Research Article

A Novel Multiobjective Quantum-Behaved Particle Swarm Optimization Based on the Ring Model

Di Zhou,¹ Yajun Li,¹ Bin Jiang,¹ and Jun Wang^{2,3,4}

¹School of Design Art & Media, Nanjing University of Science and Technology, China

²School of Digital Media, Jiangnan University, China

³Department of Radiology and BRIC, University of North Carolina at Chapel Hill, Chapel Hill, NC, USA

⁴Fujian Provincial Key Laboratory of Information Processing and Intelligent Control, Minjiang University, Fuzhou, China

Correspondence should be addressed to Di Zhou; zhoudi@njust.edu.cn

Received 20 August 2016; Revised 26 September 2016; Accepted 10 October 2016

Academic Editor: Cheng-Tang Wu

Copyright © 2016 Di Zhou et al. This is an open access article distributed under the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Due to its fast convergence and population-based nature, particle swarm optimization (PSO) has been widely applied to address the multiobjective optimization problems (MOPs). However, the classical PSO has been proved to be not a global search algorithm. Therefore, there may exist the problem of not being able to converge to global optima in the multiobjective PSO-based algorithms. In this paper, making full use of the global convergence property of quantum-behaved particle swarm optimization (QPSO), a novel multiobjective QPSO algorithm based on the ring model is proposed. Based on the ring model, the position-update strategy is improved to address MOPs. The employment of a novel communication mechanism between particles effectively slows down the descent speed of the swarm diversity. Moreover, the searching ability is further improved by adjusting the position of local attractor. Experiment results show that the proposed algorithm is highly competitive on both convergence and diversity in solving the MOPs. In addition, the advantage becomes even more obvious with the number of objectives increasing.

1. Introduction

Optimization problems with more than one objective are rather common in real-world practice, such as information system design [1], reservoir flood control operation (RFCO) problem [2], community detection [3] in social networks, and battery hybrid storage system optimization problems [4]. In such multiobjective optimization problems (MOPs), the objectives to be optimized are normally in conflict with each other, which means there is no unique solution to these problems. Instead, we are supposed to find Pareto optimal solutions that represent the best possible compromises among all the objectives.

In recent years, due to their population-based nature, a variety of evolutionary algorithms are applied to address the MOPs. Among these algorithms, particle swarm optimization (PSO) has attracted great interest for its relatively simple operation and competitive performance. Since the

first multiobjective PSO (MOPSO) proposed in 1999 [5], more than fifty variants of MOPSOs have been reported in literature, among which OMOPSO [6] proposed by Sierra and Coello is one of the most representative methods. Sierra and Coello [7] had given a survey of the existing studies on OMOPSOs before 2006, and the state-of-the-art MOPSOs are summarized by Zhou et al. [8]. Since classical PSO is designed for single-objective optimization problems and cannot be applied to multiobjective optimization problems directly, most of the existing studies have focused on how to extend PSO to its multiobjective versions, such as researches on how to select the global and local best particles [9–11], as well as how to maintain good points found so far.

However, as proved by Van Den Bergh [12], the classical PSO is not a global search algorithm, not even a local one, according to the convergence criteria provided by Solis and Wets [13]. Therefore, MOPSOs, which are derived from PSO, are unable to converge to global optima.

Quantum-behaved particle swarm optimization (QPSO) [14], first introduced by Sun et al. in 2004, is a new population-based algorithm, which is inspired by quantum mechanics and the trajectory analysis of PSO. Besides the introduction of mean best position (*mbest*), the particles in QPSO are assumed to follow a double exponential distribution in a quantum δ potential well around its local focus when a new position is sampled, which is the most significant difference between QPSO and PSO. Therefore, QPSO needs no velocity vectors for particles at all. Since its first proposal, QPSO has shown its success in solving a wide range of single-objective optimization problems [15–17].

In contrast with PSO, the global convergence of QPSO can be guaranteed if the contraction-expansion (CE) coefficient of the algorithm is properly selected [18, 19]. Sun et al. proved that the QPSO is a form of contraction mapping on the probability metric space and its orbit is probabilistic bounded, and, in turn, the algorithm converges asymptotically to the global optimum. It is the exact reason why QPSO outperforms PSO as well as most of the other evolutionary algorithms.

Although QPSO has been successfully applied in conventional single-objective optimization problems due to its global convergence and easy control, it is rarely used in solving multiobjective optimization problems [20, 21]. Although the QPSO algorithm can be global convergent, the CE coefficient is generally selected to be relatively small in order to accelerate the convergence of the algorithm for real-world problems so that premature convergence can result when the algorithm is performed for the MOPs. To eliminate this defect, we propose a novel position-update mechanism based on the ring model and combine it with the classical QPSO. This leads to MOQPSOr, an enhanced QPSO method which can be applied to the multiobjective optimization problems. The combination of the ring model with QPSO has several merits. Firstly, it employs a novel communication mechanism between particles using the ring model. This modification enables the swarm to have much larger mutate scope compared to the original QPSO, which effectively slows down the descent speed of the swarm diversity, solving the problem of premature caused by the quick convergence when applying QPSO directly into multiobjective optimization. Secondly, in this ring model, by adjusting the position of local attractor, the global searching ability is enhanced at the beginning of iteration, while the local searching ability is enhanced in the later stage of iteration. By employing this novel position-update strategy based on the ring model, the efficiency of MOQPSOr on multiobjective optimization is further improved, since there is no need for any additional mutation operation.

The rest of the paper is organized as follows: After a brief introduction of the background of PSO and QPSO in Section 2, a novel ring model for position update is proposed in Section 3 and a new version of multiobjective quantum-behaved particle swarm optimization algorithm (MOQPSOr) is presented by integrating the new position-update strategy into it accordingly. Numerical tests and performance comparison on 12 benchmark functions are provided in Section 4. Finally, the paper is concluded in Section 5.

2. Related Work

Being a heuristic search technique that simulates the sociology behaviour of an organism, particle swarm optimization (PSO) [22, 23] has become one of the most popular methods in the fields of evolutionary computation. In PSO, each particle represents a candidate solution to the problem and flies through a D -dimensional search space according to the following position-update equation:

$$\begin{aligned} V_{i,j}(t+1) &= \omega \cdot V_{i,j}(t) + c_1 \cdot r_{1,j}(t) \\ &\quad \cdot [pbest_{i,j}(t) - X_{i,j}(t)] + c_2 \cdot r_{2,j}(t) \\ &\quad \cdot [gbest_j(t) - X_{i,j}(t)], \\ X_i(t+1) &= X_i(t) + V_i(t+1), \end{aligned} \quad (1)$$

where the current position and velocity of i th particle at the t th iteration are represented, respectively, as $X_i(t) = (X_{i,1}(t), X_{i,2}(t), \dots, X_{i,D}(t))$ and $V_i(t) = (V_{i,1}(t), V_{i,2}(t), \dots, V_{i,D}(t))$. $pbest_i$ is the best previous position of particle i , while $gbest$ is the position of the best particle in the whole swarm. The parameters r_1 and r_2 are different random numbers distributed uniformly on (0,1), and c_1 as well as c_2 denote the acceleration coefficients that typically are both set to a value of 2.0, which implies that the “social” and “cognition” parts have the same influence on the velocity update. The parameter ω is known as the inertia weight and is usually set to a positive value chosen from a linear or nonlinear function of the iteration number.

