Open Journal of Optimization, 2015, 4, 51-60
Published Online September 2015 in SciRes. http://www.scirp.org/journal/ojop
http://dx.doi.org/10.4236/ojop.2015.43007

Quantum-Inspired Bee Colony Algorithm

Guorui Li[1], Mu Sun[2], Panchi Li[1]

[1]School of Computer and Information Tsechnology, Northeast Petroleum University, Daqing, China
[2]Beijing Branch of Daqing Oilfield Information Technology Company, Beijing, China
Email: lipanchi@vip.sina.com

Received 15 March 2015; accepted 22 August 2015; published 25 August 2015

Abstract

To enhance the performance of the artificial bee colony optimization by integrating the quantum computing model into bee colony optimization, we present a quantum-inspired bee colony optimization algorithm. In our method, the bees are encoded with the qubits described on the Bloch sphere. The classical bee colony algorithm is used to compute the rotation axes and rotation angles. The Pauli matrices are used to construct the rotation matrices. The evolutionary search is achieved by rotating the qubit about the rotation axis to the target qubit on the Bloch sphere. By measuring with the Pauli matrices, the Bloch coordinates of qubit can be obtained, and the optimization solutions can be presented through the solution space transformation. The proposed method can simultaneously adjust two parameters of a qubit and automatically achieve the best match between two adjustment quantities, which may accelerate the optimization process. The experimental results show that the proposed method is obviously superior to the classical one for some benchmark functions.

Keywords

Quantum Computing, Bee Colony Optimizing, Bloch Sphere Rotating, Algorithm Designing

1. Introduction

Artificial bee colony algorithm is proposed as an intelligent optimization algorithm which attempts to simulate bee colony to search food sources by scholars from Turkey in 2005 [1]. Compared with genetic algorithm, particle swarm optimization and other intelligent algorithm, the outstanding advantage of this algorithm is synchronously to perform the global and local search in each of iterations, which avoids the premature convergence greatly and increases the probability of obtaining the optimal solution. At present, this algorithm has already been successfully applied to numerical optimization [2]-[4], neural network design [5], digital filter design [6], and network reconfiguration in distributed system [7], construction of the minimum spanning tree [8] and many

How to cite this paper: Li, G.R., Sun, M. and Li, P.C. (2015) Quantum-Inspired Bee Colony Algorithm. *Open Journal of Optimization*, **4**, 51-60. http://dx.doi.org/10.4236/ojop.2015.43007

other fields. However, the progress has been slow in the algorithm improvement. For the design of the control parameters, Ding Haijun *et al*. present an improved artificial bee colony algorithm for the solution to TSP problem [9]. Kang Fei *et al*. propose a cultural annealing artificial bee colony algorithm [10]. Duan Haibin *et al*. present a new algorithm with application of the combination of artificial bee colony algorithm and quantum evolutionary algorithm [11].

As a new computing model, quantum computing catches wide attention of international and domestic academics for its polymorphism, superposition and concurrency. At present, the fusion with genetic algorithm, immune optimization, paticle swarm optimization and other intelligent optimization model is successfully applied. In the real quantum system, the qubits are described on the Bloch sphere with two adjustable parameters. However, in the existing quantum intelligent optimization algorithms, individuals are encoded by qubits described by unit circle with a single adjustable parameter. Evolutionary mechanism adopts quantum rotation gates and quantum non-gates, and it essentially rotates qubits about circle center, which only changes one parameter of qubits as well. Therefore quantum properties have not been fully reflected. Although Ref. [12] presents a quantum genetic algorithm based on the Bloch coordinates of qubit, however, this algorithm does not consider the match between two adjustment quantities, which means it doesn't go along the shortest path when the current qubit moves towards the target qubit. As a result, the optimization performance is influenced.

This paper proposes a new individual coding evolutionary mechanism, which is different from the coding of quantum genetic algorithm in Ref. [12]. In this paper, we directly use qubits described on the Bloch sphere to code (not the qubits coordinates). The evolutionary search is achieved by rotating the qubit about the rotation axis on the Bloch sphere. This method can automatically achieve the best match between two adjustment quantities of bee colony individual qubits. For the design of quantum-inspired bee colony algorithm, we elaborate design principle and implementation programs of the algorithm. Taking benchmark functions extremum optimization as example, it shows that the proposed method obviously outperforms the classical one by comparison.

