Particle Swarm Optimization based on Vector Gaussian Learning

Jia Zhao[1,2], Li Lv[1,2], Hui Wang[1,2], Hui Sun[1,2], Runxiu Wu[2], Jugen Nie[1,2] and Zhifeng Xie[2]
[1] Jiangxi Province Key Laboratory of Water Information Cooperative Sensing and Intelligent Processing, Nanchang Institute of Technology
Nanchang, 330099 - CHINA
[e-mail: zhaojia925@163.com; lvli623@163.com; huiwang@whu.edu.cn; sunhui@nit.edu.cn]
[2] School of Information Engineering, Nanchang Institute of Technology
Nanchang, 330099 - CHINA
[e-mail: wurunxiu@tom.com; niejugen@163.com; xiezhifeng_nit@163.com]
*Corresponding author: Jia Zhao

Received November 6, 2016; revised January 14, 2017; accepted February 3, 2017; published April 30, 2017

Abstract

Gaussian learning is a new technology in the computational intelligence area. However, this technology weakens the learning ability of a particle swarm and achieves a lack of diversity. Thus, this paper proposes a vector Gaussian learning strategy and presents an effective approach, named particle swarm optimization based on vector Gaussian learning. The experiments show that the algorithm is more close to the optimal solution and the better search efficiency after we use vector Gaussian learning strategy. The strategy adopts vector Gaussian learning to generate the Gaussian solution of a swarm's optimal location, increases the learning ability of the swarm's optimal location, and maintains the diversity of the swarm. The method divides the states into normal and premature states by analyzing the state threshold of the swarm. If the swarm is in the premature category, the algorithm adopts an inertia weight strategy that decreases linearly in addition to vector Gaussian learning; otherwise, it uses a fixed inertia weight strategy. Experiments are conducted on eight well-known benchmark functions to verify the performance of the new approach. The results demonstrate promising performance of the new method in terms of convergence velocity and precision, with an improved ability to escape from a local optimum.

Keywords: Particle Swarm Optimization, Gaussian Learning, Elite Particle, Vector

This research was supported by the Jiangxi Province Department of Education Science and Technology Project under Grant (No. GJJ151133), the National Natural Science Foundation of China under Grant (Nos. 51669014, 61663029, 61561035), Science Foundation of Jiangxi Province under Grant (Nos.20161BAB212037, 20151BAB207039).

https://doi.org/10.3837/tiis.2017.04.012 **ISSN : 1976-7277**

1. Introduction

Particle swarm optimization (PSO) [1] is a swarm intelligence algorithm that emulates the behavior of birds of prey and was originally presented by Kennedy and Eberhart. The simplicity of the technique, the fact that it only requires a few parameters, and its easy implementation have encouraged researchers to apply PSO to many research areas, such as to solve power system load flow problems [2], wireless sensor networks[3], multi-objective problems[4], and image processing [5,6].

The disadvantage of PSO is that it easily converges to a local optimum and has the problem of premature and slow convergence velocity. Various researchers improved the standard PSO algorithm to enhance its performance. Han et al. [7] used an example set of multiple global best particles to update the positions of the particles and presented example-based learning particle swarm optimization (ELPSO). Beheshti et al. [8] developed centripetal accelerated particle swarm optimization (CAPSO), inspired by Newton's law of motion. Particle swarm optimization with an aging leader and challengers (ALC-PSO) [9] was introduced by Chen et al. Zhang et al.[10] proposed an adaptive bare-bones particle swarm optimization algorithm (ABBPSO), in which each particle had its own disturbance value and the value was used to adaptively decide the convergence degree of the particle and the diversity of the swarm. The swarm's evolving solution is represented by the best solution; however, this best solution often limits the search area. To solve the problem, Wang et al. [11] proposed a dynamic tournament topology strategy to improve PSO. Cheng et al. [12] introduced social learning mechanisms into PSO to develop social learning PSO (SL-PSO). In order to improve the convergence and diversity, Yang et al. proposed a multi-objective PSO algorithm based on the interaction of multi-level information(MLII-MOPSO) [13], in which the optimization is divided into the standard particle optimization layer, the particle evolution and learning layer, and the archive information exchange layer. Considering a fractional calculus approach, Couceiro and Sivasundaram provided a novel fractional PSO (FPSO) [14]. Although the above-mentioned improved PSO variants can improve the performance, an improved strategy has proven difficult to realize.

Gaussian learning, a new technology in the computational intelligence area, adds a random disturbance term obeying Gaussian distribution to the individuals and generates Gaussian solutions, after which it chooses an improved solution from among the Gaussian solutions as the next generation individual. Gaussian learning has been applied to PSO [15], the harmony search algorithm [16], artificial bee colony algorithm [17, 18], and the bacterial foraging optimization algorithm [19]. However, all dimensions of the particles use Gaussian learning, which means that the effect of the technique is not strong. In addition, due to the use of the same learning strategy, the nature of the particle is such that convergence becomes easy, the population diversity is reduced, and the learning effect is sub-optimal.

Our approach to solving the above-mentioned problems is to propose PSO based on vector Gaussian learning (VGL-PSO). It can generate the vector Gaussian solution by using a vector Gaussian learning strategy, which, rather than being applied to all particles, is only applied to elite particles. In the process of learning, this strategy maintains the swarm diversity. If the algorithm converges prematurely, it can adjust the inertia weight adaptively, use the vector Gaussian learning strategy to escape from a local optimum, and enhance local exploitation. Experiments show that VGL-PSO, compared with other algorithms, has fast convergence velocity, and a strong ability to escape from a local optimum.

We organize the remainder of this paper as follows. We provide a simple introduction of PSO in Section 2. The details of VGL-PSO are described in Section 3. The effectiveness of the proposed method is revealed by numerous experiments in Section 4. Finally, we summarize the paper.

2. Particle Swarm Optimization

In PSO, the number and dimensions of aparticle are N and M, respectively. $X_i = (x_{i1}, x_{i2}, \cdots, x_{iM})$ and $V_i = (v_{i1}, v_{i2}, \cdots, v_{iM})$ represent the location and velocity of the i^{th} ($i = 1,2,...,N$) particle, respectively.

For the k^{th} iteration, the location and velocity of particle i are updated as follows.

$$x_{im}^{k} = x_{im}^{k-1} + v_{im}^{k} \tag{1}$$

$$v_{im}^{k} = w * v_{im}^{k-1} + c_1 * r_1 * (pBest_{im}^{k-1} - x_{im}^{k-1}) + c_2 * r_2 * (gBest_{m}^{k-1} - x_{im}^{k-1}) \tag{2}$$

where $m = 1,2,...,M$, w isthe inertia weightof which the function is to balance the ability to perform local and global search, c_1 and c_2 represent theacceleration factors, and r_1 and r_2 are random numbers between 0 and 1. $gBest_m^{k-1}$ is the optical location of the swarm at the $(k-1)^{th}$ iteration, $pBest_{im}^{k-1}$ is the best location of the i^{th} individual at the $(k-1)^{th}$ iteration.

