Applied Mathematics and Computation 218 (2011) 3763–3775

Contents lists available at SciVerse ScienceDirect

Applied Mathematics and Computation

journal homepage: www.elsevier.com/locate/amc

Quantum-behaved particle swarm optimization with Gaussian distributed local attractor point

Jun Sun [a,*], Wei Fang [a,*], Vasile Palade [b], Xiaojun Wu [a], Wenbo Xu [a]

[a] *Key Laboratory of Advanced Process Control for Light Industry (Ministry of Education), Department of Computer Science and Technology, Jiangnan University, No. 1800, Lihu Avenue, Wuxi, Jiangsu 214122, China*
[b] *Department of Computer Science, University of Oxford, Parks Road, Oxford, OX1 3QD, United Kingdom*

ARTICLE INFO

Keywords:
Particle swarm optimization
Gaussian probability distribution
Swarm intelligence
Quantum behavior

ABSTRACT

This paper proposes a novel variant of quantum-behaved particle swarm optimization (QPSO) algorithm with the local attractor point subject to a Gaussian probability distribution (GAQPSO). The local attractor point in QPSO plays an important in that determining the convergence behavior of an individual particle. As such, the mean value and standard deviation of the proposed Gaussian probability distribution in GAQPSO are carefully selected. The distributions and diversities of the local attractor points in GAQPSO and QPSO are evaluated and compared. For the purpose of comparison, two variants of the GAQPSO algorithm are proposed by using a mutation probability and other types of probability distribution. The GAQPSO has been comprehensively evaluated on the suite of CEC2005 benchmark functions, and the experimental results are compared with those of the PSO and QPSO algorithms based on different probability distributions. It is shown by the results that the GAQPSO algorithm is an effective approach that can improve the QPSO performance considerably, that is, the GAQPSO algorithm is less likely to be stuck in local optima and hence it can achieve better solutions in most cases.

1. Introduction

Particle swarm optimization (PSO) is a population-based stochastic optimization algorithm based on the metaphor of social interaction and communication such as bird flocking and fish schooling [1,2]. PSO algorithms can be easily implemented and is computationally inexpensive, having to adjust only a small number of parameters. In PSO, the particles, representing the potential solutions, move around in a multidimensional search space with a velocity constantly updated by the particle's own experience and the experience of the particle's neighbors or the experience of the whole swarm. PSO shares many similarities with evolutionary algorithms, and has been proven to have robust performance over a variety of difficult optimization problems [3].

In the past a few years, many improved PSO algorithms have been proposed [4–11]. Different kinds of probability distributions to generate random numbers have been used in PSO algorithm, such as Gaussian, Cauchy, Levy and exponential distributions. In [12], Kennedy replaced the particle update rule with sampling from a Gaussian distribution with the mean being the centroid of the personal and neighborhood best positions of each particle, and the standard

* Corresponding authors.
E-mail addresses: sunjun_wx@hotmail.com (J. Sun), wxfangwei@hotmail.com (W. Fang).

0096-3003/$ - see front matter © 2011 Elsevier Inc. All rights reserved.
doi:10.1016/j.amc.2011.09.021

deviation being the gap between them. In [13,14], Kennedy used double-exponential distribution and other versions of Gaussian probability distribution in PSO to achieve good results. Richer and Blackwell pointed out that the tails of Gaussian distribution function are too thin to enable escape from stagnation and then proposed the Levy distribution in PSO to improve performance of the bare bones PSO so that it becomes effectively equivalent to standard PSO [15]. In [16–19], Krohling and Coelho has studied the influence of different probability distributions to generate the weighting coefficients of PSO and they found that Gaussian, Cauchy, and the exponential probability distribution could improve the performance of the canonical PSO. In [20], Krohling proposed the jump strategy, which is implemented based on the Gaussian or the Cauchy probability distribution in the bare bone PSO. Higashi and Iba designed the Gaussian mutation in PSO to improve the search ability [21].

Recently, inspired by quantum mechanics and trajectory analysis of PSO [22], Sun et al. used a strategy based on a quantum δ potential well model to sample around the previous best points[23], and later introduced the mean best position into the algorithm and proposed a new version of PSO, quantum-behaved particle swarm optimization (QPSO)[24,25]. The iterative equation of QPSO is very different from that of PSO. Besides, unlike PSO, QPSO needs no velocity vectors for particles, and also has fewer parameters to adjust, making it easier to implement. The QPSO algorithm has been shown to successfully solve a wide range of continuous optimization problems and many efficient strategies have been proposed to improve the algorithm [26–30]. Since QPSO was proposed, the Gaussian and Cauchy probability distribution have been imported to generate the random numbers in order to avoid premature convergence. In [31–33], the random sequences in QPSO were generated using the absolute value of the Gaussian probability distribution with zero mean and unit variance. Based on the characteristic of QPSO, the variables of the global best and mean best positions are mutated with Cauchy distribution in [34], and an adaptive QPSO version was proposed in [35]. A set of different mutation operations on the personal best positions of the particles in QPSO were studied in [36]. Most of these mutation operations were executed on the global best position, the mean best position or the personal best positions. Some of the operations were used to change the sampling methods of random numbers in QPSO algorithm. Few interests were focused on the local attractor point of the QPSO algorithm. Therefore, this paper concentrates on the mutation operator for the local attractor point and proposed a QPSO with Gaussian distributed attractor (GAQPSO). The GAQPSO algorithm, along with QPSO and other variants of PSO, is tested on a set of CEC2005 benchmark functions. The proposed method outperforms its competitors in most cases, as shown by the experimental results.

The rest of the paper is organized as follows. Sections 2 and 3 describe PSO and QPSO, respectively. Section 4 presents the idea of Gaussian distribution on the local attractor point, and an analysis of distributions and the diversities of the points are provided, followed by the proposed GAQPSO algorithm and two variants of it. Section 5 experimentally compares the GAQPSO algorithm with various existing PSO algorithms and some variants of the GAQPSO using a set of benchmark functions. Finally, general conclusions are drawn in Section 6.

2. Particle swarm optimization

In the original PSO with M particles, each particle is represented as a potential solution to a problem in a D-dimensional space and its position at the tth iteration is denoted as $X_i^t = \left[X_{i,1}^t, \ldots, X_{i,j}^t, \ldots, X_{i,D}^t\right]$. Each particle remembers its own previous best position and its velocity along each dimension as $V_i^t = [V_{i,1}^t, \ldots, V_{i,j}^t, \ldots, V_{i,D}^t]$. The velocity and position of particle i at $(t+1)$th iteration are updated by the following equations:

$$V_{i,j}^{t+1} = wV_{i,j}^t + c_1 r_{i,j}^t \left(P_{ij}^t - x_{ij}^t\right) + c_2 R_{i,j}^t \left(G_j^t - X_{i,j}^t\right) \tag{1}$$

$$X_{i,j}^{t+1} = X_{i,j}^t + V_{i,j}^{t+1} \tag{2}$$

where c_1 and c_2 are two positive constants, known as the acceleration coefficients; $r_{i,j}^t$ and $R_{i,j}^t$ are two uniformly distributed random numbers on the range (0,1) for the jth dimension of particle i. Vector $P_i^t = \left[P_{i,1}^t, \ldots, P_{i,j}^t, \ldots P_{i,D}^t\right]$ is the position with the best fitness found so far for the ith particle, which is called personal best (*pbest*) position. And vector $G^t = \left[G_1^t, \ldots, G_j^t, \ldots G_D^t\right]$ records the best position discovered by the swarm so far, known as the global best (*gbest*) position. $X_{i,j}^t, V_{i,j}^t$ and $P_{i,j}^t$ are the jth dimension of vector of X_i^t, V_i^t and P_i^t, respectively. The parameter w is the inertia weight used for the balance between the global and local search abilities [2]. Usually w decreases linearly with the iteration generations as:

$$w = w_{\max} - t \cdot (w_{\max} - w_{\min})/T, \tag{3}$$

where w_{max} and w_{min} are the maximum and minimum weights and usually set to 0.9 and 0.4, respectively [2]. T is a predefined maximum number of iterations, and t represents the number of current iteration. Let f be the objective function to be minimized. The PSO algorithm can be described by the following pseudocode.