Compared with PSO, the most significant advantage of QPSO is that its global convergence can be theoretically guaranteed [18]. In addition, QPSO is much easier to be controlled, benefiting from the fact that it only has one parameter. Trajectory analyses demonstrated the fact that convergence of the whole particle swarm may be achieved if each particle converges to its local attractor $p_i = (p_{i1}, p_{i2}, \dots, p_{iD})$ [24]:

$$\begin{aligned} p_{i,j}(t) &= \frac{[c_1 \cdot r_{1,j}(t) \cdot pbest_{i,j}(t) + c_2 \cdot r_{2,j}(t) \cdot gbest_j(t)]}{[c_1 \cdot r_{1,j}(t) + c_2 \cdot r_{2,j}(t)]}, \end{aligned} \quad (2)$$

$$\text{or: } p_{i,j}(t) = \varphi \cdot pbest_{i,j}(t) + (1 - \varphi) \cdot gbest_j(t),$$

where φ is a sequence of uniformly distributed random numbers in (0,1).

Unlike PSO, each individual particle in QPSO moves in the search space with a δ potential on each dimension, of whose center is point $p_{i,j}$. When a particle x_i evolves its position in this δ potential, the new position $X_i(t+1)$ is subject to an exponential distribution whose probability density function is

$$\begin{aligned} F(|X_i(t+1) - p_i(t)|) &= \frac{1}{L_i(t)} \exp\left(-\frac{2|X_i(t+1) - p_i(t)|}{L_i(t)}\right), \end{aligned} \quad (3)$$

where L_i determines the distribution scope. In QPSO, the distribution scope of each particle is set elaborately to relate to its relative position in the whole swarm:

$$L_{i,j} = 2\beta \cdot |mbest_j(t) - X_{i,j}(t)|, \quad (4)$$

where $mbest$ is the mean of the personal best positions among all particles:

$$\begin{aligned} mbest(t) &= (mbest_1(t), mbest_2(t), \dots, mbest_D(t)) \\ &= \left(\frac{1}{M} \sum_{i=1}^M pbest_{i,1}(t), \frac{1}{M} \sum_{i=1}^M pbest_{i,2}(t), \dots, \frac{1}{M} \sum_{i=1}^M pbest_{i,D}(t) \right). \end{aligned} \quad (5)$$

In this way, particles far away from the center of the whole swarm will have a larger searching scope, while those particles close to the middle can only search in a relatively limited small space. Therefore, the position of the particle in QPSO is updated according to the following iteration equation:

$$X_{i,j}(t+1) = p_{i,j} \pm \beta \cdot |mbest_j - X_{i,j}(t)| \cdot \ln\left(\frac{1}{\mu}\right), \quad (6)$$

where μ is a random number uniformly distributed in (0,1) and β is called Contraction-Expansion Coefficient, which is employed to control the convergence speed of the algorithm. As proved by Sun et al. [14], β must be set as $\beta < 1.782$ to guarantee convergence of the particle.

3. Proposed Method

3.1. Novel Ring Model Based Position-Update Strategy. From the perspective of both empirical evidence and theory analysis, the global search ability as well as the convergence rate of QPSO and its variants has been fully discovered on the single-objective optimization problems. However, this advantage of QPSO leads to premature convergence when it is applied directly to the multiobjective optimizations. Without the loss of generality, a multiobjective optimization problem can be formulated as follows:

$$\min F(x) = [f_1(x), f_2(x), \dots, f_M(x)]^T, \quad (7)$$

where f_i ($i = 1, 2, \dots, M$) are the objective functions, while $x = [x_1, x_2, \dots, x_D]^T \in \Omega$ is the vector of decision variable. The optimization performance is generally measured by two aspects: closeness to the ideal Pareto front and distribution of the approximated solutions [8]. However, the quick convergence property of QPSO is apt to lead rapid decline of the swarm's diversity, which becomes a serious problem that must be addressed when it is extended into multiobjective optimization.

Each particle in QPSO is located in an exponential distributed potential, with the center p_i and distribution scope L_i , respectively. Figure 1 illustrates the relationship between the particle position and its distribution scope in

FIGURE 1: The dependence of the search scope of the particle on the distance of the particle from the mean best position.

FIGURE 2: All the particles in the swarm are arranged in circle in MOQPSOr.

QPSO. Here, x_1 represents the particle far away from the mean best position ($mbest$) of the swarm and its corresponding distribution at next iteration with the center p_1 visualised on upper right; x_2 denotes the particle near the mean best position of the swarm with p_2 being the center of the exponential distribution of its position at next iteration. According to the iteration equation of QPSO, we can see in the figure that x_2 has a much smaller variation scope than x_1 . That is, the closer x_i to the mean best position, the smaller the scope of the variation. Only those particles away from the $mbest$, like x_1 , have the large variation scope. That implies that, in original QPSO, a certain number of particles in the swarm are supposed to have small, or even very close to zero, distribution scopes.

In order to control the descent speed of the swarm diversity, we propose a novel position-update strategy based on the ring model. In this model, for a swarm with M particles, all the particles are arranged in a circle like Figure 2, numbered as $x_1, x_2, \dots, x_{i-1}, x_i, x_{i+1}, \dots, x_M$. Different from the way of deciding particle's variation scope according to its location in the swarm in original QPSO, in our proposed method, when the particle x_i evolves, its variation scope is decided by the distance to its next-numbered particle x_{i+1} . Accordingly, for the iteration equation of particle x_i , we replace $mbest$ by $pbest_{i+1}$, which represents the personal best position of particle x_{i+1} . Since particles in the swarm are distributed randomly and independently, the position of x_{i+1} can locate everywhere in the search space theoretically. That means that particles with continuous indices are not necessarily adjacent in position. The particles around the center of the swarm could also have the opportunity to mutate

FIGURE 3: The relationship between the location of the local attractor p and the value of r for different values of α .

with large scopes. Therefore, by this novel position-update strategy, the swarm of MOQPSOr can mutate more than the original QPSO, which subsequently leads to the slowdown of the descent speed of the swarm diversity.

In terms of the local attractor p_i , in QPSO, it is set to lie uniform-randomly in the hyperrectangle with $pbest_i$ and $gbest$ being two ends of its diagonal. Generally speaking, the local searching ability will be enhanced when p_i moves towards $gbest$, and when p_i moves towards $pbest_i$, the global searching ability will be enhanced. Therefore, in MOQPSOr, p_i is given larger probabilities locating near $pbest_i$ in the beginning and near $gbest$ in the later stage of iteration, respectively.