3. Particle swarm optimization based on vector Gaussian learning

3.1 Vector Gaussian learning strategy

Definition 1---Gaussian Solution (GS): Given a point in the M dimensions as a candidate solution $X_i = (x_{i1}, x_{i2}, \cdots, x_{iM})$, the corresponding Gaussian solution is defined as follows.

$$x_{ij}^{*} = x_{ij} + Gauss\_random(\mu, \sigma^2) \tag{3}$$

where, x_{ij}^{*} is the new location of the i^{th} individual at the j^{th} dimension after Gaussian learning, x_{ij} is the previous location of the i^{th} individual at the j^{th} dimension, $Gauss\_random(\mu, \sigma^2)$ is a Gaussian random function, and μ and σ^2 are the mean and variance, respectively.

The optimal location of the swarm, namely, the location of an elite particle, is the leader and is used as learning example for the other particles and the elite particle itself has no learning example. If the elite particle converges to a local optimum, the algorithm is considered to have converged prematurely. An individual particle learns from the advantages and disadvantages of other particles, which do not have a strong learning ability [20]. Therefore, we introduce vector Gaussian learning, which can balance the global exploration and local exploitation ability. The Gaussian learning dimension of an elite particle decreases linearly with the evolution. In the early stage of evolution, we select a larger dimension space to study, enhance the ability to explore the algorithm, and improve the probability of elite particles to search for the global optimal position. This algorithm requires smaller exploration ability and larger development ability with each iterative cycle, especially in the later stages of evolution. One or

a few dimensions converge to a local optimum, and the algorithm selects a smaller dimension to learn in the later stage of the iteration. As the other dimensions of information remain the same, the strategy can retain as much advantageous information as possible, enable the algorithm to escape from a local optimum and improve the accuracy of the solution.

Definition 2---Vector Gaussian Solution (VGS) --- Given an elite particle in the M dimensions as $gBest$, the corresponding vector Gaussian solution $gBest^*$ is defined as follows. where, x_{ij}^* is the new location of the i^{th} individual at the j^{th} dimension after Gaussian learning, x_{ij} is the old location of the i^{th} individual at the j^{th} dimension, $Gauss\_random(\mu,\sigma^2)$ is a Gaussian random function, μ and σ^2 are the mean and variance, respectively.

The optimal location of swarm, namely, the location of elite particle, is the leader and learning example of other particles and the elite particle itself has no learning example. If the elite particle falls into local optimum, the algorithm will be premature. Individual learns from the advantages and disadvantages of other particles, whose learning is not strong [20]. Therefore, we introduce the vector Gaussian learning, which can balance global exploration and local exploitation ability. The Gaussian learning dimension of the elite particle is linearly decreasing with the evolution. In the early stage of evolution, we select the larger dimension space to study, enhance the ability to explore the algorithm, and improve the probability of searching the global optimal position of elite particles. Algorithm requires smaller exploration ability and larger development ability with the iteration, especially in the later stage of evolution. One or a few dimensions fall into local optimum, and algorithm selects a smaller dimension to learn in later stage, other dimensions of information remains the same, so it can retain the advantage information as most as possible, help the algorithm to escape from local optimum and improve the accuracy of the solution.

Definition 2---Vector Gaussian Solution (VGS) --- Given a elite particle in the M dimensions as $gBest$, the corresponding vector Gaussian solution $gBest^*$ is defined as follows.

$$num = Int((1.0 - i*1.0/iterNum)*M)+1 \tag{4}$$

$$m = rand()\%M \tag{5}$$

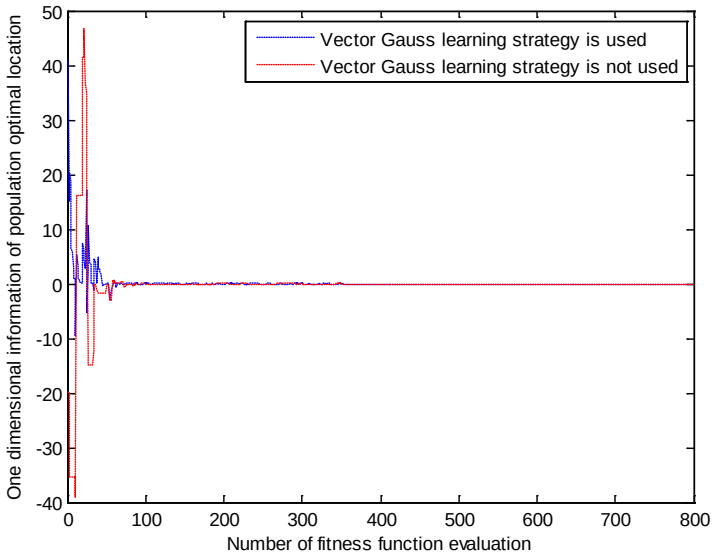
$$gBest_m^* = gBest_m + Gauss\_random(\mu,\sigma^2) \tag{6}$$

where, M is the dimension of the particle; i and $iterNum$ represent the current iteration number and maximum iteration number, respectively; num is the dimension of vector Gaussian learning(as the number of iterative cycles increases, the dimension of learning becomes increasingly smaller, until it becomes one dimension); $Int()$ is an integral function; $m(0 \le m < M-1)$ is a random number, which means the m^{th} dimension; $Gauss\_random(\mu,\sigma^2)$ is a Gaussian random function; $gBest_m$ represents the location information of the elite particle in the m^{th} dimension.

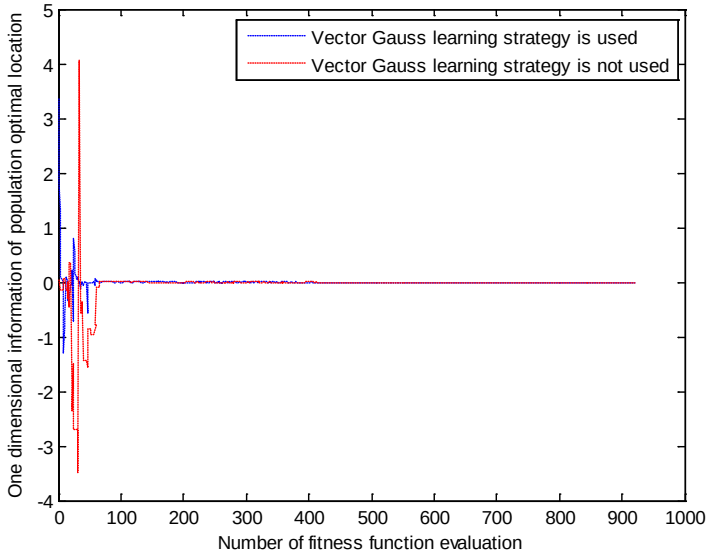
In order to verify whether $gBest^*$ is a more optimal solution than $gBest$, we choose two unimodal functions (Sphere and Schwefel's P2.22) and multimodal functions (Rastrigin and Noncontinuous Rastrigin) to test. PSO is used to calculate the test functions, for which two different operations are used: the first uses the vector Gaussian learning strategy, and the other does not use any learning strategies in the optimal position of the population. The influence of the vector Gaussian learning strategy on the algorithm is demonstrated by randomly choosing

one dimension to display in the optimal position of the population. **Fig. 1** and **2** display the influence of the Gaussian learning strategy on the unimodal and multimodal functions, respectively. The horizontal and vertical axes represent the number of iterative cycles and the information of the optimal location of the population in one dimension, respectively.