Pseudocode of the PSO algorithm:

```
Randomly generate an initial population with positions and velocities
Repeat
    For $i = 1$ to population size do
      if $f(X_i) < f(P_i)$ then $P_i = X_i$;
      $G = \arg\min(f(P_i))$;
      For $j = 1$ to $D$ do
        Velocity update with Eq. (1);
        Position update with Eq. (2);
      End//end for loop $j$
    End//end for loop $i$
Until termination criterion is met.
```

3. Quantum-behaved particle swarm optimization (QPSO)

Trajectory analyses in [22] demonstrated the fact that convergence of PSO algorithm may be achieved if each particle converges to its local attractor $p_i^t = \left[p_{i,1}^t, p_{i,2}^t, \ldots p_{i,j}^t, \ldots, p_{i,D}^t\right]$ with coordinates

$$p_{i,j}^t = \frac{c_1 r_{i,j}^t P_{ij}^t + c_2 R_{i,j}^t G_j^t}{c_1 r_{i,j}^t + c_2 R_{i,j}^t}, \tag{4}$$

or

$$p_{i,j}^t = \varphi P_{i,j}^t + (1-\varphi) G_j^t, \tag{5}$$

where

$$\varphi = \frac{c_1 r_{i,j}^t}{c_1 r_{1j}^t + c_2 R_{i,j}^t}. \tag{6}$$

Assume that a PSO system is a quantum system, and each particle has a quantum behavior with its quantum state formulated by a wave functionψ. $|\psi|^2$ is the probability density function of the position of the particle. Inspired by analysis of convergence of the traditional PSO in [22], we further assume that, at iteration t, particle i moves in D-dimensional space with a δ potential well centered at $p_{i,j}^t$ on the jth dimension. Correspondingly, the wave function at iteration $t+1$ is

$$\psi(X_{i,j}^{t+1}) = \frac{1}{\sqrt{L_{ij}^t}} \exp\left(-\left|X_{i,j}^t - p_{i,j}^t\right| / L_{i,j}^t\right), \tag{7}$$

where L_{ij}^t is the standard deviation of the double exponential distribution, varying with iteration number t. Hence, the probability density function Q is a double exponential distribution as follows

$$Q(X_{i,j}^{t+1}) = \left|\psi\left(X_{i,j}^{t+1}\right)\right|^2 = \frac{1}{L_{i,j}^t} \exp\left(-2\left|X_{i,j}^{t+1} - p_{i,j}^t\right| / L_{i,j}^t\right) \tag{8}$$

and thus the probability distribution function F is

$$F\left(X_{i,j}^{t+1}\right) = 1 - \exp\left(-2\left|X_{i,j}^{t+1} - p_{i,j}^t\right| / L_{i,j}^t\right). \tag{9}$$

Using Monte Carlo method, we can obtain the jth component of position X_i at iteration $t+1$ as:

$$X_{i,j}^{t+1} = p_{i,j}^t \pm \frac{1}{2} L_{i,j}^t \ln\left(1/u_{i,j}^{t+1}\right), \tag{10}$$

where $u_{i,j}^{t+1}$ is a random number uniformly distributed over $(0,1)$. The value of $L_{i,j}^t$ is calculated as:

$$L_{i,j}^t = 2\alpha\left|C_j^t - X_{i,j}^t\right|, \tag{11}$$

where C^t is known as the mean best (*mbest*) position defined as the mean of the *pbest* positions of all particles. That is

$$C^t = (C_1^t, C_2^t, \ldots, C_D^t) = \left(\frac{1}{M}\sum_{i=1}^M P_{i,1}^t, \frac{1}{M}\sum_{i=1}^M P_{i,2}^t, \ldots, \frac{1}{M}\sum_{i=1}^M P_{i,j}^t, \ldots, \frac{1}{M}\sum_{i=1}^M P_{i,D}^t\right), \tag{12}$$

where M is the population size and P_i^t is the personal best position of particle i. Hence, the position of the particle updates according to the following equation:

$$X_{i,j}^{t+1} = p_{i,j}^t \pm \alpha\left|C_j^t - X_{i,j}^t\right| \ln\left(1/u_{i,j}^{t+1}\right), \tag{13}$$

where parameter α is known as the contraction–expansion (CE) coefficient, which can be tuned to control the convergence speed of the algorithms. Generally, we call the PSO with Eq. (13) quantum-behaved particle swarm Optimization (QPSO), where parameter α must be set as $\alpha < 1.781$ to guarantee convergence of the particle [37]. When using QPSO in practical applications, the CE coefficient should be properly controlled. Generally, there are two methods to control this parameter. One is fixing the value of α during the search process. In [37], it is shown that setting α to be a number in the (0.5, 0.8) interval can generate satisfactory results for most benchmark functions. Particularly, when $\alpha = 0.75$, QPSO can obtain good performance in general. However, the method fixing α is sensitive to population size and maximum number of iterations. If these two parameters are changed, the value of α to get desirable algorithmic performance may be very different. The method that can surmount this problem is to use a time-varying CE coefficient. In [37], it is suggested that decreasing the value of α linearly from α_1 to $\alpha_0(\alpha_0 < \alpha_1)$ in the course of the search process is a simple but effective way. In this approach, the value of α is computed by:

$$\alpha = \alpha_0 + (T - t) \cdot (\alpha_1 - \alpha_0)/T, \tag{14}$$

where α_1 and α_0 are the final and initial values of α, respectively, T is the maximum number of iterations, t is the current iteration number. In [37], it is suggested that setting α_1 larger than 0.8 and smaller than 1.2 while α_0 smaller than 0.6 can generate the acceptable algorithmic performance. In most of the papers in the literature on QPSO and in [37], it recommended that linearly decreasing α_0 from 1.0 to 0.5 can lead the QPSO algorithm to good performance in general.

QPSO has some characteristics that are different from those of PSO. First of all, the introduced exponential distribution of positions makes QPSO global convergent. Secondly, the introduction of the mean best position into QPSO is another improvement of QPSO. In original PSO, each particle converges to the global best position independently. On the other hand, in the QPSO with mean best position, each particle cannot converge to global best position without considering its colleagues. It is indicated by Eq. (13) that the distance between particle's current position and the *mbest* position determines the position distribution of the particle for the next iteration. If the personal best positions of several particles are far from the global best position G^t (these particle called lagged particles) while those of the other particles are near the global best position, the *mbest* position may be pulled away from G^t by lagged particles. When the lagged particles are chasing after their colleagues, say converging to G^t, the *mbest* position will be approaching G^t slowly. The distances between the *mbest* position and the personal best positions of particles near G^t do not decrease quickly, which can decelerate the convergence of the particles near G^t and make them explore globally around G^t until the lagged ones are close to G^t. Therefore, in the QPSO with mean best position, the particle swarm does never abandon any lagged particle and thus seems to be more intelligent and more cooperative social organism. In a word, the wait among particles enhances the global search ability of QPSO greatly.

Fig. 1 illustrates the waiting phenomena among the particles in QSPO. In Fig. 1, the big circle with black background represents the particle with the global position, the little circles with white background and black dots represent the other particles, and the little circles with white background and vertical lines represents the lagged particles; the arrows around the little circles represent the possible directions of the particles; the big arrowhead points to the direction in which the particle moves with high probability. In the PSO algorithm, each particle converges to the global best position independently without waiting its colleagues so that the particle flies in the direction toward the global best position with a high probability, as shown by Fig. 1(a). The influence of the lagged particles on the other particles is very little since the only linkage among the particles is the global best position. If the lagged particles fail to find out positions better than the current *gbest* position during a certain number of iterations, their influence on the other particle is null. On the other hand, in QPSO, the lagged particles exert greater influence on the other particles through the *mbest* position except for the *gbest* position, and thus the particles around the *gbest* position may fly in any direction, as shown by Fig. 1(b). However, they move in the direction toward the *gbest* position with higher probability since the overall tendency of the particles' movements is convergence to the *gbest* position.

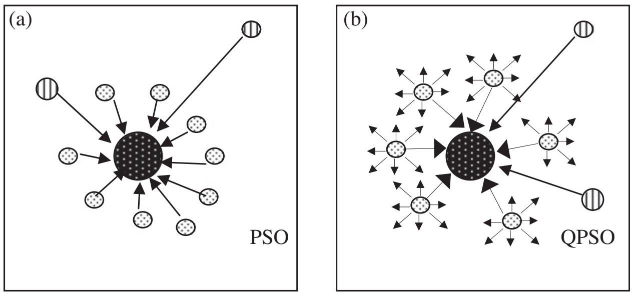

Fig. 1. The movements of particles in PSO and QPSO; (a) PSO (b) QPSO.

