International Journal of Computational Intelligence and Applications

A COMPARISON BETWEEN QUANTUM INSPIRED BACTERIAL FORAGING ALGORITHM AND GA-LIKE ALGORITHM FOR GLOBAL OPTIMIZATION

Sha-Rina Huang*, Guoliang Zhao

School of Science, Heilongjiang Institute of science and technology, Harbin 150027, People's Republic of China†

Received (received date)
Revised (revised date)

Bacterial foraging algorithm(BFA) is a population-based stochastic search technique for solving various scientific and engineering problems. However, it is inefficient in some practical situations, in order to improve the performance of the BFA, we propose a novel optimization algorithm, named quantum inspired bacterial foraging algorithm(QBFA), which applies several quantum computing principles, and a new mechanism is proposed to encode and observe the population. The algorithm has been evaluated on the standard high-dimensional benchmark functions in comparison with GA, PSO, GSO and FBSA, respectively. The proposed algorithm is then used to tune a PID controller of an automatic voltage regulator(AVR) system. Simulation results clearly illustrate that the proposed approach is very efficient and could be easily extended to 300 or higher dimensional problems.

Keywords: Quantum algorithm; bacterial foraging optimization; hybrid optimization; numeric optimization.

1. Introduction

In the last two decades, many bio-inspired computational methodologies have received increased attention from the academic and communities in various domains of industry, such as Particle Swarm Optimization(PSO) [1], Genetic Algorithm(GA) [2]. More recently, a new and rapidly growing subject—Bacterial Foraging Algorithm(BFA), which is inspired by the behavior of bacterial called *E coli* searching for food in human intestines, since its advent in 2002, has attracted many researchers from various domains of knowledge. A few variant of the classical algorithm as well as hybridizations of BFA with other naturally inspired algorithms has been introduced [3,4,5]. BFA has been applied to many kinds of real world optimization problems. Such as optimal power-system stabilizers design [6,7], multi-class wavelet SVM classifiers [8], high-dimensional function optimization [9], optimal power flow

*E-mail:hsrn1982@163.com.
†

[10,11,12], **parametric modelling of flexible manipulator systems**[13]**,** and economic load dispatch [14]. BFA also has prominent and encouraging performance for global optimization problems [3,4,5,15].

Quantum physics and quantum computing principles have also been widely seen as a source of inspiration, quantum algorithm is based on probability to search the best solution randomly, it has the drawback of premature and stagnation in the late evolution stage. Q-bit chromosomes of individuals can overcome the premature phenomenon and maintain the solution diversity, usually, the update mechanism of the quantum individual is called quantum rotation gate, i.e. Q-gate. It should be noted that NOT gate, Controlled NOT gate, Hadamard gate and $H\epsilon$ gate [16] could be used as a Q-gate. For more information, refer to the state-of-the art outline of quantum computation and quantum theory [17]. By injecting the power of quantum parallelism into the well known computational intelligence such as evolutionary algorithms(EAs), swarm intelligence(SI) **and genetic algorithm (GA). Many hybridized algorithms have been introduced. Sun gives an theoretical Convergence analysis of quantum-behaved particle swarm optimization**[18]**. Wang studied the numerical optimization and parameter estimation problems based on quantum computing and genetic algorithm (QGA)**[19]**. Also, the QGA is used to solve economic dispatch problem** [20,21]**, job shop scheduling problem**[22]**. These hybrid algorithms are also used in the category of control engineering, such as parameters optimization of PID controller for automatic voltage regulator(AVR)** [23,24,25]**, neuro-fuzzy controller design**[26] **and power systems commitment problems**[27]**. There are also many other related field, such as, k-means clustering** [28]**, fuzzy clustering design**[29]**, multi-objective flow shop scheduling** [30]**, image segmentation** [31]**, global optimization** [32]**, DNA encoding** [33]**, benignancy/malignancy detection**[34]**, etc.**

In this paper, a quantum-inspired bacterial foraging algorithm is proposed to solve the global optimization problems. The rest of the paper is organized as follows, in section 2, we provide a brief overview of the quantum algorithm and bacterial foraging algorithm followed by the quantum inspired bacterial algorithm(QBFA). In section 3, the proposed algorithm QBFA is validated using 18 benchmark functions [35] and PID controller tuning. At the end of this paper, some conclusions are obtained.

2. Hybrid system of quantum algorithm and bacterial foraging algorithm

2.1. *Quantum computing principles*

1) Quantum Individuals

In digital computers, the smallest information unit is one bit being either in the state '0' or '1' at any given time. The corresponding analogue on a quantum

computer is represented by a quantum bit or Q-bit. Mathematically a qubit is represented by a unit vector in the two-dimensional complex Hilbert space, and can be written in the Dirac notation as follows:

$$|\phi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{1}$$

where $|0\rangle$ and $|1\rangle$ are two basis states, and α and β are complex numbers with $\alpha^2 + \beta^2 = 1$. The state $|0\rangle$ and $|1\rangle$ are called computational basis states of qubits. The number α and β are called probability amplitudes of the state $|\phi\rangle$. An example state of qubit is $|-\rangle = \frac{1}{\sqrt{2}}(|0\rangle + |1\rangle)$.

A ith Q-bit individual with a string of m-qubit is defined as

$$\begin{bmatrix} \alpha_{i1} & \alpha_{i2} & \cdots & \alpha_{im} \\ \beta_{i1} & \beta_{i2} & \cdots & \beta_{im} \end{bmatrix} \tag{2}$$

where $|\alpha_{ij}|^2 + |\beta_{ij}|^2 = 1, j = 1, 2, \cdots, m$, for instance, following three-Q-bit:

$$\begin{bmatrix} \frac{\sqrt{3}}{2} & \frac{1}{3} & \frac{\sqrt{2}}{2} \\ \frac{1}{2} & \frac{2\sqrt{3}}{3} & -\frac{\sqrt{2}}{2} \end{bmatrix} \tag{3}$$

The state can be described as

$$\frac{1}{24}|000\rangle - \frac{1}{24}|001\rangle + \frac{1}{3}|010\rangle - \frac{1}{3}|011\rangle + \frac{1}{72}|100\rangle - \frac{1}{72}|101\rangle + \frac{1}{9}|110\rangle - \frac{1}{9}|111\rangle$$

the above three-Q-bit string includes eight states and every state has a corresponding probability. This Q-bit representation has the advantage that it is able to represent a linear superposition of states probabilistically. The Q-bit representation has a better characteristic of generation diversity then any other representations.

