Hindawi Publishing Corporation
Mathematical Problems in Engineering
Volume 2013, Article ID 469723, 10 pages
http://dx.doi.org/10.1155/2013/469723

Research Article

Quantum Behaved Particle Swarm Optimization with Neighborhood Search for Numerical Optimization

Xiao Fu,[1] Wangsheng Liu,[1] Bin Zhang,[2] and Hua Deng[1]

[1] *Department of Fundamental Courses, Air Force Aviation University, Changchun 130022, China*
[2] *Department of Aviation Survival, Air Force Aviation University, Changchun 130022, China*

Correspondence should be addressed to Xiao Fu; fuxiao_cq@126.com

Received 22 April 2013; Accepted 13 September 2013

Academic Editor: Yang Xu

Quantum-behaved particle swarm optimization (QPSO) algorithm is a new PSO variant, which outperforms the original PSO in search ability but has fewer control parameters. However, QPSO as well as PSO still suffers from premature convergence in solving complex optimization problems. The main reason is that new particles in QPSO are generated around the weighted attractors of previous best particles and the global best particle. This may result in attracting too fast. To tackle this problem, this paper proposes a new QPSO algorithm called NQPSO, in which one local and one global neighborhood search strategies are utilized to balance exploitation and exploration. Moreover, a concept of opposition-based learning (OBL) is employed for population initialization. Experimental studies are conducted on a set of well-known benchmark functions including multimodal and rotated problems. Computational results show that our approach outperforms some similar QPSO algorithms and five other state-of-the-art PSO variants.

1. Introduction

Many real-world problems can be formulized into optimization problems over continuous or discrete search space. With development of economic, optimization problems are increasingly complex, and more efficient optimization algorithms are needed. Over the past several years, some population-based random optimization techniques have been widely used to solve optimization problems, such as genetic algorithms (GA) [1], evolutionary programming (EP) [2], particle swarm optimization (PSO) [3], differential evolution (DE) [4], ant colony optimization (ACO) [5], and artificial bee colony (ABC) [6]. Due to PSO's simple concept, easy implementation, yet effective, has been widely applied to various optimization areas [7–11].

PSO was firstly introduced by Kennedy and Eberhart in 1995. It is a new optimization technique inspired by swarm intelligence. Compared to GA, PSO is also a population-based algorithm, but it does not contain any crossover or mutation operator. In PSO, the movement of particles is determined by their corresponding previous best particles ($p$best) and the global best particle ($g$best). Due to the attraction of these best particles ($p$best and $g$best), PSO shows fast convergence rate. However, it easily converges to local minima when solving complex problems. The main reason is that the attraction search pattern of PSO greatly depends on $p$best and $g$best. Once these best particles ($p$best and $g$best) get stuck, all particles in the swarm will quickly converge to the trapped position. To help trapped particles jump out, many improved PSO algorithms have been proposed. In [12], Shi and Eberhart introduced a inertia weight w into the original PSO to achieve a balance between the global and local search. Reported results show that a linearly decreased w is a good choice for the test suite. Bergh and Engelbrecht [13] proposed a cooperative approach for PSO (CPSO-H) for solving multimodal problems. Liang et al. [14] proposed a comprehensive learning PSO (CLPSO), in which each particle can learn other particles' experiences in different dimensions. Simulation results show that CLPSO outperforms seven other PSO algorithms. Li et al. [15] presented an adaptive learning PSO for function optimization, in which the learning mechanism of each particle is

separated into three components: its own historical best position, the closest neighbor, and the global best one. By using this individual level adaptive technique, a particle can well control its well-balanced behavior of exploration and exploitation. Zhan et al. [16] presented an adaptive PSO (APSO) by employing two following strategies. The first one evaluates the population distribution and particle fitness and identifies the current search status. The second one utilizes an elitist learning strategy to help the global best particle jump out of the likely local optima. Wang et al. [17] proposed a new PSO algorithm with generalized opposition-based learning (GOBL) and Cauchy mutation. GOBL is an enhanced opposition-based learning (OBL) [18], which is helpful to accelerate the evolution. The Cauchy mutation focuses on improving the global search ability. In [19], Wang et al. introduced a diversity enhanced PSO algorithm (DNSPSO) which employs a diversity enhancing mechanism and neighborhood search strategies to achieve a tradeoff between exploration and exploitation abilities.

Like other population-based stochastic algorithms, the performance of PSO is greatly influenced by its control parameters, such as initial weight (w) and acceleration coefficients (c_1 and c_2). Different parameter settings may result in different performance. To minimize the effects of these parameters, some adaptive parameter mechanisms have been designed [15, 16]. Recently, Sun et al. [20] proposed a novel PSO algorithm called quantum-behaved PSO (QPSO), in which a quantum model is used to depict the state of particles. Compared to the original PSO, QPSO eliminates the velocity term and does not contain parameters w, c_1, and c_2. In QPSO, new particles are generated around the weighted positions of previous best particles and the global best particle. This may result in attracting too fast. To tackle this problem, some improved QPSO algorithms have been proposed [21–24]. Sun et al. [21] proposed a diversity-guided QPSO (DGQPSO), which employs a mutation operator on the global best particle. In [22], chaotic search is introduced into QPSO to increase the diversity of swarm in the latter period of the search, so as to help the algorithm escape from local minima. Zhao et al. [23] proposed a fuzzy QPSO, in which the center of potential particle is influenced by more than two particles in the neighborhood and the influence is defined by a fuzzy variable. Wang and Zhou [24] presented a local QPSO (LQPSO) as a generalized local search operator. The LQPSO is incorporated into a main QPSO to construct a hybrid algorithm QPSO-LQPSO. Results show that QPSO-LQPSO achieves better results than PSO and QPSO.

In this paper, we also proposed a new QPSO algorithm called NQPSO, which employs one local and one global neighborhood search strategies are utilized to balance exploitation and exploration. In addition, a concept of opposition-based learning (OBL) [18] is employed for population initialization. To verify the performance of our approach, twelve well-known benchmark functions, including multimodal and rotated problems, are used in the experiments. Simulation results show that NQPSO achieves better results than some similar QPSO algorithms and other state-of-the-art PSO variants.