The procedure for implementing the QPSO is given by the following pseudo-codes:

```
Initialize the population size ($M$), the positions and the dimensions of the particles;
**For** $t = 1$ to Maximum Iteration $T$
  Compute the mean best position $C$ by Eq. (12);
  $\alpha = (\alpha_1 - \alpha_0) \cdot (T - t)/T + \alpha_0$;
  **For** $i = 1$ to population size $M$
    **If** $f(X_i) < f(P_i)$ then $P_i = X_i$; **Endif**
    $G = \mathrm{argmin}\ (f(P_i))$;;
    **For** $j = 1$ to $D$
      $\varphi = \mathrm{rand}\ (0,1)$; $u = \mathrm{rand}\ (0,1)$;
      $p_{ij} = \varphi \cdot P_{ij} + (1 - \varphi)G_j$;
      **If** (rand (0,1) > 0.5)
        $X_{ij} = p_{ij} + \alpha \cdot \mathrm{abs}\ (C_j - X_{ij}) \cdot \log\ (1/u)$;
      **Else**
        $X_{ij} = p_{ij} - \alpha \cdot \mathrm{abs}\ (C_j - X_{ij}) \cdot \log\ (1/u)$;
      **Endif**
    **Endfor// end for loop j**
  **Endfor//end for loop i**
**Endfor//end for loop t**
```

4. QPSO with Gaussian distributed attractor (GAQPSO)

4.1. Analysis of the local attractor point

It is evident from the Eq. (13) that each component of the particle's updated position is determined by $p_{i,j}^t$ and the disturbing part $\alpha\left|C_j^t - C_{i,j}^t\right| \ln\left(1/u_{i,j}^{t+1}\right)$. It was proved in [37] that if $\alpha < 1.781$, the disturbing part converges to zero so that each component of particle's position converges to $p_{i,j}^t$. Like the original PSO algorithm, convergence of the particles' positions to their local stochastic attractors can guarantee the convergence of the algorithm [37]. Eq. (5) indicates that p_i^t, the local stochastic attractor of particle i, lies in the hyper-rectangle with P_i^t and G^t being the two ends of its diagonal so that it moves following P_i^t and G^t. In fact, as the particles are converging to their own local attractors, their personal best positions are converging to the global best position, leading the QPSO algorithm to converge. As a result, the local attractor of each particle will gather toward the global best position, which in turn makes also the current position of the particle converge to the global best position. Thus we may find that the global best position guides the movement of p_i^t, which influence the convergence behavior of the particle's current position. If the global best position is trapped into a local optimal point, p_i^t, the particle's current position X_i^t will also be dragged into that point, leading the algorithm to premature convergence. Therefore, it may happen that using other probability distribution, instead of the uniform distribution as in Eq. (5), to determine the local attractor may be a way of improving the QPSO algorithm.

4.2. The local attractor point with Gaussian probability distribution

As the local attractor plays an important role in the QPSO algorithm, the misleading local attractor point may make the QPSO algorithm trap into the local optimal solution frequently. Here we propose to use Gaussian distribution to determine the local attractor. The mean value and the standard deviation of Gaussian distribution are the key parameters, which determine the shape of the distribution. In the proposed method, we denote the new local attractor point as *np*. The value of the original defined local attractor point in QPSO is chosen as the mean of the distribution. The standard deviation is equal to the distance between the mean best particle and the personal best particle, that is $C^t - P_i^t$. According to the definition of the *mbest* position, at the early stage of the search, the particles may be scattered in a wider space, and then the defined standard deviation is relatively larger. During the search procedure, as the particles converge, P_i^t careens toward C^t, and the standard deviation declines to zero. The proposed method can diversify the QPSO swarm through the new local attractor point and improve the performance in escaping the local minima. The new local attractor point *np* is generated by

$$np_i^t = N\left(p_i^t, C^t - P_i^t\right). \tag{15}$$

Therefore the particle position is updated according to the following equation:

$$X_{i,j}^{t+1} = np_{i,j}^t \pm \alpha\left|C_j^t - X_{i,j}^t\right| \ln\left(1/u_{i,j}^{t+1}\right), \tag{16}$$

where α and $u^t_{i,j}$ have the same meanings as those in QPSO algorithm. We call the proposed method Gaussian distributed local attractor QPSO (GAQPSO) algorithm. The procedure for implementing the GAQPSO is given by the following pseudo-codes.

```
Initialize the population size, the positions and the dimensions of the particles;
**For** $t = 1$ to Maximum Iteration $T$
  Compute the mean best position $C$ by Eq. (12);
  $\alpha = (\alpha_1 - \alpha_0) \cdot (T - t)/T + \alpha_0$;
  **For** $i = 1$ to population size $M$
    **If** $f(X_i) < f(P_i)$ then $P_i = X_i$; **Endif**
    Find the gbes position $G$;
    **For** $j = 1$ to $D$
      $\varphi = \text{rand}(0,1)$; $u = \text{rand}(0,1)$;
      $p_{ij} = \varphi \cdot P_{ij} + (1 - \varphi)G_j$;
      Generate the new local attractor point $np_i$ by Eq. (15)
      **If** (rand (0,1) > 0.5)
        $X_{ij} = np_{ij} + \alpha \cdot |C_j - X_{ij}| \cdot \log(1/u)$;
      **Else**
        $X_{ij} = np_{ij} - \alpha \cdot |C_j - X_{ij}| \cdot \log(1/u)$;
      **Endif**
    **Endfor// end for loop j**
  **Endfor//end for loop i**
**Endfor//end for loop t**
```

4.3. Analysis of distributions and the diversities

To illustrate the details of the distribution of the local attractor point in the QPSO and GAQPSO during the search process, we herein take two 2-dimensional test functions as an example. One is Sphere function and the other is Griewank function. Sphere function is a unimodal function while Griewank function is a multimodal one.

$$\text{Sphere function}: \ f_1(x) = \sum_{i=1}^{D} x_i^2, \quad \text{with} \ -100 \leqslant x_i \leqslant 100 \tag{17}$$

$$\text{Griewank function}: \ f_4(x) = \frac{1}{4000}\sum_{i=1}^{D} x_i^2 - \prod_{i=1}^{n}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1, \quad \text{with} \ -600 \leqslant x_i \leqslant 600. \tag{18}$$

Firstly, QPSO algorithm and GAQPSO algorithm are used to solve the two minimization problems. The population size is 50 and the maximum number of iterations is 10 for both algorithms. The distributions of the local attractor points in QPSO and GAQPSO during the search process are observed and shown in Fig. 2. In Fig. 2, the square points represent the local attractor p and the circle points represent the new local attractor np and they are plotted from initialization to the final iteration. It can be seen from Fig. 2 that whether in solving the unimodal function or the multimodal function, GAQPSO algorithm produces a

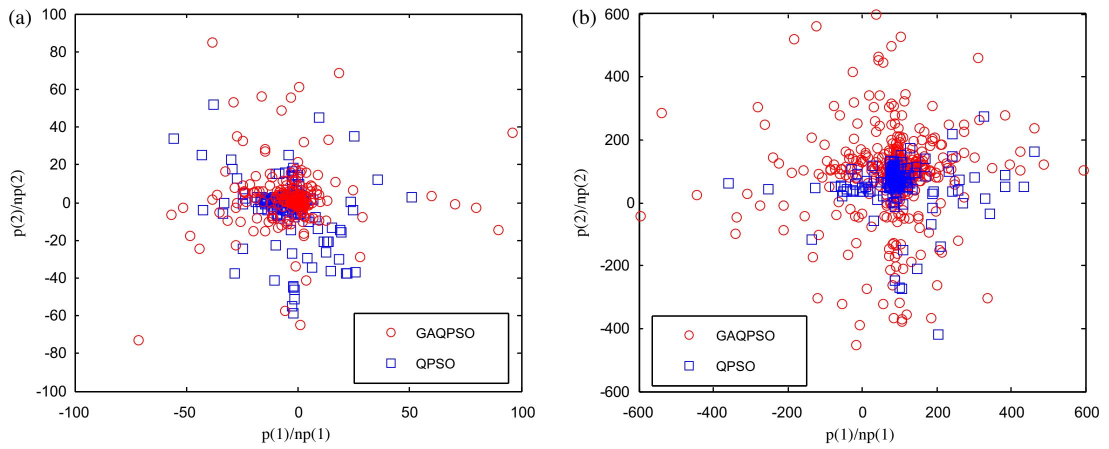

Fig. 2. Distribution comparison between QPSO and GAQPSO algorithm on Sphere function (a) and Griewank function (b) in 2 dimensions.

wide range of attractor points than QPSO algorithm. Therefore, GAQPSO may have stronger abilities to lead the swarm escape from the local minima.