Vector Gaussian learning has a certain influence on the performance of the algorithm on the unimodal functions in **Fig. 1**. In the first 100 cycles of the iteration, the location dimension information of the particle is characterized by a small amplitude and slow frequency. As the number of iterative cycles increases, the influence of the vector Gaussian learning strategy diminishes, and the advantage of the vector Gaussian learning is not obvious after the 100th cycle, especially. Compared with unimodal functions, the influence on multimodal functions is greater in **Fig. 2**. As shown in **Fig. 2** (a), regardless as to whether we adopt vector Gaussian learning, the dimensional information of a particle experiences repeated shocks during the first 25 iterations. The dimensional information that uses vector Gaussian learning gradually becomes closer in the vicinity of the optimum; however, without using the learning strategies, the dimension information of the particle remains unaffected by arbitrary shock. Without vector Gaussian learning, the dimension information of a particle is located near the optimum between 40 and 60 cycles and escapes from the optimum after the 60^{th} iterative cycle. **Fig. 2** (b) shows that, without vector Gaussian learning the particles converges to the local optimum and is unable to escape from it within 20 iterative cycles. In contrast, a particle with vector Gaussian learning found the optimal location after fewer iterative cycles. This comparative analysis enables us to conclude that the use of the vector Gaussian learning strategy makes it easier for the algorithm to approach the optimal solution and the search efficiency is higher than that of particles without the vector Gaussian learning strategy. Therefore, the vector Gaussian learning strategy can broaden the active region of the group, improve the diversity, and avoid local optimization.

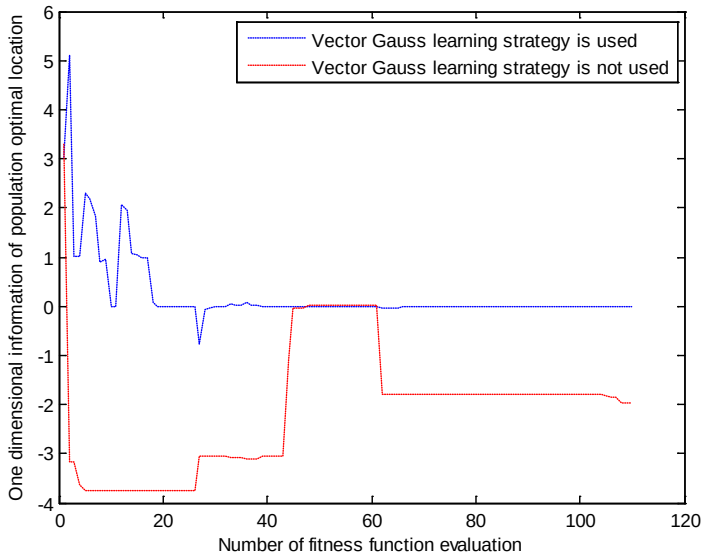


(a)Sphere function

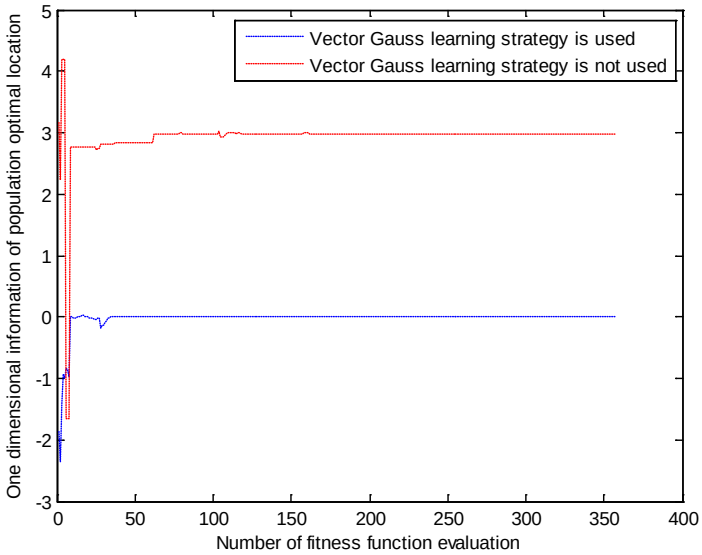


(b)Schwefel's P2.22 function

Fig. 1. Dimension information change curve on unimodal functions



(a)Rastrigin function



(b)Noncontinuous Rastrigin function

Fig. 2. Dimension information change curve on multimodal functions

3.2 Equation of VGL-PSO

Balancing global exploration and the local exploitation ability requires us to dynamically adjust the variance of vector Gaussian learning, i.e., the greater the value of σ^2, the greater the search space and the stronger the global exploration ability in the early stage of evolution. The smaller the value of σ^2, the smaller the search space and the stronger the local exploitation ability in the later stage of evolution.

The dimension of vector Gaussian learning is $num(1 \le num \le M)$, and the chosen values of the dimensions are $j_1, j_2, \cdots, j_{num}$, respectively, $1 \le j_1$ and $j_{num} \le M$. The definition of the variance adjustment strategy is as follows.

$$gBest_{\min} = \min(|gBest_{j_1}|, |gBest_{j_2}|, \cdots |gBest_{j_{num}}|) \tag{7}$$

$$\sigma^2 = gBest_{\min} * r_3 \tag{8}$$

where, $\min(\,)$ is the minimum function, r_3 is a random number between 0 and 0.5.

Meanwhile, we also adjust the inertia weight w. If the algorithm converges prematurely, w decreases with an increase in the number of iterative cycles of the algorithm; otherwise, the value of w is kept constant.

$$w = \begin{cases} 0.5 & , \quad tag \le limit \\ 1.2 - (1.2 - 0.02) * i / iterNum & , \quad otherwise \end{cases} \tag{9}$$

where, w is an inertia weight, tag signifies the number of optimal locations of the population that are not updated, and $limit$ represents the threshold of premature state.

3.3 Steps of VGL-PSO

1) Initialize the parameters, including c_1, c_2, w, tag, and $limit$.

2) Calculate and evaluate the fitness value.

3) If $tag \le limit$, we regard the evolution state as a normal state, proceed to step (4); otherwise, regard it as a premature state, proceed to step (5).

4) Adopt the fixed inertia weight, update velocity and location by formula (1) and (2), and update the individual optimal location and global optimum information. Judge whether the global optimum is updated; if updated, set $tag = 0$, otherwise, $tag++$ then proceed to step (6).

5) Use vector Gaussian learning for elite particles by formula (4), (5), (6), (7) and (8), and update particle by (9), (1) and (2); meanwhile, update the optimal location of individual and global optima. Judge whether the global optimum is updated, if updated, set $tag = 0$, otherwise, $tag++$, then proceed to step (6).

6) If the algorithm satisfies the ending condition, output the global optimum $GBest$ and fitness value; otherwise, please return to step (3) and continue.

4. Experiments

4.1 Benchmark Functions

The performance of VGL-PSO was verified by selecting eight benchmark functions as follows, where $f_1 \sim f_4$ are unimodal functions with only one extreme point in the fixed search range to test the convergence velocity and precision, and functions $f_5 \sim f_8$ are multimodal functions with many extreme points to adopt to verify the global search ability and the ability to escape from a local optimum. These functions are shown as follows.