The rest of the paper is organized as follows. The original PSO and QPSO are briefly introduced in Sections 2 and 3, respectively. Our approach NQPSO is described in Section 4. Experimental results and discussions are presented in Section 5. Finally, the work is summarized in Section 6.

2. Particle Swarm Optimization

PSO is a population-based optimization technique, which is motivated by the behaviors of fish schooling or birds flocking. In PSO, a population is called a swarm, and each member in the swarm is called a particle which is a potential solution to the optimization task. During the evolution, the search direction of one particle is determined by its own previous best particle and the global best particle found so far by all particles.

Let N be the swarm size. Each particle i $(1 \le i \le N)$ has two vectors, velocity (V) and position (X). At each iteration, each particle in the swarm updates its velocity and position as follows [12]

$$\begin{aligned} V_{i,j}(t+1) &= w \cdot V_{i,j}(t) + c_1 \cdot r_1 \cdot \left(p\text{best}_{i,j} - X_{i,j}(t)\right) \\ &\quad + c_2 \cdot r_2 \cdot \left(g\text{best}_j - X_{i,j}(t)\right) \\ X_i(t+1) &= X_i(t) + V_i(t+1), \end{aligned} \tag{1}$$

where X_i and V_i are the position and velocity vectors of the ith particle, respectively. $p\text{best}_i$ represents the previous best particle of the ith particle and $g$best is the global best particle found so far by all particles. r_1 and r_2 are two independently generated random numbers with the range of $[0, 1]$. The parameter w is called inertia weight. c_1 and c_2 are known as acceleration coefficients.

3. Quantum-Behaved Particle Swarm Optimization

A recent theoretical study [25] reported that each particle converges to its local attractor, $p_i = (p_{i,1}, p_{i,2}, \ldots, p_{i,D})$ defined as follows:

$$p_{i,j} = \varphi \cdot p\text{best}_{i,j} + (1 - \varphi) \cdot g\text{best}_j, \tag{2}$$

where $\varphi \in (0, 1)$. It can be seen that p_i is a stochastic attractor of particle i that lies in a hyperrectangle with $p\text{best}_i$ and $g$best.

Based on the above characteristic, Sun et al. [20] proposed a quantum-behaved PSO (QPSO) algorithm. In QPSO, each particle only has position vector and does not have the velocity vector. During the evolution, each particle updates its position as follows:

$$X_{i,j}(t+1) = \begin{cases} p_{i,j}(t) + \beta \cdot \left(M\text{best}_j(t) - X_{i,j}(t)\right) \cdot \ln\left(\frac{1}{u}\right), & \text{if } h > 0.5 \\ p_{i,j}(t) - \beta \cdot \left(M\text{best}_j(t) - X_{i,j}(t)\right) \cdot \ln\left(\frac{1}{u}\right), & \text{otherwise,} \end{cases} \tag{3}$$

```
Begin
  while FEs <= MAX_FEs do
    for each particle i do
      Update the position according to (3);
      Calculate the fitness value of the new particle;
      FEs++;
    end for
    Update the pbest, gbest and p in the population;
  end while
End
```

Algorithm 1: The QPSO algorithm.

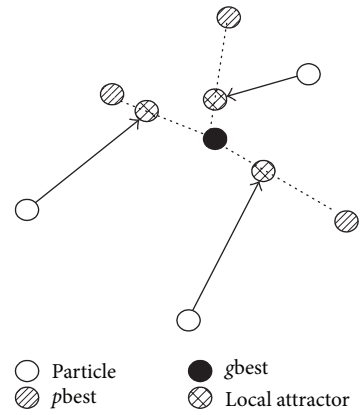

Figure 1: The search behavior of QPSO.

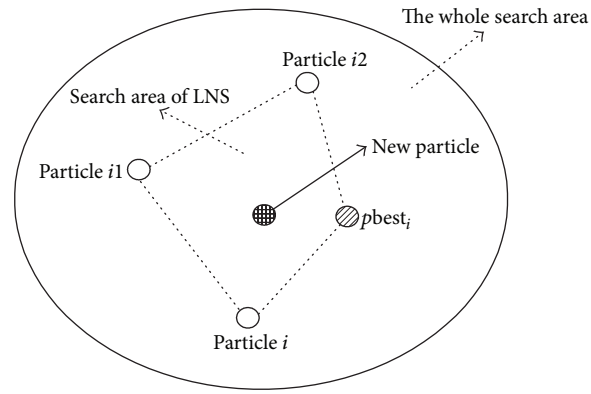

Figure 2: The local neighborhood search (LNS) strategy.

where h and u are two random numbers distributed uniformly with the range of (0,1), respectively. The parameter β is called contraction-expansion coefficient which can be tuned to control the convergence speed of the algorithm. Mbest is called mean best position of the population which is calculated by

$$M\text{best}_j(t) = \frac{1}{N}\sum_{i=1}^{N} p\text{best}_{ij}(t), \tag{4}$$

where N is the population size.

The main steps of the QPSO are described in Algorithm 1, where p is the local attractor, FEs is the number of fitness evaluations, and MAX_FEs is the maximum number of FEs. Compared to the original PSO, QPSO does not have the velocity term and the parameters, w, c_1, and c_2. But QPSO introduced a new parameter β which is linearly deceased from 1.0 to 0.5 reported in some of the literature [20, 21].

4. Proposed Approach

According to (3), new particles are generated around the local attractor p_i. It means particles move to the local attractors during the search process. The local attractors are weighted positions of $p$best and $g$best. Then, the local attractors are in the neighborhood of the $g$best. It indirectly demonstrates that particles move to the neighborhood of the $g$best. This search mechanism can obtain fast convergence speed by generating new particles in the neighborhood of $g$best. However, it may result in premature convergence because of fast attraction. Figure 1 illustrates the search behavior of QPSO.

