A Swarm Optimization Genetic Algorithm Based on Quantum-Behaved Particle Swarm Optimization

Tao Sun1,2 and Ming-hai Xu1

1College of Pipeline and Civil Engineering, China University of Petroleum, Qingdao 266580, China
2Shengli College, China University of Petroleum, Dongying, Shandong 257000, China

Correspondence should be addressed to Ming-hai Xu; minghai@upc.edu.cn

Received 30 January 2017; Revised 10 April 2017; Accepted 20 April 2017; Published 25 May 2017

Quantum-behaved particle swarm optimization (QPSO) algorithm is a variant of the traditional particle swarm optimization (PSO). The QPSO that was originally developed for continuous search spaces outperforms the traditional PSO in search ability. This paper analyzes the main factors that impact the search ability of QPSO and converts the particle movement formula to the mutation condition by introducing the rejection region, thus proposing a new binary algorithm, named swarm optimization genetic algorithm (SOGA), because it is more like genetic algorithm (GA) than PSO in form. SOGA has crossover and mutation operator as GA but does not need to set the crossover and mutation probability, so it has fewer parameters to control. The proposed algorithm was tested with several nonlinear high-dimension functions in the binary search space, and the results were compared with those from BPSO, BQPSO, and GA. The experimental results show that SOGA is distinctly superior to the other three algorithms in terms of solution accuracy and convergence.

1. Introduction

Particle swarm optimization (PSO) algorithm is a population-based optimization method, which was originally introduced by Eberhart and Kennedy in 1995 [1]. In PSO, the position of a particle is represented by a vector in search space, and the movement of the particle is determined by an assigned vector called the velocity vector. Each particle updates the velocity based on its current velocity, the best previous position of the particle, and the global best position of the population. PSO is extensively used for the optimization problems because it has simple structures and is easy to implement. However, it has some disadvantages, such that it easily falls into local optima when solving the complex and high-dimension problems [2, 3]. Hence a number of variant algorithms have been proposed to overcome the disadvantages of PSO [4, 5].

The particle swarm algorithm based on the probability movement way. The Bare Bones PSO (BBPSO) family is a typical class of probabilistic PSO algorithms [6–8]. The Gaussian distribution was used in the original version of BBPSO, which was proposed by Kennedy [6], then several new BBPSO variants used other distributions which seem to generate better results [7–9].

Inspired by the quantum theory and the trajectory analysis of PSO [10], Sun et al. proposed a new probabilistic algorithm, quantum-behaved particle swarm optimization (QPSO) algorithm [11]. In QPSO, each particle has a target point, which is defined as a linear combination of the best previous position of the particle and the global best position. The particle appears around the target point following a double exponential distribution. The QPSO algorithm essentially belongs to the BBPSO family, and its update equation uses an adaptive strategy and has fewer parameters to be adjusted [12–14]. The QPSO has been shown to perform well in finding the optimal solutions for continuous optimization problems and successfully applied to a wide range of areas such as multiobjective optimization [15, 16], clustering [17–19], neural network training [20–22],
image processing [23, 24], engineering design [25], and dynamic optimization [26].

PSO and QPSO have been effective tools for solving global optimization problems, but they were originally developed for continuous search spaces. Kennedy and Eberhart introduced a binary version of PSO for discrete problems named binary PSO (BPSO) [27], where the trajectories are defined as changes in the probability that each particle changes its state to 1. Binary PSO has simple structure and is easy to implement; hence, it is extensively employed in the optimization problems [28–30]. But it also suffers from some disadvantages when solving the complex and high-dimension problems [28]. Sun et al. proposed binary QPSO (BQPSO), in which the target point is obtained by using the crossover operator at the best previous position of the particle and the global best position. Experiment results show that BQPSO can find better solution generally than BPSO [31].

In recent years, BQPSO has been used successfully in many fields [32–34]. However, although BQPSO broadens the application fields of QPSO, it did not show the same advantage as in the continuous space. QPSO algorithm should have better performance in solving the problems based on discrete space. This paper analyzes the main factors that impact the search ability of QPSO and converts the particle movement formula to the mutation condition by the introduction of rejection region. It then designed a new binary coding QPSO, which has crossover and mutation operator and is like genetic algorithm (GA) in form; that is, the proposed algorithm is a new genetic algorithm but incorporates the core idea of QPSO. So it was named swarm optimization genetic algorithm (SOGA).

Compared with the GA, the SOGA has no selection operator, and each individual participates in evolution based on the information of the population and its own information. At the same time, the mutation probability of the SOGA is not fixed. In the early stage of the algorithm, the probability of mutation is large and the population can keep the diversity, with the iteration of the algorithm, the mutation probability tends to zero, and the algorithm can finally converge.

The rest of this paper is organized as follows. Section 2 is a brief introduction of PSO and binary PSO; Section 3 summarizes QPSO and binary QPSO; Section 4 introduces the mutation condition of binary coding converted from the particle movement formula in QPSO; Section 5 proposes the new binary QPSO algorithm, SOGA, and then discusses the difference between this algorithm and QPSO, GA; Section 6 presents the experiment results from the benchmark functions; finally, the paper is concluded in Section 7.

2. Particle Swarm Optimization

Particle swarm optimization (PSO) algorithm is a population-based optimization technique used in continuous spaces. It can be mathematically described as follows.

Assume the size of the population is \( n \) and the dimension of the search space is \( q \); then the \( i \)th particle of the swarm can be represented by a position vector \( X_i = (x_{i1}, x_{i2}, \ldots, x_{iq}) \); the velocity of a particle \( i \) is denoted by vector \( V_i = (v_{i1}, v_{i2}, \ldots, v_{iq}) \); vector \( P_i = (p_{i1}, p_{i2}, \ldots, p_{iq}) \) is the best previous position of particle \( i \), called personal best position, and \( P_g = (p_{g1}, p_{g2}, \ldots, p_{gq}) \) is the best position of the population, called global best position.