1) Sphere function (value space: $[-100,100]^M$, optimum: 0)

$$f_1(x) = \sum_{i=1}^{M} x_i^2$$

2) Schwefel's P2.22 function (value space: $[-10,10]^M$, optimum: 0)

$$f_2(x) = \sum_{i=1}^{M} |x_i| + \prod_{i=1}^{M} |x_i|$$

3) Quadric function(value space: $[-100,100]^M$, optimum: 0)

$$f_3(x) = \sum_{i=1}^{M} (\sum_{j=1}^{i} x_j)^2$$

4) Quadric Noise function (value space: $[-1.28,1.28]^D$, optimum: 0) :

$$f_4(x) = \sum_{i=1}^{D} i \cdot x_i^4 + random[0,1)$$

5) Rastrigin function (value space:$[-5.12,5.12]^D$, optimum:0)

$$f_5(x) = \sum_{i=1}^{D} [x_i^2 - 10\cos(2\pi x_i) + 10]$$

6) Noncontinuous Rastrigin function (value space:$[-5.12,5.12]^D$, optimum:0)

$$f_6(x) = \sum_{i=1}^{D} [y_i^2 - 10\cos(2\pi y_i) + 10] \quad where \quad y_i = \begin{cases} x_i & |x_i| < 0.5 \\ round(2x_i)/2 & |x_i| \ge 0.5 \end{cases}$$

7) Ackley function (value space:$[-32,32]^D$, optimum:0)

$$f_7(x) = -20\exp(-0.2\sqrt{1/D\sum_{i=1}^{D} x_i^2}) - \exp(1/D\sum_{i=1}^{D}\cos 2\pi x_i) + 20 + e$$

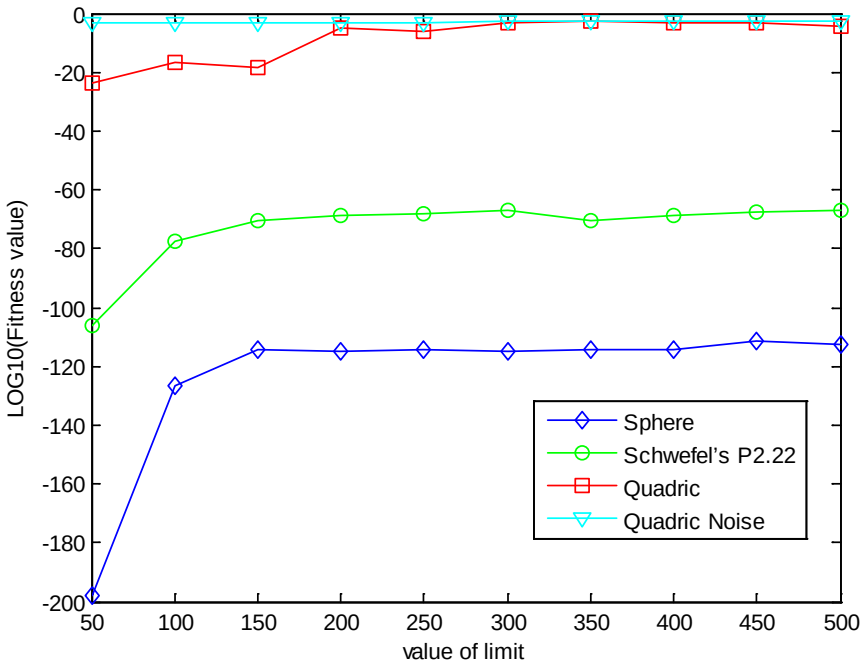
8) Generalized Penalized function (value space: $[-50,50]^D$, optimum: 0)

$$f_8(x) = \frac{\pi}{D}\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})] + (y_D-1)^2\} + \sum_{i=1}^{D} u(x_i,10,100,4)$$

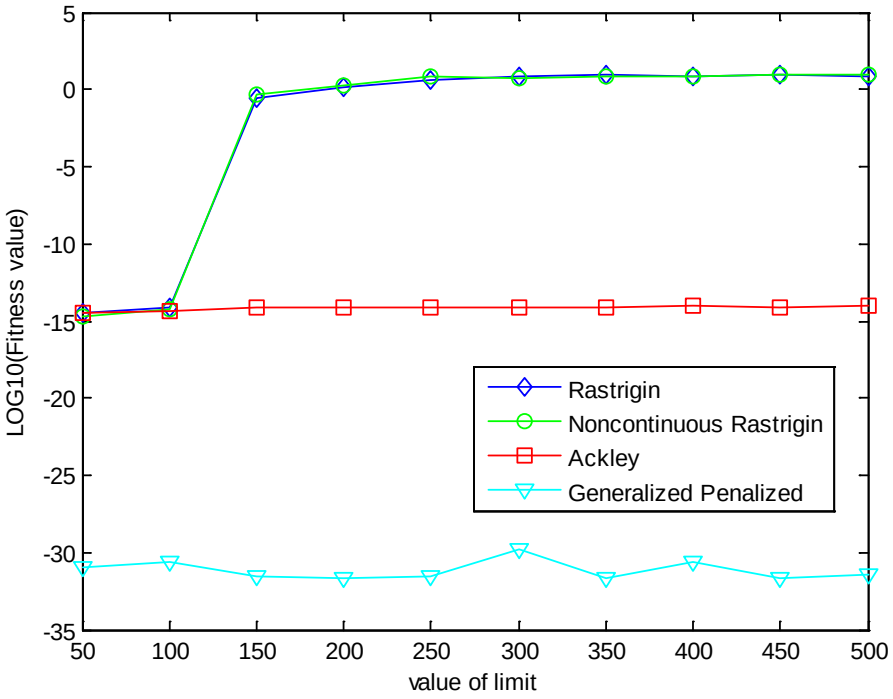
$$\text{where } y_i = 1+\frac{1}{4}(x_i+1) \quad u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m, & x_i > a \\ 0, & -a \le x_i \le a \\ k(-x_i-a)^m, & x_i < -a \end{cases}$$

4.2 Judgment of evolution state

The proposed algorithm judges the current evolution state according to the value of *limit*, adjusts the inertia weight adaptively, and chooses a different learning strategy according to the current state. Thus, the value of *limit* is important for the performance of the algorithm. If *limit* is too large, then the particle cannot escape from the local optimum; if it is too small, the particle is forced to converge to the local optimum, which affects the learning of the particle. Therefore, we choose different values for *limit*, and calculate the average optimal fitness value of eight classical test functions for 30 dimensions with 20 particles. The curves of the average optimal fitness values under different *limit* conditions are as follows.



(a) umimodel functions



(b) multimodel functions

Fig. 3. Curves of test functions with different limit

In addition to the Noise Quadric function, the average optimal fitness values of the other three functions are significantly different with the change of limit from **Fig. 3** (a). When the value of the limit is between 50 and 150, the logarithm of the average optimal fitness increases with an increase in the limit, and the convergence precision of the algorithm gradually decrease. When the value of the limit is greater than 150, the curve of the function exhibits almost no change, and the convergence precision of the algorithm has no obvious change. For the multimodal functions, only two functions are sensitive to changes in the limit. There is no obvious change in the average optimal fitness of the function when the limit is between 50 and 100.An increase in the value of the limit gradually reduces the average optimal fitness value of the function. The change curve of the function is smooth, and the convergence precision of the algorithm is not changed obviously with $limit > 150$. In summary, we set limit to be 50 to balance the performance of unimodal and multimodal functions.