Based on the above analysis, particles in MOQPSOr that move according to the position-updating strategy can be described as follows:

$$X_{i,j}(t+1) = p_{i,j} \pm \beta \cdot |pbest_{i+1,j}(t) - X_{i,j}(t)| \cdot \ln\left(\frac{1}{\mu}\right), \quad \mu \sim U(0,1), \quad (8)$$

$$\text{where: } p_{i,j} = gbest_j + (pbest_{i,j} - gbest_j) \cdot r^\alpha, \quad (9)$$

$$r \sim U(0,1),$$

where parameter α is called Searching Coefficient, by adjusting which, local attractor $p_{i,j}$ can be controlled to appear near $pbest_{i,j}$ or $gbest_j$. Figure 3 plots the distribution of p 's location formulated in (9), where the horizontal axis denotes the random number r and the vertical axis denotes p 's location between $pbest$ and $gbest$. $\alpha = 0.3$ and $\alpha = 3$ are used as examples to demonstrate situations when $\alpha < 1$ and $\alpha > 1$, respectively. From the red dotted lines, it could be seen that when $\alpha = 0.3$, p has half probability to locate in $[pbest, 0.812pbest + 0.188gbest]$, which is much closer to $pbest$ than to $gbest$. In Figure 3(b), when $\alpha = 3$, p has

half probability to locate in $[0.125pbest + 0.875gbest, gbest]$, which is closer to $gbest$ than to $pbest$. That is, when $\alpha < 1$, the local attractor p would appear near $pbest$ with large probability. The smaller the value of α is, the closer the point p gathers towards $pbest$. On the contrary, when $\alpha > 1$, p would appear with large probability near $gbest$. The bigger the value of α is, the closer the point p gathers towards $gbest$. Therefore, the algorithm's global searching ability could be enhanced by setting $\alpha < 1$, while the local searching ability could be enhanced by setting $\alpha > 1$.

3.2. Multiobjective QPSO with Ring Model (MOQPSOr). In MOQPSOr, we adopt the concept of crowding distance [25] for the leader selection. Whenever a leader particle needs to be selected as the global best position from the external archive, the crowding factor of each leader is calculated, followed by the subsequent selection by means of a binary tournament based on these crowding factors. A particle with larger crowding distance has more chances to be chosen as leader.

Crowding distance of each individual is also used to decide which leaders would keep over generations when the maximum external archive size is exceeded in MOQPSOr. The particle in the archive with the smallest crowding distance will be removed first whenever needed.

Since MOQPSOr could have already slowed down the descent speed of the swarm diversity by the novel position-update strategy based on ring model, there is no need for any additional mutation operation. The procedure of the MOQPSOr algorithm could be described as follows.

Step 1 (initialization).

Step 1.1. Parameter settings are as follows: the swarm size M , the external archive size E_{size} , the stopping criterion T_{max} , iteration time $t = 0$, Searching Coefficient α , and Contraction-Expansion Coefficient β .

Step 1.2. Initialize the swarm randomly within the feasible solution space $X = (X_1, X_2, \dots, X_M)$, as well as the personal best positions $pbest = (pbest_1, pbest_2, \dots, pbest_M)$, where $pbest_i = X_i$, $i \in [1, M]$.

Step 1.3. Initialize the external archive E as the nondominated solution in $pbest$.

Step 2 (termination). If termination condition is met, stop and return all the individuals in the current E . Otherwise, go to Step 3.

Step 3 (reproduction). For each particle X_i , $i \in [1, M]$, note the following.

Step 3.1. Select a global best position $gbest_i$ from external archive.

Step 3.2. Update position by (8) and (9).

Step 3.3. Update personal best position $pbest_i$.

Step 4 (external archive update).

Step 4.1. Consider $E(t+1) = E(t) \cup pbest$.

Step 4.2. Remove dominated solutions in $E(t+1)$.

Step 4.3. If the size of the current E is larger than E_{size} , calculate the crowding distance of each individual in E , sort them in descending order of crowding distance, and keep the first E_{size} individuals in $E(t+1)$.

Step 5. Consider $t = t + 1$; go to Step 2.

4. Experiments and Analysis

4.1. Test Functions. Walking-Fish-Group (WFG) [26], a well-designed multiobjective test suite which provides a truer means of assessing the performance of optimization algorithms on a wide range of different problems, is used to validate the performance of our approach in 2-objective space. Compared with the other two commonly used suite of ZDT [27] and DTLZ [28], WFG test suite is more challenging and contains a number of problems that exhibit properties not evident in ZDT and DTLZ, including nonseparable problems, deceptive problems, a truly degenerate problem, a mixed shape Pareto front problem, problems scalable in the number of position-related parameters, and problems with dependencies related to position and distance parameters.

Besides WFGs, another three 3-objective benchmark functions, which are acknowledged for the extreme difficulty to optimize in the DTLZ test suite, are also involved in the comparison test. DTLZ2 tests the ability of global convergence by providing a spherical Pareto front. DTLZ4 assesses the maintainability of a good distribution of solutions by generating a nonuniform distribution of points along the true Pareto front. The Pareto front of DTLZ7 is the intersection of a straight line and a hyperplane. All these twelve benchmark problems are listed in Table 1.

4.2. Performance Metrics. To assess the performance of algorithms in this experiment, three quality indicators are considered: Additive Unary ε -indicator ($I_{\varepsilon+}^1$) [29], hypervolume (I_{HV}) [30], and the Inverted Generational Distance (IGD) [31].

Additive Unary ε -Indicator ($I_{\varepsilon+}^1$). It measures the convergence of the resulting Pareto fronts. A lower value indicates a better approximation set. For an approximation set X , the additive Unary ε -indicator is defined as [32]

$$I_{\varepsilon+}^1(X) = \inf_{\varepsilon \in \mathbb{R}} \left\{ \forall z^2 \in P \exists z^1 \in X : z^1 \succeq z^2 \right\}, \quad (10)$$

where P is the ideal Pareto front.

Hypervolume (I_{HV}). This metric measures both convergence and diversity of the solutions. The higher the I_{HV} values are, the better the algorithm performs. Generally speaking, the hypervolume measures the volume of the space dominated by the approximation set, bounded by a reference point. $I_{HV}(X)$ of an approximation set X can be mathematically defined as

$$I_{HV}(X) = \Lambda \left(\bigcup_{(x_1, \dots, x_d) \in X} [r_1, x_1] \times \dots \times [r_d, x_d] \right), \quad (11)$$

where $r = (r_1, r_2, \dots, r_d)$ is the reference point and Λ is the usual Lebesgue measure.

Inverted Generational Distance (IGD). Inverted Generational Distance is the average distance from every solution in the reference set to the nearest solution in the approximation set; it, therefore, reflects convergence of the solutions. The fewer the IGD values, the better the algorithm's performance. The IGD metric is calculated for the solution set X using the reference point set Z as follows:

$$IGD(Z, X) = \frac{1}{|Z|} \sum_{i=1}^{|Z|} \min_{j=1}^{|X|} d(z_i, x_j), \quad (12)$$

where $d(z_i, x_j)$ is the distance between z_i and x_j in the objective space.

