Research Article

Pareto-Ranking Based Quantum-Behaved Particle Swarm Optimization for Multiobjective Optimization

Na Tian¹ and Zhicheng Ji²

¹*Institute of Educational Informatization, Jiangnan University, Wuxi 214122, China*

²*Institute of Electrical Automation, Jiangnan University, Wuxi 214122, China*

Correspondence should be addressed to Na Tian; tianna329@hotmail.com

Received 27 April 2015; Accepted 25 June 2015

Academic Editor: Fabio Tramontana

Copyright © 2015 N. Tian and Z. Ji. 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.

A study on pareto-ranking based quantum-behaved particle swarm optimization (QPSO) for multiobjective optimization problems is presented in this paper. During the iteration, an external repository is maintained to remember the nondominated solutions, from which the global best position is chosen. The comparison between different elitist selection strategies (preference order, sigma value, and random selection) is performed on four benchmark functions and two metrics. The results demonstrate that QPSO with preference order has comparative performance with sigma value according to different number of objectives. Finally, QPSO with sigma value is applied to solve multiobjective flexible job-shop scheduling problems.

1. Introduction

Most real-world optimization problems have more than one objective, with at least two objectives conflicting with each other. The conflicting objectives lead to a problem where a single solution does not exist. Instead, a set of optimal trade-off solutions exists, which are referred to as the pareto-optimal front or pareto front. This kind of optimization problems is referred to as multiobjective optimization problems.

In the past ten years, a wide variety of algorithms have been proposed to address such problems. Deb et al. [1] gave rise to a fast and elitist multiobjective genetic algorithm (NSGA-II) to reduce the computational complexity and adopted an elitism strategy. In [2], Konak et al. gave a tutorial about several variants of genetic algorithms and compared their performance. A particle swarm optimization (PSO) with pareto dominance was proposed in [3], in which a secondary repository and a special mutation operator were incorporated. A survey of the state-of-the-art multiobjective PSOs is presented in [4]. From the above literatures, the great difference between genetic algorithm (GA) and PSO exists for solving the multiobjective optimization problems. In GA, a set of particles are needed to be selected into next generation, while, in QPSO, one particle has to be chosen from the

nondominated optima as the global best position (*gbest*). Therefore, different pareto-ranking strategies are required.

Inspired by the quantum mechanics, QPSO was proposed by Sun et al. in 2004 [5], which has been proved to outperform PSO in both convergence rate and global search ability [6, 7]. Similar to other population-based algorithms (GA, PSO), maintaining a diverse population is an important consideration in multiobjective optimization to ensure solutions uniformly distributed over the pareto front. Therefore, in this paper, QPSO with three pareto-ranking strategies is compared on some benchmark functions and two metrics are used to test the performance of each strategy.

The research on the multiobjective FJSP is not as widely as that on the monoobjective FJSP. Brandimarte [8] was first to apply the decomposition method to FJSP and solved the routing subproblem by using some existing dispatching rules and addressed the scheduling subproblem by using a tabu search algorithm. Davarzani et al. [9] used artificial immune system to solve the multiobjective FJSP and compared it with Approach by Localization [10], AL + CGA [11]. Considering the advantages and disadvantages of stochastic algorithms and local search methods, some hybrid approaches were also proposed [12, 13]. In this paper, QPSO with sigma value is applied to solve multiobjective flexible job-shop scheduling.

The remainder of this paper is organized as follows. In Section 2, some basic concepts for multiobjective optimization are provided. Section 3 describes the three variants of pareto-ranking based QPSO in detail. Numerical tests on benchmark functions are presented in Section 4. In Section 5, QPSO with sigma value is applied to solve multiobjective flexible job-shop scheduling. Finally, conclusion is given in Section 6.

2. Basic Concepts for Multiobjective Optimization

Without loss of generality, only minimization problem is assumed here:

$$\begin{aligned} \text{minimize} \quad & \vec{f}(\vec{x}) = [f_1(\vec{x}), f_2(\vec{x}), \dots, f_n(\vec{x})]^T \\ \text{subjected to} \quad & g_i(\vec{x}) \leq 0, \quad i = 1, 2, \dots, q, \\ & h_i(\vec{x}) = 0, \quad i = 1, 2, \dots, p, \end{aligned} \quad (1)$$

where $\vec{x} = [x_1, x_2, \dots, x_D]^T \in \Omega$ is the vector of decision variables. $f_i, i = 1, 2, \dots, n$, are the objective functions, g_i and h_j ($i = 1, 2, \dots, q, j = 1, 2, \dots, p$) are the constraint functions of the problem, D is the dimension size of the search space, and n is the number of objectives.

Definition 1 (pareto dominance). A vector $\vec{u} = (u_1, u_2, \dots, u_n)$ dominates $\vec{v} = (v_1, v_2, \dots, v_n)$ (denoted by $\vec{u} \leq \vec{v}$) if and only if u is partially less than v ; that is, $\forall i \in \{1, 2, \dots, n\}, u_i \leq v_i$, and $\exists i \in \{1, 2, \dots, n\}, u_i < v_i$.