2) Rotation Gate Operation

Variation operators like crossover or mutation operations are used to explore the search space. The Quantum analog for exploring the search space is called quantum gate. Because of the real number encoding mechanism, rotation operation used in this paper is defined as follows:

$$\begin{bmatrix} \alpha_i^{'} \\ \beta_i^{'} \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta_i) & -\sin(\Delta\theta_i) \\ \sin(\Delta\theta_i) & \cos(\Delta\theta_i) \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} = U(\Delta\theta_i) \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} \tag{4}$$

where $[\alpha_i, \beta_i]^T$ denotes the present solution. $U(\Delta\theta_i)$ represents a Q-gate. The $[\alpha_i^{'}, \beta_i^{'}]^T$ denotes the solution after performing rotation gate operation, $\Delta\theta_i$ is a rotation angle determined by the bacterial position x and the best position of the N_b bacterial. The lookup table is shown in Table 1. In this Table, $f(x) < f(b)$ means that the position x is better than the position b. x_i and $best_i$ are the ith bit of the position x and the best position of all the bacteria.

2.2. *Bacterial foraging algorithm*

Inspired by the bacteria's behavior, passino introduced the bacterial inspired computation [5]. This bacterial foraging system consists of four principal mechanisms,

Table 1. Lookup table of rotation angle.

$x_i > best_i$	$x_i = best_i$	$x_i < best_i$	$f(x) > f(best)$	$\Delta\theta_i$
false	false	true	false	-0.01π
false	false	true	true	-0.001π
false	true	false	false	-0.001π
false	true	false	true	0.001π
true	false	false	false	-0.001π
true	false	false	true	0.001π

chemotaxis, swarming, reproduction and elimination . Following is an brief introduction of the Bacterial Foraging Algorithm(BFA).

2.2.1. *Chemotaxis and Swarming*

The chemotaxis process simulate the process of the movement of an *E coli* cell through swimming and tumbling via flagella, during the life time of the bacterial, it can swim for a period of time in the same direction, also it may display another pattern named tumble, and it will alternate these two modes according to the environment change. Let $\theta^i(j,k,l)$ be the ith bacterium at jth chemotactic, kth reproduction and lth elimination dispersal step. $C(i)$ is the step size taken in the random direction specified by the tumble(swim).

If bacteria near the optima or very close to the optima, and the chemotactic steps does not stop, this bacterium will be in the state of oscillation, this crisis can be remedied by following method [4].

$$C = \frac{|J(\theta)|}{|J(\theta)| + \lambda} \tag{5}$$

where λ is a positive constant,$\lambda = 4000$ is a suitable parameter.

The swarming pattern of the cell-to-cell attraction and repellence in BFA has negative effects [11] and verified by Liu [31]. As showed in Liu's paper [31], the repellent effect reduces the precision of the optimization. The cell-to-cell attraction leads to the phenomenon that the bacteria in the local optima attract those in the global optimum and lower the convergence speed. A novel principle of swarming for BFA named FBSA [9] was introduced. It is assumed that the bacteria have the similar ability like birds to follow the best bacterium in the optimization domain. The position of each bacterium after each step (either tumble or run) is updated as following:

$$\theta^i(j+1,k,l) = \theta^i(j,k,l) + C(i)(\theta^b(j,k,l) - \theta^i(j,k,l)), \\ if J^i(j,k,l) > J_{min}(j,k,l) \tag{6}$$

where $\theta^i(j,k,l)$ is the position of the ith bacterium at the jth chemotaxis, kth reproduction and lth elimination and disperse stage. $\theta^b(j,k,l)$ is the best position the bacterium have at the moment. $J^i(j,k,l)$ is the health status of the ith bacterium at the jth chemotaxis, kth reproduction and lth elimination and disperse stage.

2.2.2. *Reproduction*

The least healthy bacteria will die and the healthier bacteria have more opportunity to reproduce themselves, which asexually split into two bacterial cells, they are then placed in the same position as their parents. Meanwhile keep the same value. That is, after N_c chemotactic steps, the fitness value for the ith bacterium in the chemotactic loop are accumulated and calculated by

$$J^i_{health} = \sum_{j=1}^{Nc+1} J^i(j,k,l) \tag{7}$$

where $J^i(j,k,l)$ represents the health of the ith bacterium at the jth chemotaxis, kth reproduction and lth elimination and dispersion stage.

2.2.3. *Elimination and Dispersal*

For the purpose of improving the global search ability, after N_{re} steps of reproduction. Elimination-dispersal event is applied to the algorithm. The bacterium are eliminated and dispersed to random positions in the optimization domain according to the probability P_{ed} and their health status. This will help the bacterium avoid being trapped into local optimum.

2.3. *Procedure of the Quantum Inspired Bacterial Foraging Algorithm*

The main goal of quantum inspired bacterial foraging algorithm is to find the minimum of the function $f(x), x \in \Re^n$. As to the bacterial foraging algorithm (BFA), x is characterized as the position of a bacterium, which is denoted by $\theta\,(\theta \in \Re^n)$ in the quantum bacterial foraging algorithm. In our method, initial population denoted as follows:

$$P_Q(t) = \{P_{Q_1}(t), P_{Q_2}(t), \cdots, P_{Q_{Nb}}(t)\}^T \tag{8}$$

where $P_Q(t)$ has $2N_b \times N$ Q-bit and $P_{Q_i}(t)$ is defined as follows:

$$P_{Q_i}(t) = \begin{bmatrix} \cos(\theta_{i1}) & \cos(\theta_{i2}) & \cdots & \cos(\theta_{iN}) \\ \sin(\theta_{i1}) & \sin(\theta_{i2}) & \cdots & \sin(\theta_{iN}) \end{bmatrix} \tag{9}$$

where $\theta_{ij} = \frac{\pi(i-1)}{2(N_b-1)}, i = 1, 2, \cdots, N_b; j = 1, 2, \cdots, N$. N_b is the number of bacteria and N is the dimension of the search space. Unlike the usual technique used in QA which are initialized with $\theta_{ij} = \frac{\pi}{4}$, i.e., α^0_{ij} and β^0_{ij} is $\frac{1}{\sqrt{2}}$. The newly proposed technique placed the N_b individuals equally distributed in the search space.

2.3.1. *Convert $P_Q(t)$ to the position $P_P(t)$*

As mentioned above, the position of each bacterium is real number, denoted by x, however, QA is encoded as a Q-bit string. Traditionally, the most used observational

manner of QA is to convert a Q-bit string to binary string. Which is not suitable for global optimization. In this paper, A new observational manner is proposed, which can be described as follows:

$$P_P(t) = \{P_{P_1}(t), P_{P_2}(t), \cdots, P_{P_{Nb}}(t)\}^T$$

$$P_{P_i}(t) = \left[\cos^2(t_{i1}), \cos^2(t_{i2}), \cdots, \cos^2(t_{iN})\right](UB - LB) + LB \tag{10}$$

Where UB denote the upper bound of the vector x, LB is the lower bound. For example, for the *kowalik's function*, $-5 \leq x_i \leq 5\,(i = 1, 2, 3, 4)$. That is , $LB = -5$, $UB = 5$, and $N = 4$. The ith bacterium with Q-bit string is described as follows:

$$\begin{aligned} P_{Q_i}(t) &= \begin{bmatrix} \cos(t_{i1}) & \cos(t_{i2}) & \cos(t_{i3}) & \cos(t_{i4}) \\ \sin(t_{i1}) & \sin(t_{i2}) & \sin(t_{i3}) & \sin(t_{i4}) \end{bmatrix} \\ &= \begin{bmatrix} 0.7071 & 0.5 & 0.2588 & 0 \\ 0.7071 & 0.8660 & 0.9659 & 1 \end{bmatrix} \end{aligned} \tag{11}$$

This Q-bit string can be converted to real number. **Then the** position of the bacterium can be calculated as follows:

$$\begin{aligned} P_{P_i}(t) &= [0.7071^2 \times 10 - 5, 0.5^2 \times 10 - 5, 0.2588^2 \times 10 - 5, 0 \times 10 - 5] \\ &= [0, -2.5, -4.3301, -5] \end{aligned}$$

This converting method guarantees each position of bacterium is feasible and can be used to evaluate the fitness of the bacterium easily.