4.3. Algorithm for Comparison and Parameter Setting. In this experiment, five state-of-the-art multiobjective optimization algorithms are chosen for comparison, including the most efficient and widely used multiobjective particle swarm optimizer OMOPSO [6] and another two well-known PSO-based multiobjective optimization algorithms: σ MOPSO [33] and pdMOPSO [34], as well as two competitive evolutionary multiobjective optimizers: NSGA-II [25] and PESA-II [35].

To make a fair comparison, the population size and the leader archive in MOQPSOr and all the other five comparison algorithms are fixed to 100 for all test instances. The stopping condition is set to 250 iterations, which means a total of 25000 function evaluations. For MOQPSOr, α increases linearly from 0.6 to 1.2, and β is set to 0.3. For OMOPSO, σ MOPSO, and pdMOPSO, the sets of C_1 , C_2 , and ω , as well as the mutation method and the mutation probability are all the

TABLE 1: Description of the test functions.

Code	Objective functions	Feature	Objectives
WFG1	$\min f_{m=1:M-1}(x) = x_M + 2m \left(\prod_{i=1}^{M-m} \left(1 - \cos \left(\frac{x_i \pi}{2} \right) \right) \right) \cdot \left(1 - \sin \left(\frac{x_{M-m+1} \pi}{2} \right) \right)$	Mixed, convex/concave, Unimodal, separable	2
	$\min f_M(x) = x_M + 2m \left(1 - x_1 - \frac{\cos(2A\pi x_1 + \pi/2)}{2A\pi} \right)^\alpha$		
	$x = \{x_1, \dots, x_M\} = \{\max(t_M^p, A_1)(t_M^p - 0.5) + 0.5, \dots, \max(t_M^p, A_{M-1})(t_{M-1}^p - 0.5) + 0.5, t_M^p\}$		
	$t^p = \{t_1^p, \dots, t_M^p\} \leftarrow t^{p-1} \leftarrow \dots \leftarrow t^1$		
	$t_{i=1:k}^1 = y_i \quad t_{i=k+1:n}^1 = \text{s_linear}(y_i, 0.35)$		
	$t_{i=1:k}^2 = y_i \quad t_{i=k+1:n}^2 = b_flat(y_i, 0.8, 0.75, 0.85)$		
	$t_{i=1:n}^3 = b_poly(y_i, 0.02)$		
	$t_{i=1:M-1}^4 =$		
	$r_sum \left(\{y_{(i-1)k/(M-1)+1}, \dots, y_{ik/(M-1)}\}, \left\{ 2 \left(\frac{(i-1)k}{(M-1)} + 1 \right), \dots, \frac{2ik}{(M-1)} \right\} \right)$		
	$t_{i=M}^4 = r_sum(\{y_{k+1}, \dots, y_n\}, \{2(k+1), \dots, 2n\})$		
WFG2	$f_{m=1:M-1} \text{ and } t^1 \text{ are as those from WFG1,}$	Convex, disconnected, Unimodal, multimodal, nonseparable	2
	$\min f_M(x) = x_M + 1 - (x_1)^\alpha \cos^2(A(x_1)^\beta \pi), \quad \alpha = \beta = 1, \quad A = 5$		
	$t_{i=1:k}^2 = y_i \quad t_{i=k+1:n}^2 = r_nonsep(\{y_{k+2(i-k)-1}, y_{k+2(i-k)}\}, 2)$		
	$t_{i=1:M-1}^3 = r_sum(\{y_{(i-1)k/(M-1)+1}, y_{ik/(M-1)}\}, \{1, \dots, 1\})$		
	$t_M^3 = r_sum(\{y_{k+1}, \dots, y_{k+1/2}\}, \{1, \dots, B\})$		
	$t_{i=M}^4 = r_sum(\{y_{k+1}, \dots, y_n\}, \{2(k+1), \dots, 2n\})$		
	$\min f_1(x) = x_M + 2m \prod_{i=1}^{M-1} x_i$		
	$\min f_{m=2:M-1}(x) = x_M + 2m \left(\prod_{i=1}^{M-m} x_i \right) \cdot (1 - x_{M-m+1})$		
	$\min f_M(x) = 1 - x_1$		
	$t^{1:3} \text{ are as those from WFG2}$		
WFG3	$\min f_1(x) = x_M + 2m \left(\prod_{i=1}^{M-1} \sin \left(\frac{x_i \pi}{2} \right) \right)$	Linear, degenerate, unimodal, nonseparable	2
	$\min f_{m=2:M-1}(x) = x_M + 2m \left(\prod_{i=1}^{M-m} \sin \left(\frac{x_i \pi}{2} \right) \right) \cdot \cos \left(\frac{x_{M-m+1} \pi}{2} \right)$		
	$\min f_M(x) = x_M + 2m \cos \left(\frac{x_1 \pi}{2} \right)$		
	$t_{i=1:n}^1 = \text{s_multi}(y_i, 30, 10, 0.35)$		
	$t_{i=1:M-1}^2 = r_sum(\{y_{(i-1)k/(M-1)+1}, \dots, y_{ik/(M-1)}\}, \{1, \dots, 1\})$		
	$t_M^2 = r_sum(\{y_{k+1}, \dots, y_n\}, \{1, \dots, 1\})$		
	$f_{m=1:M} \text{ and } t^2 \text{ are as those from WFG4,}$		
	$t_{i=1:n}^1 = \text{s_decept}(y_i, 0.35, 0.001, 0.05)$		
WFG5	$f_{m=1:M} \text{ and } t^2 \text{ are as those from WFG4,}$	Concave, deceptive, separable	2

TABLE 1: Continued.