Definition 2 (pareto optimal set). One has $P^* = \{x \in \Omega \mid \neg \exists x' \in \Omega, \vec{f}(x') \leq \vec{f}(x)\}$.

Definition 3 (pareto front). For a given pareto optimal set P^* , the pareto front is defined as $PF^* = \{\vec{u} = \vec{f} = (f_1(x), f_2(x), \dots, f_n(x)) \mid x \in P^*\}$.

Generally, the analytical expression of the line or surface which contain those points does not exist. It is only possible for us to determine the nondominated points and to produce the pareto front.

3. Pareto-Ranking Based Quantum-Behaved Particle Swarm Optimization

3.1. Particle Swarm Optimization. PSO is a population-based optimization technique originally proposed by Kennedy and Eberhart in 1995 [14]. A PSO system simulates the knowledge evolvement of a social organism, in which particles represent candidate solutions of an optimization problem. The position of each particle is evaluated according to the objective function, and particles share memories of their “best” positions. These memories are used to adjust the particles’ own velocities and their subsequent positions. It has already been shown that PSO is comparable in performance with and may be considered as an alternative to GA [15].

In a PSO system with M particles in the D -dimensional space, the position, and velocity vector of particle i at the k th

iteration is represented as $X_i(t) = (X_{i1}(t), X_{i2}(t), \dots, X_{iD}(t))$ and $V_i(t) = (V_{i1}(t), V_{i2}(t), \dots, V_{iD}(t))$. The particle moves according to the following equations:

$$\begin{aligned} V_i(t+1) &= \omega V_i(t) + c_1 r_1 (P_i(t) - X_i(t)) \\ &\quad + c_2 r_2 (P_g(t) - X_i(t)), \\ X_i(t+1) &= X_i(t) + V_i(t+1), \end{aligned} \quad (2)$$

where, $i = 1, 2, \dots, M$, $j = 1, 2, \dots, D$, c_1 and c_2 are acceleration coefficient. r_1 and r_2 are random numbers distributed uniformly in $(0, 1)$. ω is inertia weight. Vector $P_i = (P_{i1}, P_{i2}, \dots, P_{iD})$ is the previous best position of particle i ($pbest$), and vector $P_g = (P_{g1}, P_{g2}, \dots, P_{gD})$ is the position of the best particle g in the swarm ($gbest$).

3.2. Quantum-Behaved Particle Swarm Optimization. The main disadvantage of PSO is that it is not guaranteed to be global convergent [16]. Concepts of QPSO were developed to address this disadvantage and first reported by Sun et al. in 2004 [5]. Trajectory analysis in [17] demonstrated the fact that the convergence of PSO may be achieved if each particle converges to its local attractor, $p_i = (p_{i1}, p_{i2}, \dots, p_{iD})$, defined as

$$p_i(t) = \varphi P_i(t) + (1 - \varphi) P_g(t), \quad (3)$$

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_i and P_g being two ends of its diagonal and moves following P_i and P_g .

In quantum world, the velocity of the particle is meaningless, so, in QPSO system, position is the only state to depict the particles, which moves according to the following equation [5]:

$$X_i(t+1) = p_i(t) \pm \alpha \left| mbest(t) - X_i(t) \right| \ln \left(\frac{1}{u} \right), \quad (4)$$

where u is a random number uniformly distributed in $(0, 1)$ and $mbest(t)$, called mean best position, is defined as the mean value of personal best positions of the swarm:

$$\begin{aligned} mbest(t) &= \frac{1}{M} \sum_{i=1}^M P_i \\ &= \left(\frac{1}{M} \sum_{i=1}^M P_{i1}(t), \frac{1}{M} \sum_{i=1}^M P_{i2}(t), \dots, \frac{1}{M} \sum_{i=1}^M P_{iD}(t) \right). \end{aligned} \quad (5)$$

The parameter α in (4) is named as contraction-expansion (CE) coefficient, which can be adjusted to control convergence rate. The most commonly used method to control CE is linearly decreasing from $\alpha_{\max}$ to $\alpha_{\min}$:

$$\alpha = \frac{(\alpha_{\max} - \alpha_{\min})(t_{\max} - t)}{t_{\max}} + \alpha_{\min}, \quad (6)$$

where t is the current iteration step, $t_{\max}$ is the predefined maximum iteration steps, and $\alpha_{\max}$ and $\alpha_{\min}$ are the maximum and minimum value of CE.

QPSO does not require velocity vectors for the particles and has fewer parameters to control, making the algorithm easier to implement. Experimental results performed on some well-known benchmark functions show that QPSO has better performance than PSO [5–7].

A great deal of efforts has been made to PSO for solving multiobjective optimization problems. A survey of state-of-the-art works is presented in [4]. The most contributive work is proposed by Ceollo et al. [3], in which the objective space is divided into hypercubes using an adaptive grid. The non-dominated solutions are distributed into grids and then the grid is chosen according to a roulette wheel selection strategy. However, the grid is problems dependent and computation complicated. While in [18] a new density measure strategy named as sigma method was introduced to multiobjective PSO, so as to choose a $gbest$ from the nondominated archive, the sigma method can guide particles to the pareto optimal front. However, if the initialized solutions in nondominated archive are bad, the algorithm will fall into local optimum due to the fast convergence. Yang et al. [19] proposed a hybrid method which combined the density information and sigma method so as to achieve a balance between global search and local search. Considering the calculation of the density value and crowding distance is limited by two objectives. When the number of objectives becomes large, the measure of density and crowding distance will become invalid. Therefore, the idea of preference order is introduced [20].