The velocity of particle \( i \) is calculated accordingly:

\[
v_{ij}^{k+1} = \omega v_{ij}^{k} + c_1 r_1 (p_{ij}^{k} - x_{ij}^{k}) + c_2 r_2 (p_{gj}^{k} - x_{ij}^{k}),
\]

where \( i = 1, 2, \ldots, n \), \( j = 1, 2, \ldots, q \), \( n \) is population size, \( k \) is the number of iterations, \( \omega \) is inertia weight, \( c_1 \) and \( c_2 \) are acceleration coefficients, and \( r_1 \) and \( r_2 \) are random numbers in the interval [0, 1].

Then the next position is updated as follows:

\[
x_{ij}^{k+1} = x_{ij}^{k} + v_{ij}^{k+1}.
\]

The PSO algorithm is applied to solve optimization problems in the real search space, but many optimization problems are set in discrete space. Kennedy and Eberhart proposed a discrete binary version of PSO, named binary PSO (BPSO), where the particle position has two possible values, “0” or “1.” The velocity formula in BPSO remains unchanged, and the particle position is updated as follows:

\[
x_{ij}^{k+1} = \begin{cases} x_{ij}^{k} & \text{if } S(v_{ij}^{k+1}) \leq \text{rand} \\ 0 & \text{otherwise}, \end{cases}
\]

where rand is a random number in the interval [0, 1] and the function \( S(v) \) is a Sigmoid function as

\[
S(v) = \frac{1}{(1 + e^{-v})}.
\]

3. Quantum-Behaved Particle Swarm Optimization

Inspired by trajectory analyses of PSO in [10], Sun et al. proposed a novel variant of PSO, named quantum-behaved particle swarm optimization (QPSO), which outperforms the traditional PSO in search ability.

QPSO sets a target point for each particle; denote \( G_i = (g_{i1}, g_{i2}, \ldots, g_{iq}) \) as the target point for particle \( i \), of which the coordinates are

\[
g_{ij} = \phi_{ij} p_{ij} + (1 - \phi_{ij}) p_{gj},
\]

where \( \phi_{ij} \) is a random number in the interval [0, 1]. The trajectory analysis in [10] shows that \( G_i \) is the local attractor of particle \( i \); that is, in PSO, particle \( i \) converges to it.

The position of particle \( i \) is updated as follows:

\[
x_{ij}^{k+1} = g_{ij}^{k} + \frac{L_{ij}^{k}}{2} \ln \left( \frac{1}{u} \right),
\]

where \( u \) is a random number in the interval [0, 1] and \( C = [c_1, c_1, \ldots, c_q] \) is known as the mean best position that is
defined by the average of the personal best position of all particles, accordingly,

\[ c_j = \frac{1}{n} \sum_{i=1}^{n} p_{ij}, \quad j = 1, 2, \ldots, q. \]  

Parameter \( \alpha \) is called Contraction-Expansion Coefficient, which can be tuned to control the convergence speed of the algorithms.

Because the iterations of QPSO are different from those of PSO, the methodology of BPSO cannot be applied to QPSO. Sun et al. introduced the crossover operator of GA into QPSO and proposed binary QPSO (BQPSO). In BQPSO, the methodology of BPSO cannot be applied to QPSO. However, the methodology of PSO can be tuned to control the convergence speed of the algorithms. This should also be reflected in the construction of binary QPSO algorithm.

The target point \( G_i \) for particle \( i \) is generated through crossover operator; that is, BQPSO exerts crossover operation on the personal best position \( P_i \) and the global best position \( G \) to generate two offspring binary strings, and \( G_i \) is randomly selected from them.

Define

\[ p_m = \alpha \ast d_H (c_j, x^{k}_i) \ast \ln \left( \frac{1}{u} \right), \quad u \sim U [0, 1], \]  

where \( k \) is the number of iterations and \( d_H (c_j, x^{k}_i) \) is the Hamming distance between \( c_j \) and \( x^{k}_i \). Compared with the two bit strings, the Hamming distance is the count of bit difference in the two strings. \( c_j \) is the jth substring of the mean best position, and the \( d \)th bit of \( c_j \) is determined by the states of the \( d \)th bit of all particles’ personal best positions. If more particles take on 0 at the \( d \)th bit, the \( d \)th bit of \( c_j \) is 0; otherwise the bit will be 0.

For each bit of \( g_{ij} \), when \( p_m > \text{rand} \) execute operations as follows: if the state of the bit is 1, then set its state to 0; else set its state to 0.

4. A Mutation Condition Using in Binary Space

The reason why the QPSO algorithm has better global search capability than the traditional PSO algorithm is that it changes the velocity-displacement model of the traditional PSO algorithm; in QPSO, the movement of particle to its target point has no determined trajectory; it can appear at any position in the whole feasible search space with a certain distribution, which is the double exponential distribution [13, 14]. Such a position can be far from the target point and may be superior to the current global best position of the population. This should also be reflected in the construction of binary QPSO algorithm.

The probability density function of particle \( i \) in QPSO is

\[ f (x_i) = \frac{1}{L_i} e^{-2|x_i-G_i|/L_i}, \quad \text{for} \quad x_i \in \mathbb{R}, \quad i = 1, 2, \ldots, n. \]  

Set \( \lambda_i = 2/L_i \) and \( y_i = x_i - G_i \); then (9) can be rewritten as

\[ f (y_i) = \frac{2}{\lambda_i} e^{-\lambda_i |y_i|}, \quad \lambda_i > 0. \]  

That is, \( y_i \) obeys the double exponential distribution, of which the mean and variance are \( E(y_i) = 0 \) and \( D(y_i) = 2/\lambda_i^2 \). The graph of probability density function (10) is Figure 1. Since the domain of \( y_i \) is \((-\infty, +\infty)\), particle can appear in any position of the search space, but the probability that a particle appears in a position far away from its target point is small. When \( \lambda_i \to +\infty \), the variance \( D(y_i) = 2/\lambda_i^2 \to 0 \) which means that \( X_i \) converge to \( G_i \) with probability 1.