2. Bee Colony Algorithm

Assuming the number of bees is N_s, and the number of employed bees and onlooker bees are N_e and N_u, respectively. Individual dimension is D, and individual searching space is $S = R^D$. The employed bee colony is $\boldsymbol{X} = \left(\boldsymbol{X}_1, \boldsymbol{X}_2, \cdots, \boldsymbol{X}_{N_e}\right)$ and its initial the *n*-th generation colony is $\boldsymbol{X}(0)$ and $\boldsymbol{X}(n)$, respectively. The objective function is $f: S \to R^+$. Taking minimum optimization as example, the artificial bee colony algorithm can be described as follows:

(a) Randomly generating N_s solutions, where the *i*-th solution $\boldsymbol{X}_i$ is written as:

$$X_i^j = X_{\min}^j + rand(0,1)\left(X_{\max}^j - X_{\min}^j\right), \tag{1}$$

where $j = 1, 2, \cdots, D$. Calculate the target values, and then initialize $\boldsymbol{X}(0)$ by the first N_e solutions;

(b) For the employed bee $\boldsymbol{X}_i$ in the current generation, the new position in its neighborhood can be obtained from the following equation:

$$V_i^j = X_i^j + \phi_i^j\left(X_i^j - X_k^j\right), \tag{2}$$

where $j \in \{1, 2, \cdots, D\}$, $k \in \{1, 2, \cdots, N_e\}$, and $k \neq i$, and φ_i^j is a random number in the range (0, 1);

(c) We use greedy selection operator to select the better ones in $\boldsymbol{V}_i$ and $\boldsymbol{X}_i$ for the next generation. This operator is denoted as $T_s: S^2 \to S$, and its probability distribution is written as:

$$P\left\{T_s\left(\boldsymbol{X}_i, \boldsymbol{V}_i\right) = \boldsymbol{V}_i\right\} = \begin{cases} 1, & f\left(\boldsymbol{V}_i\right) < f\left(\boldsymbol{X}_i\right) \\ 0, & f\left(\boldsymbol{V}_i\right) \geq f\left(\boldsymbol{X}_i\right) \end{cases}; \tag{3}$$

(d) Using the strategy of roulette, randomly select an employed bee, and search a new position in its neighborhood. This operator is written as $T_{s1}: S^{N_e} \to S$, and its probability distribution is written as:

$$P\left\{T_{s1}\left(\boldsymbol{X}\right) = \boldsymbol{X}_i\right\} = f\left(\boldsymbol{X}_i\right) \Big/ \sum_{m=1}^{N_e} f\left(\boldsymbol{X}_m\right); \tag{4}$$

(e) Let f_{best} denote the minimum objective function value, and the corresponding individual be $\left(x_1, x_2, \cdots, x_D\right)$. We initialize the employed bee if its searching time *Bas* reach to the threshold value *Limit* and it does not find a better position;

(f) If algorithm meet the stopping criteria, it outputs f_{best} and the corresponding individual $(x_1, x_2, \cdots, x_D)$, else go back to (b).

3. Quantum-Inspired Bee Colony Algorithm

This paper studies a new method of quantum-inspired searching with the fusion of bee algorithm. This method is named quantum-inspired bee colony algorithm (QIBC).

3.1. Description of Qubits on the Bloch Sphere

During the quantum computing, a qubit is a two-level quantum system which could be described in two-dimension complex Hilbert space. Based on principle of superposition, a qubit can be defined as:

$$|\varphi\rangle = \cos\frac{\theta}{2}|0\rangle + e^{i\varphi}\sin\frac{\theta}{2}|1\rangle, \tag{5}$$

where $0 \le \theta \le \pi$ and $0 \le \phi \le 2\pi$.

Owing to the continuity of θ and φ, a qubit can be in infinitely many different states and described by a point on the Bloch sphere. As shown in **Figure 1**.

3.2. The Rotation of Qubits about the Axis

In this paper, we create a search mechanism on the Bloch sphere which is used to rotate the qubit about the rotation axis to the target qubit on the Bloch sphere. The rotation can simultaneously change two parameters of a qubit, and automatically achieve the best match between two adjustment quantities, thus we can enhance the performance of optimization. The key to the above rotation is the design of rotation axis. The design method given by this paper can be expressed as the follows [13] [14].

Assuming $\boldsymbol{P} = [p_x, p_y, p_z]$ and $\boldsymbol{Q} = [q_x, q_y, q_z]$ denote two points on the Bloch sphere respectively, the rotation axis $\boldsymbol{R}_{axis}$ about which $\boldsymbol{P}$ rotates to $\boldsymbol{Q}$ along the shortest path is the vector product of $\boldsymbol{P}$ and $\boldsymbol{Q}$, namely, $\boldsymbol{R}_{axis} = \boldsymbol{P} \times \boldsymbol{Q}$. As shown in **Figure 2**.