3.3. Preference Order. As we all know, when the optimization problem has two objectives, the pareto optimal solutions can be plotted on a curve. Though the trade-off surface can also be visualized for three objectives, it is not easy to pick the final point. For problems with more than three objectives, it becomes extremely difficult to find the optimized solution through visualization.

The idea of preference order was firstly proposed in 1999 [21]. A point $x^* \in \Omega$ is said to be efficiency of order k if $f(x^*)$ is not dominated by any of the k -element subsets of f_Ω .

Consider a minimization example with three objectives in Table 1; combination of all subsets of the objectives is $\{f_1, f_2, f_3, (f_1, f_2), (f_2, f_3), (f_1, f_3)\}$.

The dominant relation of all points in 2-element subsets is shown in Table 2, from which only Point 3 is nondominated by any points on all 2-element subsets. So Point 3 is said to be efficiency of order 2.

About efficiency of order, [22] gave three claims.

Claim 1. In a space with n objectives, there is no more than one point with efficiency of order $n - 1$.

Claim 2. x^* is efficient of order $k + 1$, if it is efficient of order k .

Claim 3. x^* is not efficient of order $k - 1$, if it is not efficient of order k .

The pseudocode of QPSO with preference order is shown below.

TABLE 1: Nondominated set.

Points	f_1	f_2	f_3
Point 1	0.5	2.1	11.2
Point 2	6	1.1	1.3
Point 3	3.14	0	3.76
Point 4	1.1	0.5	4.32
Point 5	3.89	0.1	3.23

TABLE 2: Dominant relation.

Points	Point 1	Point 2	Point 3	Point 4	Point 5
Point 1	×	×	×	×	×
Point 2	(f_2, f_3)	×	×	×	×
Point 3	(f_2, f_3)	(f_1, f_2)	×	(f_2, f_3)	(f_1, f_2)
Point 4	(f_2, f_3)	(f_1, f_2)	×	×	(f_1, f_2)
Point 5	(f_2, f_3)	(f_1, f_2)	×	(f_2, f_3)	×

Preference-Order Based QPSO

- (1) Initialization: initialize the swarm size M , the external archive size ND , and the maximum number of iterations t_{max} , CE α . Position of each particle X_i , $i = 1, 2, \dots, M$, is randomly initialized. Personal best position of each particle $pbest_i$ is equal to X_i . Construct external nondominated set nds .
- (2) Update the following:
 - (a) Identify global best position using preference-order method.
 - (i) Get combination of all subsets for n objectives.
 - (ii) Compute the efficiency order for each non-dominated solution.
 - (iii) Sort nondominated solutions in descending order according to their efficiency order.
 - (iv) Get $gbest$.
 - (b) Update position of each particle according to (3), (4), and (5).
 - (c) Update $pbest$ according to dominance rule.
 - (d) Update external archive and remove solution with lower efficiency order if archive size ND is exceeded.
 - (e) Update α according to (6).
- (3) Repeat (2) until the maximum number of iterations is reached.

3.4. Sigma Method. The sigma method was first proposed in [18] for finding the best local guide for each particle. Each nondominated solution in the external repository is assigned a value σ_i , $i = 1, 2, \dots, ND$, which is defined as (for two objectives shown in Figure 1.)

$$\sigma_i = \frac{f_{i,1}^2 - f_{i,2}^2}{f_{i,1}^2 + f_{i,2}^2}. \quad (7)$$

FIGURE 1: Sigma value for a two-objective space.

In the general case, σ is a vector of $\binom{N}{2}$ elements, and each element is the combination of two coordinates. Take a problem of three objectives; for example,

$$\sigma_i = \frac{\begin{pmatrix} f_{i,1}^2 - f_{i,2}^2 \\ f_{i,2}^2 - f_{i,3}^2 \\ f_{i,3}^2 - f_{i,1}^2 \end{pmatrix}}{(f_{i,1}^2 + f_{i,2}^2 + f_{i,3}^2)}. \quad (8)$$

The pseudocode of QPSO with sigma value is shown below.

QPSO with Sigma Method

- (1) Initialization: initialize the swarm size M , the external archive size ND , and the maximum number of iterations $t_{\max}$, CE α . Position of each particle X_i , $i = 1, 2, \dots, M$, is randomly initialized. Personal best position of each particle $pbest_i$ is equal to X_i . Construct external nondominated set nds .
- (2) Update the following:
 - (a) Identify local best position using sigma method.
 - (i) Assign the value σ_j to each particle in the external archive.
 - (ii) Calculate the value σ_i for each particle in the swarm.
 - (iii) Calculate the distance between σ_i and σ_j .
 - (iv) The particle k in the archive with σ_k has the minimum distance to σ_i selected as the best local guide for the particle i .
 - (b) Update position of each particle according to (3), (4), and (5).
 - (c) Update $pbest$ according to dominance rule.
 - (d) Update α according to (6).
- (3) Repeat (2) until the maximum number of iterations is reached.