When the position of a particle uses binary encoding, it is hard to describe the relative position of two points using the measure of two binary strings. Similar to set a rejection region, we set a threshold value \( \nu \) \((\nu > 0)\). When the value of \( y_i \) falls into the rejection region, as shown in Figure 2, set \( y_i = 0 \); that is, \( X_i = G_i \), else \( X_i = \text{mutation}(G_i) \). mutation\( (G_i) \) means mutation operation on \( G_i \).
For any \( u \), which is a random number in the interval \([0, 1]\), the condition that \( y_i \) does not fall into the rejection region is

\[
P(\|y_i\| > v) > u.
\]  

(11)

The left side of Condition (11) can be written as

\[
P(\|y_i\| > v) = \int_{-\infty}^{-v} \frac{\lambda_i}{2} e^{\lambda_i t} dt + \int_{v}^{\infty} \frac{\lambda_i}{2} e^{-\lambda_i t} dt
\]

\[= 2 \int_{v}^{\infty} \frac{\lambda_i}{2} e^{-\lambda_i t} dt = e^{-\lambda_i v}.
\]

(12)

Thus Condition (11) means \( e^{-\lambda_i v} > u \), accordingly:

\[
\lambda_i v < \ln \left( \frac{1}{u} \right).
\]

(13)

In order to ensure that the algorithm can converge, set

\[
\lambda_i = \frac{1}{d_H(C, X_i)},
\]

(14)

where \( C \) is the mean best position of the population.

Then Condition (13) is

\[
\frac{v}{d(C, X_i)} < \ln \left( \frac{1}{u} \right),
\]

(15)

where \( d(\cdot) \) is used to measure the difference of two binary strings. Hamming distance can be used here.

Assume \( y = \ln(1/u) \); then

\[
u = \exp(-y), \quad y > 0.
\]

(16)

For (16), when the value of \( y \) is small, the function has fast rates of change as shown in Figure 3, so Condition (15) suffers from the effect of the initial value of \( d(C, X_i) \). So Condition (15) can be changed into its equivalent form:

\[
\sigma d(C, X_i) > \ln \left( \frac{1}{u} \right),
\]

(17)

where parameter \( \sigma \) is a constant that is greater than zero.

### 5. Swarm Optimization Genetic Algorithm

Based on the mutation condition (17), mutation operator is introduced into BQPSO. \( X_i \) still represents the position of particle \( i \), \( P_i \) is the personal best position of particle \( i \), \( P_g \) is the global best position, and \( C \) is the mean best position which is defined the same as in BQPSO.

Different from BQPSO, crossover or mutation operation process is applied to the whole binary string, instead of bits. Because the procedure of the algorithm is similar to GA, it is named as swarm optimization genetic algorithm (SOGA). The process can be described as follows.

1. Initialize a population of particles \( X_i \) in binary space;
2. Set personal best position \( P_i = X_i \), and compute \( C \);
3. Evaluate the fitness of particles \( f(X_i) \) and determine the global best position \( P_g \);
4. while terminate condition is not reached do
5. for each particle \( i \) do
(6) Exert crossover operation on \( P_i \) and \( P_g \) to generate two offspring binary strings,

(7) \( G_i \) is randomly selected from them.

(8) if condition (17) is true,

(9) Exert mutation operation on \( G_i \);

(10) endif

(11) Set \( X_i = G_i \);

(12) Compute the fitness of particles \( f(X_i) \), and update \( P_i \),

(13) end for

(14) Update \( P_g \) and the mean best position \( C \);

(15) endwhile

Compared to the GA with the same crossover and mutation operator, SOGA has the following characteristics:

(1) SOGA does not have selection operator and crossover probability and its crossover operator is exerted directly on \( P_i \) and \( P_g \). Therefore, the form of the fitness function \( f(X_i) \) has no effect on the algorithm, and the target function of the maximization problem can be set as the fitness function.

(2) Condition (17) can be turned into \( u > \exp(-\sigma d(C, X_i)) \).

Since \( \sigma d(C, X_i) \geq 0 \), the range of \( \exp(-\sigma d(C, X_i)) \) is \((0, 1)\), and \( u \) is a random number in the interval \([0, 1]\); thus Condition (18) is equivalent to an adaptive mutation probability:

\[
p_m = 1 - \exp(-\sigma d(C, X_i)),
\]

where \( \sigma \) is a constant that is greater than zero and \( d(C, X_i) \) decreases with the increase of iteration times. Therefore, \( p_m \) is shrunk, which causes the algorithm to converge.

(3) \( \sigma \) is the only parameter of SOGA, which can be tuned to control the convergence speed of the algorithms as Contraction-Expansion Coefficient \( \alpha \) in BQPSO.

When the value of \( \sigma \) is 0.5, 1, and 2, the curves of mutation probability \( p_m \) changing with \( d() \) are shown in Figure 4. The figure demonstrates that the smaller the value of \( \sigma \), the faster the convergence speed of the algorithm. It also can be seen that the global searching ability of the algorithm is reduced when \( \sigma \) is too small. So set \( \sigma = 1 \) in SOGA.

6. Experimental Results

The proposed SOGA is compared with BPSO, BQPSO, and GA. They are tested on the following 10 benchmark problems to be minimized [28, 35]:

(1) Sphere Function

\[
F_1(X) = \sum_{i=1}^{n} x_i^2, \quad |x_i| \leq 100.
\]

(2) Schwefel’s Problem 2.22

\[
F_2(X) = \sum_{i=1}^{n} |x_i| + \prod_{i=1}^{n} |x_i|, \quad |x_i| \leq 10.
\]

(3) Schwefel’s Problem 1.2

\[
F_3(X) = \sum_{i=1}^{n} \left( \sum_{j=1}^{i} x_j \right)^2, \quad |x_i| \leq 100.
\]

(4) Step Function

\[
F_4(X) = \sum_{i=1}^{n} (|x_i + 0.5|)^2, \quad |x_i| \leq 100.
\]

(5) Schwefel’s Problem 2.21

\[
F_5(X) = \max_i \{|x_i|, \quad i = 1, 2, \ldots, n, \quad |x_i| \leq 100.
\]

(6) 2^n Minima Function

\[
F_6(X) = \frac{1}{n} \sum_{i=1}^{n} (x_i^4 - 16x_i^2 + 5x_i), \quad |x_i| \leq 5.
\]
The parameters of algorithms are listed in Table 1, where population is 50 and the total number of iterations is set to 2000 in Table 3 to ensure that the algorithm is fully convergent.