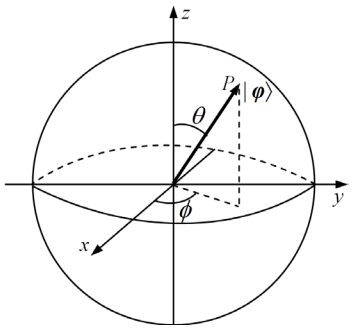

Figure 1. A qubit description on the Bloch sphere.

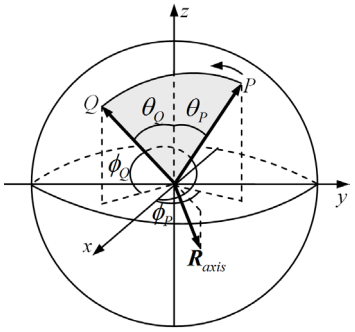

Figure 2 Rotation axis of qubit on the Bloch sphere.

Based on the principle of quantum computing, the rotation matrix makes the qubit rotate about a rotation axis along the unit vector $\boldsymbol{n}=\left[n_x, n_y, n_z\right]$ with radian of rotation δ, and it's defined by

$$\boldsymbol{R}_{\boldsymbol{n}}(\delta)=\cos\frac{\delta}{2}\boldsymbol{I}-\mathrm{i}\sin\frac{\delta}{2}(\boldsymbol{n}\times\boldsymbol{\sigma}), \tag{6}$$

where $\boldsymbol{I}$ is the unit matrix, and $\boldsymbol{\sigma}=\left[\boldsymbol{\sigma}_x, \boldsymbol{\sigma}_y, \boldsymbol{\sigma}_z\right]$, $\boldsymbol{\sigma}_x, \boldsymbol{\sigma}_y, \boldsymbol{\sigma}_z$ are Pauli matrices [15].

Therefore, on the Bloch sphere, the rotation matrix that rotates the $\left|\boldsymbol{\varphi}_{ij}(t)\right\rangle$ about $\boldsymbol{R}_{axis}(i, j)$ to $\left|\boldsymbol{\varphi}_{bj}(t)\right\rangle$ can be described as follows:

$$\boldsymbol{M}_{ij}=\cos\frac{\delta_{ij}(t)}{2}\boldsymbol{I}-\mathrm{i}\sin\frac{\delta_{ij}(t)}{2}\left(\boldsymbol{R}_{axis}(i, j)\times\boldsymbol{\sigma}\right). \tag{7}$$

The rotation operation is given by

$$\left|\boldsymbol{\varphi}_{ij}(t)\right\rangle=\boldsymbol{M}_{ij}\left|\boldsymbol{\varphi}_{ij}(t)\right\rangle, \tag{8}$$

where t is iteration step.

3.3. QIBC Coding Method

In QIBC, the individual coding adopts qubit based on the Bloch sphere. Let N_s denote population size, D denote the number of variables, and $\boldsymbol{P}=\left[\boldsymbol{p}_1(t), \boldsymbol{p}_2(t), \cdots, \boldsymbol{p}_{N_s}(t)\right]$ denote the t-th generation colony. During in itialization, the i-th individual can be coded as follows:

$$\boldsymbol{p}_i(0)=\left[\left|\boldsymbol{\varphi}_{i1}(0)\right\rangle, \left|\boldsymbol{\varphi}_{i2}(0)\right\rangle, \cdots, \left|\boldsymbol{\varphi}_{iD}(0)\right\rangle\right], \tag{9}$$

where $i=1, 2, \cdots, N_s$.

3.4. The Projection Measurement of Qubits

On the basis of the principle of quantum computing, by applying Pauli matrices to $\left|\boldsymbol{\varphi}\right\rangle$, we can get the Bloch coordinates of $\left|\boldsymbol{\varphi}\right\rangle$. The qubits projection measurement is denoted by the following equations:

$$x=\left\langle\boldsymbol{\varphi}\middle|\boldsymbol{\sigma}_x\middle|\boldsymbol{\varphi}\right\rangle=\left\langle\boldsymbol{\varphi}\middle|\begin{bmatrix}0 & 1\\ 1 & 0\end{bmatrix}\middle|\boldsymbol{\varphi}\right\rangle, \tag{10}$$

$$y=\left\langle\boldsymbol{\varphi}\middle|\boldsymbol{\sigma}_y\middle|\boldsymbol{\varphi}\right\rangle=\left\langle\boldsymbol{\varphi}\middle|\begin{bmatrix}0 & -\mathrm{i}\\ \mathrm{i} & 0\end{bmatrix}\middle|\boldsymbol{\varphi}\right\rangle, \tag{11}$$

$$z=\left\langle\boldsymbol{\varphi}\middle|\boldsymbol{\sigma}_z\middle|\boldsymbol{\varphi}\right\rangle=\left\langle\boldsymbol{\varphi}\middle|\begin{bmatrix}1 & 0\\ 0 & -1\end{bmatrix}\middle|\boldsymbol{\varphi}\right\rangle, \tag{12}$$

where $i=1, 2, \cdots, N_s$ and $j=1, 2, \cdots, D$.