4. Numerical Experiments

4.1. Test Functions. Seven common test functions [20–23] are used to test the proposed algorithm (DTLZ1, DTLZ2, DTLZ3, and DTLZ4), as listed in Table 3. We call $k = |\vec{x}_N|$ difficulty factor. Table 4 gives the difficulty factor, number of runs, number of iterations, number of particles, and size of nondominated archive used in our test. The reason why these functions are chosen is due to the following features:

- (1) The relatively small implementation effort.
- (2) The ability to be scaled to any number of objectives and decision variable.
- (3) The global pareto front being known analytically.
- (4) Convergence and diversity difficulties that can be easily controlled.

4.2. Performance Metrics. Usually, pareto-based multiobjective algorithms consider two aspects: closeness to the global pareto front (*distance*) and spread along the pareto front (*diversity*) [2], which are defined as follows.

Generational Distance (GD). It is a way of estimating how far the elements in the nondominated set are found so far from those in the pareto optimal set, which is defined as

$$GD = \frac{\sqrt{\sum_{i=1}^{ND} d_i^2}}{ND}, \quad (9)$$

where ND is the number of points in nondominated set and d_i is the Euclidean distance between a point and the nearest member of Pareto optimal set.

Spacing (SP). It is a metric measuring the distance variance of neighboring points in the nondominated set, which is defined as

$$S = \sqrt{\frac{1}{n-1} \sum_{i=1}^{ND} (\bar{d} - d_i)^2}, \quad (10)$$

where $d_i = \min_j (\sum_{k=1}^N |f_k^i(\vec{x}) - f_k^j(\vec{x})|)$, $i, j = 1, 2, \dots, ND$, $\bar{d}$ is the mean of all d_i , and N is the number of objectives.

4.3. Experiments Results. Tables 5 and 6 show the mean values of generational distance and spacing for three methods (QPSO + PO, QPSO + Rand, and QPSO + Sigma) with different number of objectives (2, 3, 4, 5, 6, 7, and 8). It is easy to note that QPSO with preference order has comparative performance with sigma value. When the number of objectives is small (e.g., 2, 3, 4, and 5), QPSO with sigma value performs better. When the number of objectives becomes larger (e.g., 6, 7, and 8), QPSO with preference order obtains solutions closer to the true pareto front, but it is very time-consuming.

When N becomes larger, the number of nondominated particles in external archive ND and the number of subsets $\binom{N}{i}$ ($i = 1, 2, \dots, N-1$) will accordingly become large. Therefore, the computing complexity of QPSO with preference order is $O(N^3)$.

TABLE 3: Description of the tested functions.

Name	Expression	Optimal solution
DTLZ1	$\min f(\vec{x}), \vec{x} \in \Omega$ $\Omega = \{\vec{x} \mid 0 \leq x_i \leq 1 \ \forall i = 1, 2, \dots, D\}$ $f_1(\vec{x}) = 0.5(1 + g(\vec{x}_N))x_1x_2 \cdots x_{N-1}$ $f_2(\vec{x}) = 0.5(1 + g(\vec{x}_N))x_1x_2 \cdots (1 - x_{N-1})$ $\vdots$ $f_N(\vec{x}) = 0.5(1 + g(\vec{x}_N))(1 - x_1)$ $g(\vec{x}_N) = 100 \left[\vec{x}_N + \sum_{x_i \in \vec{x}_N} ((x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))) \right]$	$x_i = 0.5,$ $\forall i \in \vec{x}_N$
DTLZ2	$\min f(\vec{x}), \vec{x} \in \Omega$ $\Omega = \{\vec{x} \mid 0 \leq x_i \leq 1 \ \forall i = 1, 2, \dots, D\}$ $f_1(\vec{x}) = (1 + g(\vec{x}_N)) \cos(0.5\pi x_1) \cdots \cos(0.5\pi x_{N-2}) \cos(0.5\pi x_{N-1})$ $f_2(\vec{x}) = (1 + g(\vec{x}_N)) \cos(0.5\pi x_1) \cdots \cos(0.5\pi x_{N-2}) \sin(0.5\pi x_{N-1})$ $\vdots$ $f_N(\vec{x}) = (1 + g(\vec{x}_N)) \sin(0.5\pi x_1)$ $g(\vec{x}_N) = \sum_{x_i \in \vec{x}_N} (x_i - 0.5)^2$	$x_i = 0.5,$ $\forall i \in \vec{x}_N$
DTLZ3	$\min f(\vec{x}), \vec{x} \in \Omega$ $\Omega = \{\vec{x} \mid 0 \leq x_i \leq 1 \ \forall i = 1, 2, \dots, D\}$ $f_1(\vec{x}) = (1 + g(\vec{x}_N)) \cos(0.5\pi x_1) \cdots \cos(0.5\pi x_{N-2}) \cos(0.5\pi x_{N-1})$ $f_2(\vec{x}) = (1 + g(\vec{x}_N)) \cos(0.5\pi x_1) \cdots \cos(0.5\pi x_{N-2}) \sin(0.5\pi x_{N-1})$ $\vdots$ $f_N(\vec{x}) = (1 + g(\vec{x}_N)) \sin(0.5\pi x_1)$ $g(\vec{x}_N) = 100 \left[\vec{x}_N + \sum_{x_i \in \vec{x}_N} ((x_i - 0.5)^2 - \cos(20\pi(x_i - 0.5))) \right]$	$x_i = 0.5,$ $\forall i \in \vec{x}_N$
DTLZ4	$\min f(\vec{x}), \vec{x} \in \Omega$ $\Omega = \{\vec{x} \mid 0 \leq x_i \leq 1 \ \forall i = 1, 2, \dots, D\}$ $f_1(\vec{x}) = (1 + g(\vec{x}_N)) \cos(0.5\pi x_1^\alpha) \cdots \cos(0.5\pi x_{N-2}^\alpha) \cos(0.5\pi x_{N-1}^\alpha)$ $f_2(\vec{x}) = (1 + g(\vec{x}_N)) \cos(0.5\pi x_1^\alpha) \cdots \cos(0.5\pi x_{N-2}^\alpha) \sin(0.5\pi x_{N-1}^\alpha)$ $\vdots$ $f_N(\vec{x}) = (1 + g(\vec{x}_N)) \sin(0.5\pi x_1^\alpha)$ $g(\vec{x}_N) = \sum_{x_i \in \vec{x}_N} (x_i - 0.5)^2$	$x_i = 0.5,$ $\forall i \in \vec{x}_N$