Moreover, the statistical test is conducted in order to determine whether the average best results are different with a statistical significance. The confidence level is fixed at 0.95, and the tests return $p$ value which are shown in Tables 4 and 5. We use the SAS for statistical testing; in the SAS system, if the $p$ value is less than 0.0001, the system displays < 0.0001. The value of $h$ in Tables 4 and 5 shows the result of pairwise comparison; $h = 1$ indicates the previous comparison algorithm is significantly better than the latter; $h = 0$ represents no significant difference between the two compared algorithms; $h = -1$ indicates the previous comparison algorithm is significantly worse than the latter.

The results of SOGA compared with BPSO and BQPSO are listed in Tables 2 and 4. The results show that SOGA surpasses BPSO and BQPSO in minimizing the ten benchmark functions except $F_5$. Figure 5 illustrates the convergence process of the best target function value of population in one running. As shown in Figure 5, the SOGA converges faster than BPSO and BQPSO.

Since SOGA has almost the same form as GA, the same crossover and mutation operator, single-point crossover and single-point mutation, are used in both algorithms. In GA, the elitist strategy is applied to improve the convergence and optimization results. It should be noted that the GA can not converge after 500 iterations for most of the functions; for better comparison, the number of iterations of the GA is set to 2000 in Table 3 to ensure that the algorithm is fully convergent.

For high-dimension functions, assume $X_i = (x_{i1}, x_{i2}, \ldots, x_{iq})$ is the binary string of the particle (or individual) $i$, where $q$ is the number of dimensions and $x_{ij}$ is the $j$th substring of $X_i$. It is easy for GA or SOGA to exert crossover and mutation operation on each substring $x_{ij}$ in turn, instead of on the whole $X_i$. For instance, in SOGA, Condition (17) can be written as

$$
\sigma d(c_j, x_{ij}) > \ln \left( \frac{1}{u} \right), 
$$

### Table 1: Parameters of algorithms applied in the experiments.

| Algorithm | Parameter settings |
|-----------|--------------------|
| SOGA     | $\sigma = 1$      |
| BPSO     | $\omega = 0.7, c_1 = c_2 = 2, V_{max} = 6$ |
| BQPSO    | $\alpha = 1.1$      |
| GA       | $p_c = 0.90, p_m = 0.10$ to 0.15 |

In these functions, $F_1$ to $F_6$ are unimodal and $F_6$ to $F_{10}$ are multimodal. Their optimum values are all zeros except $F_8$ and $F_9$. The minimum values of $F_8$ and $F_9$ are $-78.3323 \pm 32$ and $-418.9829 \pm 32$, respectively, where $n$ is the dimension of a function.

In the experiments, the dimension of each function is 8, and the binary code length of each continuous variable is 15, so the length of particle is 120 for each function. The size of population is 50 and the total number of iterations is set to 500. The parameters of algorithms are listed in Table 1, where $p_c$ is crossover probability and $p_m$ is mutation probability in GA.

Four algorithms ran independently 30 times on the benchmark functions, and the best target function value was recorded at each run. To compare the four algorithms, 30 data sets were analyzed using the following statistic parameters: the mean, the standard deviation (STD), the best, the worst, and the median; these results are reported in Tables 2 and 3.

Moreover, the statistical test is conducted in order to determine whether the average best results are different with a statistical significance. The confidence level is fixed at 0.95, and the tests return $p$ value which are shown in Tables 4 and 5. We use the SAS for statistical testing; in the SAS system, if the $p$ value is less than 0.0001, the system displays < 0.0001. The value of $h$ in Tables 4 and 5 shows the result of pairwise comparison; $h = 1$ indicates the previous comparison algorithm is significantly better than the latter; $h = 0$ represents no significant difference between the two compared algorithms; $h = -1$ indicates the previous comparison algorithm is significantly worse than the latter.

The results of SOGA compared with BPSO and BQPSO are listed in Tables 2 and 4. The results show that SOGA surpasses BPSO and BQPSO in minimizing the ten benchmark functions except $F_5$. Figure 5 illustrates the convergence process of the best target function value of population in one running. As shown in Figure 5, the SOGA converges faster than BPSO and BQPSO.

Since SOGA has almost the same form as GA, the same crossover and mutation operator, single-point crossover and single-point mutation, are used in both algorithms. In GA, the elitist strategy is applied to improve the convergence and optimization results. It should be noted that the GA can not converge after 500 iterations for most of the functions; for better comparison, the number of iterations of the GA is set to 2000 in Table 3 to ensure that the algorithm is fully convergent.

For high-dimension functions, assume $X_i = (x_{i1}, x_{i2}, \ldots, x_{iq})$ is the binary string of the particle (or individual) $i$, where $q$ is the number of dimensions and $x_{ij}$ is the $j$th substring of $X_i$. It is easy for GA or SOGA to exert crossover and mutation operation on each substring $x_{ij}$ in turn, instead of on the whole $X_i$. For instance, in SOGA, Condition (17) can be written as

$$
\sigma d(c_j, x_{ij}) > \ln \left( \frac{1}{u} \right), 
$$

### Ackley Function

$$
F_8 (X) = -20 \exp \left( -0.2 \sqrt{\frac{1}{n} \sum_{i=1}^{n} x_i^2} \right) - \exp \left( \frac{1}{n} \sum_{i=1}^{n} \cos \left( \frac{2\pi x_i}{n} \right) \right) + 20 + e, 
$$

### Generalized Penalized Function

$$
F_{10} (X) = \frac{\pi}{4000} \sum_{i=1}^{n} x_i^2 - \sum_{i=1}^{n} \cos \left( \frac{x_i}{\sqrt{i}} \right) + 1, 
$$

where

$$
y_1 = 1 + \frac{(x_1 + 1)}{4}, \quad u(x_i, a, k, m) = \begin{cases} 
 k(x_i - a)^m & x_i > a, \\
 0 & -a \leq x_i \leq a, \\
 k(-x_i - a)^m & x_i < -a.
\end{cases}
$$
Figure 5: Continued.
for each substring $x_{ij}$ of $X_i$, where $c_j$ are the $j$th substring of $C$. Then the process of SOGA when the crossover and mutation operation act on substring can be described as follows: the same operation can also be used in GA.