In QIBC, each qubit are regarded as three paratactic genes, each one contains three paratactic gene chains, and each of gene chains represents an optimization solution. Therefore, each entity represents three optimization solutions at the same time.

3.5. The Solution Space Transformation

In QIBC, three optimization solutions given by each entity can be expressed by Bloch coordinates. Because the value of coordinate is in (−1, 1), we must map it to solutions to the problems. Assuming the j-th variable is $X_j\in\left[\mathrm{Min}_j, \mathrm{Max}_j\right]$, the solution transformation can be expressed as the following three equations:

$$X_{ij}=\left[\mathrm{Max}_j\left(1-x_{ij}\right)+\mathrm{Min}_j\left(1+x_{ij}\right)\right]/2, \tag{13}$$

$$Y_{ij}=\left[\mathrm{Max}_j\left(1-y_{ij}\right)+\mathrm{Min}_j\left(1+y_{ij}\right)\right]/2, \tag{14}$$

$$Z_{ij} = \left[\mathrm{Max}_j\left(1 - z_{ij}\right) + \mathrm{Min}_j\left(1 + z_{ij}\right)\right] / 2, \tag{15}$$

where $i = 1, 2, \cdots, N_s$ and $j = 1, 2, \cdots, D$.

3.6. Employed Bee Search

Taking minimum optimization as example, according to the value of target function, we rank the optimized population in descending order, and select the first N_e entities to compose employed bee colony. The rest of them compose onlooker colony. For the *i*-th employed bee $\boldsymbol{p}_i(t)$, first of all, randomly select employed bees $\boldsymbol{p}_j(t)$ and $\boldsymbol{p}_k(t)$ $(i \neq j \neq k)$, and also randomly select a dimension *d*. Then calculate the rotation axis $\boldsymbol{R}_{axis}(i, j) = \left|\varphi_{id}(t)\right\rangle \otimes \left|\varphi_{jd}(t)\right\rangle$ and the rotation angle $\delta_{ik}(t)$ between $\left|\varphi_{id}(t)\right\rangle$ and $\left|\varphi_{kd}(t)\right\rangle$. Taking $\left|\varphi_{kd}(t)\right\rangle$ as target, rotate $\left|\varphi_{id}(t)\right\rangle$ about $\boldsymbol{R}_{axis}(i, j)$ through $\delta_{ik}(t)$. Let $\hat{\boldsymbol{p}}_i(t)$ denote the employed bee after rotation. By greedy selection operator, we select the better ones between $\boldsymbol{p}_i(t)$ and $\hat{\boldsymbol{p}}_i(t)$ for the next generation from the following equations:

$$\boldsymbol{p}_i(t) = \begin{cases} \hat{\boldsymbol{p}}_i(t), & f\left(\hat{\boldsymbol{p}}_i(t)\right) < f\left(\boldsymbol{p}_i(t)\right) \\ \boldsymbol{p}_i(t), & f\left(\hat{\boldsymbol{p}}_i(t)\right) \geq f\left(\boldsymbol{p}_i(t)\right) \end{cases}, \tag{16}$$

where,

$$f\left(\boldsymbol{p}_i(t)\right) = \max\left(f\left(\boldsymbol{X}_i(t)\right), f\left(\boldsymbol{Y}_i(t)\right), f\left(\boldsymbol{Z}_i(t)\right)\right), \tag{17}$$

$$f\left(\hat{\boldsymbol{p}}_i(t)\right) = \max\left(f\left(\hat{\boldsymbol{X}}_i(t)\right), f\left(\hat{\boldsymbol{Y}}_i(t)\right), f\left(\hat{\boldsymbol{Z}}_i(t)\right)\right). \tag{18}$$

3.7. Onlooker Bee Search

It is similar with employed bee search. First, we calculate selective probability of each employed bee with the equation as follows:

$$P\left(\boldsymbol{p} = \boldsymbol{p}_i\right) = f\left(\boldsymbol{p}_i\right) / \sum_{m=1}^{N_e} f\left(\boldsymbol{p}_m\right). \tag{19}$$

To each onlooker bee, first, we select employed bee $\boldsymbol{p}_i$ according to the roulette method, and in its neighborhood we search for new position $\hat{\boldsymbol{p}}_i$ in the same method as the employed bee search. If $\hat{\boldsymbol{p}}_i$ is better than $\boldsymbol{p}_i$, set $\boldsymbol{p}_i = \hat{\boldsymbol{p}}_i$, and set the number of searches $Bas = 0$. If Bas is less than threshold $Limit$, set $Bas = Bas + 1$, otherwise, we initialize $\boldsymbol{p}_i$ and set $Bas = 0$.