TABLE 4: Difficulty factor, number of runs, iterations, particles, and external archive.

Name	k	Number of runs	Number of iterations	M	ND
DTLZ1	5	10	600	80	80
DTLZ2	10	10	600	80	80
DTLZ3	10	10	600	80	80
DTLZ4	10	10	600	80	80

Furthermore, in QPSO with sigma value, each particle flies towards its nearest nondominated particle in external archive, which will easily result in a premature convergence and the particles will get trapped in a local optimum.

5. QPSO for Flexible Job-Shop Scheduling Problems

5.1. Flexible Job-Shop Scheduling Problems. The FJSP could be formulated as follows. There is a set of n jobs and a set

of m machines. M denotes the set of all machines. Each job i consists of a sequence of n_i operations. Each operation $O_{i,j}$ ($i = 1, 2, \dots, n; j = 1, 2, \dots, n_i$) of job i has to be processed on one machine M_k out of a set of given compatible machines $M_{i,j}$ for ($M_k \in M_{i,j}, M_{i,j} \subseteq M$). The execution of each operation requires one machine selected from a set of available machines. In addition, the FJSP needs to set each operation's starting and ending time. The FJSP is needed to determine both an assignment and a sequence of the operations on the machines in order to satisfy the given criteria. If there is $M_{i,j} \subset M$ for at least one operation, it is partial flexibility FJSP (P-FJSP); while there is $M_{i,j} = M$ for each operation, it is total flexibility FJSP (T-FJSP) [10, 11].

In this paper, the following criteria are to be minimized:

- (1) The maximal completion time of machines, that is, the makespan.
- (2) The maximal machine workload, that is, the maximum working time spent on any machine.

TABLE 5: Mean value of generational distance.

Name	N	QPSO + PO	QPSO + Rand	QPSO + Sigma
DTLZ1	2	299.14	273.50	270.71
	3	957.39	321.18	287.55
	4	548.28	408.88	298.15
	5	616.97	525.00	466.76
	6	332.29	359.63	538.87
	7	398.67	472.19	424.32
	8	298.87	415.31	830.97
DTLZ2	2	$3.8E-03$	$3.7E-03$	$1.5E-03$
	3	$1.02E-02$	$1.22E-02$	$4.9E-03$
	4	$2.76E-02$	$3.26E-02$	$1.52E-02$
	5	$6.64E-02$	$6.55E-02$	$4.48E-02$
	6	$7.34E-02$	$9.54E-02$	$7.49E-02$
	7	$9.65E-02$	0.116	0.111
	8	$9.87E-02$	0.128	0.117
DTLZ3	2	$1.31E04$	$2.40E04$	$2.59E04$
	3	$2.76E04$	$2.38E04$	$1.70E04$
	4	$3.93E04$	$2.86E04$	$2.05E04$
	5	$5.95E04$	$3.81E04$	$2.68E04$
	6	$5.35E04$	$3.98E04$	$3.19E04$
	7	$1.20E04$	$5.52E04$	$3.72E04$
	8	$4.04E04$	$4.73E04$	$4.98E04$
DTLZ4	2	$3.98E-08$	$2.64E-14$	0
	3	0	$1.1E-03$	0
	4	$9.74E-04$	$1.62E-09$	0
	5	$1.38E-06$	$2.55E-05$	$5.84E-18$
	6	$2.32E-02$	0.17	$5.3E-03$
	7	$7.11E-02$	0.16	0.21
	8	$4.11E-02$	$9.09E-02$	$1.43E-01$

TABLE 6: Mean value of spacing.