Secondly, as mentioned above, GAQPSO could diversify the population as an overall result, which is achieved by diversification of the local attractor points. Herein we use a diversity measure to analyze the diversity changes of local attractor points in QPSO and GAQPSO. The diversity measure for D-dimensional numerical problems is the "distance-to-average-point" measure defined as [38]

$$diversity(P) = \frac{1}{LM} \cdot \sum_{i=1}^{M} \sqrt{\sum_{j=1}^{D} (s_{ij} - \overline{s_j})^2}, \tag{19}$$

where L is the length of the diagonal in the search space, M is the population size, D is the dimensionality of the problem, s_{ij} is the jth value of the ith individual, and $\overline{s_j}$ is the jth value of the average point. This diversity measure is dependent on swarm size, the dimensionality of the problem as well as the search range in each dimension. Low population diversity indicates that the swarm has clustered in a small region. Conversely, high population diversity indicates that the swarm has scattered in a wide region. Low population diversity is always taken the blame for the local convergence. However, high diversity may cause the algorithm not to converge. Thereupon the diversity should be considered together with the problem and the search process of the algorithm.

In this test, the population size is 20, the dimension is 30, and the maximum number of iterations is 2000 for QPSO and GAQPSO. Both the two algorithm are used to solve the two minimization problems. The diversity of the attractor points is recorded during the search process. For QPSO, the diversity is calculated as:

$$diversity(p) = \frac{1}{LM} \cdot \sum_{i=1}^{M} \sqrt{\sum_{j=1}^{D} (p_{ij} - \overline{p_j})^2} \tag{20}$$

and for GAQPSO, the diversity is calculated as:

$$diversity(np) = \frac{1}{LM} \cdot \sum_{i=1}^{M} \sqrt{\sum_{j=1}^{D} (np_{ij} - \overline{np_j})^2}. \tag{21}$$

Each algorithm was tested 50 times on each function. Fig. 3 compares the diversities between the QPSO and GAQPSO algorithm on Sphere function and Griewank function with dimensionality 30. The graph for both algorithms is an average over 50 test runs. We observe that, for both the functions the diversity of GAQPSO are higher than that of QPSO during the early phase of the search process and this is helpful for the attractor points located in a wide search space, which can prevent the algorithm from being trapped in the local minima. As far as Sphere function is concerned, since it is a simple unimodal function, during the later stage of search the low diversity can help reach high-precision solutions. From Fig. 3(a), it can be seen that GAQPSO has lower diversity in the later stage than QPSO. As for Griewank function, which is a difficult multimodal function, keeping diversity at certain level during the later phase of search is helpful to find better regions. As can be seen from Fig. 3(b), GAQPSO has higher diversity than QPSO in the final phase. The disadvantage of keeping high diversity in the final phase is that the convergence speed may be slowed.

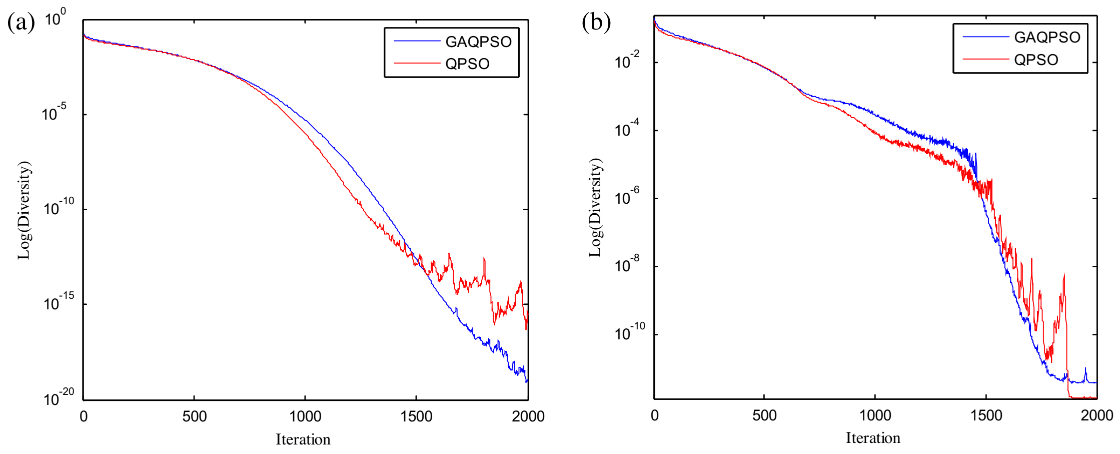

Fig. 3. Diversity comparisons between the QPSO and the GAQPSO algorithm on Sphere function (a) and Griewank function (b) in 30 dimensions.

4.4. Variants of GAQPSO

4.4.1. GAQPSO under a certain mutation probability

In this section, we propose a variant of GAQPSO by using a mutation probability which is used to determine whether the local attractor is determined by Eq. (5) or Eq. (15) during the iterative process. In this variant of GAQPSO, a random number uniformly distributed on (0,1) is generated for each particle during each iteration. If the random number is smaller than the given mutation probability p_m, then the local attractor of the particle is determined by Eq. (15); otherwise by Eq. (5). Thus, if p_m is set to be 1, the algorithm is essentially the GAQPSO algorithm. On the other hand, if p_m is zero, the algorithm is identical to the original QPSO. As a result, the update of the particle's position in the proposed variant of GAQPSO is measured by the following procedure:

```
IF rand (0,1) < $p_m$ Then $X_{ij} = np_{ij} \pm \alpha \cdot |C_j - X_{ij}| \cdot \log(1/u)$;
ELSE $X_{ij} = p_{ij} \pm \alpha \cdot |C_j - X_{ij}| \log(1/u)$; ENDIF
```

where p_m is the user-defined mutation probability.

4.4.2. GAQPSO using other methods to set up the standard deviation

In order to investigate the effectiveness of the proposed standard deviation in GAQPSO, we present two different strategies for generating the Gaussian distribution that is used to generate the local attractor of each particle during the search process. The mean value for both strategies is still p_i^t. The standard deviation for the first strategy is set to the distance between C^t and the middle point between P_i^t and G^t, and for the second strategy it is set to the distance between C^t and G^t. The algorithms based on the two strategies are denoted as GAQPSO-Type I and GAQPSO-Type II, respectively.

$$\text{GAQPSO} - \text{Type I} : np_i^t = N(p_i^t, C^t - A_i^t), A_i^t = 0.5 \cdot (P_i^t + G^t) \quad (22)$$

$$\text{GAQPSO} - \text{TypeII} : np_i^t = N(p_i^t, C^t - G^t). \quad (23)$$

5. Experimental results and analysis

5.1. Benchmark functions and experimental setup

In this section, GAQPSO is used to optimize the functions F_1 to F_{12} of the CEC2005 benchmark functions proposed by Suganthan et al. [39], in order to determine whether GAQPSO algorithm can be as effective as QPSO and other variants of PSO, including PSO with inertia weight (PSO-In) [2], PSO with constriction factor (PSO-Co), standard PSO [22], Gaussian PSO [17], Gaussian Bare Bones PSO [12], Levy PSO [15], dynamic multiple swarm PSO [40] and fully-informed particle swarm [11]. The expressions of the benchmarks function are not provided in this text due to the space limitation. One can refer to reference [30] and find the MATLAB source code on Suganthan's website. Of the used twelve benchmark functions, F_1 to F_6 are unimodal functions, F_7 to F_{12} are functions with many local minima. The magnitude of upper and lower bounds as well as the initialization ranges of benchmark functions can also be found in [39].

