the favorable performances on all multimodal functions. However, it is noteworthy that the effect of proposed adaptive strategies is not obvious on F4, F6, and F9, which may cause the low diversity in the early period and fail to find the global optima. This is demonstrated by the result of F6 in Table 3.

From the performances for F11 to F20, the similar problems of rapid convergence in the initial period can be observed on F11, F14, and F15. Furthermore, AGMPSO fails

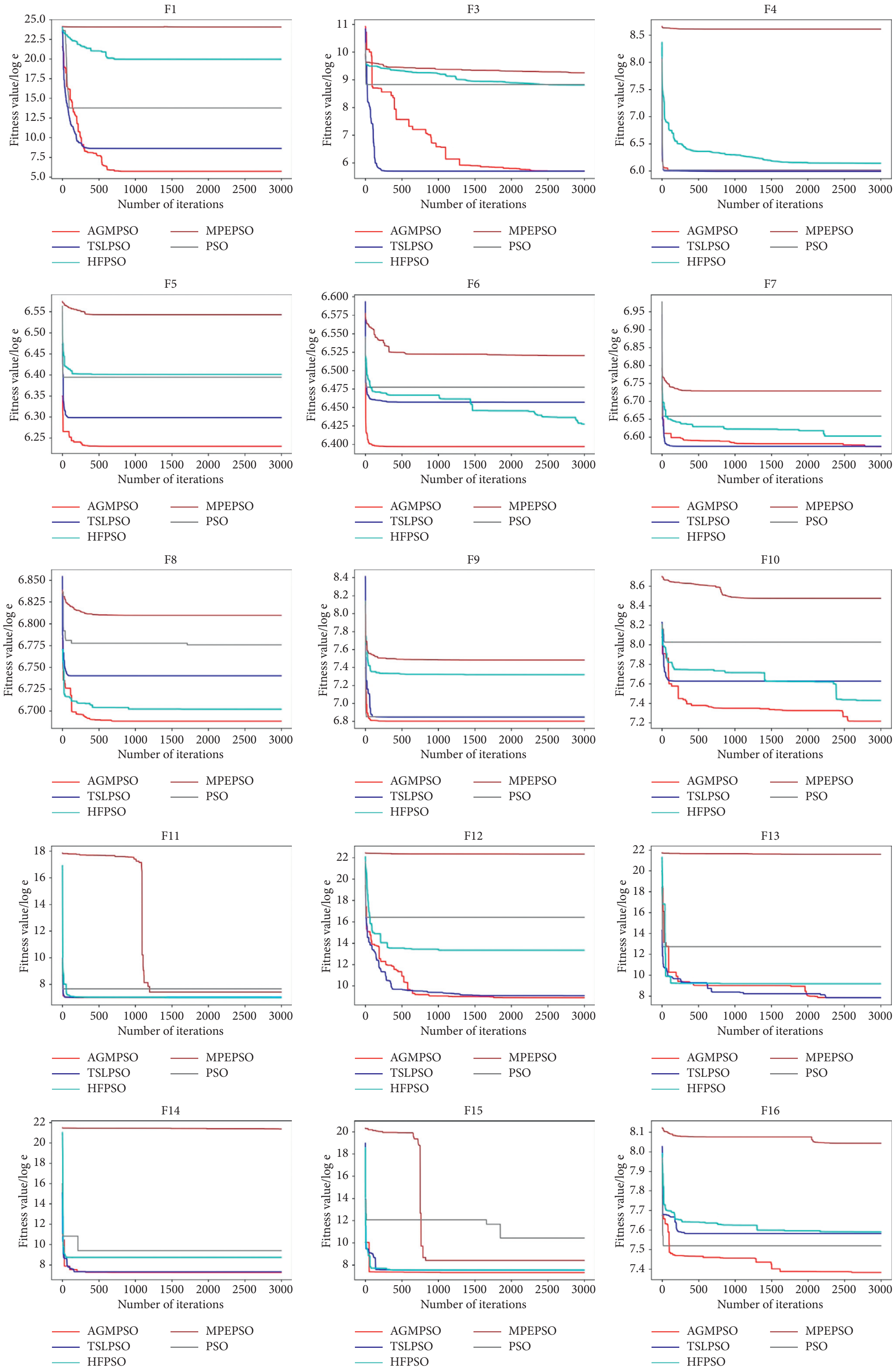


Figure 6: Convergence curves of comparison algorithms on F1–F16.