2.3.2. *The Quantum-inspired Bacterial Foraging Algorithm*

The algorithm to search optimal value of parameters is described as follows:

[step 1] Initialize parameters $N_b, N, \theta_{i\cdot}, P_Q(t), N_c, N_s, N_{re}, N_{ed}, P_{ed}, IterMax$, $C(i)(i = 1, 2, \cdots, N_b)$. Where

N_b : Total number of bacteria in the population.

N : Dimension of the search space.

$\theta_{i\cdot}$: Angle specialized in the (9).

$P_Q(t)$: In the step of initialize $P_Q(t)$, i.e., α_{ij}^0 and $\beta_{ij}^0, (i = 0, 1, \cdots, N_b,\ j = 0, 1, \cdots, N)$ are initialized by (9).

N_c : Number of chemotactic steps.

N_s : Swimming length.

N_{re} : The number of reproduction steps.

N_{ed} : The number of elimination-dispersal steps.

P_{ed} : Elimination-dispersal events.

$C(i)$: The size of the step taken in the random specified by the tumble.

$IterMax$: The maximum iteration steps algorithm can perform.

[step 2] Make a observation of the quantum states of $P_Q(t)$, obtained the position of the bacteria $P_P(t)$ and save the best position P_{best} and the best fitness J_{best}.

[step 3] Elimination-dispersal loop: For $l = 1, 2, \cdots, N_{ed}$.

[step 4] Reproduction loop: For $k = 1, 2, \cdots, N_{re}$.

[step 5] Chemotaxis loop: For $j = 1, 2, \cdots, N_c$.

(a) For $i = 1, 2, \cdots, N_b$, update the quantum gate for ith bacterium. In this paper, Q-bit individuals in $P_Q(t)$ are updated by (4).

(b) Compute fitness of the ith bacterium $J(i, j, k, l)$.

(c) Let $J_{last} = J^i(j, k, l)$ to save this value, since we may find a better position via a run.

(d) Tumble: Let

$$\theta^i(j + 1, k, l) = \theta^i(j, k, l) + C(i)(\theta^b(j, k, l) - \theta^i(j, k, l)),$$

this result in a step size specified by $C(i)$ in the tumble direction for ith bacterium.

(e) Swim:

i) set $m = 0$ (counter for swim steps) and compute $J(i, j + 1, k, l)$.

ii) while $m < N_s$ (if have not climbed down enough steps).

iii) $m = m + 1$.

iv) if $J^i(j + 1, k, l) < J_{last}$ (if doing better), let $J_{last} = J^i(j + 1, k, l)$ and update $\theta^i(j + 1, k, l)$ by Table 1.

v) else, let $m = N_s$, if $J_{last} < J_{best}$, update the J_{best} and P_{best}, end the while loop.

(f) Go to next bacterium $i = i + 1$, if $i < N_b$, go to step (b) .

[step 6] If $j < N_c$, go to step 5. Since the life time of bacterial is not over.

[step 7] Reproduction:

(a) For the given k and l, and for the ith bacterium ($i = 1, 2, \cdots, N_b$), compute the health status by (7). Let J^i_{health} be the health status of the ith bacterium. Sort bacteria and chemotactic parameters $C(i)$ in order of ascending health status J_{health} (higher cost means lower health).

(b) The S_r bacteria with the best fitness values survive and split.

[step 8] If $k < N_{re}$, go to step 3. In this case, we have not reached the number of specified reproduction steps $IterMax$, so we start the next generation of the chemotactic loop.

[step 9] Elimination-dispersal: For $i = 1, 2, \cdots, N_b$ with probability P_{ed}, eliminate and disperse the bacteria (this keeps the number of bacteria in the same size). To do this, if a bacterium is eliminated, simply generate a random location on the optimization domain, if $l < N_{ed}$, go to step 2, otherwise end the algorithm.

3. Experiment results

3.1. *Benchmark functions for the quantum inspired bacterial foraging algorithm*

This section illustrates some comparisons between the proposed QBFA(Quantum inspired Bacterial foraging algorithm), GA, PSO, GSO and FBSA. The tested benchmark functions [35] are depicted in Table 2.

Table 2. The tested benchmark functions($f_1 - f_{18}$) for comparison.