Name	N	QPSO + PO	QPSO + Rand	QPSO + Sigma
DTLZ1	2	12.51	2.58	9.56
	3	19.32	7.78	7.42
	4	8.42	10.76	9.16
	5	19.42	18.50	15.78
	6	7.80	11.61	18.24
	7	28.45	23.16	19.75
	8	5.63	17.90	27.53
DTLZ2	2	0.099	0.108	0.098
	3	0.165	0.155	0.148
	4	0.267	0.275	0.225
	5	0.429	0.360	0.305
	6	0.452	0.441	0.425
	7	0.436	0.461	0.477
	8	0.332	0.569	0.438
DTLZ3	2	104.69	22.13	1.16
	3	54.21	87.79	53.11
	4	180.30	82.14	116.29
	5	124.35	137.43	108.40
	6	219.13	143.94	111.38
	7	158.78	209.16	171.57
	8	112.1	147.65	123.78
DTLZ4	2	0	1.15	0
	3	0	$2.79E-04$	0.223
	4	$4.76E-04$	0	0.25
	5	0	0	0.23
	6	$4.5E-03$	0.12	1.10
	7	0.47	1.19	1.05
	8	$4.9E-03$	$4.81E-01$	$9.17E-01$

- (3) The total workload of machines, which represents the total working time over all machines.

During the process of solving this problem, the following assumptions are made:

- (1) Each operation cannot be interrupted during its performance (nonpreemptive condition).
- (2) Each machine can perform at most one operation at any time (resource constraint).
- (3) The precedence constraints of the operations in a job can be defined for any pair of operations.
- (4) Setting up time of machines and move time between operations are negligible.
- (5) Machines are independent from each other.
- (6) Jobs are independent from each other.
- (7) All machines are available at time 0.
- (8) Each job has its own release date.
- (9) The order of operations for each job is predefined and invariant.

The model of the multiobjective FJSP is then given as follows:

$$\min f_1 = \max_{1 \leq k \leq m} (C_k) \quad (11)$$

$$\min f_2 = \max_{1 \leq k \leq m} (W_k) \quad (12)$$

$$\min f_3 = \sum_{k=1}^m W_k \quad (13)$$

$$\text{s.t. } C_{i,j} - C_{i,(j-1)} \geq P_{i,j,k} x_{i,j,k}, \quad j = 1, 2, \dots, n_i, \forall i, j; \quad (14)$$

$$\begin{aligned} & [(C_{h,g} - t_{h,g,k} - C_{i,j}) x_{h,g,k} x_{i,j,k} \geq 0] \\ & \vee [(C_{i,j} - t_{i,j,k} - C_{h,g}) x_{h,g,k} x_{i,j,k} \geq 0], \end{aligned} \quad (15)$$

$$\forall (i, j), (h, g), k;$$

$$\sum_{k \in M_{i,j}} x_{i,j,k} = 1, \quad \forall i, j. \quad (16)$$

Inequation (14) ensures the operation precedence constraint. Inequation (15) ensures that each machine could process only

TABLE 7: Example of 3 jobs in 4 machines.

Job	Operation	M_1	M_2	M_3	M_4
J_1	O_{11}	1	3	4	1
	O_{12}	3	8	2	1
	O_{13}	3	5	4	7
J_2	O_{21}	4	1	1	4
	O_{22}	2	3	9	3
	O_{23}	9	1	2	2
J_3	O_{31}	8	6	3	5
	O_{32}	4	5	8	1

1 2 3 3 1 1 2 2 1000|0001|1000|1000|1000|0100|0010|0001

FIGURE 2: Encoding of a particle.

one operation at each time. Equation (16) states that one machine could be selected from the set of available machines for each operation.

5.2. Encoding Scheme. Particle's position representation is the most important task for successful application of PSO or QPSO to solve the FJSP. In this paper, the particle's position is represented by a vector which consists of two parts (take Table 7, e.g.). The first part represents the sequence of the operations ($O_{1,1}, O_{2,1}, O_{3,1}, O_{3,2}, O_{1,2}, O_{1,3}, O_{2,2}, O_{2,3}$). The second part represents the machines on which the operations in Table 7 are processed; take Figure 2 for example, 1000 represents $O_{1,1}$ is processed on M_1 and 0001 represents $O_{1,2}$ is processed on M_4 . The length of the vector is equal to $\sum_{i=1}^n n_i(1+m)$. The advantage of this encoding method is that it is easy to apply crossover and mutation on operation part and machine part separately, while the disadvantage is the storage complexity.

5.3. Swarm Initialization. The initial population has great influence on the performance of the algorithm. Here in this paper, a guided initialization method is used. In order to ensure the diversity of the population, 50% of the particles are initialized randomly, and the others are initialized according to the following rules.

Guided Initialization

Step 0: Initialize. It includes the following:

- Create an array for a random job order J (e.g., $J_4 J_2 J_3 J_1$).
- Assign the first operation $O_{x,1}$ of each job to a suitable machine following the order $O_{4,1} O_{2,1} O_{3,1} O_{1,1}$. This machine is selected by calling Step 1(b); $O_{x,1}$ is assigned directly to the selected machine. If there is no operation assigned to the selected machine, $O_{x,1}$ starts from time 0. Otherwise, it starts from the stop time of the last operation on the selected machine.