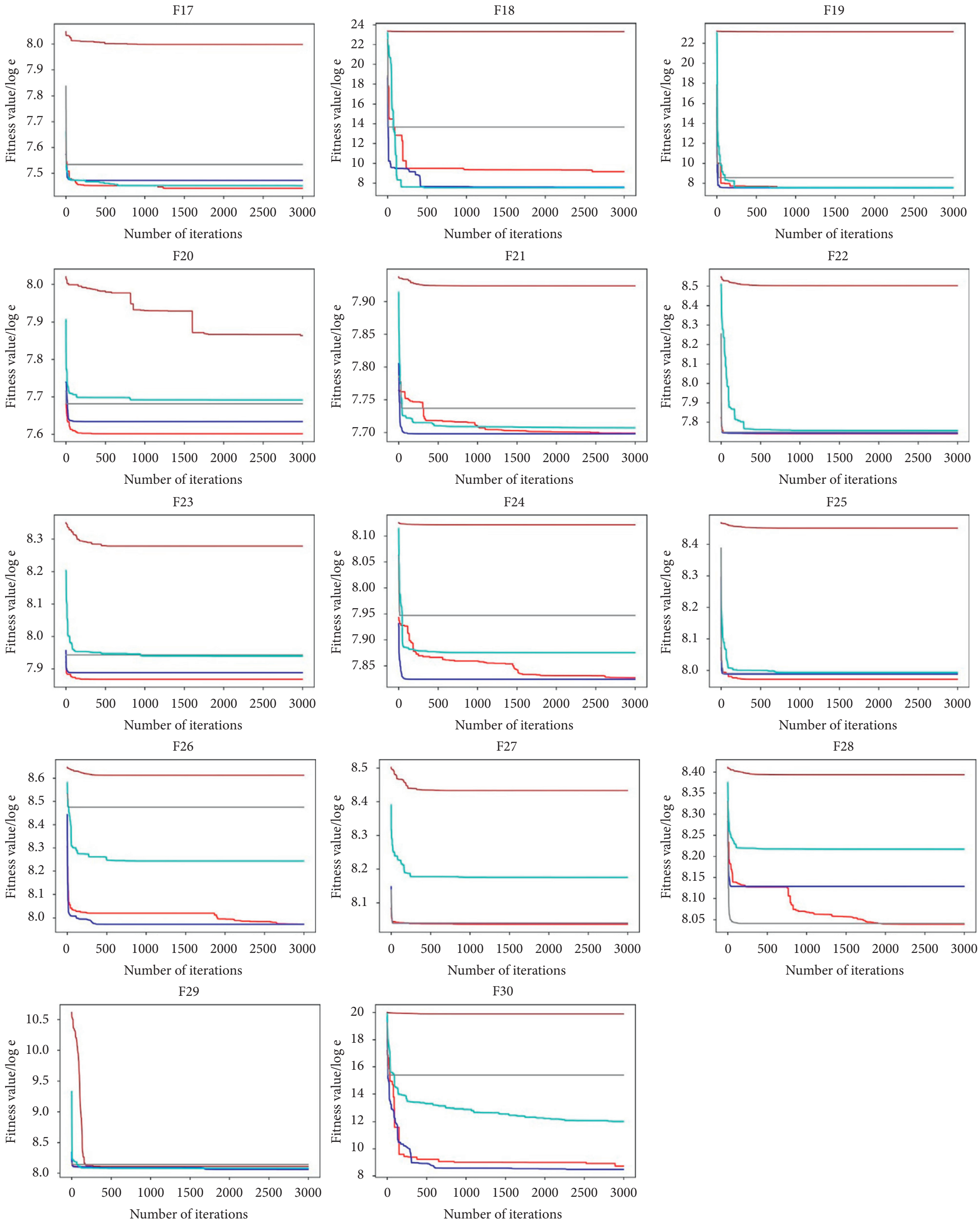


Figure 7: Convergence curves of comparison algorithms on F17–F30.