Each algorithm ran 100 times on each problem. At each run, the particles used in the algorithm start in new and randomly-generated positions. The parameters of each algorithm are chosen as follows:

Population size: $M = 20$;
Problem dimension: $D = 30$;
Maximum iteration: $T = 3000$.

For the GAQPSO algorithms and QPSO, the CE coefficient decreases linearly from 1.0 to 0.5 during the search process according to equation (14), that is, $\alpha_1 = 1.0$ and$\alpha_0 = 0.5$. The other parameter configurations of the other PSO variants were set according to their corresponding reference.

The best fitness value for each run was recorded and then the average best fitness value can be compared with one another, and the standard deviations of the best fitness values can be calculated.

5.2. Experimental results

5.2.1. Comparison between GAQPSO and other PSO Variants

The mean best fitness values and standard deviations by the algorithms on each problem are presented in Table 3. For a thorough comparison the t-test has also been carried out. Table 1 shows the t values on every function of this unpaired test with a significance level of 0.05 between GAQPSO and another algorithm. In Table 2, the signs that GAQPSO performs significantly better than or significantly worse than the compared algorithm are given respectively.

Table 1
Experimental results of mean best fitness values and standard deviations by algorithms and problems (best results in bold).

Algorithms	F_1	F_2	F_3	F_4	F_5	F_6
In-PSO (Std. Dev.)	3.8773e−013 (1.6083e−012)	785.0932 (661.2154)	3.9733e + 07 (4.6433e + 07)	1.1249e + 04 (5.4394e + 03)	6.0547e + 03 (2.0346e + 03)	263.7252 (437.4145)
Co-PSO	1.5713e−026 (1.4427e−025)	0.1267 (0.3796)	8.6472e + 06 (9.1219e + 06)	1.3219e + 04 (6.0874e + 03)	7.6892e + 03 (2.3917e + 03)	123.0243 (266.2520)
Standard PSO	8.2929e−026 (1.2289e−025)	78.2831 (52.3272)	6.6185e + 06 (3.0124e + 06)	1.3312e + 04 (4.1076e + 03)	6.2884e + 03 (1.4318e + 03)	153.5178 (246.1049)
Gaussian PSO	7.3661e−026 (5.9181e−025)	**0.0988** **(0.3362)**	1.1669e + 07 (2.5153e + 07)	2.3982e + 04 (1.2512e + 04)	8.0279e + 03 (2.3704e + 03)	150.7872 (303.3368)
Gaussian Bare Bones PSO	1.7869e−025 (8.4585e−025)	16.8751 (16.2021)	7.7940e + 06 (4.3240e + 06)	1.1405e + 04 (6.7712e + 03)	9.5814e + 03 (3.0227e + 03)	144.1377 (165.2616)
PSO-E	5.2531e−024 (2.2395e−023)	20.2750 (15.2414)	6.2852e + 06 (2.8036e + 06)	8.2706e + 03 (3.6254e + 03)	7.2562e + 03 (1.8666e + 03)	189.8292 (375.8636)
Lévy PSO	1.1880e−024 (1.1455e−023)	36.9986 (29.1360)	1.7366e + 07 (1.9001e + 07)	7.4842e + 03 (6.6588e + 03)	8.2543e + 03 (2.2297e + 03)	133.9526 (293.8460)
CLPSO	3.5515e−008 (2.2423e−008)	5.3394e + 03 (1.2207e + 03)	5.1434e + 07 (1.3489e + 07)	1.6069e + 04 (3.4776e + 03)	5.4958e + 003 (888.9618)	117.3987 (54.8846)
DMS-PSO	7.2525e−006 (2.2114e−005)	844.9978 (350.2620)	1.2841e + 07 (4.9745e + 06)	2.7125e + 003 (972.8958)	2.9189e + 003 (811.5164)	296.0911 (347.1682)
FIPS	3.3157e−027 (2.5732e−028)	75.4903 (76.1305)	1.0409e + 07 (4.4786e + 06)	1.0529e + 04 (3.8510e + 03)	4.3452e + 003 (978.6149)	188.8304 (294.0374)
QPSO	**1.2672e−027** **(3.7147e−028)**	120.6051 (62.2340)	4.4257e + 06 (2.3302e + 06)	4.0049e + 03 (2.7218e + 03)	3.3684e + 003 (975.6551)	88.0494 (159.7481)
GAQPSO	8.5433e−027 (1.6695e−027)	15.2620 (16.9682)	**4.1685e + 06** **(1.8723e + 06)**	**2.3503e + 03** **(1.8573e + 03)**	2.8864e + 03 (732.4688)	**56.6257** **(90.6767)**
GAQPSO-Type I	7.6789e−027 (1.9185e−027)	113.2691 (83.1571)	1.1177e + 07 (6.3152e + 06)	3.0692e + 03 (1.9989e + 03)	**2.5286e + 03** **(947.8740)**	72.3946 (110.9107)
GAQPSO-Type II	4.8949e−012 (2.2871e−011)	2.5788e + 03 (1.9398e + 03)	3.6754e + 07 (3.1630e + 07)	7.7058e + 03 (1.5084e + 03)	2.6231e + 03 (935.7576)	116.9730 (148.0414)

Algorithms	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}
In-PSO (Std. Dev.)	0.9907 (4.7802)	0.0414 (0.2393)	39.5528 (16.1654)	239.5814 (72.2521)	41.0529 (6.0318)	3.6785e + 04 (4.0943e + 04)
Co-PSO	0.0255 (0.0327)	5.1120 (4.5667)	96.7296 (28.0712)	171.6488 (58.5713)	36.0339 (7.2659)	9.9648e + 03 (1.6158e + 04)
Standard PSO	0.0218 (0.0165)	0.2744 (0.6795)	79.1219 (20.2619)	128.9865 (32.3662)	30.3424 (2.7409)	1.8178e + 04 (1.4866e + 04)
Gaussian PSO	0.0224 (0.0178)	2.7722 (1.4603)	103.6245 (28.6113)	184.2657 (57.3675)	33.5448 (6.5823)	(6.5610e + 04)
Gaussian Bare Bones PSO	0.0205 (0.0208)	3.5460 (6.1929)	80.9496 (22.0621)	164.2914 (72.8542)	29.8088 (3.2671)	3.4327e + 04 (6.2435e + 04)
PSO-E	0.0493 (0.0538)	3.5881 (5.5286)	66.5112 (20.9853)	163.7187 (55.0921)	29.2666 (3.2083)	1.7161e + 04 (1.0862e + 04)
Lévy PSO	0.0446 (0.1182)	2.2168 (1.3575)	74.0446 (21.6913)	154.3838 (76.3070)	28.9923 (5.0212)	1.6282e + 04 (2.5184e + 04)
CLPSO	2.4151 (0.7533)	1.1582e−04 (6.7878e−05)	**0.6990** **(0.7983)**	151.2854 (23.4628)	30.9500 (1.6697)	5.4400e + 04 (1.2508e + 04)
DMS-PSO	0.3985 (0.2502)	0.1213 (0.3716)	39.9694 (10.2384)	**112.8426** **(71.2957)**	**25.8903** **(3.1488)**	1.3714e + 04 (8.8995e + 03)
FIPS	0.0330 (0.0464)	0.3843 (0.5713)	64.6289 (14.5907)	198.3699 (21.7958)	35.4586 (2.7216)	4.6334e + 04 (2.4690e + 04)
QPSO	0.0208 (0.0130)	2.0961e−14 (1.9099e−14)	29.9218 (10.5736)	118.4549 (53.0216)	28.1887 (6.2233)	1.2938e + 04 (1.3787e + 04)
GAQPSO	0.0161 (0.0141)	**1.5632e−014** **(3.1128e−015)**	25.4653 (20.9819)	169.7682 (32.8371)	33.7577 (7.6365)	**6.8676e + 03** **(6.3181e + 03)**
GAQPSO-Type I	**0.0152** **(0.0125)**	1.9895e−014 (5.5316e−015)	97.9204 (46.1955)	202.1704 (13.7047)	40.6849 (1.3981)	2.4744e + 04 (2.1769e + 04)
GAQPSO-Type II	0.0700 (0.0571)	5.7137e−07 (1.2202e−06)	124.9709 (45.8213)	215.3553 (19.1182)	41.1852 (1.1642)	3.4988e + 04 (3.3402e + 04)

Table 2
t-test[a] value for all the twelve functions: comparison of GAQPSO with other algorithms (GAQPSO-other algorithm).