Step 1: Find a Suitable Machine to Process an Operation. It includes the following:

FIGURE 3: Crossover on operation part.

- Identify the machine with earliest stop time M_s and the last operation $O_{i,j}$ on M_s . If more than one machine is satisfied, a random machine of them is selected. Let t be the stop time of M_s .
- Find a suitable machine to process $O_{i,j+1}$ by the following procedure:
 - Identify all the machines that can process $O_{i,j+1}$: $P(O_{i,j+1})$.
 - Calculate the waiting time of each machine $M_k \in P(O_{i,j+1})$: if M_k does not process any operation at time t , it is equal to processing time of $O_{i,j+1}$ on M_k . Otherwise, if it is processing an operation $O_{z,y}$, its waiting time is the total processing times of operations on M_k 's waiting list + (remaining time of $O_{z,y}$) + processing time of $O_{i,j+1}$ on machine M_k .
 - The machine with shortest waiting time is chosen to process $O_{i,j+1}$. If this machine does not process any operation at time t , assign $O_{i,j+1}$ to it to be processed. Otherwise, add $O_{i,j+1}$ to the waiting list of this machine.

Step 2: Find a Suitable Order of Operations on a Machine. from Step 1(a), M_s and $O_{i,j}$ are identified, and then assign $O_{i,j+1}$ to a machine. Now an operation is needed to find on the waiting list to be processed on M_s . If there is more than one operation on the waiting list of M_s , select one of them depending on one of the following dispatching rules: shortest processing time, longest processing time, or first in first out.

Step 3. Return to Step 1.

5.4. Crossover and Mutation. For combinational optimization problems, effective information exchanging is able to help find the better solution. The crossover operator on the operation part is shown in Figure 3, in which the offspring C1 and C2 inherit some operations from the parents P1 ($\{1\}$) and P2 ($\{2, 3\}$) directly, and the other operations are copied from another parent.

The crossover operator on the machine part is shown in Figure 4, in which two positions are selected randomly, the same as that in GA.

The mutation of a particle is also divided into two steps; one is on operation part (Figure 5), in which, two positions are randomly chosen, and the operations between them are reversed.

Another mutation is on machine part, but the operation to be mutated cannot be randomly chosen. Because

TABLE 8: Comparison of results in four problems.

Problem	Objective	AL + CGA	PSO + SA	GA	QPSO + Sigma
4×5	f_1	16	11	12	11
	f_2	10	10	11	10
	f_3	34	32	33	32
8×8	f_1	16	16	15	14
	f_2	15	13	14	13
	f_3	75	73	73	73
10×10	f_1	7	7	7	7
	f_2	5	6	6	5
	f_3	45	44	43	44
15×10	f_1	24	12	12	11
	f_2	11	11	11	11
	f_3	91	91	92	91

FIGURE 4: Crossover on machine part.

FIGURE 5: Mutation on operation part.

the makespan and workload are determined by the critical path [13] in the scheduling sequence, the feasible mutation method to reduce the makespan is to choose a critical operation on the critical path, and then try to move it to another machine. Taking Figure 6, for example, the critical path is constructed by $\{O_{4,1}, O_{2,1}, O_{2,2}, O_{2,3}\}$. The operation $O_{4,1}$ is moved from M_1 to M_4 , and then the makespan is reduced from 12 to 11.

5.5. Experimental Results. As the benchmark results analysis in Section 4, QPSO with sigma value outperforms QPSO with preference order for smaller number of objectives. In this section, only three objectives are defined (makespan, max machine workload, and total workload), so QPSO with sigma value is adopted.

To illustrate the performance of QPSO with sigma value, four representative examples based on practical data have been selected [8–11]: problem 4×5 , problem 8×8 , problem 10×10 , and problem 15×10 .

The parameters are set as population size $M = 100$; size of the external archive $ND = 100$; dimensional size equals the total number of operations $\sum_{i=1}^n n_i$; the maximum number of iterations $k_{\max} = 2000$.

From Table 8, it is easy to notice that, for the problems of 4×5 , 8×8 , and 15×10 , the results obtained by QPSO with sigma dominate other algorithms. For problem

(a)

(b)

FIGURE 6: The Gantt graph of moving operation.

of 10×10 , the result obtained by QPSO with sigma is not dominated by other algorithms. Therefore, QPSO with sigma value is competitive for solving multiobjective flexible job-shop scheduling problems.

Figures 7–10 show the Gantt graph of the optimized scheduling sequence by QPSO with sigma.

6. Conclusion and Future Works

In this paper, QPSO with three elitist selection strategies are used to solve multiobjective optimization problems. The results show that QPSO with preference order has comparative performance with sigma value. When the number of objectives is small (e.g., 2, 3, 4, and 5), QPSO with sigma value performs better. However, when the number of objectives becomes larger (e.g., 6, 7, and 8), QPSO with preference order obtains solutions closer to the true pareto front but is very time-consuming. Therefore, in the future work, we will try

FIGURE 7: Gantt graph of problem 4×5 . FIGURE 8: Gantt graph of problem 8×8 . FIGURE 9: Gantt graph of problem 10×10 . FIGURE 10: Gantt graph of problem 15×10 .

to combine these two methods to achieve a balance between global search and local search.