Objective function	n	S	f_{min}
$f_1(x) = \sum_{i=1}^{30} x_i^2$	30	$[-100, 100]^n$	0
$f_2(x) = \sum_{i=1}^{30} \|x_i\| + \prod_{i=1}^{30} \|x_i\|$	30	$[-10, 10]^n$	0
$f_3(x) = \sum_{i=1}^{30} (\sum_{j=1}^{i} x_j)^2$	30	$[-100, 100]^n$	0
$f_4(x) = \max\{\|x_i\|, 1 \le i \le 30\}$	30	$[-100, 100]^n$	0
$f_5(x) = \sum_{i=1}^{29} [100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$	30	$[-100, 100]^n$	0
$f_6(x) = \sum_{i=1}^{30} (\lfloor x_i + 0.5 \rfloor)^2$	30	$[-30, 30]^n$	0
$f_7(x) = \sum_{i=1}^{30} i x_i^4 + random([0, 1])$	30	$[-1.28, 1.28]^n$	0
$f_8(x) = -\sum_{i=1}^{30} (x_i \sin(\sqrt{\|x_i\|}))$	30	$[-500, 500]^n$	-12569.5
$f_9(x) = \sum_{i=1}^{30} (x_i^2 - 10\cos(2\pi x_i) + 10)$	30	$[-5.12, 5.12]^n$	0
$f_{10}(x) = -20\exp(-0.2\sqrt{\frac{1}{30}\sum_{i=1}^{30} x_i^2}) +$ $20 + e - \exp(\frac{1}{30}\sum_{i=1}^{30} \cos 2\pi x_i)$	30	$[-32, 32]^n$	0
$f_{11}(x) = \frac{1}{4000}\sum_{i=1}^{30} x_i^2 - \prod_{i=1}^{30} \cos(\frac{x_i}{\sqrt{i}}) + 1$	30	$[-600, 600]^n$	0
$f_{12}(x) = \frac{\pi}{30}\{10\sin^2(\pi y_1) + \sum_{i=1}^{29}(y_i - 1)^2$ $[1 + 10\sin^2(\pi y_{i+1})] + (y_n - 1)^2\}$ $+ \sum_{i=1}^{30} u(x_i, 10, 100, 4)$	30	$[-50, 50]^n$	0
$f_{13}(x) = 0.1\{\sin(3\pi x_1)^2 + \sum_{i=1}^{29}(x_i - 1)^2$ $[1 + 10\sin(3\pi x_{i+1})^2] + (x_n - 1)^2$ $[1 + 10\sin(2\pi x_{30})^2]\} + \sum_{i=1}^{30} u(x_i, 5, 100, 4)$	30	$[-50, 50]^n$	0
$f_{14}(x) = \left[\frac{1}{500} + \sum_{j=1}^{25} \frac{1}{j + (x_i - a_i)^6 + (x_i - b_i)^6}\right]^2$, $a_i = 16((i \bmod 5) - 2)$, $b_i = 16(\lfloor (i/5) \rfloor - 2)$	30	$[-65.536, 65.536]^n$	0.998
$f_{15}(x) = \sum_{i=1}^{11} \left[a_i - \frac{x_1(b_i^2 + b_i x_2)}{b_i^2 + b_i x_3 + x_4}\right]^2$	4	$[-5, 5]^n$	0.0003075
$f_{16}(x) = 4x_1^2 - 2.1x_1^4 + \frac{x_1^6}{3} + x_1x_2 - 4x_2^2 + 4x_2^4$	2	$[-5, 5]^n$	-1.0316285
$f_{17}(x) = (x_2 - \frac{5.1x_1^2}{4\pi^2} + \frac{5x_1}{\pi} - 6)^2$ $+10(1 - \frac{1}{8\pi})\cos x_1 + 10$	2	$[-5, 10] \times [0, 15]$	0.398
$f_{18}(x) = [1 + (x_1 + x_2 + 1)^2(19 - 14x_1 + 2x_1^2$ $-14x_2 + 6x_1x_2 + 3x_2^2)] \times [30 + (2x_1$ $-3x_2)^2 \times (18 - 32x_1 + 12x_1^2 + 48x_2$ $-36x_1x_2 + 27x_2^2)]$	2	$[-2, 2]^n$	3

3.2. *Parameter Settings*

In the simulation studies, QBFA is evaluated on the 18 benchmark functions in comparison with GA, PSO, GSO and FBSA. The parameter settings of the QBFA is summarized as follows. The initial population of QBFA is generated by (9). The number of bacteria is 50. The common probability of elimination-dispersal event is $P_{ed} = 0.25$. $N_{cs} = 50, N_s = 50, N_{re} = 50, N_{ed} = 50$. **The existed results of GA, PSO GSO and FBSA are directly adopted [40,35] to give a comparison with QBFA**. The maximal function evaluations are depicted in Table 3. Also, a set of two tailed t-test were adopted. Based on the central limit theorem, one may note that as the sample size larger than 50, the sampling distribution of the mean approaches a normal distribution regardless of the distribution of the original population. the t-test[40] assumed the variances of the two populations to be equal, it is inappropriate for different algorithms. Although the original t-test with roughly

equal sample sizes is highly robust to the presence of unequal variance. It would be better to use Welch's t-test [36], which defines the statistic t by the following formula:

$$t = \frac{\overline{X}_1 - \overline{X}_2}{\sqrt{\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}}}$$

where $\overline{X}_i, s_i^2$ and N_i are the ith sample mean, sample variance and sample size, respectively. Unlike in Student's t-test, the denominator is not based on a pooled variance estimate.

The degrees of freedom associated with this variance estimate is approximated using the Welch-Satterthwaite equation:

$$\nu = \frac{\left(\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}\right)^2}{\frac{s_1^4}{N_1^2 \cdot \nu_1} + \frac{s_2^4}{N_2^2 \cdot \nu_2}} = \frac{\left(\frac{s_1^2}{N_1} + \frac{s_2^2}{N_2}\right)^2}{\frac{s_1^4}{N_1^2 \cdot (N_1 - 1)} + \frac{s_2^4}{N_2^2 \cdot (N_2 - 1)}} \tag{12}$$

Here $\nu_i = N_i - 1, (i = 1, 2)$, the degrees of freedom associated with the ith variance estimate.

Table 3. The maximal number of function evaluations of GA, PSO and GSO for $f_1 - f_{18}$.

Function	QBFA	GA	PSO	GSO
$f_1 - f_3, f_5, f_6, f_{11} - f_{12}$	1.50E5	1.50E5	1.50E5	1.50E5
f_4	2.50E5	1.50E5	1.50E5	1.50E5
$f_7 - f_{10}$	1.50E5	2.50E5	2.50E5	2.50E5
f_{13}	1.80E5	1.50E5	1.50E5	1.50E5
f_{14}	1.50E5	7.50E3	7.50E3	7.50E3
f_{15}	4.00E5	2.50E5	2.50E5	2.50E5
f_{16}	2.00E5	1.25E3	1.25E3	1.25E3
f_{17}	1.50E5	5.00E3	5.00E3	5.00E3
f_{18}	2.00E5	1.00E4	1.00E4	1.00E4

3.3. *Unimodal Functions*

Unimodal functions can be solved efficiently by many deterministic algorithms. We tested the QBFA on a set of unimodal functions in comparison with the other four algorithms. Table 4 lists the mean and standard deviations of the function values obtained over 1000 runs, t-test values between these algorithms. The results of GA, PSO, GSO are adopted from He et al [40]. The results of FBSA is adopted from H. Mi[9]. Results from QBFA, GA, PSO and GSO and FBSA are tabulated in Table 4.

In Table 4, we can concluded that QBFA generate significantly better results than GA on all the unimodal functions $f_1 - f_7$. FBSA had better performance on

$f_1 - f_3$. QBFA had significantly better performance on $f_4 - f_7$. In summary, the search ability of these algorithms can be ordered as

$$QBFA \geq FBSA \geq GSO \geq PSO \geq GA$$

The processor is 2.16 GHz Intel Core2 Duo CPU and 2-GB RAM in Windows 7 environment. As shown in Table 4, no algorithm could find out the global minimum

Table 4. Comparison of means and standard deviations for $f_1 - f_7$. All results have been averaged over 1000 runs.