3.8. Algorithm Termination Criterion

To this algorithm, the termination criterion is the number of iterations. Whether it meets the pre-set accuracy or not, the algorithm will stop when the limited number of iterations reaches.

4. Analysis of Experimental Results

Taking functions extremum optimization as example, by comparing with bee colony [1], Bloch quantum genetic algorithm [12], elite genetic algorithm, quantum delta potential-based particle swarm optimization [16], we verify the superiority of QIBC. All of algorithms use Matlab R2009a to implement on the computer (P-II 2.0 GHz) with 1.0 G memory.

4.1. Benchmark Functions

To verify the superiority of QIBC, the following ten Benchmark functions are used in this experiment. All of ten functions are for the minimum optimization, and $\boldsymbol{X}^*$ is the minimum extreme value point.

(1) $f_1(\boldsymbol{X}) = \sum_{i=1}^{D}\left(\sum_{j=1}^{i} x_j\right)^2$, $-100 \leq x_i \leq 100$, $x_i^* = 0$, $f\left(\boldsymbol{X}^*\right) = 0$;

(2) $f_2(\boldsymbol{X})=\sum_{i=1}^{D-1}\left(100\left(x_{i+1}-x_i^2\right)^2+\left(x_i-1\right)^2\right)$, $-100\le x_i\le 100$, $x_i^*=1$, $f\left(\boldsymbol{X}^*\right)=0$;

(3) $f_3(\boldsymbol{X})=\sum_{i=1}^{D} i*x_i^4\left(1+rand\left(0,1\right)\right)$, $-100\le x_i\le 100$, $x_i^*=0$, $f\left(\boldsymbol{X}^*\right)=0$;

(4) $f_4(\boldsymbol{X})=-20\mathrm{e}^{-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}}-\mathrm{e}^{\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)}+20+\mathrm{e}$, $-100\le x_i\le 100$, $x_i^*=0$, $f\left(\boldsymbol{X}^*\right)=0$;

(5) $f_5(\boldsymbol{X})=\frac{D(D+4)(D-1)}{6}+\sum_{i=1}^{D}\left(x_i-1\right)^2-\sum_{i=2}^{D}x_i x_{i-1}$, $-D^2\le x_i\le D^2$, $x_i^*=i(D-i+1)$, $f\left(\boldsymbol{X}^*\right)=0$;

(6) $f_6(\boldsymbol{X})=\sum_{k=1}^{D}\sum_{j=1}^{D}\left(\frac{y_{jk}^2}{4000}-\cos\left(y_{jk}\right)+1\right)$, $y_{jk}=100\left(x_k-x_j^2\right)^2+\left(1-x_j\right)^2$, $-100\le x_i\le 100$, $x_i^*=1$, $f\left(\boldsymbol{X}^*\right)=0$;

(7) $f_7(\boldsymbol{X})=g\left(x_1,x_2\right)+\cdots+g\left(x_{D-1},x_D\right)+g\left(x_D,x_1\right)$, $g(x,y)=\left(x^2+y^2\right)^{0.25}\left[\sin^2\left(50\left(x^2+y^2\right)^{0.1}\right)+1\right]$, $-100\le x_i\le 100$, $x_i^*=0$, $f\left(\boldsymbol{X}^*\right)=0$;

(8) $f_8(\boldsymbol{X})=\sum_{i=1}^{D}x_i^2+\left(\sum_{i=1}^{D}0.5ix_i\right)^2+\left(\sum_{i=1}^{D}0.5ix_i\right)^4$, $-100\le x_i\le 100$, $x_i^*=0$, $f\left(\boldsymbol{X}^*\right)=0$;

(9) $f_9(X)=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$, $-500\le x_i\le 500$, $x_i=0$, $f\left(X^*\right)=0$;

(10) $f_{10}(X)=10D+\sum_{i=1}^{D}\left(y_i^2-10\cos\left(2\pi y_i\right)\right)$, $y_i=\begin{cases}x_i, |x_i|<1/2\\ round\left(2x_i\right)/2, |x_i|\ge 1/2\end{cases}$, $-100\le x_i\le 100$, $x_i=0$, $f\left(X^*\right)=0$.