To enhance the global search ability and avoid premature convergence, some mutation techniques have been introduced into QPSO algorithm. In [21], Sun et al. proposed a diversity-guided QPSO, in which a mutation operation is conducted on the $g$best if the diversity of swarm is smaller than a predefined value. In [26], Jamalipour et al. proposed another mutation operator inspired by the mutation scheme of DE, in which each particle updates the position according to the original quantum model or the DE mutation with equal probability.

Although the above mutation techniques can improve the global search ability of QPSO, they show poor search for local search. To make a tradeoff between global and local search, this paper proposes one local and one global neighborhood search strategies.

In the local neighborhood search (LNS) strategy, we focus on searching the local neighborhood of the current particle. This can help find more accurate solutions. The local neighborhood search strategy is defined by

$$\begin{aligned} X_i(t+1) = {} & a_1 \cdot X_i(t) + a_2 \cdot \left(p\text{best}_i - X_i(t)\right) \\ & + a_3 \cdot \left(X_{i1}(t) - X_{i2}(t)\right), \end{aligned} \tag{5}$$

where X_{i1} and X_{i2} are the position vectors of two randomly selected particles, a_1, a_2, and a_3 are three random numbers with the range of (0,1), and $a_1 + a_2 + a_3 = 1$. Figure 2 illustrates the mechanism of the local neighborhood search strategy. The proposed LNS strategy is similar to the local search operator used in [19], but they are different. The local search operator used in [19] is based on an assumed ring topology, while our approach is based on the population.

In the global neighborhood search (GNS) strategy, we concentrate on searching the global neighborhood of the current particle. This can enhance the global search and avoid premature convergence. The global neighborhood search strategy is defined by

$$X_i(t+1) = X_i(t) + \text{Levy}(), \tag{6}$$

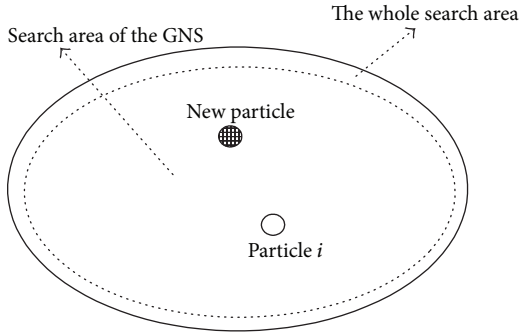

Figure 3: The global neighborhood search (GNS) strategy.

where Levy() is a random number generated by Lévy distribution with a parameter $\alpha = 1.3$ [15]. The main reason of using Lévy mutation is that the Lévy probability distribution has an infinite second moment and is, therefore, more likely to generate a new particle that is farther away from its parent than the commonly employed Gaussian mutation. Figure 3 illustrates the mechanism of the global neighborhood search strategy.

When conducting the neighborhood search, two new particles are generated by the local and global neighborhood search strategies, respectively. Then, there are three particles, the current particle and two other new particles. A greedy selection method is employed to choose the best one among the three particles as the new current particle.

Population initialization, as an important step in population-based stochastic algorithms, can affect the convergence speed and quality of solutions. In General, randomly initialization is used to generate initial population when lacking prior information. By the suggestions of [27], replacing the random initialization with opposition-based learning (OBL) can obtain better initial solutions and accelerate convergence speed. So, this paper also employs OBL for population initialization. This method is described as follows.

(1) Randomly generate N particles to initialize the population P.

(2) Calculate the boundaries $[a_j, b_j]$ of the current population according to (7)

$$a_j = \min\left(X_{i,j}\right), \qquad b_j = \max\left(X_{i,j}\right). \tag{7}$$

(3) For each particle in P, an opposite particle is generated by

$$X^*_{i,j} = a_j + b_j - X_{i,j}, \tag{8}$$

where X^*_i is the opposite position of X_i. After conduct the opposition, there are N opposite particles, which form an opposite population OP.

(4) Select N fittest particles from P and OP as the initial population.

In our approach NQPSO employs one local and one global neighborhood search strategies into the original QPSO. The neighborhood search strategies focus on searching the local and global neighbors of particles and making a balance between the local and global search. The opposition-based population initialization can generate high quality of initial solutions and accelerate the convergence speed.

The main steps of NQPSO are described in Algorithm 2, where rand(0,1) is a random number with the range of $[0, 1]$, the parameter q is the probability of neighborhood search, FEs is the number of fitness evaluations, and MAX_FEs is the maximum number of FEs. Compared to the original QPSO, our approach NQPSO does not add extra loop operations. Therefore, both NQPSO and QPSO have the same computational time complexity.

5. Experimental Study

5.1. Test Problems. There are twelve benchmark functions used in the following experiments. These problems were utilized in previous studies [14, 17, 19]. According to their properties, they are divided into three groups: unimodal and simple multimodal problems (f_1-f_2), unrotated multimodal problems (f_3–f_8), and rotated multimodal problems (f_9–f_{12}). For rotated problems, the original variable x is left multiplied by the orthogonal matrix $\mathbf{M}$ to get the new rotated variable $\mathbf{y} = \mathbf{M} * \mathbf{x}$. All problems used in this paper are minimization problems. The brief descriptions of these problems are presented in Table 1.

5.2. Comparison of NQPSO with Other Similar QPSO Algorithms. Since the introducing of QPSO, some improved QPSO algorithms have been proposed. In order to verify the effectiveness our approach, this section compares NQPSO with similar QPSO algorithms, including QPSO, diversity-guided QPSO (DGQPSO) [21], and QPSO with weighted mean best position (WQPSO) [28].

To have a fair competition, the same settings are used for common parameters. For all algorithms, the population size N is set to 40. The parameter β is linearly deceased from 1.0 to 0.5. For DGQPSO, the parameter d_{low} is set to 0.0001, and the coefficient γ used in the mutation is equal to 0.00001. For NQPSO, the probability q of the neighborhood search is set to 0.2 based on empirical studies. When the number of fitness evaluations (FEs) reaches to the maximum value MAX_FEs, the algorithm stops running. In the experiment, MAX_FEs is set to $2.0e + 05$. All algorithms are run 30 times for each test problem. Throughout the experiments, the mean fitness error values and standard deviation are reported (the mean error value is defined as $f(x) - f(\text{opt})$, where $f(x)$ is the fitness value found in the last generation, and $f(\text{opt})$ is the global optimum of the problem).