Function	Algorithms	Mean	Std.	t-test
f_1	GA	3.1711	1.6621	$-59.96^{\dagger}$
	PSO	3.6927E-37	2.4598E-36	$665.67^{\dagger}$
	GSO	1.9481E-8	1.1629E-8	$665.67^{\dagger}$
	FBSA	**2.50E-135**	4.22E-136	$665.67^{\dagger}$
	QBFA	1.9672E-02	9.3452E-04	N/A
f_2	GA	0.5771	0.1306	$-9.57^{\dagger}$
	PSO	2.9818E-24	1.1362E-23	$2504.49^{\dagger}$
	GSO	3.7039E-5	8.6185E-5	$2503.24^{\dagger}$
	FBSA	**2.01E-29**	3.31E-31	$2504.49^{\dagger}$
	QBFA	7.4013E-02	1.0199E-02	N/A
f_3	GA	9749.9145	2594.9593	$-118.76^{\dagger}$
	PSO	1.1979E-3	2.1109E-3	$30.13^{\dagger}$
	GSO	5.7829	3.6813	$-8.88^{\dagger}$
	FBSA	**1.49E-4**	2.93E-5	$30.14^{\dagger}$
	QBFA	4.4670E+00	4.6865E+00	N/A
f_4	GA	7.9610	1.5063	$-167.13^{\dagger}$
	PSO	0.4123	0.2500	$-52.15^{\dagger}$
	GSO	0.1078	3.9981E-2	$-85.26^{\dagger}$
	FBSA	6.5E-3	2.3E-4	$-893.69^{\dagger}$
	QBFA	**2.5413E-12**	7.4761E-14	N/A
f_5	GA	338.5616	361.497	$-29.62^{\dagger}$
	PSO	37.3582	32.1436	$-36.75^{\dagger}$
	GSO	48.8359	30.1771	$-51.18^{\dagger}$
	FBSA	3.02E1	1.28	$-746.10^{\dagger}$
	QBFA	**6.6277E-08**	3.7657E-16	N/A
f_6	GA	3.6970	1.9517	$-59.90^{\dagger}$
	PSO	0.1460	0.4182	$-11.04^{\dagger}$
	GSO	1.6000E-2	0.1333	$-3.80^{\dagger}$
	FBSA	0	0	-
	QBFA	**0**	0	N/A
f_7	GA	0.1045	3.6217E-2	$-89.49^{\dagger}$
	PSO	9.9024E-3	3.5380E-2	$-7.12^{\dagger}$
	GSO	7.3773E-2	9.2557E-2	$-24.54^{\dagger}$
	FBSA	5.63E-2	2.72E-2	-0.37
	QBFA	**1.9351E-03**	1.4124E-03	N/A

The value of t with 999 degree of freedom calculated by Eq. (12) is significant at $\alpha = 0.05$ by a two-tailed test.

of Rosenbrock function except QBFA, in Han's paper [16], there is a detailed description of the reason. The proposed algorithm QEA[16] **doesn't** converge to the optimal value in most of the experiment. **The reason is that the Hϵ gate, that prevents each Q-bit converging to the final state (0 or 1). The results of the average Q-bit convergence show that the final values for QEAs converge to $1-2\epsilon$ and 1. The tactics involved in QBFA (new initial strategy and observational mechanism, global search by quantum algorithm, and local search by bacterial foraging algorithm) make QBFA more powerful than other algorithms in terms of convergence speed, fitness, and robustness.**

The function $f_1 - f_7$ are non-linear, separable and scalable, with no shift. And the Generalized Rosenbrock function is difficult in finding the global optimum. In order to illustrate the bacteria's trace in the process of finding the optimum. We use the shifted f_5 $(n = 2)$ as **an example**. i.e., $f_5(x) = 100(z_2 - z_1^2)^2 + (z_1 - 1)^2$, where $z = x - o$. From Fig. 1, the start point selected from $P_P(t)$ is $(2.00294, 3.09096)$, which is the start position of the initial population. After the N_b bacteria performed the tumble and swim steps, the best position is recorded and plotted. As shown in Figure 1, the proposed QBFA has the ability to reach the optimization regardless of the initial angle θ.

We can see that, from the results from Table 4, for the four test functions f_4, f_5, f_6, f_7, QBFA gets the much smaller results than other algorithms. For the average best fitness value, FBSA achieves the best results on f_1, f_2, f_3, and GA has largest STD. If QBFA adopt the strategy like GSO and PSO, we believe QBFA will get much better results.

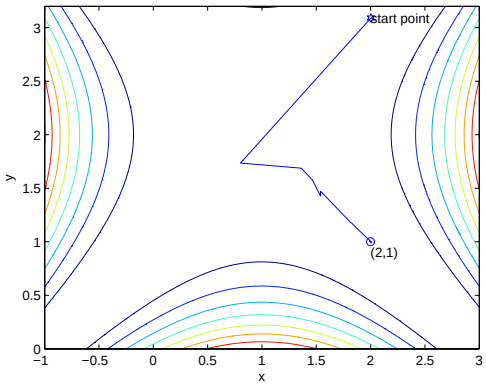

Fig. 1. Path of the shifted function $f_5(x) = 100((x_2 - 2) - (x_1 - 1)^2)^2 + (x_1 - 2)^2$ at every stage, where $z = x - o(o = (1, 2))$.

3.4. *Multi-modal Functions with many local minimal*

This set of benchmark functions($f_8 - f_{13}$) are the most difficult functions to optimize since the number of local minimal increases exponentially as the function dimension increases. The experiment runs 1000 times. We listed the mean, standard deviations, t-test of the benchmark function in the Table 5.

From the Table 5, we can conclude that for the six benchmark functions, GSO markedly outperformed GA and PSO. For the functions f_9, QBFA performs better than the other algorithms. QBFA has a comparable result with functions f_{11} and f_{13}. **Since QBFA observes the population by rotation gate which is a function of θ_i, it is more likely for QBFA to avoid the local minimum based on a small $\Delta\theta_i$, Furthermore, among these five algorithms, QBFA can find the sub-optimal value on every test function.**

In Table 6, we increase the function evaluations. The mean and standard deviations are tabulated. As the function evaluations increase from 1.00E5 to 1.00E6, the mean of 50 runs of function $f_{10} - f_{13}$ approximately reach 0 while the standard deviation approach 0.

3.5. *Multi-modal Functions with a few local minimal*

This set of benchmark functions $f_{14} - f_{18}$ are multi-modal but in low dimension($n \leq 6$) and they have a few local minimal. In Table 7 , we can see that QBFA has better performance than the GA and PSO on most of the functions except the function f_{15}. **The Std of QBFA shows that the the proposed algorithm is more robust than the GA-like algorithm, the employed quantum algorithm guarantee the algorithm reach the global minimum, while the BFA is efficient in finding the local minimum**. Under the function evaluations of 1E5, GSO are statistically generated better average results than other algorithms.