Algorithms	F_1	F_2	F_3	F_4	F_5	F_6
PSO-In	−2.4108[b]	−11.6388[b]	−7.6531[b]	−15.4821[b]	−14.6516[b]	−4.6361[b]
PSO-Co	−0.4969[c]	8.9176[d]	−4.8096[b]	−17.0772[b]	−19.2009[b]	−2.3607[b]
Standard PSO	−6.0525[b]	−11.4564[b]	−6.9076[b]	−24.3162[b]	−21.153[b]	−3.6942[b]
Gaussian PSO	−1.1003[c]	8.9345[d]	−2.9737[b]	−17.1014[b]	−20.7236[b]	−2.9741[b]
Gaussian Bare Bones PSO	−2.0115[b]	−0.6876[c]	−7.6943[b]	−12.896[b]	−21.5261[b]	−4.6425[b]
PSO-E	−2.3418[b]	−2.1979[b]	−6.2786[b]	−14.5338[b]	−21.7927[b]	−3.4451[b]
Lévy PSO	−1.0296[c]	−6.4468[b]	−6.9122[b]	−7.4265[b]	−22.872[b]	−2.5145[b]
CLPSO	−15.8386[b]	−43.6112[b]	−34.7073[b]	−34.797[b]	−22.6539[b]	−5.7337[b]
DMS-PSO	−3.2796[b]	−23.6613[b]	−16.3165[b]	−1.7275[b]	−0.2973[c]	−6.6738[b]
FIPS	30.9469[d]	−7.7217[b]	−12.8558[b]	−19.1293[b]	−11.9342[b]	−4.2965[b]
QPSO	42.5421[d]	−16.3308[b]	−0.8604[d]	−5.0214[b]	−3.9508[b]	−1.7107[b]
GAQPSO-Type I	3.3989[d]	−11.5478[b]	−10.6401[b]	−2.6347[b]	2.9869[d]	−1.1007[c]
GAQPSO-Type II	−2.1402[b]	−13.215[b]	−10.2841[b]	−22.383[b]	2.2157[d]	−3.4761[b]

Algorithms	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}
PSO-In	−2.0388[b]	−1.73[b]	−5.3187[b]	−8.7966[b]	−7.4966[b]	−7.2216[b]
PSO-Co	−2.6397[b]	−11.1941[b]	−20.3344[b]	−0.2801[c]	−2.1594[b]	−1.7852[b]
Standard PSO	−2.6263[b]	−4.0383[b]	−18.3956[b]	8.845[d]	4.2094[d]	−7.0021[b]
Gaussian PSO	−2.7744[b]	−18.9838[b]	−22.0289[b]	−2.1932[b]	0.2112[c]	−9.4074[b]
Gaussian Bare Bones PSO	−1.751[b]	−5.7259[b]	−18.2237[b]	0.6853[c]	4.7543[d]	−4.3757[b]
PSO-E	−5.9694[b]	−6.4901[b]	−13.8317[b]	0.9432[c]	5.422[d]	−8.1915[b]
Lévy PSO	−2.3942[b]	−16.33[b]	−16.0972[b]	1.8519[d]	5.2141[d]	−3.6259[b]
CLPSO	−31.841[b]	−17.063[b]	11.7951[d]	4.5797[d]	3.5918[d]	−33.9198[b]
DMS-PSO	−15.2596[b]	−3.2643[b]	−6.2125[b]	7.2522[d]	9.5245[d]	−6.2729[b]
FIPS	−3.4849[b]	−6.7268[b]	−15.3244[b]	−7.257[b]	−2.0981[b]	−15.4858[b]
QPSO	−2.4507[b]	−2.7539[b]	−1.8967[b]	8.2277[d]	5.6531	−4.0027[b]
GAQPSO-Type I	0.4776[c]	−6.7162[b]	−14.2805[b]	−9.1063[b]	−8.9229[b]	−7.8864[b]
GAQPSO-Type II	−9.1643[b]	−4.6826[b]	−19.7445[b]	−11.9975[b]	−9.6152[b]	−8.2721[b]

[a] The *t* value of 99 degrees of freedom is significant at 0.05 level of significance by an unpaired *t*-test.
[b] GAQPSO is significantly better than the compared algorithm.
[c] GAQPSO has no significant performance difference with the compared algorithm.
[d] GAQPSO is significantly worse than the compared algorithm.

For function F_1, shifted sphere function, all the algorithms could get a solution near the global optimal solution, but the solution accuracy differs widely among them. QPSO offered the highest accuracy and GAQPSO and GAQPSO-Type I performed better than all the other algorithms. GAQPSO was the fourth best performing algorithm for this function. F_2 is shifted Schwefel's problem 1.2. For this function, the mean best fitness value of Gaussian PSO outperforms all the other algorithms. From a statistical point of view, GAQPSO is significantly better than the other algorithms except PSO-Co and Gaussian PSO. For function F_3, shifted rotated high conditioned elliptic function, the mean best fitness value of GAQPSO was the best among all the compared algorithms. Tables 1 and 2 show that GAQPSO performed significantly better than any other algorithms except QPSO. The results for F_4, shifted Schwefel's problem 1.2 with noise in fitness, shows that GAQPSO outperformed any other competitor in a significant manner. The fifth benchmark function F_5 is known as Schwefel's problem 2.6 with global optimum on bounds. It can be observed that GAQPSO-Type I and GAQPSO-Type II had the better performances than all the other methods. The third best performed algorithm was GAQPSO, which outperformed the rest of the algorithms in a significant manner, except DMS-PSO. Table 1 also shows that for F_6, shifted Rosenbrock function, GAQPSO obtained the best solution among all the algorithms and its advantages over other competitors, except QPSO, were statistically significant. Results obtained for F_7, shifted rotated Griewank's function without bounds, indicate that GAQPSO and GAQPSO-Type I generated better solutions than their competitors but the difference between them was not statistically significant. F_8 is shifted rotated Ackley's function with global optimum on bounds. For this function, the performance of GAQPSO was superior to the other algorithms significantly. F_9, shifted Rastrigrin function, is separable. For this function, CLPSO had significant better performance than the others, and GAQPSO was the second best methods among the tested algorithms. F_{10} is shifted rotated Rastrigrin function, for which DMS-PSO was the winner, and GAQPSO along with its variants didn't show better performance than the others. The similar thing to F_{10} happened for F_{11}, shifted rotated Weierstrass function. For F_{12}, Schwefel's problem 2.13, GAQPSO had the significant advantage over all the other methods.

Ranking the algorithmic performances (i.e. the mean best fitness values), as shown in Table 3, provided an overall comparison among all the algorithms. It can be observed that GAQPSO had the best total rank, which means that it had the best overall performance on the suite of CEC 2005 benchmark functions. QPSO is showed to have the second best overall performance, as indicated by its total rank. GAQPSO-Type II, however, didn't perform so effectively as GAQPSO, and GAQPSO-Type II, probably due to its slower convergence speed resulting from the method of determining the local attractor. From Table 3, it can be seen that DMS-PSO and standard PSO showed the better overall performance than the other algorithms except GAQPSO, QPSO and GAQPSO-Type I.

Table 3
Ranking of the algorithmic performances.

Algorithms	F_1	F_2	F_3	F_4	F_5	F_6	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}	Total rank
PSO-In	11	11	13	10	8	13	13	6	4	14	13	11	127
PSO-Co	5	2	6	11	11	6	7	14	11	9	11	2	95
Standard PSO	7	8	4	12	8	10	5	8	9	3	6	7	87
Gaussian PSO	6	1	9	14	12	9	6	11	13	10	8	14	113
Gaussian Bare Bones PSO	8	4	5	9	14	8	3	12	10	7	5	9	94
PSO-E	10	5	3	7	10	12	10	13	7	6	4	6	93
Lévy PSO	9	6	11	5	13	7	9	10	8	5	3	5	91
CLPSO	13	14	14	13	7	5	14	5	1	4	7	13	110
DMS-PSO	14	12	10	2	4	14	12	7	5	1	1	4	86
FIPS	2	7	7	8	6	11	8	9	6	11	10	12	97
QPSO	1	10	2	4	5	3	4	3	3	2	2	3	42
GAQPSO	4	3	1	1	3	1	2	1	2	8	9	1	**36**
GAQPSO-Type I	3	9	8	3	1	2	1	2	12	12	12	8	73
GAQPSO-Type II	12	13	12	6	2	4	11	4	14	13	14	10	115

Table 4
Experimental results of mean best fitness values and standard deviations for the GAQPSO algorithm under a set of mutation probabilities (best results in bold).