Table 2 presents the computational results of QPSO, DGQPSO, WQPSO, and NQPSO on the test suite, where "Mena" indicates the mean fitness error value, and "Std" represents the standard deviation. For each problem, the best result (the minimal value) is shown in bold. It can be seen that DGQPSO outperforms QPSO on f_5. In this problem, all four

```
Begin
  Use opposition-based learning to generate initial population;
  while FEs <= MAX_FEs do
    for each particle i do
      Update the position according to (3);
      Calculate the fitness value of the new particle;
      FEs++;
      if rand(0, 1) < q then
        Generate a new particle according to (5);
        Generate a new particle according to (6);
        Calculate the fitness values of the two new particles;
        Fes = Fes + 2;
        Select the fittest one among particle i and two new particles as the new particle i;
      end if
      Update the pbest, gbest and p in the population;
    end for
  end while
End
```

Algorithm 2: The proposed NQPSO algorithm.

Table 1: Benchmark problems used in the experiments.

Problems	D	Search range
$f_1(x) = \sum_{i=1}^{D} x_i^2$	30	$[-100, 100]$
$f_2(x) = \sum_{i=1}^{D-1} \left[100\left(x_i^2 - x_{i+1}\right)^2 + \left(x_i - 1\right)^2\right]$	30	$[-2.048, 2.048]$
$f_3(x) = -20\exp\left(-0.2\sqrt{1/D\sum_{i=1}^{D} x_i^2}\right) - \exp\left(1/D\sum_{i=1}^{D}\cos\left(2\pi x_i\right)\right) + 20 + e$	30	$[-32.768, 32.768]$
$f_4(x) = \sum_{i=1}^{D} x_i^2/4000 - \prod_{i=1}^{D}\cos\left(x_i/\sqrt{i}\right) + 1$	30	$[-600, 600]$
$f_5(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k\max}\left[a^k\cos\left(2\pi b^k\left(x_i + 0.5\right)\right)\right]\right) - D\sum_{k=0}^{k\max}\left[a^k\cos\left(2\pi b^k\cdot 0.5\right)\right]$, $a = 0.5$, $b = 3$, $k\max = 20$	30	$[-0.5, 0.5]$
$f_6(x) = \sum_{i=1}^{D}\left(x_i^2 - 10\cos\left(2\pi x_i\right) + 10\right)$	30	$[-5.12, 5.12]$
$f_7(x) = \sum_{i=1}^{D}\left(y_i^2 - 10\cos\left(2\pi y_i\right) + 10\right)$ $y_i = \begin{cases} x_i, & \lvert x_i\rvert < 1/2 \\ \text{round}(2x_i)/2, & \lvert x_i\rvert \geq 1/2 \end{cases}$	30	$[-5.12, 5.12]$
$f_8(x) = 418.9829\cdot D - \sum_{i=1}^{D}\left(x_i\sin\left(\sqrt{\lvert x_i\rvert}\right)\right)$	30	$[-500, 500]$
$f_9(x) = -20\exp\left(-0.2\sqrt{1/D\sum_{i=1}^{D} y_i^2}\right) - \exp\left(1/D\sum_{i=1}^{D}\cos\left(2\pi y_i\right)\right) + 20 + e$ $\mathbf{y} = \mathbf{M} * \mathbf{x}$	30	$[-32.768, 32.768]$
$f_{10}(x) = \sum_{i=1}^{D} y_i^2/4000 - \prod_{i=1}^{D}\cos\left(y_i/\sqrt{i}\right) + 1$ $\mathbf{y} = \mathbf{M} * \mathbf{x}$	30	$[-600, 600]$
$f_{11}(x) = \sum_{i=1}^{D}\left(\sum_{k=0}^{k\max}\left[a^k\cos\left(2\pi b^k\left(y_i + 0.5\right)\right)\right]\right) - D\sum_{k=0}^{k\max}\left[a^k\cos\left(2\pi b^k\cdot 0.5\right)\right]$ $\mathbf{y} = \mathbf{M} * \mathbf{x}$	30	$[-0.5, 0.5]$
$f_{12}(x) = \sum_{i=1}^{D}\left(y_i^2 - 10\cos\left(2\pi y_i\right) + 10\right)$ $\mathbf{y} = \mathbf{M} * \mathbf{x}$	30	$[-5.12, 5.12]$

QPSO algorithms can find the global optimum. Although the diversity-guided method can improve the performance of QPSO, it only achieves small advantages on most test problems. Like DGQPSO, WQPSO performs better than QPSO on all problems except for f_5, but the weighted best position method can significantly improve the performance of QPSO on f_{10} and f_{11}. NQPSO significantly improves the performance of QPSO on f_1, f_4, f_6, and f_{10}–f_{13}. Especially for f_4, f_6, and f_{10}–f_{13}, only NQPSO can converge to the global optimum, while other algorithms fall into local minima.

To observe the evolutionary processes of the algorithms, Figure 4 lists the convergence characteristics of QPSO, DGQPSO, WQPSO, and NQPSO on some representative

Table 2: Computational results achieved by QPSO, DGQPSO, WQPSO, and NQPSO.