3.6. *300-Dimension Multi-modal Functions*

Many real world optimization problems usually involve hundreds or even thousands of variables. Algorithms such as GA, PSO and EAs have shown excellent search abilities but often lose their efficacy when applied to higher dimensional problems, such as those with more than one hundred decision variables. Many optimization methods suffer from the 'curse of dimensionality' [37,38], their accuracy deteriorates quickly as the dimensionality of the search space increases. The reasons for this phenomenon appear to be two-fold. First, the solution space of a problem often increases exponentially with the problem dimension and more efficient search strategies are needed. Second, the characteristics of a problem may change with the scale. Rosenbrock function f_5 is unimodal for two dimension but becomes multi-modal for higher ones [39]. A previously successful algorithm for low dimension problems may no longer be capable of finding the optimal solution. Due to the hybridization of quantum computing and bacterial foraging algorithm, the proposed QBFA has the

Table 5. Comparison of QBFA with GA, PSO and GSO for $f_8 - f_{13}$. All results have been averaged over 1000 runs.

Function	Algorithms	Mean	Std.	t-test
f_8	GA	-12569.0977	2.1088	1.47
	PSO	-9659.6993	463.7825	$-198.37^{\dagger}$
	GSO	**-12569.4882**	2.2140E-2	$697.30^{\dagger}$
	FBSA	5.53E3	3.02E2	$-1895.17^{\dagger}$
	QBFA	-1.2569.4861	1.9109E-10	N/A
f_9	GA	0.6509	0.3594	$-17.21^{\dagger}$
	PSO	20.7863	5.9400	$-24.68^{\dagger}$
	GSO	1.0179	0.9509	$-7.11^{\dagger}$
	FBSA	3.54E1	9.30	$-26.88^{\dagger}$
	QBFA	**4.2079E-02**	1.9094E-01	N/A
f_{10}	GA	0.8678	0.2805	$-20.50^{\dagger}$
	PSO	1.3404E-3	4.2388E-2	$8.71^{\dagger}$
	GSO	2.6548E-5	3.0820E-5	$52.48^{\dagger}$
	FBSA	**7.99E-15**	0	$52.50^{\dagger}$
	QBFA	5.4310E-02	7.3145E-03	N/A
f_{11}	GA	1.0038	6.7545E-2	$-59.57^{\dagger}$
	PSO	0.2323	0.4434	$-3.13^{\dagger}$
	GSO	3.0792E-2	3.0867E-2	0.06
	FBSA	**0**	0	$2.39^{\dagger}$
	QBFA	3.1675E-02	9.3549E-02	N/A
f_{12}	GA	4.3572E-2	5.0579E-2	$-6.09^{\dagger}$
	PSO	3.9505E-5	9.1424E-2	0.00
	GSO	2.7648E-11	9.1674E-11	$500.04^{\dagger}$
	FBSA	**3.87E-31**	8.47E-32	$500.04^{\dagger}$
	QBFA	3.3356E-05	4.7131E-07	N/A
f_{13}	GA	0.1681	7.0681E-2	$-16.82^{\dagger}$
	PSO	5.0519E-2	0.5691	-0.63
	GSO	4.6948E-5	1.005E-4	$-3.29^{\dagger}$
	FBSA	1.35E-4	9.72E-6	$-98.10^{\dagger}$
	QBFA	**1.5393E-07**	0	N/A

The value of t with 999 degree of freedom calculated by Eq. (12) is significant at $\alpha = 0.05$ by a two-tailed test.

ability to scale up to 300 or higher dimension problems.

3.7. *Intelligent tuning of PID controller for automatic voltage regulator(AVR) using QBFA*

The block diagram of an AVR system [23,24,25] with a PID controller is given by

$$\mathrm{PID}(s) = k_p + \frac{k_i}{s} + K_d s$$

Table 6. Result of different function evaluations for $f_{10} - f_{13}$. All results have been averaged over 50 runs.

Function	Function Evaluations			
	1.5E5	2.0E5	3.0E5	1.0E6
	Mean (Standard Deviation)			
f_{10}	5.4310E-02 (7.3145E-03)	3.2693E-12 (3.6720E-27)	3.2694E-12 (3.67E-27)	1.1479E-12 (1.4184E-12)
f_{11}	3.1675E-02 (9.3549E-02)	0 (0)	0 (0)	0 (0)
f_{12}	3.3356E-05 (4.7131E-07)	1.0912E-08 (1.4388E-17)	1.0912E-08 (7.1102E-17)	1.0782E-08 (9.1307E-10)
f_{13}	4.6871E-04 (3.3232E-05)	1.5393E-07 (0)	1.5299E-07 (4.7317E-09)	1.5000E-07 (1.7837E-08)

Table 7. Comparison of QBFA with GA,PSO and GSO for $f_{14} - f_{18}$. All results have been averaged over 1000 runs.

Function	Algorithms	Mean	Std.	t-test
f_{14}	GA	0.9989	4.4333E-03	-1.44
	PSO	1.0239	0.1450	-1.26
	GSO	**0.9980**	0	0.00
	QBFA	0.9980	1.6194E-11	N/A
f_{15}	GA	7.0878E-03	7.8549E-03	$-4.57^{\dagger}$
	PSO	3.8074E-04	2.5094E-04	$27.07^{\dagger}$
	GSO	**3.7713E-04**	2.5973E-04	$26.80^{\dagger}$
	QBFA	2.0063E-03	3.4259E-03	N/A
f_{16}	GA	-1.0298	3.1314E-3	$-4.13^{\dagger}$
	PSO	-1.0160	1.2786E-2	$-8.64^{\dagger}$
	GSO	-1.031628	0	$-28.69^{\dagger}$
	QBFA	**-1.0316283811**	9.39125E-08	N/A
f_{17}	GA	0.4040	1.0385E-2	$-4.09^{\dagger}$
	PSO	0.4040	6.8805E-2	-0.62
	GSO	0.3979	0	$3.48^{\dagger}$
	QBFA	**0.39799**	1.8275E-04	N/A
f_{18}	GA	7.5027	10.3978	$-3.06^{\dagger}$
	PSO	3.0050	1.2117E-2	$-2.92^{\dagger}$
	GSO	**3.0**	0	$3.75^{\dagger}$
	QBFA	3.0000020399	3.8425E-06	N/A

The block diagram is shown in Fig. 2. The performance index of control response is defined by

$$\begin{aligned} \min F(k_p, k_i, k_d) &= \frac{e^{-\beta\, t_s}/max(t)}{(1-e^{-\beta})\|1 - t_r/max(t)\|} + e^{-\beta} Mo + ess \\ &= \frac{e^{-\beta}(t_s/max(t) + \alpha Mo)}{\alpha} + ess \end{aligned} \tag{13}$$

Table 8. Comparison of QBFA with GA, PSO and GSO for $f_8(x)^{300} - f_{13}(x)^{300}$, all results have been averaged over 50 runs.