1. Initialize a population of particles $X_i$ in binary space;
2. Set personal best position $P_i = X_i$, and compute $C$;
3. Evaluate the fitness of particles $f(X_i)$ and determine the global best position $P_g$;
4. while terminate condition is not reached do
5. for each particle $i$ do
6. for each substring of particle $j$ do
   (7) Exert crossover operation on $P_{ij}$ and $P_{gj}$ to generate two offspring binary strings,
   (8) $G_{ij}$ is randomly selected from them,
   (9) if condition (30) is true,
   (10) Exert mutation operation on $G_{ij}$;
   (11) end if
   (12) Set $X_{ij} = G_{ij}$;
   (13) end for $j$
   (14) Compute the fitness of particles $f(X_i)$, and update $P_i$,
   (15) end for $i$

Figure 5: Figures of the convergence processes of BPSO, BQPSO, and SOGA.
Figure 6: Continued.
(16) Update $P_g$ and the mean best position $C$;
(17) endwhile

The convergence processes of SOGA and GA, when crossover and mutation operation act on substrings of particles (or individual), are shown in Figure 6; the results of SOGA and GA are listed in Tables 3, 4, and 5. The experimental results show that SOGA is obviously superior to the GA on the solution accuracy and the convergence. For high-dimension functions, it is effective to improve the convergence speed and optimization ability, exerting crossover and mutation operation on substrings; as shown in Tables 3 and 4, it significantly improves the convergence rate of GA. But its influence is not significant for SOGA; according to Table 5, it has better performance in minimizing functions $F_3$, $F_7$, $F_8$, and $F_9$, especially for function $F_3$.

7. Conclusions
In this study, SOGA, a binary swarm intelligence algorithm, which is based on QPSO and binary QPSO, is introduced.
### Table 2: Minimization results for BPSO, BQPSO, and SOGA.

| Function | Algorithm | The best | Mean   | SD     | The worst | Median  |
|----------|-----------|----------|--------|--------|-----------|---------|
|          | BPSO      | 65.1674  | 278.1128 | 160.0441 | 702.2221 | 244.4259 |
| $F_1$    | BQPSO     | 1.4096   | 3.3647  | 1.4272  | 6.5327    | 3.1107  |
|          | SOGA      | 7.4510e−05 | 1.6641e−04 | 4.8945e−04 | 0.0028    | 7.4510e−05 |
| $F_2$    | BPSO      | 1.5918   | 3.0268  | 0.9300  | 5.3401    | 2.9463  |
|          | BQPSO     | 0.2155   | 0.4080  | 0.1226  | 0.7025    | 0.4025  |
|          | SOGA      | 0.0024   | 0.0026  | 0.0005  | 0.0049    | 0.0024  |
| $F_3$    | BPSO      | 121.8395 | 1171.3480 | 911.2165 | 3261.0044 | 860.7055 |
|          | BQPSO     | 8.2112   | 30.8752 | 25.8621 | 120.6034 | 21.4084 |
|          | SOGA      | 263.9170 | 1919.6076 | 1078.9650 | 3782.1623 | 2175.0068 |
| $F_4$    | BPSO      | 56       | 312.7667 | 248.9619 | 1154      | 250.5   |
|          | BQPSO     | 1        | 4.8667  | 2.4174  | 11        | 5       |
|          | SOGA      | 0        | 0.1000  | 0.4026  | 2         | 0       |
| $F_5$    | BPSO      | 9.0243   | 16.3089 | 5.5677  | 27.5491   | 14.3406 |
|          | BQPSO     | 0.9430   | 1.8437  | 0.5141  | 3.3296    | 1.8342  |
|          | SOGA      | 0.0031   | 0.2228  | 0.6241  | 3.1281    | 0.0275  |
| $F_6$    | BPSO      | −77.5179 | −74.5146 | 1.8144  | −70.6644  | −74.5941 |
|          | BQPSO     | −72.7114 | −69.3385 | 1.7327  | −66.4601  | −68.9703 |
|          | SOGA      | −77.7357 | −76.2789 | 0.6355  | −75.1676  | −76.2048 |
| $F_7$    | BPSO      | −3136.9403 | −2724.3377 | 194.8185 | −2263.5189 | −2752.3562 |
|          | BQPSO     | −1810.3356 | −1498.0555 | 157.2211 | −1272.7048 | −1454.7974 |
|          | SOGA      | −3248.4725 | −2913.3245 | 218.1944 | −2467.9053 | −2952.6730 |
| $F_8$    | BPSO      | 4.5456   | 7.6315  | 2.0230  | 15.5215   | 7.8838  |
|          | BQPSO     | 0.9446   | 1.9230  | 0.3842  | 2.5818    | 1.9273  |
|          | SOGA      | 0.0040   | 1.3614  | 1.1703  | 3.1276    | 1.8407  |
| $F_9$    | BPSO      | 2.6925   | 12.6817 | 7.3535  | 31.3502   | 10.9785 |
|          | BQPSO     | 0.7294   | 1.7635  | 0.5748  | 2.9455    | 1.7687  |
|          | SOGA      | 0.0742   | 1.4639  | 1.4386  | 5.2594    | 0.9397  |
| $F_{10}$ | BPSO      | 1.5798   | 3.6606  | 1.5582  | 7.7680    | 3.4519  |
|          | BQPSO     | 0.3552   | 0.8360  | 0.2063  | 1.1561    | 0.8923  |
|          | SOGA      | 0.0546   | 0.3702  | 0.1933  | 0.6738    | 0.4183  |