Code	Objective functions	Feature	Objectives
WFG6	$f_{m=1:M}$ are as those from WFG4, t^1 as those from WFG1, $t^2_{i=1:M-1} = r_nonsep\left(\{y_{(i-1)k/(M-1)+1}, \dots, y_{ik/(M-1)}\}, \frac{k}{(M-1)}\right)$ $t^2_M = r_nonsep(\{y_{k+1}, \dots, y_n\}, l)$	Concave, unimodal, nonseparable	2
WFG7	$f_{m=1:M}$ are as those from WFG4, t^2 as t^1 from WFG1, t^3 as t^2 from WFG4, $t^1_{i=1:k} = b_param\left(y_i, r_sum(\{y_{i+1}, \dots, y_n\}, \{1, \dots, 1\}), \frac{0.98}{49.98}, 0.02, 50\right)$ $t^1_{i=k+1:n} = y_i$	Concave, unimodal, separable	2
WFG8	$f_{m=1:M}$ are as those from WFG4, t^2 as t^1 from WFG1, t^3 as t^2 from WFG4, $t^1_{i=1:k} = y_i$ $t^1_{i=k+1:n} = b_param\left(y_i, r_sum(\{y_{i+1}, \dots, y_{i-1}\}, \{1, \dots, 1\}), \frac{0.98}{49.98}, 0.02, 50\right)$	Concave, unimodal, nonseparable	2
WFG9	$f_{m=1:M}$ are as those from WFG4, t^3 as t^2 from WFG6, $t^1_{i=1:n-1} = b_param\left(y_i, r_sum(\{y_{i+1}, \dots, y_n\}, \{1, \dots, 1\}), \frac{0.98}{49.98}, 0.02, 50\right)$ $t^1_{i=n} = y_n$ $t^2_{i=1:k} = s_decept(y_i, 0.35, 0.001, 0.05)$ $t^2_{i=k+1:n} = s_multi(y_i, 30, 95, 0.35)$	Concave, multimodal, deceptive, nonseparable	2
DTLZ2	$\min f_1(x) = (1 + g(x_M)) \prod_{i=1}^{M-1} \cos\left(\frac{x_i \pi}{2}\right)$ $\min f_{m=2:M-1}(x) = (1 + g(x_M)) \sin\left(\frac{x_{M-m+1} \pi}{2}\right) \prod_{i=1}^{M-m} \cos\left(\frac{x_i \pi}{2}\right)$ $\min f_M(x) = (1 + g(x_M)) \sin\left(\frac{x_1 \pi}{2}\right)$ $s.t. g(x_M) = \sum_{x_i \in x_M} (x_i - 0.5)^2$	Concave, unimodal	3
DTLZ4	$f_{1:M}$ and $g(x_M)$ are both the same with DTLZ2, besides x_i^α is used to replace x_i	Concave, nonuniform	3
DTLZ7	$\min f_{m=1:M-1}(x) = x_m$ $\min f_M(x) = (1 + g(x_M)) \left\{ M - \sum_{i=1}^{M-1} \left[\frac{f_i}{1 + g(x_M)} (1 + \sin(3\pi f_i)) \right] \right\}$ $s.t. g(x_M) = 1 + \frac{9}{ x_M } \sum_{x_i \in x_M} x_i$	Disconnected, unimodal, multimodal	3

TABLE 2: Comparison results in terms of $I_{\varepsilon+}^1$ between MOQPSOr and other algorithms.

Function	MOQPSOr	OMOPSO	σ MOPSO	pdMOPSO	NSGA-II	PESA-II
WFG1	0.6662	0.7697 (+)	0.7058 (+)	0.7247 (+)	0.5662 (–)	0.7149 (+)
WFG2	0.005945	0.01156 (+)	0.05869 (+)	0.08135 (+)	0.006621 (+)	0.004953 (–)
WFG3	0.3327	0.3341 (+)	0.3345 (+)	0.3378 (+)	0.3338 (+)	0.3348 (+)
WFG4	0.02752	0.03035 (+)	0.04921 (+)	0.03004 (+)	0.01162 (–)	0.01002 (–)
WFG5	0.03067	0.03071 (+)	0.03070 (+)	0.05671 (+)	0.03210 (+)	0.03367 (+)
WFG6	0.006673	0.007467 (+)	0.01973 (+)	0.01244 (+)	0.03534 (+)	0.04072 (+)
WFG7	0.005080	0.006404 (+)	0.008138 (+)	0.01039 (+)	0.01455 (+)	0.01318 (+)
WFG8	0.05696	0.05381 (–)	0.05714 (+)	0.05401 (–)	0.05475 (–)	0.06613 (+)
WFG9	0.01266	0.01331 (+)	0.01308 (+)	0.01573 (+)	0.01898 (+)	0.01667 (+)
DTLZ2	0.05071	0.05176 (+)	0.05150 (+)	0.06394 (+)	0.1153 (+)	0.1201 (+)
DTLZ4	0.05191	0.05700 (+)	0.05681 (+)	0.06437 (+)	0.1069 (+)	0.1007 (+)
DTLZ7	0.05656	0.06321 (+)	0.06229 (+)	0.06870 (+)	0.07233 (+)	0.1459 (+)
Better (+)		11	12	11	9	10
Worse (–)		1	0	1	3	2
Score		10	12	10	6	8

TABLE 3: Comparison results in terms of I_{HV} between MOQPSOr and other algorithms.

Function	MOQPSOr	OMOPSO	σ MOPSO	pdMOPSO	NSGA-II	PESA-II
WFG1	0.02904	0 (+)	0 (+)	0 (+)	0.2046 (–)	0.1447 (–)
WFG2	0.5608	0.5557 (+)	0.5502 (+)	0.5489 (+)	0.5626 (–)	0.5619 (–)
WFG3	0.44197	0.4408 (+)	0.4401 (+)	0.4399 (+)	0.4399 (+)	0.4399 (+)
WFG4	0.1915	0.1888 (+)	0.1898 (+)	0.1910 (+)	0.2161 (–)	0.2166 (–)
WFG5	0.1979	0.1975 (+)	0.1976 (+)	0.1954 (+)	0.1949 (+)	0.1961 (+)
WFG6	0.2082	0.2074 (+)	0.2062 (+)	0.2003 (+)	0.1699 (+)	0.1698 (+)
WFG7	0.2107	0.2089 (+)	0.2065 (+)	0.2043 (+)	0.2075 (+)	0.2088 (+)
WFG8	0.1596	0.1624 (–)	0.1595 (+)	0.1629 (–)	0.1610 (–)	0.1631 (–)
WFG9	0.2324	0.2314 (+)	0.2310 (+)	0.2309 (+)	0.2308 (+)	0.2317 (+)
DTLZ2	0.41174	0.4111 (+)	0.4102 (+)	0.4069 (+)	0.3725 (+)	0.4019 (+)
DTLZ4	0.4104	0.4080 (+)	0.4061 (+)	0.3921 (+)	0.3765 (+)	0.4101 (+)
DTLZ7	0.2642	0.2740 (–)	0.2623 (+)	0.2602 (+)	0.2638 (+)	0.2538 (+)
Better (+)		10	12	11	8	8
Worse (–)		2	0	1	4	4
Score		8	12	10	4	4

same as in Durillo et al.'s work [36]. For NSGA-II and PESA-II, the crossover rate *sbx.rate* and the distribution index *sbx.distributionIndex* for simulated binary crossover are set to 1.0 and 15.0, respectively, while the mutation rate and the distribution index for polynomial mutation are set to *pm.rate* = 1/*N* and *pm.distributionIndex* = 20.0, where *N* is the number of decision variables. For PESA-II, in addition, the number of bisections in the adaptive grid archive is set to 8.

All the experiments are implemented using MOEA framework [37], an open source Java library for developing and experimenting with multiobjective evolutionary algorithms.

Every algorithm runs on each problem over 30 independent trials; and the average results of $I_{\varepsilon+}^1$, I_{HV} and IGD are recorded.

4.4. Experimental Results. Tables 2–4 tabulate the performance results on $I_{\varepsilon+}^1$, I_{HV} , and IGD, respectively. For each test function, the best result is bolded. Markers “+” and “–” in Tables 2, 3, and 4 are used to indicate the performance comparison results. “+” means MOQPSOr outperforms its rivals, while “–” means MOQPSOr underperforms. The summaries of the comparison results on each metric are also shown in each table.