Algorithms	F_1	F_2	F_3	F_4	F_5	F_6
GAQPSO	1.2783e−027	104.8460	6.6256e + 006	4.3192e + 003	3.2724e + 003	153.5163
(*pm* = 0.1)	(3.1594e−028)	(101.1204)	(2.7293e + 006)	(2.7817e + 003)	(945.2200)	(302.3906)
GAQPSO	1.2103e−027	77.7572	6.6095e + 006	3.4578e + 003	3.3525e + 003	133.2310
(*pm* = 0.2)	(4.7711e−028)	(58.0841)	(2.7952e + 006)	(2.1091e + 003)	(987.8831)	(256.6336)
GAQPSO	1.1345e−027	64.8749	5.9896e + 006	3.5549e + 003	3.2236e + 003	118.0402
(*pm* = 0.3)	(3.3074e−028)	(47.7864)	(2.7173e + 006)	(2.5534e + 003)	(896.0692)	(208.8569)
GAQPSO	1.1433e−027	57.7256	6.0165e + 006	2.8639e + 003	3.0920e + 003	105.9820
(*pm* = 0.4)	(2.9646e−028)	(42.2141)	(2.6675e + 006)	(1.9578e + 003)	(886.4175)	(201.6067)
GAQPSO	**1.1372e−027**	56.4884	6.2019e + 006	2.8592e + 003	2.9851e + 003	89.6964
(*pm* = 0.5)	**(2.8156e−028)**	(43.2032)	(3.0493e + 006)	(1.8087e + 003)	(883.6004)	(143.4129)
GAQPSO	1.2410e−027	40.8899	5.3771e + 006	2.8937e + 003	3.1567e + 003	88.0263
(*pm* = 0.6)	(3.7061e−028)	(29.3929)	(2.2465e + 006)	(1.7840e + 003)	(723.9084)	(156.9850)
GAQPSO	1.2789e−027	34.0822	5.0023e + 006	2.7342e + 003	2.9450e + 003	70.8822
(*pm* = 0.7)	(2.9639e−028)	(26.5946)	(2.4197e + 006)	(1.7869e + 003)	(820.2506)	(121.2040)
GAQPSO	1.5273e−027	26.0977	4.6965e + 006	2.4609e + 003	3.0789e + 003	74.5598
(*pm* = 0.8)	(4.2974e−028)	(19.2088)	(2.0126e + 006)	(2.1051e + 003)	(766.4450)	(107.4096)
GAQPSO	2.0827e−027	24.0650	4.3909e + 006	2.4301e + 003	2.9061e + 003	75.5166
(*pm* = 0.9)	(4.6317e−028)	(17.8373)	(2.0442e + 006)	(1.6011e + 003)	(805.02370)	(130.6647)
GAQPSO	8.5433e−027	**15.2620**	**4.1685e + 06**	**2.3503e + 03**	**2.8864e + 003**	**56.6257**
(*pm* = 1.0)	(1.6695e−027)	**(16.9682)**	**(1.8723e + 06)**	**(1.8573e + 03)**	**(732.4688)**	**(90.6767)**

Algorithms	F_7	F_8	F_9	F_{10}	F_{11}	F_{12}
GAQPSO	0.0211	1.4317e−014	30.7548	123.0231	28.5835	1.1428e + 04
(*pm* = 0.1)	(0.0127)	(5.9845e−015)	(9.3873)	(58.0714)	(6.9448)	(1.1116e + 04)
GAQPSO	0.0223	1.5596e−014	30.7836	124.8172	28.6232	1.0962e + 04
(*pm* = 0.2)	(0.0135)	(8.8892e−015)	(9.0599)	(52.8876)	(7.1778)	(9.7002e + 03)
GAQPSO	0.0185	1.3998e−014	29.5468	126.4855	28.1537	1.0108e + 04
(*pm* = 0.3)	(0.01280)	(5.1452e−015)	(8.7679)	(58.5137)	(7.4035)	(8.0463e + 03)
GAQPSO	0.0227	1.2292e−014	28.8157	125.2887	**26.4654**	9.5308e + 03
(*pm* = 0.4)	(0.0174)	(4.5001e−015)	(14.7901)	(52.4135)	**(7.5398)**	(7.2507e + 03)
GAQPSO	0.0198	**1.2363e−014**	29.2997	124.9129	27.6080	1.0648e + 04
(*pm* = 0.5)	(0.0157)	**(3.7271e−015)**	(10.7621)	(53.1230)	(7.3283)	(8.6593e + 03)
GAQPSO	0.0194	1.3287e−014	27.7833	**122.6634**	27.8604	1.1344e + 04
(*pm* = 0.6)	(0.0158)	(3.6280e−015)	(9.8191)	**(51.2455)**	(7.3706)	(8.6655e + 03)
GAQPSO	0.0176	1.2612e−014	27.0451	123.4124	29.0839	1.0366e + 04
(*pm* = 0.7)	(0.0120)	(3.9560e−015)	(8.1948)	(50.8396)	(8.2908)	(8.8476e + 03)
GAQPSO	0.0177	1.3038e−014	27.1810	124.2773	29.5899	1.0708e + 04
(*pm* = 0.8)	(0.0123)	(3.1150e−015)	(12.4430)	(48.6487)	(7.5390)	(1.0381e + 04)
GAQPSO	0.0194	1.3785e−014	27.9978	145.0926	30.8242	9.6254e + 03
(*pm* = 0.9)	(0.0147)	(2.6374e−015)	(16.5692)	(49.9601)	(7.8822)	(8.8004e + 03)
GAQPSO	**0.0161**	1.5632e−014	**25.4653**	169.7682	33.7577	**6.8676e + 03**
(*pm* = 1.0)	**(0.0141)**	(3.1128e−015)	**(20.9819)**	(32.8371)	(7.6365)	**(6.3181e + 03)**

5.2.2. Experimental results for GAQPSO with mutation probability

In order to test the GAQPSO with mutation probability, different values of p_m were chosen from 0.1 to 1.0 with the step size 0.1. The algorithm ran 100 times on each problem for each value of p_m. The algorithmic parameters such as the population size, problem dimension, maximum number of iterations, and CE coefficient were the same as those specified in Section 5.1. The mean best fitness values and standard deviations obtained from all the tests are presented in Table 4.

It is evident from the results that the GAQPSO without mutation probability (i.e. $p_m = 1.0$) outperformed those with a certain mutation probability except on F_1, F_8, F_{10} and F_{11}. For F_1, the best result was obtained by setting p_m to 0.5. For F_8, although the algorithm performed best when p_m was 0.5, it is shown that the value of p_m did not exert remarkable influence on the algorithmic performance. For F_{10}, setting p_m to 0.6 was able to generate the best results, but larger value of p_m did not result in better performance of the algorithm. For F_{11}, the best value of p_m is 0.4 and larger p_m also weakened the algorithmic performance. For all the other benchmark functions, it can be observed that the mean best fitness values become better and better as the mutation probability increases, implying that using Eq. (5) to determine the local attractor point of the particle can indeed improve QPSO in most cases by enhancing the global search ability of the algorithm.

6. Conclusion

In this paper, a variant of QPSO, namely GAQPSO, is proposed by using a Gaussian probability distribution to generate the local attractor point of each particle. In QPSO, each particle should converge to the local attractor point in order to guarantee that all the particles converge. Considering the effect of the local attractor point, we used a Gaussian probability distribution in the GAQPSO algorithm to generate the local attractor points. The mean of Gaussian distributed local attractor used is identical to the local attractor point in the QPSO algorithm, and its standard deviation is chosen to be the position distance between the mean best particle and the personal best. As the particle swarm evolves, the value of standard deviation goes to zero and the local attractor point in the GAQPSO algorithm careen toward that in the QPSO algorithm.