4.2. Parameters Setting

For convenient comparison, based on the complexity of benchmark functions, we set up precision thresholds ε for each function. Only when optimization effect is less than the thresholds, the algorithm is call converges. In this experiment, the precision thresholds are set as follows. for f_1, f_6, $\varepsilon=1.0$; for f_2 $\varepsilon=10$; for f_3, $\varepsilon=10^{-5}$; for f_4, f_9, f_{10}, $\varepsilon=10^{-10}$; for f_5, $\varepsilon=10^2$; for f_7, $\varepsilon=10^{-8}$; for f_8, $\varepsilon=10^4$. If the algorithm converges, the current optimization steps are called the number of iteration. If not, the number of iteration is equal to the pre-set limited number of iterations.

The dimensions of ten functions are set to $D=30$, and population sizes of five algorithms are equal to 40. For QIBC and BC, we have $N_e=20$, $N_u=20$, $Limit=50$. For BQGA, on the basis of Ref. [12], the initial value of rotation angle is 0.05π, and the mutation probability is 10^{-3}. For QDPSO, based on Ref. [16], the control parameter is $\lambda=1.2$. For EGA, the crossover probability is 0.8, the mutation probability is 0.01. The limited number of iterations of five algorithms is $G=10^4$.

4.3. Comparison and Analysis of Simulation

To enhance the objectivity of comparisons, each algorithm independently runs 50 times, and we have comparisons of statistical results. The average time for each of iteration are shown in **Table 1**. To make comparisons be sufficient, besides showing the average results, the best and the worst ones are also are given. Comparison of the number of iterations, average iterations, and optimization results are shown in **Table 2**.

From **Table 1**, for the running time, we rank five algorithms in descending order as QIBC, BQGA, BC, EGA, QDPSO. The reason is that, in QIBC and BQGA, we use coding mechanism of qubit with three chains. During each iteration, each entity respectively do updating, solution to space transformation, calculating value of benchmark functions etc. and these operations take a long time. QIBC is longer than BQGA because QIBC needs the projection measurement, design of rotation axis and rotation matrices, and its calculating quantity is

Table 1. Comparison of average time for each iteration (unit: second).

No.	QIBC	BC	BQGA	QDPSO	EGA
f1	0.0281	0.0038	0.0216	0.0016	0.0026
f2	0.0219	0.0016	0.0095	0.0005	0.0015
f3	0.0223	0.0018	0.0100	0.0006	0.0015
f4	0.0218	0.0016	0.0103	0.0005	0.0014
f5	0.0221	0.0017	0.0098	0.0006	0.0014
f6	0.0596	0.0102	0.0429	0.0040	0.0126
f7	0.0245	0.0023	0.0270	0.0010	0.0011
f8	0.0222	0.0018	0.0104	0.0006	0.0015
f9	0.0176	0.0046	0.0116	0.0005	0.0003
f10	0.0183	0.0050	0.0178	0.0007	0.0005

Table 2. Comparison of optimization results in five algorithms.

No.	Algorithm	Convergence Number	Average Iteration	Average Results	The Worst Result	The Best Results
f1	QIBC	50	3083	0.1577	0.5737	0.0313
	BC	0	10,000	349.70	1075.1	29.904
	BQGA	0	10,000	131.59	246.14	36.583
	QDPSO	21	7190	17.893	414.60	2.22e−6
	EGA	0	10,000	456.36	887.14	227.14
f2	QIBC	37	4643	9.4221	80.358	0.0444
	BC	23	6250	52.786	914.20	0.0699
	BQGA	0	10,000	1046	6992	138.01
	QDPSO	32	5076	12.440	72.776	0.0024
	EGA	0	10,000	2654.1	12,494	301.56
f3	QIBC	48	5050	1.18e−6	2.10e−5	4.74e−10
	BC	45	8016	1.36e−6	3.32e−5	9.61e−10
	BQGA	0	10,000	187.72	792.30	21.370
	QDPSO	47	6471	2.17e−4	0.0107	5.2e−75
	EGA	0	10,000	1504.8	8057.3	219.92
f4	QIBC	47	5685	3.53e−11	6.29e−10	4.17e−14
	BC	35	7210	7.84e−10	1.85e−08	5.59e−14
	BQGA	0	10,000	15.894	20.742	10.280
	QDPSO	6	9730	16.377	20.005	3.81e−14
	EGA	0	10,000	19.287	21.047	3.9204
f5	QIBC	41	3561	68.415	398.62	2.4709
	BC	10	9028	245.01	663.55	488.40
	BQGA	0	10,000	5427	12,981	435.03
	QDPSO	6	9061	827.33	2992.8	3.2900
	EGA	0	10,000	6581.4	9429.7	4487.2