Since $I_{\varepsilon+}^1$ and IGD both mainly focus on reflecting the ability of converging to the global Pareto front, Tables 2 and 4 will be discussed together. It is obvious whether on $I_{\varepsilon+}^1$ or on IGD metric that MOQPSOr no doubt performs best among all the algorithms. On $I_{\varepsilon+}^1$, MOQPSOr achieves 8 best values out of the 12 problems, as well as 2 second best values. In comparison, OMOPSO, NSGA-II, and PESA-II all get no more than 2 best values each, while σ MOPSO and pdMOPSO

TABLE 4: Comparison results in terms of IGD between MOQPSOr and other algorithms.

Function	MOQPSOr	OMOPSO	σ MOPSO	pdMOPSO	NSGA-II	PESA-II
WFG1	0.576	0.6156 (+)	0.5873 (+)	0.6003 (+)	0.4060 (–)	0.5228 (–)
WFG2	0.007339	0.009023 (+)	0.008287 (+)	0.008894 (+)	0.003544 (–)	0.004050 (–)
WFG3	0.03331	0.03363 (+)	0.03398 (+)	0.03410 (+)	0.03438 (+)	0.03428 (+)
WFG4	0.01994	0.02157 (+)	0.02431 (+)	0.02386 (+)	0.005290 (–)	0.004598 (–)
WFG5	0.02733	0.02739 (+)	0.02741 (+)	0.03082 (+)	0.02766 (+)	0.02749 (+)
WFG6	0.004706	0.005259 (+)	0.01734 (+)	0.009347 (+)	0.03296 (+)	0.03480 (+)
WFG7	0.002606	0.003473 (+)	0.004735 (+)	0.004981 (+)	0.005660 (+)	0.004236 (+)
WFG8	0.03933	0.03736 (–)	0.03989 (+)	0.03756 (–)	0.03725 (–)	0.03649 (–)
WFG9	0.008916	0.009299 (+)	0.009313 (+)	0.009305 (+)	0.009429 (+)	0.008953 (+)
DTLZ2	0.04004	0.04160 (+)	0.04231 (+)	0.04583 (+)	0.07000 (+)	0.07540 (+)
DTLZ4	0.04202	0.04794 (+)	0.04737 (+)	0.05177 (+)	0.06859 (+)	0.06436 (+)
DTLZ7	0.04426	0.04523 (+)	0.04570 (+)	0.05231 (+)	0.05180 (+)	0.07088 (+)
Better (+)		11	12	11	8	8
Worse (–)		1	0	1	4	4
Score		10	12	10	4	4

do not achieve any best result at all. In terms of IGD, similarly, MOQPSOr gets the best values in 8 out of the 12 problems, while NSGA-II and PESA-II get 2 best values each. Thus, MOQPSOr claims to be able to produce solutions closer to the global Pareto front than other comparison algorithms in our study.

I_{HV} measures both the convergence and the diversity of the solutions. It could be observed clearly again from Table 3 that MOQPSOr is the best-performing algorithm, yielding the best values in 7 out of 12 problems. The next best-performing algorithms are NSGA-II and PESA-II, which achieve 2 best values each. Although MOQPSOr does not get the best I_{HV} on 3-objective DTLZ7, it is the second best performing algorithm, only a little bit inferior to OMOPSO.

Figure 4 illustrates the comparison between the final nondominated fronts obtained by MOQPSOr and those by the other algorithms on 2-objective WFG6. In order to display more clearly, each comparison pair contains an overall (Figures 4(a), 4(c), and 4(e)) figure and a sectional (Figures 4(b), 4(d), and 4(f)) figure. In each diagram in Figure 4, thin blue lines demonstrate the ideal Pareto fronts of the problems, while the red dots present the nondominated solutions obtained by MOQPSOr. The black dots in the three pairs of Figures 4(a) and 4(b), Figures 4(c) and 4(d), and Figures 4(e) and 4(f) represent the Pareto front obtained by OMOPSO, NSGA-II, and PESA-II, respectively. It could be observed from Figure 4 that, in terms of the closeness to the blue real Pareto front, NSGA-II performs the worst on 2-objective WFG6. Although the nondominated solutions obtained by OMOPSO and PESA-II are both very close to the one obtained by MOQPSOr, the latter is still a little closer to the ideal Pareto front. Besides, MOQPSOr's nondominated solution shows a much better balanced distribution than OMOPSO's and PESA-II's.

Figures 5 and 6 demonstrate the final nondominated fronts found by MOQPSOr and by other algorithms on 3-objective DTLZ2 and DTLZ7. Figures 5(a), 5(c), 5(e), 5(g),

6(a), 6(c), 6(e), and 6(g) are the overall view of the fronts; Figures 5(b), 5(d), 5(f), 6(h), 6(b), 6(d), 6(f), and 6(h) are the side views. For DTLZ2, MOQPSOr obviously achieves the best performance. It could be observed from the side views that neither NSGA-II nor PESA-II can converge to the ideal Pareto front completely, with some dots astray, the convergence of NSGA-II being even worse than PESA-II. Although the nondominated front found by OMOPSO can converge to the ideal Pareto front as MOQPSOr, its distribution is less balanced. In other words, the distribution of the nondominated front achieved by MOQPSOr is the best.

It can be seen from Figure 6 that the solutions obtained by PESA-II cannot cover the entire ideal Pareto front on every plane when it runs on DTLZ7. Similar observations can also be found when NSGA-II runs on DTLZ7. In contrast, both OMOPSO and MOQPSOr can converge to the ideal Pareto front evenly.

In a word, it could be concluded that MOQPSOr is the most effective algorithm among all the 6 algorithms discussed in our study. Whether considered on solutions' convergence or the diversity, MOQPSOr outperforms all the PSO-based algorithms here on all 12 test functions. Compared with NSGA-II and PESA-II, MOQPSOr can achieve better solution sets on 8 out of 12 problems. Moreover, MOQPSOr performs the best on all the 3-objective functions.

It is worth noting that there are 4 multimodal functions in all of the 12 test problems, which are WFG2, WFG4, WFG9, and DTLZ7. It could be seen from Tables 2–4, on all these multimodal functions, that MOQPSOr performs the best except on 2-objective WFG2 and WFG4 compared with the NSGA-II and PESA-II. However, the performances of every algorithm change when coming to more-than-2-objective optimizations. Table 5 shows the performance results for all the algorithms with different number of objectives on these four multimodal benchmarks; the 3-objective and 4-objective optimization results are in italic font. It could be noticed that although MOQPSOr is not even the third best performed

FIGURE 4: Performance comparison on 2-objective WFG6 (the blue is the true Pareto front, the red is the Pareto front obtained by MOQPSOr, and the black is the Pareto front obtained by OMOPSO, NSGA-II, and PESA-II, resp.).

FIGURE 5: Performance on 3-objective DTLZ2 (blue is the ideal Pareto front; red is the Pareto front obtained by each algorithm).

FIGURE 6: Performance on 3-objective DTLZ7 (blue is the ideal Pareto front; red is the Pareto front obtained by each algorithm).

TABLE 5: Experiment results of all the algorithms in terms of $I_{\varepsilon+}^1$, I_{HV} , and IGD on multimodal functions.