F	QPSO		DGQPSO		WQPSO		NQPSO	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
f_1	$1.88E-73$	$2.67E-73$	$4.98E-75$	$1.22E-75$	$1.74E-80$	$1.50E-80$	**2.23E − 165**	**0.00E + 00**
f_2	$2.53E+01$	$9.89E-02$	$2.14E+01$	$3.75E-01$	$2.13E+01$	$3.23E-01$	**1.92E + 01**	**4.32E − 01**
f_3	$1.12E-14$	$7.52E-15$	$8.87E-15$	$1.95E-15$	$7.69E-15$	$0.00E+00$	**5.89E − 16**	**0.00E + 00**
f_4	$2.95E-02$	$3.26E-02$	$9.85E-03$	$2.04E-03$	$7.39E-03$	$8.71E-03$	**0.00E + 00**	**0.00E + 00**
f_5	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
f_6	$9.55E+00$	$3.19E+00$	$7.92E+00$	$2.25E+00$	$4.71E+00$	$1.72E+00$	**0.00E + 00**	**0.00E + 00**
f_7	$2.63E+01$	$1.46E+01$	$1.70E+01$	$1.23E+01$	$1.55E+01$	$1.69E+01$	**1.31E + 00**	**2.27E + 00**
f_8	$7.54E+03$	$1.13E+03$	$6.70E+03$	$1.02E+03$	$7.12E+03$	$4.92E+02$	**3.80E + 03**	**9.37E + 02**
f_9	$1.32E-14$	$8.25E-15$	$9.83E-15$	$1.95E-15$	$7.69E-15$	$0.00E+00$	**5.89E − 16**	**0.00E + 00**
f_{10}	$7.41E-03$	$3.31E-03$	$1.12E-16$	$2.37E-16$	$2.24E-16$	$3.51E-16$	**0.00E + 00**	**0.00E + 00**
f_{11}	$3.76E-02$	$3.09E-02$	$1.92E-02$	$2.94E-02$	$1.09E-05$	$9.43E-06$	**0.00E + 00**	**0.00E + 00**
f_{12}	$3.09E+01$	$2.42E+01$	$2.67E+01$	$1.72E+01$	$1.36E+01$	$1.04E+01$	**0.00E + 00**	**0.00E + 00**

Table 3: Computational results achieved by NQPSO and other five PSO algorithms.

Problems	CPSO-H	CLPSO	APSO	GOCLPSO	DNSPSO	NQPSO
	Mean	Mean	Mean	Mean	Mean	Mean
f_1	$1.29E-36$	$1.23E-13$	$9.60E-66$	$3.22E-22$	$2.88E-95$	**2.23E − 165**
f_2	**1.37E + 01**	$2.08E+01$	$1.83E+01$	$2.54E+01$	$1.73E+01$	$1.92E+01$
f_3	$2.25E-14$	$1.85E-07$	$1.09E-14$	$2.93E-12$	**5.89E − 16**	**5.89E − 16**
f_4	$1.90E-02$	$4.37E-09$	$1.20E-02$	**0.00E + 00**	**0.00E + 00**	**0.00E + 00**
f_5	$4.74E-15$	$5.62E-07$	$4.77E-02$	$1.67E-10$	$1.79E-13$	**0.00E + 00**
f_6	$3.32E+00$	$1.50E-04$	$6.27E+00$	$1.48E+01$	**0.00E + 00**	**0.00E + 00**
f_7	$6.67E-01$	$1.93E-03$	$2.27E+00$	$1.85E+01$	**0.00E + 00**	$1.31E+00$
f_8	$2.65E+03$	$3.88E-04$	$3.74E+02$	**5.20E − 04**	$4.86E+03$	$3.80E+03$
f_9	$1.82E-01$	$1.07E-05$	$1.22E+00$	$8.86E-12$	**5.89E − 16**	**5.89E − 16**
f_{10}	$2.30E-02$	$6.49E-05$	$1.38E-02$	$0.00E+00$	**0.00E + 00**	**0.00E + 00**
f_{11}	$8.33E+00$	$2.99E+00$	$8.40E+00$	$5.26E-08$	$3.72E+01$	**0.00E + 00**
f_{12}	$7.33E+01$	$5.48E+01$	$7.09E+01$	$7.02E+01$	**0.00E + 00**	**0.00E + 00**

problems. As seen, NQPSO shows faster convergence speed than other three QPSO algorithms. Especially for f_4, f_{10}, and f_{12}, NQPSO find the global optimum at the beginning stage of the evolution, while other three QPSO algorithms shows slow convergence rate. Although DGQPSO and WQPSO achieve better results than the original QPSO, the convergence characteristics of them are similar.

5.3. Comparison of NQPSO with Other State-of-the-Art PSO Algorithms. To further verify the performance of our approach, this section presents a comparative study of NQPSO with other state-of-the-art PSO variants. These PSO algorithms are listed as follows.

(i) Cooperative PSO (CPSO-H) [13].

(ii) Comprehensive learning PSO (CLPSO) [14].

(iii) Adaptive learning PSO (APSO) [16].

(iv) Comprehensive learning PSO with generalized opposition-based learning (GOCLPSO) [29].

(v) Diversity enhanced PSO with neighborhood search (DNSPSO) [19].

(vi) Our approach NQPSO.

The parameter settings of CPSO-H and CLPSO are described in [14]. By the suggestions of [16], the same parameter settings of APSO are used. The parameters w is set to 0.7298, and $c_1 = c_2 = 1.49618$. For GOCLPSO, the probability of opposition is set to 0.3 and other parameters keep the same