Continued

	QIBC	49	2467	0.1926	8.9054	1.25e−09
	BC	28	5838	17.966	106.97	3.47e−08
f6	BQGA	0	10,000	491.21	829.02	196.59
	QDPSO	0	10,000	171.47	351.05	8.8842
	EGA	0	10,000	22,127	355,958	978.91
	QIBC	49	5763	3.39e−09	1.69e−07	0
	BC	36	8354	3.67e−08	1.45e−6	1.49e−10
f7	BQGA	0	10,000	110.73	169.67	32.058
	QDPSO	32	8640	0.4312	9.2145	2.42e−19
	EGA	0	10,000	164.55	231.46	93.272
	QIBC	50	5454	1412	5700	61.335
	BC	0	10,000	50,112	69,884	33,193
f8	BQGA	0	10,000	4910.5	12,882	1006.1
	QDPSO	50	7392	2003	5300	200.81
	EGA	0	10,000	7843.8	38,137	4291.6
	QIBC	50	2162	8.65e−17	3.33e−15	0
	BC	36	3424	0.0033	0.0221	0
f9	BQGA	12	7780	0.0239	0.1275	0
	QDPSO	29	5040	0.0062	0.5136	0
	EGA	0	10,000	3.1224	7.3366	1.6101
	QIBC	50	2606	1.13e−14	1.13e−13	0
	BC	49	6290	4.03e−12	1.89e−10	0
f10	BQGA	0	10,000	172.42	416.64	26.023
	QDPSO	6	9284	7.3553	15.882	0
	EGA	0	10,000	822.60	1298.1	455.30

larger.

In **Table 2**, for the ability of optimization, obviously, QIBC is better than QIBC, BQGA is better than EGA. QDPSO is worse than QIBC, but it's better than BQGA, and it's similar to BC. For the optimization performance of five algorithms, we rank them in descending order as QIBC, BC, QDPSO, BQGA, EGA.

For the above results, we present the following analysis. Firstly, we introduce the coding mechanism of qubit with three chains, and it effectively enhances the ergodicity of algorithm to solution space. Based on the geometric properties of the Bloch sphere, this mechanism can enlarge the number of global optimal solutions and the probability of attaining global optimal solutions. That is the reason why these two algorithms are better than their classical ones respectively. Secondly, QIBC has a better performance of optimization than BQGA, because they adjust qubit in the different ways.

In BQGA, qubits are updated by directly adjusting θ and φ with the same adjustment amount (namely $\Delta\theta = \Delta\varphi$). Obviously, in this method, it is hard to close to target qubits along the shortest path, because there must be some matching relation between $\Delta\theta$ and $\Delta\varphi$ when it's along the shortest path. But this matching relation is hard to be expressed clearly by analytic equations. In QIBC, we adjust qubit in indirect method. Namely, we rotate qubit to target qubit about the rotation axis on the Bloch sphere, and obviously, this path is

Table 3. Comparison of optimization results in QIBC and BC.

Function	Algorithm	Convergence Number	Average Number	Average Results	The Worst Result	The Best Result
f9	QIBC	10	6267	1.11e−16	6.66e−16	0
	BC	0	10,000	1.23e−07	6.80e−07	1.5e−08
f10	QIBC	7	9065	4.14e−08	4.02e−07	3.4e−13
	BC	2	9758	3.17e−07	1.90e−06	2.4e−11

the shortest. Although it does't adjust two parameters of qubit directly, it achieves the best match between $\Delta\theta$ and $\Delta\varphi$ automatically and accurately. Hence, this method has better optimizing efficiency than classical ones.

It's also worth pointing out that, although the QIBC has better optimizing performance, the complexity of this algorithm is obviously higher than classical ones. Comparing with classical ones, the QIBC needs some extra operations, such as calculating rotation axis, rotation angel, rotation matrices and the solution to space transformation. Considering run time and optimizing performance, QIBC earns better optimizing efficiency by sacrificing running time, which is the same as No Free Lunch. For many off-line optimizing tasks which may ignore running time, the QIBC has wide application prospects.

We need investigate and study the influence after dimensions and parameters change. Taking f_9 and f_{10} as example, while $D=100$, population size is $N_e=N_u=40$, and $Limit=100$, $G=10^4$. Precision threshold of f_9 and f_{10} is 10^{-10}. The QIBC and the BC run 10 times respectively. Optimization results is shown in **Table 3**.

As the results are shown, for the high-dimensional optimization problems, the proposed algorithm also shows its better performance than classical ones.