4.3 Experiments

The performance of VGL-PSO was verified by comparingVGL-PSO with six classical PSO variants, namely FIPS [21], HPSO-TVAC [22], DMS-PSO [23], CLPSO [24], APSO [25], and GDPSO [15]. The size N of the swarm is 20, $c_1 = c_2 = 2.0$; the times of evaluation is 200000, $c_3 = c_4 = 1.0$, the number of estimation is 200,000, dimension $M = 30$, $limit = 50$.

The results of the seven algorithms are presented in **Table 1**, where “Mean” represents the mean of the optimal fitness value and “Std.Dev” signifies the standard deviation by averaging over 50 independent experiments. Here “Mean” reflects the precision and “Std.Dev” reflects the stability and robustness.

Table 1. Results of seven algorithms on 30 dimensional experiments

Test Functions		FIPS	HPSO-TVAC	DMSPSO	CLPSO	APSO	GDPSO	VGL-PSO
f_1	**Mean**	3.21e-30	3.38e-41	3.85e-54	1.89e-19	1.45e-150	1.12e-224	**5.39e-299**
	Std.Dev	3.60e-30	8.50e-41	1.75e-53	1.49e-19	5.73e-150	0	**0**
f_2	**Mean**	1.32e-17	6.90e-23	2.61e-29	1.01e-13	5.15e-84	7.90e-226	**5.16e-278**
	Std.Dev	7.86e-18	6.89e-23	6.60e-29	6.51e-14	1.44e-83	0.00	**0.00**
f_3	**Mean**	0.77	2.89e-07	47.5	395	1.0e-10	1.12e-01	**3.78e-26**
	Std.Dev	0.86	2.97e-07	56.4	142	2.13e-10	2.90	**1.11e-24**
f_4	**Mean**	2.55e-03	5.54e-02	1.10e-02	3.92e-03	4.66e-03	5.54e-03	**8.96e-04**
	Std.Dev	6.25e-04	2.08e-02	3.94e-03	1.14e-03	1.70e-03	3.40e-02	**5.02e-03**
f_5	**Mean**	29.98	2.39	28.1	2.57e-11	5.80e-15	**0.00**	**0.00**
	Std.Dev	10.92	3.71	6.42	6.64e-11	1.01e-14	**0.00**	**0.00**
f_6	**Mean**	35.91	1.83	32.8	0.167	4.14e-16	4.87e-01	**0.00**
	Std.Dev	9.49	2.65	6.49	0.379	1.45e-15	14.38	**0.00**
f_7	**Mean**	7.69e-15	2.06e-10	8.52e-15	2.01e-12	1.11e-14	**3.43e-15**	**3.43e-15**
	Std.Dev	9.33e-16	9.45e-10	1.79e-15	9.22e-13	3.55e-15	6.61e-15	**6.26e-15**
f_8	**Mean**	1.22e-31	7.07e-30	2.05e-32	1.59e-21	3.76e-31	2.33e-31	**1.15e-31**
	Std.Dev	4.85e-32	4.05e-30	8.12e-33	1.93e-21	**1.20e-30**	1.78e-30	2.45e-30

As shown in **Table 1**, we can see that quality and stability of our method is superior to those of the other variants. The convergence accuracy of our algorithm is much higher than that of the other algorithms on $f_1 \sim f_3$. The VGL-PSO algorithm also has a very good performance on f_5, f_6 and f_7. Especially, the proposed method can reach the global optimum on the f_5 and f_6 functions, which is difficult to achieve with the optimization algorithm. Although the performance of our method is worse than that of GDPSO, there is no significant difference between our method and GDPSO through the t test.

4.4 T test

We verifiedwhether the difference between VGL-PSO and the other six variants is significant by adopting the T test, in whichthe degree of freedom was 30, and the critical value was 1.697. If $t > 1.697$, VGL-PSO outperforms all the other variants, in which case it is marked as "+"; if $t < -1.697$, the performance of VGL-PSO is less accurate than that of the other variants, in which case it is marked as "-"; otherwise, VGL-PSO has no significant difference compared with the other variants, in which case it is marked as "=". In addition, "w/t/l" means that VGL-PSO wins in w functions, ties in t functions, and loses in l functions, compared with its competitors. The T test results of the comparison are presented in **Table 2**.

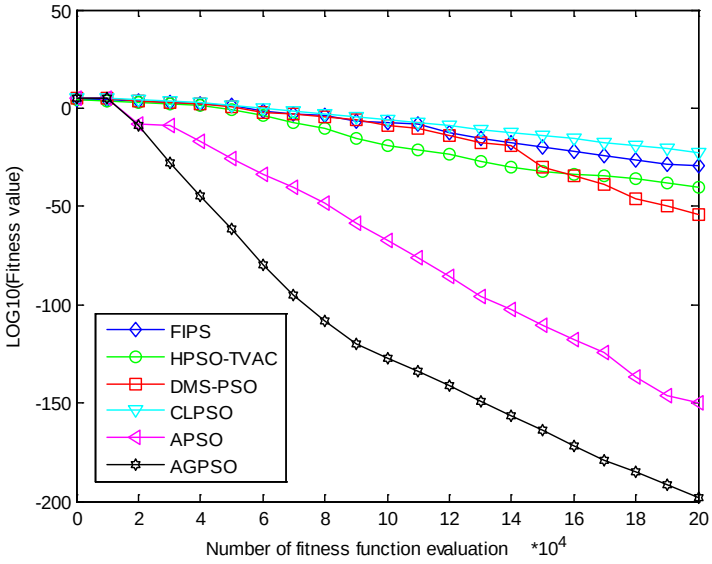
Table 2. The T test results of six algorithms

Test Functions	FIPS	HPSO-TVAC	DMS-PSO	CLPSO	APSO	GDPSO
f_1	+	+	=	+	=	=
f_2	+	+	+	+	+	=
f_3	+	+	+	+	+	+
f_4	+	+	+	+	+	+
f_5	+	+	+	+	+	=
f_6	+	+	+	+	=	+
f_7	+	=	+	+	+	=
f_8	=	+	=	+	=	=
w/t/l	7/1/0	7/1/0	6/2/0	8/0/0	5/3/0	3/5/0