It converts the movement formula of QPSO to mutation conditions, thus introducing the mutation operator of GA. SOGA has the similar form to GA but does not need to set the crossover and mutation probability, so it has fewer parameters to control. SOGA integrates strongpoint of GA and PSO. The experimental results show that SOGA is distinctly superior to BPSO, BQPSO, and GA in terms of solution accuracy and convergence. Furthermore, since SOGA has the same crossover and mutation operator as GA, many improvements on the GA can be applied to it; therefore, this algorithm has better applications and research prospects.
Table 3: Minimization results for SOGA and GA.

| Function | Algorithm | The best   | Mean       | SD          | The worst   | Median    |
|----------|-----------|------------|------------|-------------|-------------|-----------|
|          | SOGA      | 7.4510e − 05 | 1.6641e − 04 | 4.8945e − 04 | 0.0028      | 7.4510e − 05 |
| $F_1$    | GA        | 0.0013     | 76.0248     | 166.2636    | 664.4937    | 2.5246    |
|          | SOGA*     | 7.4510e − 05 | 0.0074     | 0.0189      | 0.0773      | 0.0007    |
|          | GA*       | 0.0016     | 1.1302     | 2.3717      | 11.0953     | 0.2626    |
| $F_2$    | SOGA      | 0.0024     | 0.0026     | 0.0005      | 0.0049      | 0.0024    |
|          | GA        | 0.0031     | 0.3220     | 0.4294      | 1.3587      | 0.1172    |
|          | SOGA*     | 0.0031     | 0.0069     | 0.0076      | 0.0433      | 0.1059    |
|          | GA*       | 0.0220     | 0.1423     | 0.1452      | 0.7703      | 2.5471    |
| $F_3$    | SOGA      | 263.9170   | 1919.6076  | 1078.9650   | 3782.1623   | 2175.0068 |
|          | GA        | 823.0982   | 3805.9546  | 1587.4665   | 8753.2186   | 3602.7854 |
|          | SOGA*     | 0.0238     | 27.5765    | 95.7945     | 434.9900    | 0.5557    |
|          | GA*       | 482.1519   | 4922.3981  | 2417.1198   | 10470.9239  | 4705.2068 |
| $F_4$    | SOGA      | 0.0031     | 0.2228     | 0.6241      | 3.1281      | 0.0275    |
|          | GA        | 9.3783     | 27.0193    | 11.2953     | 53.1297     | 27.1523   |
|          | SOGA*     | 0.0458     | 0.1908     | 0.0952      | 0.4913      | 0.1831    |
|          | GA*       | 6.4913     | 20.0783    | 10.5937     | 49.6017     | 18.9795   |
| $F_5$    | SOGA      | −77.7357   | −76.2789   | 0.6355      | −75.1676    | −76.2048  |
|          | GA        | −77.9857   | −75.2636   | 1.8587      | −69.1535    | −75.5698  |
|          | SOGA*     | −78.3316   | −77.2289   | 0.6240      | −75.8823    | −77.1074  |
|          | GA*       | −77.5952   | −74.8886   | 1.902       | −70.5101    | −75.457   |
| $F_6$    | SOGA      | −3248.4725 | −2642.8652 | 256.1202    | −2001.7012  | −2952.6730|
|          | GA        | −3113.4015 | −2088.3848 | 342.5752    | −1355.6609  | −2692.5141|
|          | SOGA*     | −3351.7352 | −3111.4602 | 154.6330    | −2736.5711  | −3114.6197|
|          | GA*       | −3150.5510 | −2733.6038 | 180.6351    | −2424.3367  | −2726.4449|
| $F_7$    | SOGA      | 0.0040     | 1.3614     | 1.1703      | 3.1276      | 1.8407    |
|          | GA        | 1.8409     | 3.3475     | 1.8626      | 10.2185     | 2.6024    |
|          | SOGA*     | 0.0040     | 0.0223     | 0.0216      | 0.0911      | 0.0141    |
|          | GA*       | 0.3384     | 2.7764     | 0.9262      | 4.4358      | 2.7676    |
| $F_8$    | SOGA      | 0.0742     | 1.4639     | 1.4386      | 5.2594      | 0.9397    |
|          | GA        | 1.5468     | 10.2883    | 7.6703      | 34.3495     | 8.7194    |
|          | SOGA*     | 0.0348     | 0.6195     | 0.5317      | 1.9344      | 0.4575    |
|          | GA*       | 0.6679     | 6.0007     | 4.0074      | 13.6647     | 5.7359    |
| $F_9$    | SOGA      | 0.0546     | 0.3702     | 0.1933      | 0.6738      | 0.4183    |
|          | GA        | 0.1600     | 0.9633     | 1.3811      | 7.9350      | 0.6882    |
|          | SOGA*     | 0.2550     | 0.4715     | 0.1208      | 0.7548      | 0.4758    |
|          | GA*       | 0.1805     | 0.7492     | 0.2431      | 1.2999      | 0.7741    |

*The crossover and mutation operation act on substring.
### Table 4: Comparison of SOGA with other algorithms and GA* with GA.

| Function | Test | BPSO | BQPSO | SOGA | GA | GA* | GA |
|----------|------|------|-------|------|----|-----|----|
| $F_1$    | $p$ value | $<0.0001$ | $<0.0001$ | $0.0151$ | $0.0115$ | $0.0166$ | |
| $h$      | 1 | 1 | 1 | 1 | 1 | 1 |
| $F_2$    | $p$ value | $<0.0001$ | $<0.0001$ | $0.0000$ | $<0.0001$ | $0.0340$ | |
| $h$      | 1 | 1 | 1 | 1 | 1 | 1 |
| $F_3$    | $p$ value | $0.0724$ | $<0.0001$ | $<0.0001$ | $<0.0001$ | $0.4280$ | |
| $h$      | $-1$ | $-1$ | 1 | 1 | 0 | 0 |
| $F_4$    | $p$ value | $<0.0001$ | $<0.0001$ | $0.2633$ | $0.0119$ | $0.2816$ | |
| $h$      | 1 | 1 | 0 | 1 | 0 | 0 |
| $F_5$    | $p$ value | $<0.0001$ | $<0.0001$ | $<0.0001$ | $<0.0001$ | $0.0171$ | |
| $h$      | 1 | 1 | 1 | 1 | 1 | 1 |
| $F_6$    | $p$ value | $<0.0001$ | $<0.0001$ | $0.0006$ | $<0.0001$ | $0.4430$ | |
| $h$      | 1 | 1 | 1 | 1 | 0 | 0 |
| $F_7$    | $p$ value | $0.0081$ | $<0.0001$ | $0.0004$ | $0.0080$ | $0.2582$ | |
| $h$      | 1 | 1 | 1 | 1 | 1 | 0 |
| $F_8$    | $p$ value | $<0.0001$ | $0.0002$ | $<0.0001$ | $<0.0001$ | $0.2063$ | |
| $h$      | 1 | 1 | 1 | 1 | 1 | 0 |
| $F_9$    | $p$ value | $<0.0001$ | $0.3884$ | $<0.0001$ | $<0.0001$ | $0.0043$ | |
| $h$      | 1 | 0 | 1 | 1 | 1 | 1 |
| $F_{10}$ | $p$ value | $<0.0001$ | $<0.0001$ | $0.0291$ | $<0.0001$ | $0.4065$ | |
| $h$      | 1 | 1 | 1 | 1 | 0 | 0 |