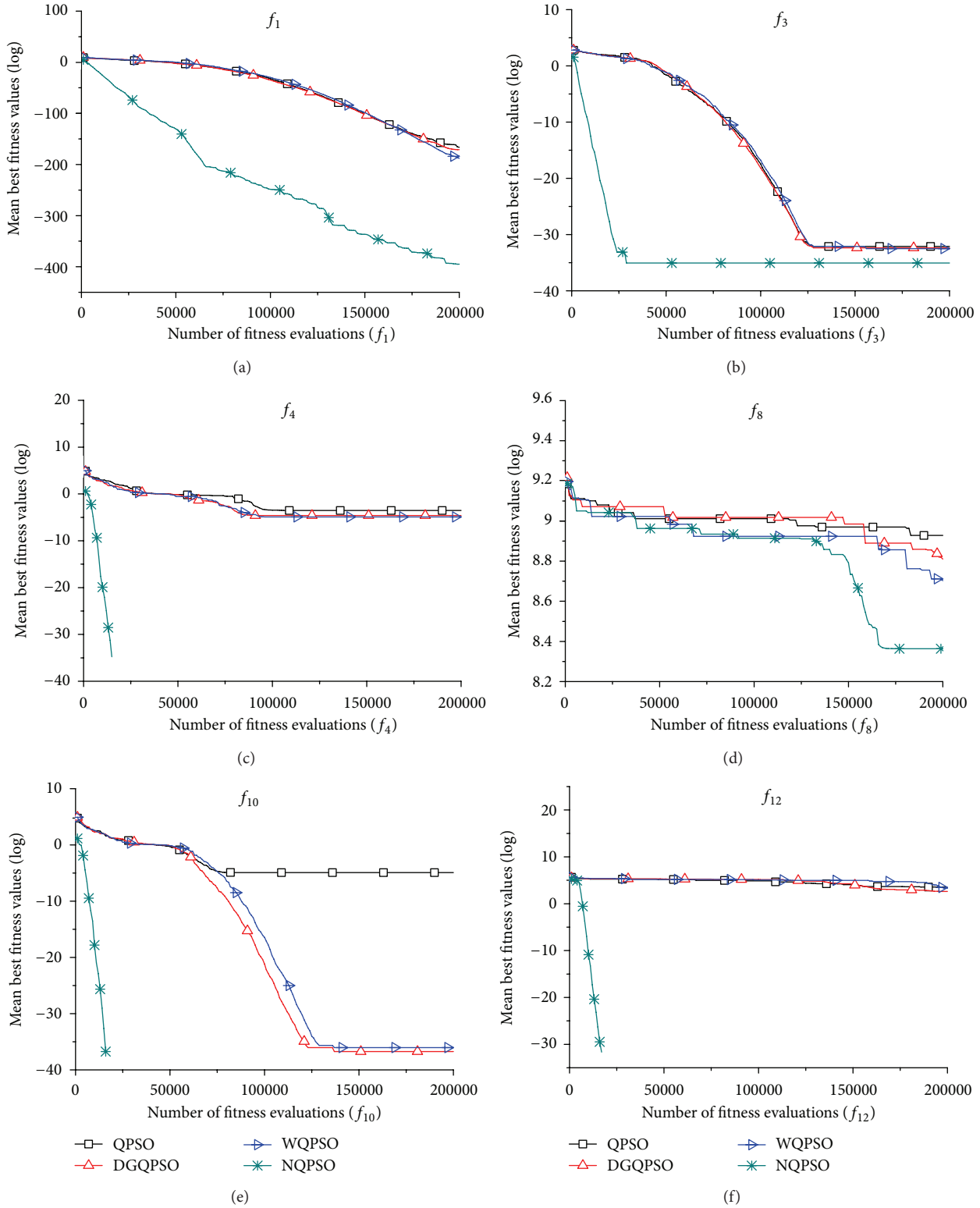

Figure 4: The convergence characteristics of four QPSO algorithms on some representative problems.

Table 4: Average rankings of the six PSO algorithms.

Algorithms	Average ranking value
NQPSO	**2.17**
DNSPSO	2.50
CLPSO	3.83
GOCLPSO	3.92
CPSO-H	4.08
APSO	4.50

Table 5: Results of Wilcoxon signed-rank test between NQPSO and other five PSO algorithms.

NQPSO versus	P values
CPSO-H	$3.88e-01$
CLPSO	$1.36e-01$
APSO	$9.95e-02$
GOCLPSO	$7.45e-02$
DNSPSO	$4.63e-01$

with CLPSO. The parameters k, p_r, and p_{ns} used in DNSPSO are set to 2, 0.9, and 0.6, respectively [19]. The above six PSO algorithms use the same population size ($N = 40$) and maximum number of fitness evaluations (MAX_FEs = $2.0e+05$). For each test problem, each algorithm is run 30 times and the mean fitness error values are recorded.

Table 3 presents the computational results achieved by CPSO-H, CLPSO, APSO, GOCLPSO, DNSPSO, and NQPSO, where "Mean" indicates the mean fitness error values. For each problem, the best result is shown in bold. From the results, it can be seen that NQPSO outperforms CPSO-H on all test problems except for f_2, f_7, and f_8. CLPSO performs better than NQPSO on f_7, and f_8, while NQPSO achieves better results on the rest 10 problems. APSO outperforms NQPSO on f_2 and f_8, while NQPSO performs better than QPSO on the rest 10 problems. GOCLPSO, DNSPSO, and NQPSO can find the global optimum on f_4. NQPSO performs better than GOCLPSO on 10 problems. Both DNSPSO and NQPSO achieve the same results on f_3, f_4, f_6, f_9, f_{10}, and f_{12}. DNSPSO outperforms NQPSO on f_2 and f_7, while NQPSO obtains better results on f_1, f_5, f_8, and f_{11}. From the comparison of DNSPSO and NQPSO, both of them employ neighborhood search strategies, but NQPSO shows better performance than DNSPSO on the majority of test problems.

In order to compare the performance of multiple algorithms on the test suite, we conduct Friedman test by the suggestions of [19]. Table 4 presents the average ranking of CPSO-H, CLPSO, APSO, GOCLPSO, DNSPSO, and NQPSO. These ranking values are calculated by SPSS software. The best ranking having the lowest ranking value is shown in bold. As seen, the performance of the six algorithms can be sorted into the following order: NQPSO, DNSPSO, CLPSO, GOCLPSO, CPSO-H, and APSO. The best average ranking was obtained by NQPSO algorithms, which outperforms the other five PSO algorithms. According to the literature [29], GOCLPSO is better than CLPSO, but our results show that CLPSO is better than GOCLPSO. The main reason is that the benchmark functions tested in this paper are different from the ones in [29]. For different test problems, one algorithm may show different performance.

Besides the Friedman test, we also conduct Wilcoxon signed-rank test to compare the performance differences between NQPSO and the other five PSO algorithms [19, 30]. Table 5 shows the P values when comparing NQPSO with other algorithms. The results show that NQPSO is not significantly better than other algorithms, but NQPSO outperforms them according to the average rankings shown in Table 4.