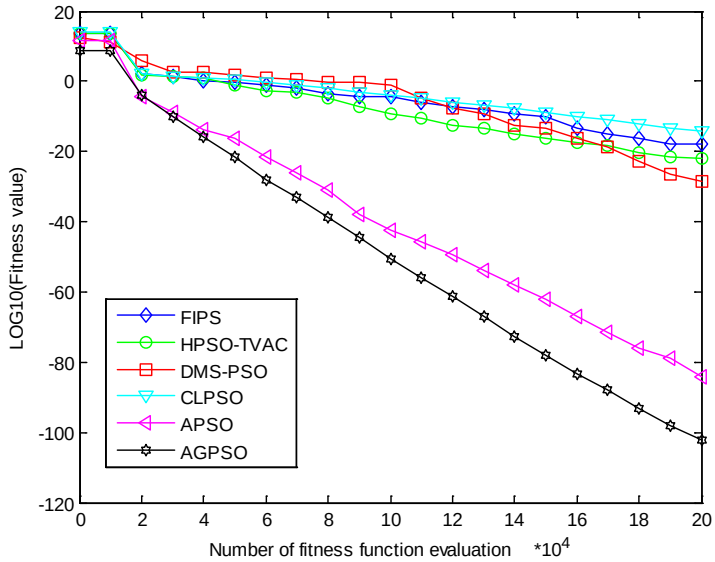
The results in **Table 2** indicate that our method outperforms CLPSO on all eight functions, whereas VGL-PSO achieves more accurate results than HPSO-TVAC on seven functions. VGL-PSO is more effective than DMS-PSO on six functions, whereas DMS-PSO is superior on the remaining two functions. APSO outperforms VGL-PSO on three functions, whereas VGL-PSO is the most effective on five functions. GDPSO outperforms VGL-PSO on five functions, whereas VGL-PSO achieves the most accurate result on the f_3, f_4 and f_6 functions.

4.5 Convergence of our method

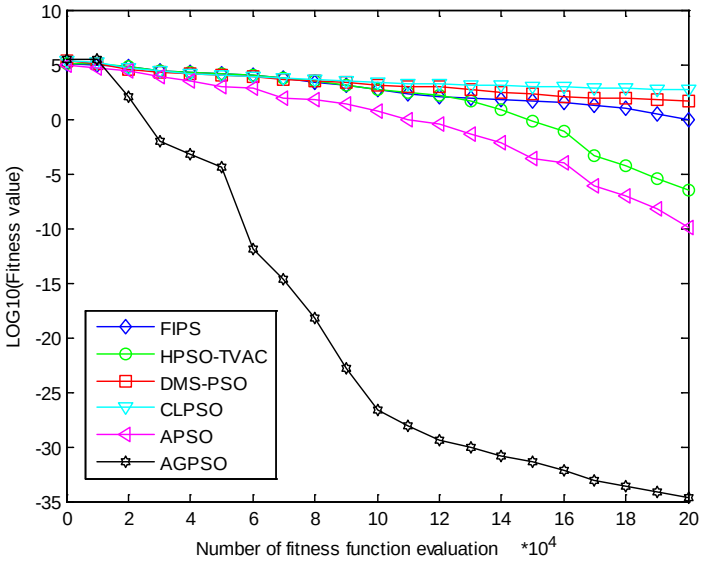
The curves of converge performance for the eight test functions over 30 dimensions are shown in **Fig. 4** and **5**. The horizontal and vertical axes represent the evaluation numbers and the logarithm of the fitness value, respectively. **Fig. 4** and **Fig. 5** are the curves of the unimodal functions and multimodal functions, respectively.