to achieve the satisfactory results on F18. Despite with the 4 outstanding achievements and the improvement in 30-dimensional experiment results, due to the rapid convergence rate at the early stage, the exploration capability of proposed algorithm in solving hybrid functions should be promoted.

The results presented in F21 to F30 depict that exploitation capability of AGMPSO is higher than most of the peers. The gradual convergence process on F21, F24, F26, F28, and F30 can be seen, which reflects the balance of exploration and exploitation of our algorithm.

The conscious summary can be drawn that AGMPSO performs well on most of functions. Meanwhile, the proposed algorithm does not yield satisfactory performances on hybrid functions, so do the other PSO variants, which suggests much room for improvement in solving these functions. It turns out from the 30-dimensional comparison results that, despite the low efficiency, increasing the number of iterations could be an improvement insight worth discussing.

By contrast with the other competitors, a relatively slow and gradual convergence curve presents in solving many functions, which exactly portrays the algorithm's intent of finding out more promising solutions.

4. Conclusion

Adaptive particle swarm optimization with Gaussian perturbation and mutation is proposed to address the existing drawback of standard PSO. To prevent trapping into local optimum, Gaussian perturbation is implemented to global optima, further increasing the exploitation capability. For the nonoptimal particles that fall into the evolutionary stagnation, the mutation is leveraged to promote the particles' diversity and utilization to improve the exploration ability. Simultaneously, the adaptive strategy regulates the interference level of Gaussian perturbation and mutation during different evolutional stages in order to balance the searching ability and accuracy. The visual result of aggregation analysis validates this dynamic process. The performance on benchmark functions of CEC 2017 test suits manifests that AGMPSO outperforms its competitors by a big margin in terms of searching accuracy, searching reliability, and searching efficiency.

In future works, considering the powerful global search ability of the particle swarm algorithm, it can be considered to optimize the topology, connection weights, and thresholds of the neural networks or combine the global optimization ability of PSO with the local optimization ability of the BPNN to improve the generalization and learning performance of the neural network.

Data Availability

No data were used to support this study.

Conflicts of Interest

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

Acknowledgments

This work was supported by the National Natural Science Foundation of China under grants 71371181 and 71672193 and by the Research Foundation of Xian International Studies University under grant BSZA2019003.

References

[1] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, IEEE, Pretoria, South Africa, June 2002.

[2] W. Han, P. Yang, H. Ren, and J. Sun, "Comparison study of several kinds of inertia weights for PSO," in *Proceedings of the 2010 IEEE International Conference on Progress in Informatics and Computing*, vol. 1, pp. 280–284, IEEE, Beijing, China, December 2010.

[3] J. C. Xu, T. H. Xu, L. Sun, and J. Y. Ren, "Feature selection for cancer classification based on neighborhood rough set and particle swarm optimization," *Journal of Chinese Computer Systems*, vol. 35, no. 11, pp. 2528–2532, 2014.

[4] R. Mendes, J. Kennedy, and J. Neves, "The fully informed particle swarm: simpler, maybe better," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 204–210, 2004.

[5] Z. H. Zhan, J. Zhang, Y. Li, and S. H. Chung, "Adaptive particle swarm optimization," *IEEE Transactions on Systems Man & Cybernetics Part B Cybernetics*, vol. 39, no. 6, pp. 1362–1381, 2009.