Function	Objective	Metric	MOQPSOr	OMOPSO	σ MOPSO	pdMOPSO	NSGA-II	PESA-II
WFG2	2	$I_{\varepsilon+}^1$	0.005945	0.01156	0.05869	0.08135	0.006621	0.004953
		I_{HV}	0.5608	0.5557	0.5502	0.5489	0.5626	0.5619
		IGD	0.007339	0.009023	0.008287	0.008894	0.003544	0.004050
	3	$I_{\varepsilon+}^1$	0.0204	0.03241	0.1060	0.1132	0.1058	0.1054
		I_{HV}	0.9146	0.9073	0.8939	0.8903	0.8950	0.8933
		IGD	0.01768	0.02480	0.02219	0.02365	0.04214	0.04180
	4	$I_{\varepsilon+}^1$	0.03501	0.04775	0.1189	0.1237	0.1143	0.07031
		I_{HV}	0.9730	0.9725	0.9401	0.9378	0.9415	0.9612
		IGD	0.01353	0.01826	0.02356	0.02405	0.04531	0.03921
WFG4	2	$I_{\varepsilon+}^1$	0.02752	0.03035	0.04921	0.03004	0.01162	0.01002
		I_{HV}	0.1915	0.1888	0.1898	0.1910	0.2161	0.2166
		IGD	0.01994	0.02157	0.02431	0.02386	0.005290	0.004598
	3	$I_{\varepsilon+}^1$	0.05139	0.06123	0.07329	0.06021	0.1328	0.1805
		I_{HV}	0.4071	0.3883	0.3893	0.3935	0.3543	0.3898
		IGD	0.02830	0.03492	0.04776	0.03954	0.06745	0.04754
	4	$I_{\varepsilon+}^1$	0.05371	0.06697	0.08031	0.06891	0.1748	0.3173
		I_{HV}	0.6413	0.6024	0.6002	0.6176	0.4175	0.4021
		IGD	0.01308	0.01865	0.02589	0.02103	0.07311	0.06605
WFG9	2	$I_{\varepsilon+}^1$	0.01266	0.01331	0.01308	0.01573	0.01898	0.01667
		I_{HV}	0.2324	0.2314	0.2310	0.2309	0.2308	0.2317
		IGD	0.008916	0.009299	0.009313	0.009305	0.009429	0.008953
	3	$I_{\varepsilon+}^1$	0.02647	0.03302	0.03223	0.05873	0.138	0.1070
		I_{HV}	0.4230	0.4206	0.4135	0.4178	0.3454	0.3793
		IGD	0.02025	0.02195	0.02310	0.02241	0.06819	0.05594
	4	$I_{\varepsilon+}^1$	0.05906	0.07056	0.07186	0.1092	0.2049	0.1818
		I_{HV}	0.6640	0.6079	0.5871	0.5950	0.4314	0.5207
		IGD	0.01371	0.03721	0.04133	0.04097	0.1374	0.08635
DTLZ7	2	$I_{\varepsilon+}^1$	0.009551	0.009263	0.009220	0.01008	0.008840	0.01073
		I_{HV}	0.3242	0.3252	0.3240	0.3227	0.3292	0.3285
		IGD	0.006785	0.006680	0.006733	0.006712	0.003786	0.003679
	3	$I_{\varepsilon+}^1$	0.05656	0.06321	0.06229	0.06870	0.07233	0.1459
		I_{HV}	0.2642	0.2740	0.2623	0.2602	0.2638	0.2538
		IGD	0.04426	0.04523	0.04570	0.05231	0.05180	0.07088
	4	$I_{\varepsilon+}^1$	0.1442	0.1598	0.1560	0.1963	0.2182	0.2603
		I_{HV}	0.1412	0.1484	0.1330	0.1289	0.1355	0.1145
		IGD	0.1156	0.1285	0.1308	0.1623	0.1579	0.2725

algorithm on 2-objective WFG2 and WFG4, with a more obvious disadvantage especially towards NSGA-II and PESA-II, it turns out to be much more effective than all of the other algorithms when dealing with the 3-objective and 4-objective optimizations. In addition, the lead swells as the number of objectives increases. In conclusion, MOQPSOr is a competitive multiobjective optimization algorithm, especially on the multimodal problems with large number of objectives.

5. Conclusion

Generally speaking, most multiobjective optimization algorithms are reformed from various single-objective optimizers, and the latter play vital roles in deciding the performance

of the former. Since the canonical PSO has been proved to be not a global search algorithm, even not a local one, there may exist the problem of not being able to converge to global optima in the multiobjective PSO-based algorithms. On the contrary, though QPSO's global convergence has been proved, the works on extending QPSO to multiobjective optimization are rare. Therefore, we have proposed a novel version of multiobjective QPSO algorithm based on the ring model (MOQPSOr) in this paper, whose position-update strategy is improved in comparison with QPSO, making it more suitable for multiobjective optimization problems. In MOQPSOr, all the particles are arranged in a circle. When a particle evolves, the distribution scope is decided by the distance to its next particle in numerical order,

which makes the swarm mutates more than the original QPSO. With a high degree of probability, the local attractor p is located near the personal best position during the early stage of the search but near the global best position $gbest$ in the later stage of iteration. Unlike most MOPSOs, there is no additional mutation operation in MOQPSO. Compared with the 5 widely used evolutionary multiobjective optimization algorithms on 12 benchmark functions, the experiment results show that the proposed algorithm is highly competitive in both convergence and diversity when solving the multiobjective optimization problems. On top of that, the advantage becomes even more obvious with the number of objectives increasing.

Competing Interests

The authors have no conflict of interests regarding the publication of this manuscript.

Acknowledgments

This work was supported by the Natural Science Foundation of Jiangsu province (Project nos. BK20130155 and BK20130160).