6. Conclusions

Quantum-behaved PSO (QPSO) is a new PSO variant, which employs a quantum model to update the positions of particles. Compared to the original PSO, QPSO eliminates the velocity term and does not contain the related parameters, w, c_1, and c_2. Although QPSO introduces a new parameter to control the step size, it still has fewer control parameters than PSO. Some recent studies show that QPSO performs better than the original PSO on many benchmark functions and real-world problems. However, both PSO and QPSO still easily fall into local minima when solving complex problems. The main reason is that particles tends to move to the neighborhood of the $g$best by the attraction of the weighted of $p$best and $g$best. If particles are attracted too fast, premature convergence will easily occur. To tackle this problem, this paper proposes a new QPSO algorithm called NQPSO, which employs one local and one global neighborhood search strategies to make a balance between exploitation and exploration. Moreover, a concept of opposition-based learning (OBL) is employed for population initialization. To verify the performance of our approach, twelve well-known benchmark functions including multimodal and rotated problems are used in the experiments. Computation results show that NQPSO outperforms some similar QPSO algorithms and five other state-of-the-art PSO variants.

Although NQPSO significantly improves the performance of QPSO on many problems, it still falls into local minima on some problems, such as f_2, f_7, and f_8. How to enhance the performance of NQPSO on these problems will be a possible research direction. In addition, a new parameter q is introduced to control the frequency of conducting neighborhood search. We have not investigated the effects of this parameter on the performance of NQPSO. How to select the best q will be another research direction in our future work.

Appendix

The Orthogonal Matrix M

The orthogonal matrix $\mathbf{M}$ is 30×30

M=[0.6576, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -0.0008, 0.3072, 0, 0, 0, 0, 0, 0, 0, -0.1604, 0, 0,
0.0993, -0.6506, 0, 0, -0.1199; 0, 0.2019, 0.1866, 0, 0, 0, -0.0393, 0, 0, -0.0114, 0, 0, 0, -0.6414,
0, 0, 0, 0, 0, 0, 0, -0.0587, 0, 0, -0.3029, 0, -0.1161, 0, 0, 0.6346; 0, -0.3973, 0.1121, 0, 0, 0,
0.0772, 0, 0.2367, 0.0224, 0.4986, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.1156, 0, 0, 0.596, 0, -0.0702, 0, 0,
0.3809; 0, 0, 0, 0.9513, 0, 0, 0, 0, 0.1644, 0, -0.0781, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.249, 0, 0,
0, 0, 0, 0; 0, 0, 0, 0, 0.5048, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.0283, 0.0106, -0.0096, 0, 0, 0, 0, 0, 0,
0, 0, -0.0179, 0.8625, 0; 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0; 0, 0, 0, 0, 0, 0, 0, 0.8361, 0, 0, -0.1042, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -0.5386, 0, 0, 0, 0, 0, 0, 0,
0; 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0; 0, 0.0096, 0,
-0.176, 0, 0, -0.0019, 0, 0.8882, -0.0005, -0.4216, 0, -0.0092, 0, 0, 0, 0, 0, 0, 0, 0, -0.0028, 0,
-0.0461, 0.0072, 0, 0, 0, 0, 0; 0, 0.3243, 0.128, 0, 0, 0, 0.2608, 0, 0, 0.0755, 0, 0, 0, 0.6295,
0.048, 0.0806, 0, 0, 0, 0, 0, 0.3902, 0.092, 0, -0.2077, 0, -0.0809, 0, 0, 0.435; 0, 0.2629,
-0.0742, 0, 0, 0, -0.0511, 0, 0.3576, -0.0148, 0.7534, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, -0.0765, 0, 0,
-0.3944, 0, 0.0464, 0, 0, -0.2521; 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.2997, 0, 0, 0, 0, 0, 0, 0, -0.954,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0; 0, -0.6198, 0, -0.0027, 0, 0, 0.1205, 0, 0.0138, 0.0349, -0.0065, 0,
0.5926, 0, 0, 0, 0, 0, 0, 0, 0, 0.1803, 0, -0.0007, -0.4651, 0, 0, 0, 0, 0; 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0.0006, 0.7852, 0, 0, 0, 0, 0, 0, 0, -0.41, 0, 0, 0, 0.4564, 0, 0, 0.0841; -0.7385, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, -0.0007, 0.2736, 0, 0, 0, 0, 0, 0, 0, -0.1429, 0, 0, -0.1115, -0.5793, 0, 0,
-0.1067; 0, -0.043, -0.017, 0, 0, 0, -0.0346, 0, 0, -0.01, 0, 0, 0, -0.0834, 0.3624, 0.6081, 0, 0, 0,
0, 0, -0.0517, 0.694, 0, 0.0275, 0, 0.0107, 0, 0, -0.0577; 0, 0, 0, 0, 0, 0, 0, 0, 0, -0.9218, 0, 0, 0,
0, 0, 0, 0.1512, 0.0564, -0.0514, 0, 0, 0.1784, 0, 0, 0, 0, 0, 0.2995, 0, 0; 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, -0.3497, 0.9369, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0; 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0.2777, 0.1036, 0.9309, 0, -0.2136, 0, 0, 0, 0, 0, 0, 0, 0, 0; 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0.954, 0, 0, 0, 0, 0, 0, 0, 0.2997, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0; 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0.0607, 0.0227, 0.2035, 0, 0.9769, 0, 0, 0, 0, 0, 0, 0, 0, 0; 0, 0, 0, 0, 0, 0, 0, 0, 0, -0.3378, 0, 0,
0, 0, 0, 0, -0.4125, -0.154, 0.1402, 0, 0, 0.0654, 0, 0, 0, 0, 0, -0.8174, 0, 0; 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0.2839, -0.7898, 0, 0, 0, 0, 0, 0, 0.5437, 0, 0, 0, 0, 0, 0, 0; 0, 0, 0, -0.2532, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.9674, 0, 0, 0, 0, 0, 0; 0, 0.4561, 0, 0, 0, 0, -0.0887, 0,
0, -0.0257, 0, 0, 0.8055, 0, 0, 0, 0, 0, 0, 0, 0, -0.1327, 0, 0, 0.3422, 0, 0, 0, 0, 0; -0.1493, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.9888, 0, 0, 0, 0; 0, 0, 0.9606, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.0503, 0, 0, -0.2733; 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
0, 0, 0, 0, 0, -0.7764, -0.2898, 0.2639, 0, 0, 0, 0, 0, 0, 0, 0, 0.4917, 0.0422, 0; 0, 0, 0, 0, 0, -0.8633,
0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0.0166, 0.0062, -0.0056, 0, 0, 0, 0, 0, 0, 0, 0, -0.0105, 0.5043, 0; 0,
0.1817, -0.0876, 0, 0, 0, 0.4464, 0, 0, 0.1292, 0, 0, 0, -0.4306, 0, 0, 0, 0, 0, 0, 0, 0.6679, 0, 0,
0.1421, 0, 0.0554, 0, 0, -0.2976;]

Acknowledgments

This work is supported by the Twelfth Five-Year Plan of Jilin Province Education Science Project (no. GH11466).