[6] R. Cheng and Y. Jin, "A social learning particle swarm optimization algorithm for scalable optimization," *Information Sciences*, vol. 291, pp. 43–60, 2015.

[7] B. Alatas, E. Akin, and A. B. Ozer, "Chaos embedded particle swarm optimization algorithms," *Chaos, Solitons & Fractals*, vol. 40, no. 4, pp. 1715–1734, 2009.

[8] X. C. Zhao, G. L. Liu, H. Q. Liu, and G. S. Zhao, "Particle swarm optimization algorithm based on non-uniform mutation and multiple stages perturbation," *Chinese Journal of Computers*, vol. 60, pp. 1–20, 2014.

[9] J. J. Liang, A. K. Qin, P. N. Suganthan, and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 3, pp. 281–295, 2006.

[10] W.-N. Chen, J. Zhang, Y. Lin et al., "Particle swarm optimization with an aging leader and challengers," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 2, pp. 241–258, 2013.

[11] X. U. Xiao-Bo, K. F. Zheng, L. I. Dan, W. U. Bin, and Y. X. Yang, "New chaos-particle swarm optimization algorithm," *Journal on Communications*, vol. 33, no. 1, pp. 24–16, 2012.

[12] V. D. B. Frans and A. P. Engelbrecht, "A cooperative approach to particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

[13] D. Tian, "Particle swarm optimization with chaos-based initialization for numerical optimization," *Intelligent Automation & Soft Computing*, vol. 24, no. 2, pp. 331–342, 2017.

[14] W. B. Du, W. Ying, G. Yan, Y. B. Zhu, and X. B. Cao, "Heterogeneous strategy particle swarm optimization," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 64, no. 4, pp. 467–471, 2016.

[15] M. Munlin and M. Anantathanavit, "Hybrid radius particle swarm optimization," in *Proceedings of the 2016 IEEE Region 10 Conference (TENCON)*, pp. 2180–2184, IEEE, Singapore, November 2016.

[16] M. S. Kiran, "Particle swarm optimization with a new update mechanism," *Applied Soft Computing*, vol. 60, pp. 670–678, 2017.

[17] K. Elumalai, M. Elumalai, K. Eluri et al., "Facile synthesis, spectral characterization, antimicrobial and in vitro cytotoxicity of novel N3, N5-diisonicotinyl-2, 6-dimethyl-4-phenyl-1, 4-dihydropyridine-3, 5-dicarbohydrazide derivatives," *Bulletin of Faculty of Pharmacy, Cairo University*, vol. 1, no. 54, pp. 77–86, 2016.

[18] S. Rastegar, R. Araújo, and J. Mendes, "Online identification of Takagi-Sugeno fuzzy models based on self-adaptive hierarchical particle swarm optimization algorithm," *Applied Mathematical Modelling*, vol. 45, pp. 606–620, 2017.

[19] A. A. Nagra, F. Han, and Q. H. Ling, "An improved hybrid self-inertia weight adaptive particle swarm optimization algorithm with local search," *Engineering Optimization*, vol. 51, no. 7, pp. 1115–1132, 2019.

[20] A. A. Nagra, F. Han, Q. H. Ling et al., "Hybrid self-inertia weight adaptive particle swarm optimisation with local search using C4.5 decision tree classifier for feature selection problems," *Connection Science*, vol. 32, no. 1, pp. 16–36, 2020.

[21] Y. Xue, B. Xue, and M. Zhang, "Self-adaptive particle swarm optimization for large-scale feature selection in classification," *ACM Transactions on Knowledge Discovery from Data*, vol. 13, no. 5, pp. 1–27, 2019.

[22] R. S. Kumar, K. Kondapaneni, V. Dixit, A. Goswami, L. S. Thakur, and M. K. Tiwari, "Multi-objective modeling of production and pollution routing problem with time window: a self-learning particle swarm optimization approach," *Computers & Industrial Engineering*, vol. 99, pp. 29–40, 2016.