GAQPSO is less susceptible to premature convergence and less likely to be stuck in local optima, owning this to the Gaussian distributed local attractor points which make the particles volatile and diversify the swarm. In order to further investigate the effectiveness of the GAQPSO algorithm, some variants of GAQPSO were proposed. One variant uses the Gaussian distributed local attractor point as a mutation according to a certain probability. Another variant is modifying the probability distribution by using other methods to set up the standard deviations in the GAQPSO. The GAQPSO algorithms, along with its variants, were tested on a suite of CEC2005 benchmark functions, and compared with QPSO and other PSO variants. The results show that the GAQPSO algorithm has superior features, both in high quality of the solution and robustness of the results in most cases.

Our further work will focus on applying the GAQPSO algorithm to the real-world optimization problems and an using some other probability distributions in the QPSO algorithm.

Acknowledgements

This work was supported by National Natural Science Foundation of China (Project Number: 61170119, 61105128, 60973094), by Natural Science Foundation of Jiangsu Province, China (Project Number: BK2010143), by the Fundamental Research Funds for the Central Universities (Project Number: JUSRP21012, JUSRP11023), and by the Foundation of Key Laboratory of Advanced Process Control for Light Industry (Jiangnan University), Ministry of Education, P.R. China.

References

[1] J. Kennedy, R.C. Eberhart, Particle swarm optimization, in: Proceedings of IEEE International Conference on Neural Networks, 1995, pp. 1942–1948.
[2] Y. Shi, R.C. Eberhart, A modified particle swarm optimizer, in: Proceedings of the IEEE International Conference on Evolutionary Computation, IEEE Press, Piscataway, NJ, 1998, pp. 69–73.
[3] R. Poli, Analysis of the publications on the applications of particle swarm optimisation, Journal of Artificial Evolution and Applications 28 (2008) 1–10.
[4] J.J. Liang, A.K. Qin, P.N. Suganthan, S. Baskar, Comprehensive learning particle swarm optimizer for global optimization of multimodal functions, IEEE Transactions on Evolutionary Computation 10 (2006) 281–295.
[5] X. Xiao-Feng, Z. Wen-Jun, Y. Zhi-Lian, Adaptive particle swarm optimization on individual level, vol. 2, 2002, pp. 1215–1218.
[6] J. Kennedy, R. Mendes, Population structure and particle swarm performance, 2002, pp. 1671–1676.
[7] S. Yuhui, R.C. Eberhart, Fuzzy adaptive particle swarm optimization, in: Proceedings of the 2001 Congress on Evolutionary Computation, 2001, pp. 101–106.
[8] P.S. Andrews, An investigation into mutation operators for particle swarm optimization, 2006, pp. 1044–1051.
[9] J.J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism, in: IEEE Congress on Evolutionary Computation, 2006, CEC 2006, 2006, pp. 9–16.
[10] F. van den Bergh, A.P. Engelbrecht, A Cooperative approach to particle swarm optimization, IEEE Transactions on Evolutionary Computation 8 (2004) 225–239.
[11] R. Mendes, J. Kennedy, J. Neves, The fully informed particle swarm: simpler, maybe better, IEEE Transactions on Evolutionary Computation 8 (2004) 204–210.
[12] J. Kennedy, Bare bones particle swarms, in: Proceedings of the 2003 IEEE Swarm Intelligence Symposium, 2003, pp. 80–87.
[13] J. Kennedy, Probability and dynamics in the particle swarm, 2004 Congress on Evolutionary Computation, 2004, pp. 340–347.
[14] K. James, Dynamic-probabilistic particle swarms, in: Proceedings of the 2005 Conference on Genetic and Evolutionary Computation, ACM, Washington DC, USA, 2005.
[15] T.J. Richer, T.M. Blackwell, The levy particle swarm, in: IEEE Congress on Evolutionary Computation, 2006, CEC 2006, 2006, pp. 808–815.

[16] L.d.S. Coelho, R.A. Krohling, Predictive controller tuning using modified particle swarm optimization based on cauchy and Gaussian distributions, in: Proceedings of the VI Brazilian Conference on Neural Networks, Sao Paulo, Brazil, 2003.
[17] R.A. Krohling, Gaussian swarm: a novel particle swarm optimization algorithm, 2004 IEEE Conference on Cybernetics and Intelligent Systems, 2004.
[18] R.A. Krohling, Gaussian particle swarm with jumps, vol. 2, 2005, pp. 226–231.
[19] R.A. Krohling, L. dos Santos Coelho, PSO-E: Particle Swarm with Exponential Distribution, 2006, pp. 1428–1433.
[20] R.A. Krohling, E. Mendel, Bare bones particle swarm optimization with Gaussian or Cauchy jumps, IEEE Congress on Evolutionary Computation, 2009, CEC'09, 2009, pp. 3285–3291.
[21] N. Higashi, H. Iba, Particle swarm optimization with Gaussian mutation, in: Proceedings of the 2003 IEEE Swarm Intelligence Symposium, 2003, pp. 72–79.
[22] M. Clerc, J. Kennedy, The particle swarm – explosion, stability, and convergence in amultidimensional complex space, IEEE Transactions on Evolutionary Computation 6 (2002) 58–73.
[23] J. Sun, B. Feng, W. Xu, Particle swarm optimization with particles having quantum behavior, in: IEEE Congress on Evolutionary Computation, 2004, pp. 325–331.
[24] J. Sun, B. Feng, W. XU, A Global search strategy of quantum-behaved particle swarm optimization, in: IEEE Conference on Cybernetics and Intelligent Systems, 2004, pp. 111–116.
[25] S. Jun, X. Wenbo, F. Bin, Adaptive parameter control for quantum-behaved particle swarm optimization on individual level, vol. 4, 2005, pp. 3049–3054.
[26] S.M. Mikki, A.A. Kishk, Quantum particle swarm optimization for electromagnetics, IEEE Transactions on Antennas and Propagation 54 (2006) 2764–2775.
[27] L. Shouyi, A new QPSO based BP neural network for face detection, Fuzzy Information and Engineering (2007).
[28] L.D. Coelho, A quantum particle swarm optimizer with chaotic mutation operator, Chaos, Solitons and Fractals 37 (2008).
[29] S.N. Omkar, K. Rahul, T.V.S. Ananth, G.N. Naik, S. Gopalakrishnan, Quantum behaved particle swarm optimization (QPSO) for multi-objective design optimization of composite structures, Expert Systems with Applications 36 (2009) 11312–11322.
[30] L. dos Santos Coelho, P. Alotto, Global optimization of electromagnetic devices using an exponential quantum-behaved particle swarm optimizer, IEEE Transactions on Magnetics 44 (2008) 1074–1077.
[31] L.S. Coelho, Novel Gaussian quantum-behaved particle swarm optimiser applied to electromagnetic design, Science, Measurement & Technology, IET 1 (2007) 290–294.
[32] L.d.S. Coelho, N. Nedjah, L.d.M. Mourelle, Gaussian quantum-behaved particle swarm optimization applied to fuzzy PID controller design studies in computational intelligence, vol. 121, 2008, pp. 1–15.
[33] L.d.S. Coelho, Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems, Expert Systems with Applications 37 (2010) 1676–1683.
[34] L. Jing, X. Wenbo, S. Jun, Quantum-behaved particle swarm optimization with mutation operator, 2005, 4.
[35] J. Liu, J. Sun, W.B. Xu, Quantum-behaved particle swarm optimization with adaptive mutation operator, Advances in Natural Computation 4221 (Pt 1) (2006) 959–967.
[36] F. Wei, S. Jun, X. Wenbo, Analysis of mutation operators on quantum-behaved particle swarm optimization algorithm, New Mathematics and Natural Computation (NMNC) 5 (2009) 487–496.
[37] J. Sun, W. Fang, X.J. Wu, V. Palade, W.B. Xu, Quantum-behaved particle swarm optimization: analysis of the individual particle's behavior and parameter selection, Evolutionary Computation, doi:10.1162/EVCO_a_00049.
[38] J. Riget, J. Vesterstroem, A diversity-guided particle swarm optimizer – The ARPSO, Department of Computer Science, University of Aarhus, 2002.
[39] P.N. Suganthan et al., Problem definitions and evaluation criteria for the CEC 2005, Technical Report, Nanyang Technological University, Singapore, 2005.
[40] J.J. Liang, P.N. Suganthan, Dynamic multi-swarm particle swarm optimizer, in: Proceedings of the 2005 IEEE Swarm Intelligence Symposium, 2005, pp. 124–129.