References

- [1] Y. Liu, F. Chen, M. Li, and J. Kou, "A multi-objective optimization model for information system design," in *Contemporary Research on E-Business Technology and Strategy*, pp. 486–495, Springer, 2012.
- [2] J. Luo, Y. Qi, J. Xie, and X. Zhang, "A hybrid multi-objective PSO-EDA algorithm for reservoir flood control operation," *Applied Soft Computing Journal*, vol. 34, pp. 526–538, 2015.
- [3] B. Amiri, L. Hossain, J. W. Crawford, and R. T. Wigand, "Community detection in complex networks: multi-objective enhanced firefly algorithm," *Knowledge-Based Systems*, vol. 46, pp. 1–11, 2013.
- [4] M. Fadaee and M. A. M. Radzi, "Multi-objective optimization of a stand-alone hybrid renewable energy system by using evolutionary algorithms: a review," *Renewable and Sustainable Energy Reviews*, vol. 16, no. 5, pp. 3364–3369, 2012.
- [5] J. Moore and R. Chapman, *Application of Particle Swarm to Multiobjective Optimization*, Department of Computer Science and Software Engineering, Auburn University, Auburn, Ala, USA, 1999.
- [6] M. R. Sierra and C. A. C. Coello, "Improving PSO-based multi-objective optimization using crowding, mutation and ϵ -dominance," in *Evolutionary Multi-Criterion Optimization*, C. A. C. Coello, A. H. Aguirre, and E. Zitzler, Eds., vol. 3410 of *Lecture Notes in Computer Science*, pp. 505–519, Springer, Berlin, Germany, 2005.
- [7] M. R. Sierra and C. A. C. Coello, "Multi-objective particle swarm optimizers: a survey of the state-of-the-art," *International Journal of Computational Intelligence Research*, vol. 2, pp. 287–308, 2006.
- [8] A. Zhou, B.-Y. Qu, H. Li, S.-Z. Zhao, P. N. Suganthan, and Q. Zhang, "Multiobjective evolutionary algorithms: a survey of the state of the art," *Swarm and Evolutionary Computation*, vol. 1, no. 1, pp. 32–49, 2011.
- [9] M. A. Abido, "Multiobjective particle swarm optimization with nondominated local and global sets," *Natural Computing*, vol. 9, no. 3, pp. 747–766, 2010.
- [10] J. Branke and S. Mostaghim, "About selecting the personal best in multi-objective particle swarm optimization," in *Parallel Problem Solving from Nature—PPSN IX*, T. P. Runarsson, H.-G. Beyer, E. Burke, J. J. Merelo-Guervós, L. D. Whitley, and X. Yao, Eds., vol. 4193 of *Lecture Notes in Computer Science*, pp. 523–532, Springer, Berlin, Germany, 2006.
- [11] Y. Wang and Y. Yang, "Particle swarm optimization with preference order ranking for multi-objective optimization," *Information Sciences*, vol. 179, no. 12, pp. 1944–1959, 2009.
- [12] F. Van Den Bergh, *An Analysis of Particle Swarm Optimizers*, University of Pretoria, 2006.
- [13] F. J. Solis and R. J. B. Wets, "Minimization by random search techniques," *Mathematics of Operations Research*, vol. 6, no. 1, pp. 19–30, 1981.
- [14] J. Sun, B. Feng, and W. Xu, "Particle swarm optimization with particles having quantum behavior," in *Proceedings of the Congress on Evolutionary Computation (CEC '04)*, Portland, Ore, USA, June 2004.
- [15] L. dos Santos Coelho and P. Alotto, "Global optimization of electromagnetic devices using an exponential quantum-behaved particle swarm optimizer," *IEEE Transactions on Magnetics*, vol. 44, no. 6, pp. 1074–1077, 2008.
- [16] Y. Cai, J. Sun, J. Wang et al., "Optimizing the codon usage of synthetic gene with QPSO algorithm," *Journal of Theoretical Biology*, vol. 254, no. 1, pp. 123–127, 2008.
- [17] C. Sun and S. Lu, "Short-term combined economic emission hydrothermal scheduling using improved quantum-behaved particle swarm optimization," *Expert Systems with Applications*, vol. 37, no. 6, pp. 4232–4241, 2010.
- [18] J. Sun, X. Wu, V. Palade, W. Fang, C.-H. Lai, and W. Xu, "Convergence analysis and improvements of quantum-behaved particle swarm optimization," *Information Sciences*, vol. 193, pp. 81–103, 2012.
- [19] J. Sun, W. Fang, X. Wu, V. Palade, and W. Xu, "Quantum-behaved particle swarm optimization: analysis of individual particle behavior and parameter selection," *Evolutionary Computation*, vol. 20, no. 3, pp. 349–393, 2012.
- [20] J. D. Schaffer, "Multiple objective optimization with vector evaluated genetic algorithms," in *Proceedings of the 1st International Conference on Genetic Algorithms*, pp. 93–100, L. Erlbaum Associates, Pittsburgh, Pa, USA, July 1985.
- [21] N. Tian and Z. Ji, "Pareto-ranking based quantum-behaved particle swarm optimization for multiobjective optimization," *Mathematical Problems in Engineering*, vol. 2015, Article ID 940592, 10 pages, 2015.
- [22] J. Kennedy, "Particle swarm optimization," in *Encyclopedia of Machine Learning*, Springer, Berlin, Germany, 2011.
- [23] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth, Australia, November–December 1995.
- [24] M. Clerc and J. Kennedy, "The particle swarm-explosion, stability, and convergence in a multidimensional complex space," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 58–73, 2002.
- [25] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

- [26] S. Huband, P. Hingston, L. Barone, and L. While, "A review of multiobjective test problems and a scalable test problem toolkit," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 5, pp. 477–506, 2006.
- [27] E. Zitzler, K. Deb, and L. Thiele, "Comparison of multiobjective evolutionary algorithms: empirical results," *Evolutionary Computation*, vol. 8, no. 2, pp. 173–195, 2000.
- [28] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, *Scalable Test Problems for Evolutionary Multiobjective Optimization*, Springer, Berlin, Germany, 2005.
- [29] E. Zitzler, L. Thiele, M. Laumanns, C. M. Fonseca, and V. G. Da Fonseca, "Performance assessment of multiobjective optimizers: an analysis and review," *IEEE Transactions on Evolutionary Computation*, vol. 7, no. 2, pp. 117–132, 2003.
- [30] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.
- [31] K. Deb and H. Jain, "An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach—part I: solving problems with box constraints," *IEEE Transactions on Evolutionary Computation*, vol. 18, no. 4, pp. 577–601, 2014.
- [32] C. M. Fonseca, J. D. Knowles, L. Thiele, and E. Zitzler, "A tutorial on the performance assessment of stochastic multiobjective optimizers," in *Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization (EMO '05)*, p. 240, Guanajuato, Mexico, March 2005.
- [33] S. Mostaghim and J. Teich, "Strategies for finding good local guides in multi-objective particle swarm optimization (MOPSO)," in *Proceedings of the IEEE Swarm Intelligence Symposium (SIS '03)*, pp. 26–33, IEEE, Indianapolis, Ind, USA, April 2003.
- [34] J. E. Alvarez-Benitez, R. M. Everson, and J. E. Fieldsend, "A MOPSO algorithm based exclusively on pareto dominance concepts," in *Proceedings of the 3rd International Conference on Evolutionary Multi-Criterion Optimization (EMO '05)*, pp. 459–473, Springer, Guanajuato, Mexico, March 2005.
- [35] D. W. Corne, N. R. Jerram, J. D. Knowles, and M. J. Oates, "PESA-II: region-based selection in evolutionary multiobjective optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO '01)*, San Francisco, Calif, USA, July 2001.
- [36] J. J. Durillo, J. García-Nieto, A. J. Nebro, C. A. C. Coello, F. Luna, and E. Alba, "Multi-objective particle swarm optimizers: an experimental comparison," in *Evolutionary Multi-Criterion Optimization*, M. Ehrgott, C. M. Fonseca, X. Gandibleux, J.-K. Hao, and M. Sevaux, Eds., vol. 5467 of *Lecture Notes in Computer Science*, pp. 495–509, Springer, Berlin, Germany, 2009.
- [37] D. Hadka, *MOEA Framework User Guide. Version 2.6*, 2015.

Hindawi

Submit your manuscripts at
<http://www.hindawi.com>