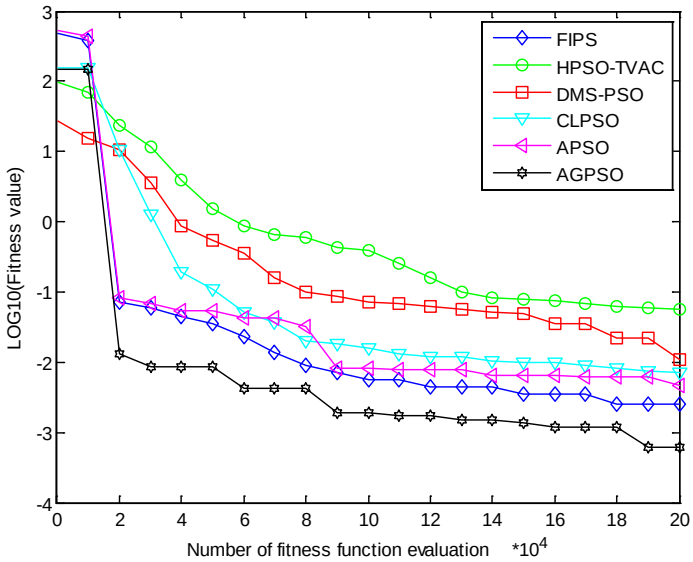
(a) f_1 function



(b) f_2 function

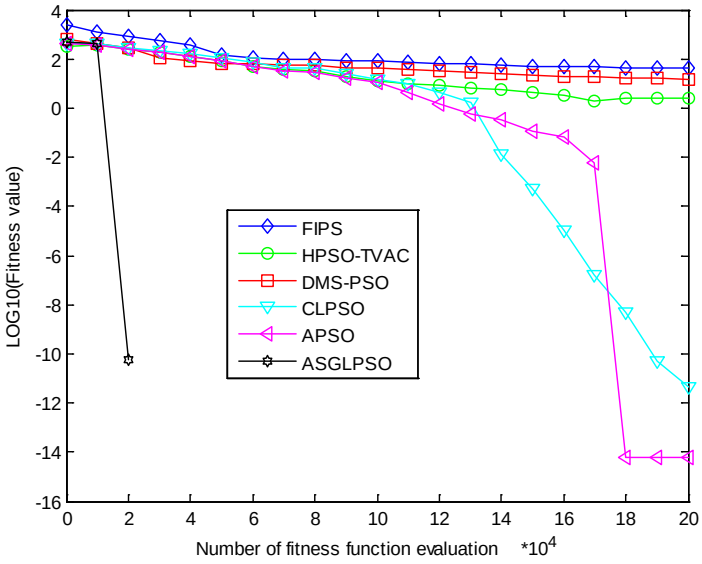


(c) f_3 function

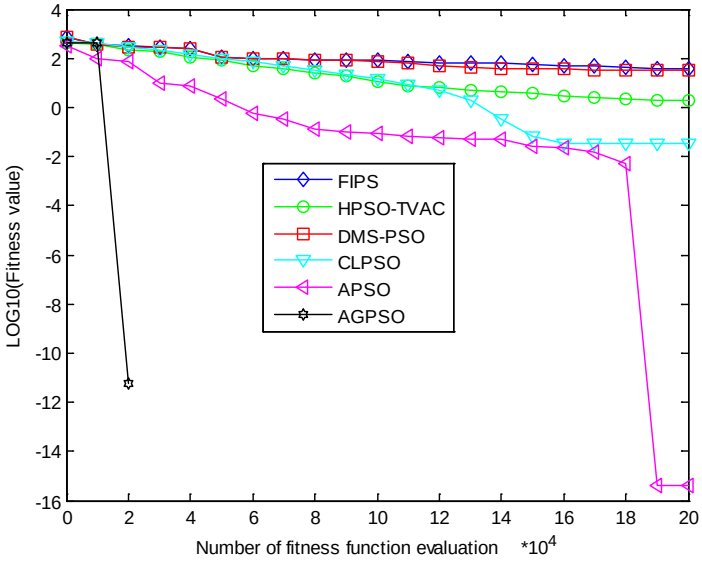


(d) f_4 function

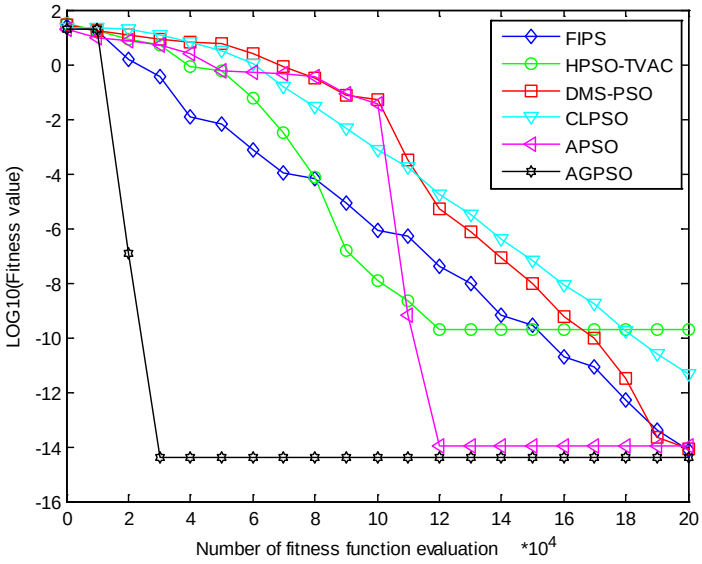
Fig. 4. Evolution curves of unimodal functions



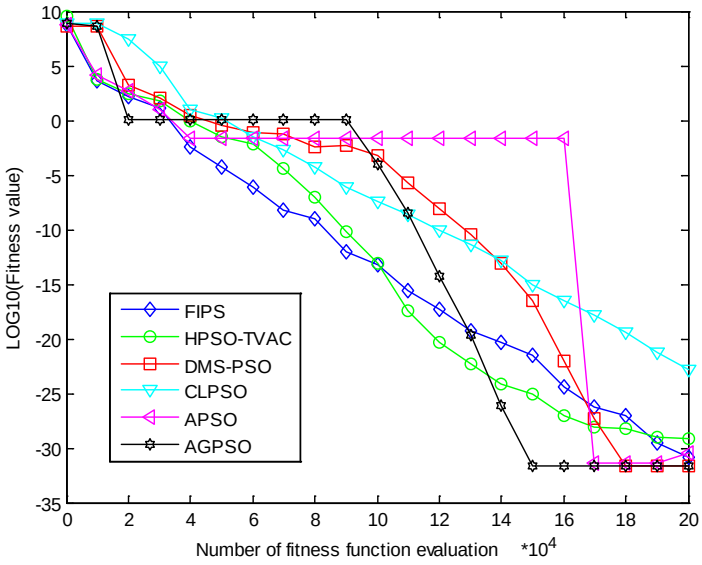
(a) f_5 function



(b) f_6 function



(c) f_7 function



(d) f_8 function

Fig. 5. Evolution curves of multimodal functions

VGL-PSO with vector Gaussian learning can enhance the ability of the algorithm to escape from a local optimum and accelerate the convergence speed, as shown in **Fig. 4** and 5. The performance of VGL-PSO is superior on unimodal functions, i.e., most notably the evolution curves of functions f_1, f_2, and f_3 decrease almost linearly in **Fig. 4**; for multimodal functions, our method performs more accurately than other variants, especially on the f_5 and f_6 functions. VGL-PSO can find the best location when the number of assessments is 20000; the other variants readily converge to a local optimum, resulting in slow or even stagnant convergence, as seen in **Fig. 5**.

5. Conclusions

Particle swarm optimization based on vector Gaussian learning is proposed to overcome the limitations of PSO and Gaussian learning strategy. The proposed method uses the threshold value *limit* to judge whether the swarm converges to a local optimum. If the algorithm is found to converge toa local optimum, we adopt the inertia weight strategy,which decreases linearly, and the vector Gaussian learning strategy, in which case the elite particlesare able to escape from the local optimum; otherwise, we adopt the fixed inertia weight strategy. The results demonstrate that VGL-PSO outperforms IPS, HPSO-TVAC, DMS-PSO, CLPSO, APSO, and GDPSO. The result of the T test shows that the VGL-PSO algorithm performs more accurately than the other six variants. Our future workintends focusing on optimizing the vector Gaussian learning strategy, adjusting the evolutionary state threshold *limit* more reasonably, and improving the performance of the algorithm [26-29].

References

[1] Kennedy J. and Berhart R., "Particle swarm optimization", in *Proc. of IEEE Conf. on Neural Networks*, pp.1942-1948, November 27- December 1, 1995. Article (CrossRef Link)

[2] Davoodi Elnaz, Hagh Mehrdad Tarafdar and Zadeh Saeid Ghassem, "A hybrid Improved Quantum-behaved Particle Swarm Optimization-Simplex method (IQPSOS) to solve power system load flow problems," *Applied Soft Computing*, vol. 21, no. 1, pp.171-179, August, 2014. Article (CrossRef Link)

[3] Jun Yang, Hesheng Zhang, Yun Ling, Cheng Pan and Wei Sun, "Task Allocation for Wireless Sensor Network Using Modified Binary Particle Swarm Optimization," *IEEE Sensors Journal*, vol. 14, no. 3, pp. 882-892, November, 2014. Article (CrossRef Link)

[4] Amir Ameli, Shahab Bahrami, Farid Khazaeli and Mahmood-Reza Haghifam, "A Multiobjective Particle Swarm Optimization for Sizing and Placement of DGs from DG Owner's and Distribution Company's Viewpoints," *IEEE Transactions on Power Delivery*, vol. 29, no. 4, pp. 1831-1840, February, 2014. Article (CrossRef Link)

[5] J. Senthilnath, S. N. Omkar, V. Mani and T. Karthikeyan, "Multiobjective Discrete Particle Swarm Optimization for Multisensor Image Alignment," *IEEE Geoscience and Remote Sensing Letters*, vol. 10, no. 5, pp. 1095-1099, February, 2013. Article (CrossRef Link)

[6] Wang Huibin, Chen Zhe, Wang Xin and Ma Yu, "Random finite sets based UPF-CPHD multi-object tracking," *Journal on Communications*, vol. 33, no. 12, pp. 147-153, December, 2012. Article (CrossRef Link)

[7] Han Huang, Hu Qin, Zhifeng Hao and Andrew Lim, "Example-based learning particle swarm optimization for continuous optimization," *Information Sciences*, vol. 182, no. 1, pp. 125-128, January, 2012. Article (CrossRef Link)

[8] Wei-Neng Chen, Jun Zhang, Ying Lin, Ni Chen, Zhi-Hui Zhan, Henry Shu-Hung Chung, Yun Li and Yu-Hui Shi, "Particle swarm optimization with an aging leader and challengers," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 2, pp. 241-258, February, 2013. Article (CrossRef Link)

[9] Zahra Beheshti and Siti Mariyam Hj, Shamsuddin, "CAPSO: Centripetal accelerated particle swarm optimization," *Information Sciences*, vol. 258, pp. 54-79, February, 2014. Article (CrossRef Link)