Function	QBFA	GSO	GA	PSO
$f_8(x)^{300}$	**-125694.86**	-125351.2	-117275.3	-87449.2
$f_9(x)^{300}$	**0.0637**	98.9	121.3	427.1
$f_{10}(x)^{300}$	**3.2694E-12**	1.3527E-3	6.24	3.9540E-6
$f_{11}(x)^{300}$	**0**	1.8239E-7	0.37	1.81
$f_{12}(x)^{300}$	**3.3803E-09**	8.2582E-8	52.82	14.56
$f_{13}(x)^{300}$	**3.38E-09**	2.0175E-7	178.34	549.2

where $\alpha = (1-e^{-\beta})\|1-t_r/\max(t)\|$, k_p, k_i, K_d are the parameters of PID controller, β the weighting factor, Mo the overshot, t_s the settling time (2%), ess the steady-state error, t the desired settling time.

Performance criterion is defined as $Mo = 10\%, ess = 0.0909, t_r = 0.2693(s)$, $t_s = 6.9834(s)$. **For the PID controller,** $0 < k_p < 1.5, 0 < k_p < 1, 0 < k_d < 1$. The parameters of QBFA are defined as: $N_b = 50, P_{ed} = 0.25, N_{cs} = 50, N_s = 50, N_{re} = 50, N_{ed} = 50, IterMax = 1E5$. The QBFA-PID controller shown in Fig. 2 replaced the PID controller. As we can see, the QBFA-PID controller can search optimal PID controller parameters efficiently. Table 9 depicts the best solutions obtained using QBFA controller for different β values. As β increase form 1 to 1.5, the overshot decrease from 4.56% to 1.45%, as indicated from Table 9.

Table 9. Best solutions obtained using QBFA controller with different β values.

β	k_p	k_i	k_d	$Mo(\%)$	ess	t_s	t_r	Evaluation vlaue
0.5	0.7713	0.7384	0.3243	4.5625	0.0187	0.310	0.217	0.1171
1	0.7052	0.5931	0.2940	1.8488	0.0126	0.351	0.240	0.0497
1.5	0.6698	0.5824	0.2804	1.5408	0.0154	0.376	0.255	0.0349

From the results obtained by QBFA-PID, we can see that overshot is smaller than that of BF-GA [23], for different β, the settling time of QBFA-PID is 0.31,0.24,0.255, while the settling time of BF-GA is 0.377,0.401, 0.418. The rise time of QBFA is smaller than BF-GA. The results show that the calculation time does not increase for QBFA-PID, The reason is that the new initial method and observational manner of QBFA guarantee that less time used on observation of the population. This practical application also indicate the QBFA is more efficient than GA-like algorithm.

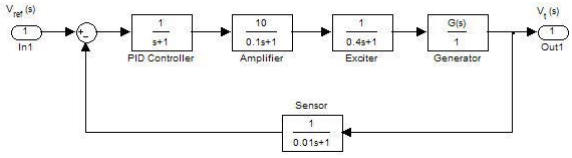

Fig. 2. Block diagram of an AVR system with PID controller.

4. Conclusions

This paper has presented **a quantum** inspired bacterial foraging algorithm. The presented scheme attempt to make a judicious balance of exploration and exploitation abilities of the search space and likely to avoid false and premature convergence in many cases. Also, the performance is illustrated using 18 benchmark functions and 8 high dimensional test functions. The overall performance of the proposed algorithm is better than a standard alone BFA and others on the most of the numerical benchmark functions. Also, the proposed algorithm is used to tune a PID parameter of an automatic voltage regulator(AVR) system. Simulation results clearly illustrate that the proposed approach is efficient. The results also suggest that the quantum computing can be applied to other evolutionary optimization methods easily.

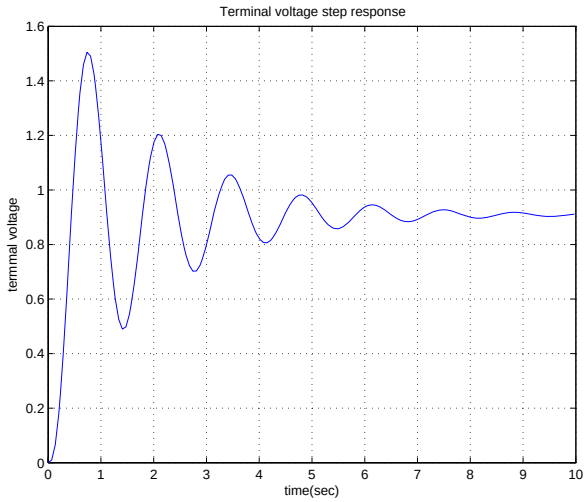

Fig. 3. Step response of terminal voltage in an AVR system without controller.

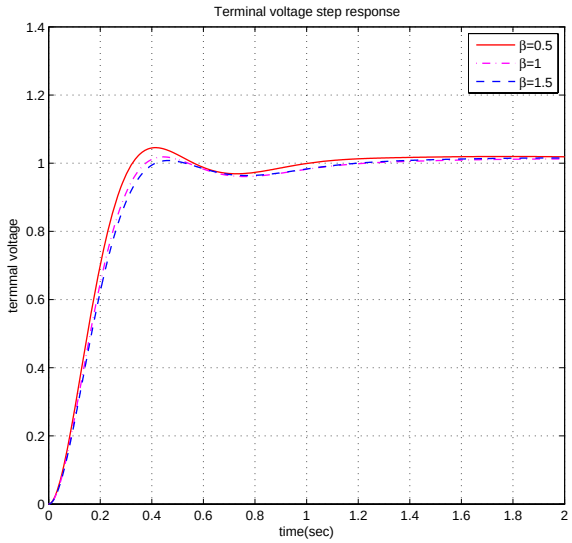

Fig. 4. Terminal voltage step response of an AVR system using QBFA algorithm.