[23] M.-C. Chen, Y.-H. Hsiao, R. Himadeep Reddy, and M. K. Tiwari, "The self-learning particle swarm optimization approach for routing pickup and delivery of multiple products with material handling in multiple cross-docks," *Transportation Research Part E: Logistics and Transportation Review*, vol. 91, pp. 208–226, 2016.

[24] M. Hu, T. Wu, and J. D. Weir, "An adaptive particle swarm optimization with multiple adaptive methods," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 5, pp. 705–720, 2012.

[25] F. Wang, H. Zhang, K. Li, Z. Lin, J. Yang, and X.-L. Shen, "A hybrid particle swarm optimization algorithm using adaptive learning strategy," *Information Sciences*, vol. 436-437, pp. 162–177, 2018.

[26] S.-F. Li and C.-Y. Cheng, "Particle swarm optimization with fitness adjustment parameters," *Computers & Industrial Engineering*, vol. 113, pp. 831–841, 2017.

[27] X. Hu and R. C. Eberhart, "Adaptive particle swarm optimization: detection and response to dynamic systems," in *Proceedings of the 2002 Congress on Evolutionary Computation CEC'02*, vol. 2, pp. 1666–1670, Honolulu, HI, USA, May 2002.

[28] X. F. Xie, W. J. Zhang, and Z. L. Yang, "Adaptive particle swarm optimization on individual level,"vol. 2, pp. 1215–1218, in *Proceedings of the 6th International Conference on Signal Processing*, vol. 2, pp. 1215–1218, IEEE, Beijing, China, August 2002.

[29] H. Wang, Z. Wu, S. Rahnamayan, Y. Liu, and M. Ventresca, "Enhancing particle swarm optimization using generalized opposition-based learning," *Information Sciences*, vol. 181, no. 20, pp. 4699–4714, 2011.

[30] M. J. Mahmoodabadi, Z. Salahshoor Mottaghi, and A. Bagheri, "Hepso: high exploration particle swarm optimization," *Information Sciences*, vol. 273, no. 18, pp. 101–111, 2014.

[31] H. Wang, H. Sun, C. Li, S. Rahnamayan, and J.-S. Pan, "Diversity enhanced particle swarm optimization with neighborhood search," *Information Sciences*, vol. 223, pp. 119–135, 2013.

[32] Y. V. Pehlivanoglu, "A new particle swarm optimization method enhanced with a periodic mutation strategy and neural networks," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 436–452, 2012.

[33] S. Shao, Y. Peng, C. He, and Y. Du, "Efficient path planning for UAV formation via comprehensively improved particle swarm optimization," *ISA Transactions*, vol. 97, pp. 415–430, 2020.

[34] X. Tao, W. Guo, Q. Li, C. Ren, and R. Liu, "Multiple scale self-adaptive cooperation mutation strategy-based particle swarm optimization," *Applied Soft Computing*, vol. 89, pp. 106–124, 2020.

[35] G. Xu, Q. Cui, X. Shi et al., "Particle swarm optimization based on dimensional learning strategy," *Swarm and Evolutionary Computation*, vol. 45, pp. 33–51, 2019.

[36] İ. B. Aydilek, "A hybrid firefly and particle swarm optimization algorithm for computationally expensive numerical problems," *Applied Soft Computing*, vol. 66, pp. 232–249, 2018.

[37] Z. Liu and T. Nishi, "Multipopulation ensemble particle swarm optimizer for engineering design problems," *Mathematical Problems in Engineering*, vol. 2020, Article ID 1450985, 30 pages, 2020.

[38] N. H. Awad, M. Z. Ali, J. J. Liang, B. Y. Qu, and P. N. Suganthan, *Problem Definitions and Evaluation Criteria for the CEC 2017 Special Session and Competition on Single Objective Bound Constrained Teal-Parameter Numerical optimization*, Nan- yang Technological University, Singapore, 2016.