*The crossover and mutation operation act on substring.

### Table 5: Comparison of SOGA* with other algorithms.

| Function | Test | BPSO | BQPSO | SOGA* | GA | GA* | SOGA |
|----------|------|------|-------|-------|----|-----|-------|
| $F_1$    | $p$ value | $<0.0001$ | $<0.0001$ | $0.0151$ | $0.0120$ | $0.0409$ | |
| $h$      | 1 | 1 | 1 | 1 | 1 | $-1$ |
| $F_2$    | $p$ value | $<0.0001$ | $<0.0001$ | $0.0002$ | $<0.0001$ | $0.0308$ | |
| $h$      | 1 | 1 | 1 | 1 | 1 | $-1$ |
| $F_3$    | $p$ value | $<0.0001$ | $0.8561$ | $<0.0001$ | $<0.0001$ | $0.0001$ | |
| $h$      | 1 | 0 | 1 | 1 | $1$ | $1$ |
| $F_4$    | $p$ value | $<0.0001$ | $<0.0001$ | $0.2629$ | $0.0104$ | $0.1555$ | |
| $h$      | 1 | 1 | 0 | 1 | $1$ | $0$ |
| $F_5$    | $p$ value | $<0.0001$ | $<0.0001$ | $<0.0001$ | $<0.0001$ | $0.2768$ | |
| $h$      | 1 | 1 | 1 | 1 | $1$ | $0$ |
| $F_6$    | $p$ value | $<0.0001$ | $<0.0001$ | $0.0002$ | $<0.0001$ | $0.3923$ | |
| $h$      | 1 | 1 | 1 | 1 | $1$ | $0$ |
| $F_7$    | $p$ value | $<0.0001$ | $<0.0001$ | $<0.0001$ | $<0.0001$ | $0.0001$ | |
| $h$      | 1 | 1 | 1 | 1 | $1$ | $1$ |
| $F_8$    | $p$ value | $<0.0001$ | $<0.0001$ | $<0.0001$ | $<0.0001$ | $0.0001$ | |
| $h$      | 1 | 1 | 1 | 1 | $1$ | $1$ |
| $F_9$    | $p$ value | $<0.0001$ | $<0.0001$ | $<0.0001$ | $<0.0001$ | $0.0044$ | |
| $h$      | 1 | 1 | 1 | 1 | $1$ | $1$ |
| $F_{10}$ | $p$ value | $<0.0001$ | $<0.0001$ | $0.0435$ | $<0.0001$ | $0.1231$ | |
| $h$      | 1 | 1 | 1 | 1 | $1$ | $0$ |

*The crossover and mutation operation act on substring.
Conflicts of Interest
The authors declare that they have no conflicts of interest.

Acknowledgments
The work was supported by the National Natural Science Foundation of China (551276199).