References

1. R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in *Proc. of the sixth international symposium on micro machine and human science*, (October 1995), Nagoya,Japan, 39-43.
2. D. Whitley, A genetic algorithm tutorial, *Statistics and computing.* **4** (1994) 65–85.
3. S. Dasgupta, A. Biswas, A. Abraham, S. Das,Adaptive computational chemotaxis in bacterial foraging algorithm, in *International Conference on Complex, Intelligent and Software Intensive Systems.* (March 2008), Barcelona Spain, 64–71.
4. S. Dasgupta, S. Das,A. Abraham, A. Biswas, Adaptive computational chemotaxis in bacterial foraging optimization: An analysis, *IEEE Transactions on Evolutionary Computation.* **13** (2009) 919–941.
5. K. Passino, Biomimicry of bacterial foraging for distributed optimization and control, *IEEE Control Systems Magazine.* **22** (2002) 52–67.
6. T.K. Das, G.K. Venayagamoorthy,U.O. Aliyu, Bio-Inspired algorithms for the design of multiple optimal power system stabilizers: SPPSO and BFA, *IEEE Transactions on Industry Applications.* **44** (2008) 1445–1457.
7. B. Sumanbabu, S. Mishra, B. Panigrahi, G. Venayagamoorthy, Robust tuning of modern power system stabilizers using bacterial foraging algorithm, in *IEEE Congress on Evolutionary Computation*, (September 2007), Singapore, 2317–2324.
8. Z. Luo, W. Zhang, Y. Zhang, M. Xiang, C. Piao, Multi-class wavelet SVM classifiers using quantum-inspired evolutionary algorithm, in *7th World Congress on Intelligent Control and Automation*, (June 2008), Chongqing, China, 7146–7150.
9. H. Mi, H. Liao, Z. Ji, Q.H. Wu, A fast bacterial swarming algorithm for high-dimensional function optimization, in *IEEE World Congress on Computational Intelligence*, (June 2008), Hong Kong, China, 3135–3140.
10. W. Tang, M. Li, S. He, Q. Wu, J. Saunders, Optimal power flow with dynamic loads using bacterial foraging algorithm, in *International Conference on Power System Technology*, (October 2006), Chongqing, China, 1–5.
11. W. Tang, Q. Wu, J. Saunders, Bacterial foraging algorithm for dynamic environments, in *IEEE International Conference on Evolutionary Computation.* (July 2006), Vancouver, BC, Canada, 1324–1330.
12. W. J. Tang, M.S. Li, Q.H. Wu, J.R. Saunders, Bacterial foraging algorithm for optimal power flow in dynamic environments, *IEEE Transactions on Circuits and Systems I: Regular Papers.* **55** (2008) 2433–2442.
13. H. Supriyono, M. O. Tokhi, Parametric modelling approach using bacterial foraging algorithms for modelling of flexible manipulator systems, *Engineering Applications of Artificial Intelligence* (2012) http://dx.doi.org/10.1016/j.engappai.2012.03.004.
14. M. Tripathy, A. Barisal, Bacteria foraging algorithm based economic load dispatch with wind energy, in *Nature Biologically Inspired Computing.* (December 2009), Sambalpur, India, 1327 – 1332.
15. S. Muller, J. Marchetto, S. Airaghi, P. Kournoutsakos, Optimization based on bacterial chemotaxis, *IEEE Transactions on Evolutionary Computation.* **6** (2002) 16–29.
16. K. Han, J. Kim, Quantum-inspired evolutionary algorithms with a new termination criterion, Hϵ Gate, and two-phase scheme, *IEEE Transactions on Evolutionary Computation.* **8** (2004) 156–169.
17. M. Ying, Quantum computation, quantum theory and AI, *Artificial Intelligence.* **174** (2010) 162–176.
18. J. Sun, X. Wu, et al., Convergence analysis and improvements of quantum-behaved particle swarm optimization, *Information Sciences* **193** (2012) 81-103.
19. L., Wang, F. Tang, et al, Hybrid genetic algorithm based on quantum computing for

numerical optimization and parameter estimation, *Applied Mathematics and Computation* **171** (2005) 1141-1156.
20. K. Meng, H.G. Wang, Z. Dong, K.P. Wong, Quantum-Inspired particle swarm optimization for Valve-Point economic load dispatch, *IEEE Transactions on Power Systems.* **25** (2010) 215–222.
21. J. C. Lee,, W. M. Lin, G. C. Liao, T. P. Tsao, Quantum genetic algorithm for dynamic economic dispatch with valve-point effects and including wind power system, *International Journal of Electrical Power and Energy Systems* **33** (2011) 189-197.
22. J. Gu, M. Gu, C. Cao, X. Gu, A novel competitive co-evolutionary quantum genetic algorithm for stochastic job shop scheduling problem, *Computers and Operations Research* **37** (2010) 927-937.
23. D.H. Kim, A. Abraham, J.H. Cho, A hybrid genetic algorithm and bacterial foraging approach for global optimization, *Information Sciences*, **177** (2007) 3918–3937.
24. Z. Gaing, A particle swarm optimization approach for optimum design of PID controller in AVR system, *IEEE Transactions on Energy Conversion.* **19** (2004) 384–391.
25. T. Glad, L. Ljung, Control theory: multivariable and nonlinear methods, (CRC Press, Taylor and Francis, London, 2000).
26. P. C. Li, K. P. Song, F. H.Shang, Double chains quantum genetic algorithm with application to neuro-fuzzy controller design, *Advances in Engineering Software* **42** (2011) 875-886.
27. Y. W. Jeong, J. B. Park, A new quantum-inspired binary PSO: Application to unit commitment problems for power systems, *IEEE Transactions on Power Systems* **25** (2010) 1486-1495.
28. J. Xiao, Y. Yan,J. Zhang, Y. Tang, A quantum-inspired genetic algorithm for k-means clustering, *Expert Systems with Applications.* **37** (2010) 4966–4973.
29. D. Devaraj, P. Ganesh Kumar, Mixed genetic algorithm approach for fuzzy classifier design, *International Journal of Computational Intelligence and Applications* **9** (2010) 49-67.
30. B. Li, L. Wang, A hybrid Quantum-Inspired genetic algorithm for multiobjective flow shop scheduling, *IEEE Transactions on Systems, Man and Cybernetics, Part B Cybernetics.* **37** (2007) 576–591.
31. F. Liu, Y. Liu, H. Hao, Unsupervised SAR image segmentation based on quantum-inspired evolutionary gaussian mixture model, in *2nd Asian-Pacific Conference on Synthetic Aperture Radar.* (October 2009), Xian, China, 809–812.
32. W. J. Tang, Q.H. Wu, J.R. Saunders, A bacterial swarming algorithm for global optimization, in *IEEE Congress on Evolutionary Computation.* Singapore, (September 2007), 1207–1212.
33. J. Xiao, J. Xu, Z. Chen, K. Zhang, L. Pan, A hybrid quantum chaotic swarm evolutionary algorithm for DNA encoding, *Computers & Mathematics with Applications.* **57** (2009) 1949–1958.
34. M. Bhattacharya, N. Sharma, V. Goyal, S. Bhatia, A. Das,, A study on genetic algorithm based hybrid softcomputing mode for benignancy/malignancy detection of masses using digital mammogram, *International Journal of Computational Intelligence and Applications* **10** (2011) 141-165.
35. X. Yao, Y. Liu, G. Lin, Evolutionary programming made faster, *IEEE Transactions on Evolutionary Computation.* **3** (1999) 82–102.
36. C. Markowski, E. Markowski, Conditions for the effectiveness of a preliminary test of variance, *American Statistician.* **44** (1990) 322–326.
37. R. Bellman, *Adaptive control processes - A guided tour* (Princeton, N. J., 1961).
38. J. Traub, *Information-based complexity*, John Wiley and Sons Ltd., Chichester, UK,

2003.
39. Y. Shang, Y. Qiu, A note on the extended Rosenbrock function, *Evolutionary Computation.* **14** (2006) 119–126.
40. S. He, Q.H. Wu, J.R.Saunders, Group search optimizer: an optimization algorithm inspired by animal searching behavior, *IEEE Transactions on Evolutionary Computation.* **13** (2009) 973–990.