Moreover, we apply QPSO with sigma value to solve multiobjective flexible job-shop scheduling problems, the results of which demonstrate the competitive performance of the algorithm.

Conflict of Interests

The authors declare that there is no conflict of interests regarding the publication of this paper.

Acknowledgments

This work is supported by Jiangsu Postdoctoral Funding (Project no. 1401004B), by National High-Technology Research Development Plan Project (Project no. 2013AA040405), and by National Natural Science Funding (Project no. 61300152).

References

- [1] 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.
- [2] A. Konak, D. W. Coit, and A. E. Smith, "Multi-objective optimization using genetic algorithms: a tutorial," *Reliability Engineering and System Safety*, vol. 91, no. 9, pp. 992–1007, 2006.
- [3] C. A. C. Coello, G. T. Pulido, and M. S. Lechuga, "Handling multiple objectives with particle swarm optimization," *IEEE Transactions on Evolutionary Computation*, vol. 8, no. 3, pp. 256–279, 2004.
- [4] M. Reyes-Sierra and C. A. Coello Coello, "Multi-objective particle swarm optimizers: a survey of the state-of-the-art," *International Journal of Computational Intelligence Research*, vol. 2, no. 3, pp. 287–308, 2006.
- [5] J. Sun, B. Feng, and W. B. Xu, "Particle swarm optimization with particles having quantum behavior," in *Proceedings of the*

- Congress on Evolutionary Computation (CEC '04)*, pp. 325–331, Portland, Ore, USA, June 2004.
- [6] J. Sun, W. Fang, X. J. 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.
 - [7] 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.
 - [8] P. Brandimarte, “Routing and scheduling in a flexible job shop by tabu search,” *Annals of Operations Research*, vol. 41, no. 3, pp. 157–183, 1993.
 - [9] Z. Davarzani, M. Akbarzadeh, and N. Khairdoost, “Multi-objective artificial immune algorithm for flexible job shop scheduling problem,” *International Journal of Hybrid Information Technology*, vol. 5, no. 3, pp. 75–88, 2012.
 - [10] I. Kacem, S. Hammadi, and P. Borne, “Approach by localization and multiobjective evolutionary optimization for flexible job-shop scheduling problems,” *IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews*, vol. 32, no. 1, pp. 1–13, 2002.
 - [11] I. Kacem, S. Hammadi, and P. Borne, “Pareto-optimality approach for flexible job-shop scheduling problems: hybridization of evolutionary algorithms and fuzzy logic,” *Mathematics and Computers in Simulation*, vol. 60, no. 3–5, pp. 245–276, 2002.
 - [12] G. H. Zhang, X. Y. Shao, P. G. Li, and L. Gao, “An effective hybrid particle swarm optimization algorithm for multi-objective flexible job-shop scheduling problem,” *Computers & Industrial Engineering*, vol. 56, no. 4, pp. 1309–1318, 2009.
 - [13] J. Gao, L. Sun, and M. Gen, “A hybrid genetic and variable neighborhood descent algorithm for flexible job shop scheduling problems,” *Computers & Operations Research*, vol. 35, no. 9, pp. 2892–2907, 2008.
 - [14] J. Kennedy and R. C. Eberhart, “Particle swarm optimization,” in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 1942–1948, Perth, Australia, December 1995.
 - [15] R. C. Eberhart and Y. Shi, “Comparison between genetic algorithms and particle swarm optimization,” in *Evolutionary Programming VII*, vol. 1447 of *Lecture Notes in Computer Science*, pp. 611–616, Springer, Berlin, Germany, 1998.
 - [16] F. Van den Bergh, *An analysis of particle swarm optimizers [Ph.D. thesis]*, University of Pretoria, Pretoria, South Africa, 2001.
 - [17] 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.
 - [18] S. Mostaghim and J. Teich, “Strategies for finding good local guides in multi-objective particle swarm optimization,” in *Proceedings of the IEEE Swarm Intelligence Symposium*, pp. 26–33, 2003.
 - [19] J. J. Yang, J. Zhou, L. Liu, and Y. Li, “A novel strategy of pareto-optimal solution searching in multi-objective particle swarm optimization (MOPSO),” *Computers & Mathematics with Applications*, vol. 57, no. 11–12, pp. 1995–2000, 2009.
 - [20] 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.
 - [21] I. Das, “A preference ordering among various pareto optimal alternatives,” *Structural Optimization*, vol. 18, no. 1, pp. 30–35, 1999.
 - [22] F. di Pierro, S.-T. Khu, and D. A. Savić, “An investigation on preference order ranking scheme for multiobjective evolutionary optimization,” *IEEE Transactions on Evolutionary Computation*, vol. 11, no. 1, pp. 17–45, 2007.
 - [23] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, “Scalable multi-objective optimization test problems,” in *Proceedings of the Congress on Evolutionary Computation (CEC '02)*, vol. 1, pp. 825–830, May 2002.