5. Conclusion

We present a quantum-inspired bee colony optimization algorithm. In proposed method, the bees are encoded with the qubits described on the Bloch sphere. The population evolutionary is achieved by rotating the qubit about the rotation axis on the Bloch sphere. The experimental results show that, the method of qubits coding on the Bloch sphere and the method of bee updating which can achieve the best match between two adjustment quantities of qubit parameters by rotating about the rotation axis, can truly enhances the performance of the intelligent optimization algorithm.

Acknowledgements

This work was supported by the Natural Science Foundation of Heilongjiang Province, China (Grant No. F2015021), the Youth Foundation of Northeast Petroleum University (Grant No. 2013NQ119) and the Science Technology Research Project of Heilongjiang Educational Committee of China (Grant No. 12541059).

References

[1] Karaboga, D. (2005) An Idea Based on Honey Bee Swarm for Numerical Optimization. Technical Report-TR06, Engineering Faculty, Computer Engineering Department, Erciyes University, Kayseri.

[2] Bahriye, A. and Dervis, K. (2012) A Modified Artificial Bee Colony Algorithm for Real-Parameter Optimization. *Information Sciences*, **192**, 120-142. http://dx.doi.org/10.1016/j.ins.2010.07.015

[3] Xiang, W.L. and An, M.Q. (2013) An Efficient and Robust Artificial Bee Colony Algorithm for Numerical Optimization. *Computers & Operations Research*, **40**, 1256-1265. http://dx.doi.org/10.1016/j.cor.2012.12.006

[4] Li, G.Q., Niu, P.F. and Xiao, X.J. (2012) Development and Investigation of Efficient Artificial Bee Colony Algorithm for Numerical Function Optimization. *Applied Soft Computing*, **12**, 320-332. http://dx.doi.org/10.1016/j.asoc.2011.08.040

[5] Karaboga, D., Akay, B. and Ozturk, C. (2007) Artificial Bee Colony (ABC) Optimization Algorithm for Training Feed-Forward Neural Networks. *Modeling Decisions for Artificial Intelligence*, **4617**, 318-329. http://dx.doi.org/10.1007/978-3-540-73729-2_30

[6] Karaboga, N. (2009) A New Design Method Based on Artificial Bee Colony Algorithm for Digital IIR Filter. *Journal of the Franklin Institute*, **346**, 328-348. http://dx.doi.org/10.1016/j.jfranklin.2008.11.003

[7] Rao, R., Narasimham, S. and Ramalingaraju, M. (2008) Optimization of Distribution Network Configuration for Loss Reduction Using Artificial Bee Colony Algorithm. *International Journal of Electrical Power and Energy Systems Engineering*, **1**, 709-715.

[8] Singh, A. (2009) An Artificial Bee Colony Algorithm for the Leaf-Constrained Minimum Spanning Tree Problem. *Applied Soft Computing*, **9**2, 625-631. http://dx.doi.org/10.1016/j.asoc.2008.09.001

[9] Ding, H.J. and Li, F.L. (2008) Bee Colony Algorithm for TSP Problem and Parameter Improvement. *China Science and Technology Information*, **25**, 241-243.

[10] Kang, F., Li, J.J. and Zu, Q. (2009) Improved Artificial Bee Colony Algorithm and Its Application in Back Analysis. *Water Resources and Power*, **27**, 126-129.

[11] Duan, H.B., Xu, C.F. and Xing, Z.H. (2010) A Hybrid Artificial Bee Colony Optimization and Quantum Evolutionary Algorithm for Continuous Optimization Problems. *International Journal of Neural Systems*, **20**, 39-50. http://dx.doi.org/10.1142/S012906571000222X

[12] Li, P.C. (2008) Quantum Genetic Algorithm Based on Bloch Coordinates of Qubits and Its Application. *Control Theory & Applications*, **25**, 985-989.

[13] Li, P.C. and Lin, J.J. (2012) Chaos Quantum Immune Algorithm Based on Bloch Sphere. *Systems Engineering and Electronics*, **34**, 2592-2598.

[14] Li, P.C., Wang, Q.C. and Shi, G.Y. (2013) Quantum Particle Swarm Optimization Algorithm Based on Bloch Spherical Search. *Chinese Journal of Computational Physics*, **30**, 454-462.

[15] Giuliano, B., Giulio, C. and Giuliano, S. (2004) Principles of Quantum Computation and Information (Vol. I: Basic Concepts). World Scientific, Singapore, 100-112.

[16] Li, P.C., Wang, H.Y. and Song, K.P. (2012) Research on Improvement of Quantum Potential Well-Based Particle Swarm Optimization Algorithm. *Acta Physica Sinica*, **61**, Article ID: 060302.