[10] Yong Zhang, Dun-wei, GongXiao-yan and SunNa Geng, "Adaptive bare-bones particle swarm optimization algorithm and its convergence analysis," *Soft Computing*, vol. 18, no. 7, pp. 1337-1352, July, 2014. Article (CrossRef Link)

[11] Wang Lin, Yang Bo and Orchard Jeff, "Particle swarm optimization using dynamic tournament topology," *Applied Soft Computing*, vol. 48, pp. 584-596, November, 2016. Article (CrossRef Link)

[12] Ran Cheng and Yaochu Jin, "A social learning particle swarm optimization algorithm for scalable optimization," *Information Sciences*, vol. 291, no. 6, pp. 43-60, January, 2015. Article (CrossRef Link)
[13] Yang Ning, Huo Ju and Yang Ming, "Multi-objective particle swarm optimization algorithm based on the interaction of multi-level information," *Control and Decision*, vol. 31, no. 5, pp. 907-912, May, 2015. Article (CrossRef Link)
[14] Couceiro M and Sivasundaram S, "Novel fractional order particle swarm optimization," *Applied Mathematics and Computation*, vol. 283, pp. 36-54, June, 2016. Article (CrossRef Link)
[15] Zhu Degang, Sun Hui, Zhao Jia and Yu Qing, "Particle swarm optimization algorithm based on Gaussian disturbance," *Journal of Computer Applications*, vol. 34, no. 3, pp. 754-759, March, 2014. Article (CrossRef Link)
[16] Duan Haibin and Li Junnan, "Gaussian Harmony Search Algorithm: A Novel Method for Loney's Solenoid Problem," *IEEE Transactions on Magnetics*, vol. 50, no. 3, pp. 83-87, March, 2014. Article (CrossRef Link)
[17] Zhou Xinyu, Wu Zhijian, Wang Hui and Rahnamayan Shahryar, "Gaussian bare-bones artificial bee colony algorithm," Soft Computing, vol. 20, no. 3, pp.907-924, March, 2016. Article (CrossRef Link)
[18] Zhang Le, Liu Zhong, Zhang Jianqiang and Ren Xiongwei, "Optimized improved Gaussian process model based on artificial bee colony algorithm," *Journal of National University of Defense Technology*, vol. 36, no. 1, pp. 154-160, January, 2014. Article (CrossRef Link)
[19] Liu Xiaolong, Li Rongjun and Yang Ping, "Bacterial Foraging optimization algorithm based on estimation of distribution," *Control and Decision*, vol. 26, no. 8, pp. 1233-1238, August, 2011. Article (CrossRef Link)
[20] Lü Li, "Vector Learning Particle Swarm Optimization," *International Journal of Advancements in Computing Technology*, vol. 4, no. 17, pp. 106-115, December, 2012. Article (CrossRef Link)
[21] Mendes R., Kennedy J. and Neves J., "The fully informed particle swarm: Simpler, maybe better," *IEEE Transaction on Evolutionary Computation*, vol. 8, no. 3, pp.204-210, March, 2004. Article (CrossRef Link)
[22] Rathaweera A., Halgamuge S. and Watson H., "Self-organizing hierarchical particle swarm optimizer with time-varying acceleration coefficients," *IEEE Transaction on Evolutionary Computation*, vol. 8, no. 3, pp. 240-255, March, 2004. Article (CrossRef Link)
[23] Liang J. J. and Suganthan P. N., "Dynamic multi-swarm particle swarm optimizer," in *Proc. of IEEE conf. on Swarm Intelligence Symposium*, pp. 124-129, June 8-10, 2005. Article (CrossRef Link)
[24] Liang J. J., Qin A. K., Suganthan P N and S. Baskar, "Comprehensive learning particle swarm optimizer for global optimization of multimodal functions," *IEEE Transaction on Evolutionary Computation*, vol.10, no. 3, pp.281-295, March, 2006. Article (CrossRef Link)
[25] Zhi-hui Zhan, Jun Zhang, Yun Li and Henry Shu-Hung Chung, "Adaptive Particle Swarm Optimization," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 39, no. 6, pp. 1362-1381, June, 2009. Article (CrossRef Link)
[26] Xiao R-b, Zhang Y-f, Huang Z-d, "Emergent computation of complex systems: a comprehensive review," *International Journal of Bio-Inspired Computation*, vol.7, no. 2, pp. 75-97, March, 2015. Article (CrossRef Link)
[27] Cai X, Wang L, Kang Q and Wu Q-d, "Bat algorithm with Gaussian walk," *International Journal of Bio-Inspired Computation*, vol. 6, no. 3, pp. 166-174, May, 2014. Article (CrossRef Link)
[28] Wang G-G, Deb S, Gao X-Z and Coelho LdS, "A new metaheuristic optimization algorithm motivated by elephant herding behavior," *International Journal of Bio-Inspired Computation*, vol. 8, no. 6, pp. 394-409, October, 2016. Article (CrossRef Link)
[29] Cai X, Gao X-z and Xue Y, "Improved bat algorithm with optimal forage strategy and random disturbance strategy," *International Journal of Bio-Inspired Computation*, vol. 8, no. 4, pp. 205-214, June, 2016. Article (CrossRef Link)

Zhao Jia received the M.A. degree in technology of computer application form Nanchang Hangkong University, Nanchang, China. He has been an assistant professor in the School of Information Engineering, Nanchang Institute of Technology, since 2011. His research interests include evolutionary computation, swarm intelligence, wireless sensor networks, Big Date processing.

Lv Li is currently pursuing her PhD with the College of Computer and Information Engineering, Hohai University, Nanjing, China and received her MA in Computer Architecture from Jiangxi Normal University, Nanchang, China. She has been an Associate Professor in the School of Information Engineering, Nanchang Institute of Technology, since 2013. Her research interests include evolutionary computation, object tracking, wireless sensor networks and embedded system.

Wang Hui received his B.Sc. and M.Sc. from School of Computer, China University of Geosciences (Wuhan) in 2005 and 2008, respectively. He completed his Ph.D. in the State key Lab of Software Engineering at Wuhan University, Wuhan, China in June 2011. Now, he is an associate professor at the School of Information Engineering, Nanchang Institute of Technology, China. His current research interests focus on evolutionary optimization, swarm intelligence, large-scale optimization, GPU-based computing, evolutionary learning and applications.

Sun Hui received his M.Sc. in Mechanics from the Tsinghua University, Beijing, China in 1988, and PhD in Metal Plastic Working from the Nanchang University in 2002. He is currently a Professor with Nanchang Institute of Technology, Nanchang, China. His research interests include intelligent algorithms, rough sets and granular computing and variational inequality principle.

Wu Runxiu is currently a Professor with Nanchang Institute of Technology, Nanchang, China. His research interests include swarm intelligence optimization algorithm and its applications.

Nie Jugen received the PhD of Communication and Information system in 2012 from Wuhan University, Wuhan, China. He is currently associate professor in School of Information Engineering, Nanchang Institute of Technology. His research interests include System on Wireless Sensor Network, Internet of Things and UnderWater Acoustic Networks.

Xie Zhifeng is currently pursuing his M.A. degree from Nanchang Institute of Technology, Nanchang, China. His research interests include evolutionary computation, swarm intelligence, wireless sensor networks, Big Date processing.