References

[1] D. E. Goldberg, *Genetic Algorithms in Search Optimization and Machine Learning*, Addison-Wesley, Reading, Mass, USA, 1989.

[2] L. J. Fogel, "Evolutionary programming in perspective: the top-down view," in *Computational Intelligence: Imitating Life*, IEEE Press, Piscataway, NJ, USA, 1994.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the 1995 IEEE International Conference on Neural Networks*, pp. 1942–1948, December 1995.

[4] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, Natural Computing Series, Springer, Berlin, Germany, 1st edition, 2005.

[5] M. Dorigo, V. Maniezzo, and A. Colorni, "Ant system: optimization by a colony of cooperating agents," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 26, no. 1, pp. 29–41, 1996.

[6] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," Tech. Rep. TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005.

[7] L. Ali, S. L. Sabat, and S. K. Udgata, "Particle swarm optimisation with stochastic ranking for constrained numerical and engineering benchmark problems," *International Journal of Bio-Inspired Computation*, vol. 4, no. 3, pp. 155–166, 2012.

[8] C. C. Tseng, J. G. Hsieh, and J. H. Jeng, "Active contour model via multi-population particle swarm optimization," *Expert Systems with Applications*, vol. 36, no. 3, pp. 5348–5352, 2009.

[9] K. W. Yu and Z. L. Huang, "LQ regulator design based on particle swarm optimization," in *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, pp. 4142–4145, Taipei, Taiwan, October 2006.

[10] C. Priya and P. Lakshmi, "Particle swarm optimisation applied to real time control of spherical tank system," *International*

Journal of Bio-Inspired Computation, vol. 4, no. 4, pp. 206–216, 2012.

[11] H. Wang, "Opposition-based barebones particle swarm for constrained nonlinear optimization problems," *Mathematical Problems in Engineering*, vol. 2012, Article ID 761708, 12 pages, 2012.

[12] Y. Shi and R. Eberhart, "Modified particle swarm optimizer," in *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC '98)*, pp. 69–73, May 1998.

[13] F. van den Bergh and A. P. Engelbrecht, "A cooperative approach to participle swam optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 225–239, 2004.

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

[15] C. Li, S. Yang, and T. T. Nguyen, "A self-learning particle swarm optimizer for global optimization problems," *IEEE Transactions on Systems, Man, and Cybernetics B*, vol. 42, no. 3, pp. 627–646, 2012.

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

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

[18] H. Tizhoosh, "Opposition-based reinforcement learning," *Journal of Advanced Computational Intelligence and Intelligent Informatics*, vol. 10, no. 4, pp. 578–585, 2006.

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

[20] J. Sun, B. Feng, and W. B. Xu, "Particle swarm optimization with particles having quantum behavior," in *Proceedings of the IEEE Congress on Evolutionary Computation (CEC '04)*, pp. 325–331, June 2004.

[21] J. Sun, W. B. Xu, and W. Fang, "A diversity-guided quantum-behaved particle swarm optimization algorithm," in *Simulated Evolution and Learning*, vol. 4247 of *Lecture Notes in Computer Science*, pp. 497–504, Springer, New York, NY, USA, 2006.

[22] K. Yang and H. Nomura, "Quantum-behaved particle swarm optimization with chaotic search," *IEICE Transactions on Information and Systems*, vol. 91, no. 7, pp. 1963–1970, 2008.

[23] W. Zhao, Y. San, and H. Shi, "Fuzzy quantum-behaved particle swarm optimization algorithm," in *Proceedings of the International Symposium on Computational Intelligence and Design (ISCID '10)*, pp. 49–52, Hangzhou, China, October 2010.

[24] J. Wang and Y. Zhou, "Quantum-behaved particle swarm optimization with generalized local search operator for global optimization," in *Advanced Intelligent Computing Theories and Applications: With Aspects of Artificial Intelligence*, vol. 4682 of *Lecture Notes in Computer Science*, pp. 851–860, Springer, New York, NY, USA, 2007.

[25] F. van den Bergh and A. P. Engelbrecht, "A study of particle swarm optimization particle trajectories," *Information Sciences*, vol. 176, no. 8, pp. 937–971, 2006.

[26] M. Jamalipour, R. Sayareh, M. Gharib, F. Khoshahval, and M. R. Karimi, "Quantum behaved particle swarm optimization with differential mutation operator applied to WWER-1000 in-core fuel management optimization," *Annals of Nuclear Energy*, vol. 54, pp. 134–140, 2013.

[27] S. Rahnamayan, H. R. Tizhoosh, and M. M. A. Salama, "A novel population initialization method for accelerating evolutionary algorithms," *Computers and Mathematics with Applications*, vol. 53, no. 10, pp. 1605–1614, 2007.

[28] M. Xi, J. Sun, and W. Xu, "An improved quantum-behaved particle swarm optimization algorithm with weighted mean best position," *Applied Mathematics and Computation*, vol. 205, no. 2, pp. 751–759, 2008.

[29] W. Wang, H. Wang, and S. Rahnamayan, "Improving comprehensive learning particle swarm optimiser using generalised opposition-based learning," *International Journal of Modelling, Identification and Control*, vol. 14, no. 4, pp. 310–316, 2011.

[30] H. Wang, S. Rahnamayan, H. Sun, and M. G. H. Omran, "Gaussian bare-bones differential evolution," *IEEE Transactions on Cybernetics*, vol. 43, no. 2, pp. 634–647, 2013.