References
[1] R. C. Eberhart and J. Kennedy, ”A new optimizer using particle swarm theory,” in Proceedings of the 6th International Symposium on Micromachine and Human Science, pp. 39–43, Nagoya, Japan, October 1995.
[2] F. Van den Bergh and A. P. Engelbrecht, ”A new locally convergent particle swarm optimiser,” in Proceedings of the International Conference on Systems, Man and Cybernetics, pp. 94–99, October 2002.
[3] F. van den Bergh and A. P. Engelbrecht, ”A convergence proof for the particle swarm optimiser,” Fundamenta Informaticae, vol. 105, no. 4, pp. 341–374, 2010.
[4] R. Poli, J. Kennedy, and T. Blackwell, ”Particle swarm optimisation: an overview,” Swarm Intelligence, vol. 1, no. 1, pp. 33–57, 2007.
[5] A. A. A. Esmin, R. A. Coelho, and S. Matwin, ”A review on particle swarm optimization algorithm and its variants to clustering high-dimensional data,” Artificial Intelligence Review, vol. 44, no. 1, pp. 23–45, 2015.
[6] J. Kennedy, ”Bare bones particle swarms,” in Proceedings of the IEEE Swarm Intelligence Symposium (SIS ’03), pp. 80–87, Indianapolis, Ind, USA, 2003.
[7] J. Kennedy, ”Probability and dynamics in the particle swarm,” in Proceedings of the Congress on Evolutionary Computation, CEC ’04, pp. 340–347, June 2004.
[8] T. J. Richer and T. M. Blackwell, ”’The Lévy particle’ particle swarm optimizer with distribution adaptation,” Applied Soft Computing Journal, vol. 47, pp. 534–552, 2016.
[9] R. Vafashoar and M. R. Meybodi, ”Multi swarm bare bones particle swarm optimization with distribution adaptation,” Applied Soft Computing Journal, vol. 47, pp. 534–552, 2016.
[10] M. Clerc and J. Kennedy, ”The particle swarm-explosion, stability, and convergence in a multi-dimensional complex space,” IEEE Transactions on Evolutionary Computation, vol. 6, no. 1, pp. 58–73, 2002.
[11] J. Sun, B. Feng, and W. Xu, ”Particle swarm optimization with particles having quantum behavior,” Congress on Evolutionary Computation, vol. 70, no. 3, pp. 1571–1580, 2004.
[12] J. Sun, W. Xu, and B. Feng, ”Adaptive parameter control for quantum-behaved particle swarm optimization on individual level,” in Proceedings of IEEE International Conference on Systems, Man and Cybernetics, pp. 3049–3054, October 2005.
[13] J. Sun, W. Xu, and B. Feng, ”A global search strategy of quantum-behaved particle swarm optimization,” in Proceedings of the Conference on Cybernetics Intelligent Systems, vol. 1, pp. III–116, 2005.
[14] 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.
[15] S. N. Omkar, R. Khandelwal, T. V. S. Ananth, G. Narayana Naik, and S. Gopalakrishnan, ”Quantum behaved Particle Swarm Optimization (QPSO) for multi-objective design optimization of composite structures,” Expert Systems with Applications, vol. 36, no. 8, pp. 11312–11322, 2009.
[16] T. Zhang, T. Hu, J. W. Chen, Z. Wan, and X. Guo, ”Solving bilevel multiobjective programming problem by elite quantum behaved particle swarm optimization,” Abstract and Applied Analysis, vol. 2012, no. 5, Article ID 102482, pp. 97–112, 2012.
[17] J. Sun, W. Xu, and B. Ye, ”Quantum-Behaved particle swarm optimization clustering algorithm,” in Proceedings of the International Conference on Advanced Data Mining and Applications, pp. 4093, pp. 340–347, 2006.
[18] K. Lu, K. Fang, and G. Xie, ”A hybrid quantum-behaved particle swarm optimization algorithm for clustering analysis,” in Proceedings of the 5th International Conference on Fuzzy Systems and Knowledge Discovery, FSKD, vol. 1, pp. 21–25, China, October 2008.
[19] C. Zhang and W. Chen, ”Quantum-behaved particle swarm optimization dynamic clustering algorithm,” Advanced Materials Research, vol. 694–697, pp. 2757–2760, 2013.
[20] S. Li, R. Wang, W. Hu, and J. Sun, ”A new QPSO based bp neural network for face detection,” in Proceedings of the International Conference of Fuzzy Information and Engineering, pp. 355–363, 2007.
[21] G. Y. Lian, K. L. Huang, J. H. Chen, and F. Q. Gao, ”Training algorithm for radial basis function neural network based on quantum-behaved particle swarm optimization,” International Journal of Computer Mathematics, vol. 87, no. 1–3, pp. 629–641, 2010.
[22] C.-T. Cheng, W.-J. Niu, Z.-K. Feng, J.-J. Shen, and K.-W. Chau, ”Daily reservoir runoff forecasting method using artificial neural network based on quantum-behaved particle swarm optimization,” Water, vol. 7, no. 8, pp. 4322–4246, 2015.
[23] X. Lei and A. Fu, ”Two-dimensional maximum entropy image segmentation method based on quantum-behaved particle swarm optimization algorithm,” in Proceedings of the 4th International Conference on Natural Computation, ICNC ’08, vol. 3, pp. 692–696, October 2008.
[24] X. Su, W. Fang, Q. Shen, and X. Hao, ”An image enhancement method using the quantum-behaved particle swarm optimization with an adaptive strategy,” Mathematical Problems in Engineering, vol. 2013, no. 3, Article ID 824787, pp. 211–244, 2013.
[25] L. D. S. Coelho, ”Gaussian quantum-behaved particle swarm optimization approaches for constrained engineering design problems,” Expert Systems with Applications, vol. 37, no. 2, pp. 1676–1683, 2010.
[26] W. Fang, M. Wang, and C. Li, ”Solving dynamic optimization problems based on an improved clustering quantum-behaved particle swarm optimizer,” Journal of Computational Theoretical Nanoscience, vol. 13, no. 6, pp. 3540–3547, 2016.
[27] J. Kennedy and R. C. Eberhart, ”A discrete binary version of the particle swarm algorithm,” in Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics. Computational Cybernetics and Simulation, vol. 5, pp. 4104–4108, Orlando, Fla, USA, October 1997.
[28] Z. Beheshti, S. M. Shamsuddin, and S. Hasan, ”Mimetic binary particle swarm optimization for discrete optimization problems,” Information Sciences, vol. 299, pp. 58–84, 2015.
[29] H. Banka and S. Dara, ”A Hamming distance based binary particle swarm optimization (HDBPSO) algorithm for high
dimensional feature selection, classification and validation,” *Pattern Recognition Letters*, vol. 52, pp. 94–100, 2015.

[30] K. K. Bharti and P. K. Singh, "Opposition chaotic fitness mutation based adaptive inertia weight BPSO for feature selection in text clustering," *Applied Soft Computing Journal*, vol. 43, pp. 20–34, 2016.

[31] J. Sun, W. Xu, W. Fang, and Z. Chai, in *Adaptive and Natural Computing Algorithms*, vol. 4431 of *Lecture Notes in Computer Science*, pp. 376–385, 2007.

[32] J. Zhang, Z. Zhou, W. Gao, Y. Ma, and Y. Ye, "Cognitive Radio adaptation decision engine based on binary quantum-behaved particle swarm optimization," *Chinese Journal of Scientific Instrument*, vol. 32, no. 2, pp. 221–225, 2011.

[33] M. Xi, J. Sun, L. Liu, F. Fan, and X. Wu, “Cancer feature selection and classification using a binary quantum-behaved particle swarm optimization and support vector machine,” *Computational and Mathematical Methods in Medicine*, vol. 2016, no. 9, Article ID 3572705, pp. 1–9, 2016.

[34] J. Yan, S. Duan, T. Huang, and L. Wang, “Hybrid feature matrix construction and feature selection optimization-based multi-objective QPSO for electronic nose in wound infection detection,” *Sensor Review*, vol. 36, no. 1, pp. 23–33, 2016.

[35] J. G. Digalakis and K. G. Margaritis, "An experimental study of benchmarking functions for genetic algorithms," *International Journal of Computer Mathematics*, vol. 79, no. 4, pp. 403–416, 2002.