

Research Article

Function Optimization Based on Quantum Genetic Algorithm

Ying Sun and Hegen Xiong

College of Machinery and Automation, B.O.X 242, Wuhan University of Science and Technology,
Wuhan, 430081, China

Abstract: Optimization method is important in engineering design and application. Quantum genetic algorithm has the characteristics of good population diversity, rapid convergence and good global search capability and so on. It combines quantum algorithm with genetic algorithm. A novel quantum genetic algorithm is proposed, which is called Variable-boundary-coded Quantum Genetic Algorithm (vbQGA) in which qubit chromosomes are collapsed into variable-boundary-coded chromosomes instead of binary-coded chromosomes. Therefore much shorter chromosome strings can be gained. The method of encoding and decoding of chromosome is first described before a new adaptive selection scheme for angle parameters used for rotation gate is put forward based on the core ideas and principles of quantum computation. Eight typical functions are selected to evaluate the effectiveness and performance of vbQGA against standard Genetic Algorithm (sGA) and Genetic Quantum Algorithm (GQA). The simulation results show that vbQGA is significantly superior to sGA in all aspects and outperforms GQA in robustness and solving velocity, especially for multidimensional and complicated functions.

Keywords: Function optimization, optimization algorithm, quantum genetic algorithm, variable-boundary coding

INTRODUCTION

Quantum computation is a new and developing interdisciplinary integrating information science and quantum mechanics. Benioff (1980) and Feynman (1982) proposed the concepts of quantum computing. Shor (1994) presented a quantum algorithm used for factoring very large numbers, Grover (1996) developed a quantum mechanical algorithm to search unsorted database. Since then, quantum computing has attracted serious attention and been widely investigated by researchers. Narayanan and Moore (1996) and Han (2000) proposed respectively quantum inspired genetic algorithm and genetic quantum algorithm. These algorithms are inspired by certain concept and principles of quantum computing such as qubits and superposition of states. Chromosomes in these algorithms are probabilistically represented by qubits and so can represent a linear superposition of solutions. Many researchers have found that these algorithms have excellent performance such as population diversity, rapid convergence and global search capability. Wang *et al.* (2005) have put effective applications in shop scheduling.

In classical quantum genetic algorithms, chromosomes are generally represented by two types, qubits and binary, during the algorithm procedure. Binary chromosomes are generated by observing (equating quantum collapsing in quantum mechanics)

qubit chromosomes. The two types of chromosomes have the same length. As the more of dimension of optimization problems, the bigger of range of variables and the higher of precision of variables, the chromosome strings will become longer and then result in big memory requirement and long run time for a computer.

In order to improve this condition, this study presents a novel quantum genetic algorithm, in which chromosomes are encoded by qubit and variable-boundary, to expect to short the length of chromosome strings and then cut down the memory requirement and speed up the run velocity of algorithm.

RESULTS AND DISCUSSION

Variable-boundary-coded quantum genetic algorithm: Han (2000) proposed a novel evolutionary computing method called a Genetic Quantum Algorithm (GQA) and applied it to a well-known combinatorial optimization problem, knapsack problem. His research shows that GQA is superior to other genetic algorithm. Based on the GQA, we propose a novel quantum genetic algorithm called variable-boundary-coded quantum genetic algorithm, vbQGA, which we will introduce in this section.

Representation in vbQGA: In GQA, the smallest unit of information is qubit. A qubit may be in the '0' state,

Corresponding Author: Ying Sun, College of Machinery and Automation, B.O.X 242, Wuhan University of Science and Technology, Wuhan, 430081, China

This work is licensed under a Creative Commons Attribution 4.0 International License (URL: <http://creativecommons.org/licenses/by/4.0/>).

in the ‘1’ state, or in any superposition of the two. Based on the idea, in vbQGA, we represent the state of a qubit as follow:

$$|\Psi\rangle = \alpha \cdot |x^l\rangle + \beta \cdot |x^u\rangle \quad (1)$$

where, x^l and x^u are respectively the lower bound and the upper bound of some variable x , α and β are complex numbers that specify the probability amplitudes of the corresponding states. Obviously, a qubit may be in the ‘ x^l ’ state, in the ‘ x^u ’ state, or in any superposition of the two. The $|\alpha|^2$ and $|\beta|^2$ give respectively the probability that the qubit will be found in ‘ x^l ’ state and in ‘ x^u ’ state. Normalization of the state to unity guarantees:

$$|\alpha|^2 + |\beta|^2 = 1 \quad (2)$$

Now suppose we have an N -dimension function optimization problem described as:

$$\begin{aligned} \min: f(X) &= f(x_1, x_2, \dots, x_i, \dots, x_N) \\ \text{s.t.}: x_i^l &\leq x_i \leq x_i^u \quad i = 1, 2, \dots, N \end{aligned} \quad (3)$$

With respect to the chromosome k in generation t , the substring of variable x_i can be represented by qubit as follow:

$$q_{k,i}^t = \begin{bmatrix} \alpha'_{k,i1} & \alpha'_{k,i2} \\ \beta'_{k,i1} & \beta'_{k,i2} \end{bmatrix} \quad (4)$$

then, a whole qubit chromosome string for the N -dimension function optimization problem can be defined as:

$$q_k^t = \begin{bmatrix} \alpha'_{k,11} & \alpha'_{k,12} & \alpha'_{k,21} & \alpha'_{k,22} & \dots & \alpha'_{k,i1} & \alpha'_{k,i2} & \dots & \alpha'_{k,N1} & \alpha'_{k,N2} \\ \beta'_{k,11} & \beta'_{k,12} & \beta'_{k,21} & \beta'_{k,22} & \dots & \beta'_{k,i1} & \beta'_{k,i2} & \dots & \beta'_{k,N1} & \beta'_{k,N2} \end{bmatrix} \quad (5)$$

Apparently, the length of a qubit chromosome is $L = 2N$. Let s be the population size, then chromosome population in generation t can be described as:

$$Q^t = \{q_k^t \mid k = 1, 2, \dots, s\} \quad (6)$$

Observation of qubit chromosomes in vbQGA: Observation in quantum genetic algorithm is similar to quantum collapse in quantum mechanics. In GQA, a probabilistic qubit chromosome will “collapse” into a binary chromosome through observation. However, in vbQGA, a qubit chromosome will “collapse” into a variable-boundary coded chromosome. For any a qubit

$[\alpha'_{k,ij}, \beta'_{k,ij}]^T$ ($j = 1, 2$) we generate a random number between 0 and 1, $r'_{k,ij}$, if $r'_{k,ij} \leq |\alpha'_{k,ij}|^2$ the qubit will be found in the ‘ x^l ’ state, otherwise, the qubit will be found in the ‘ x^u ’ state. With the substring of variable x_i of chromosome k in Eq. (4), we can “collapse” it into a substring of a variable-boundary coded chromosome, which we denote as $vb'_{k,i}$. A $vb'_{k,i}$ can be one of the four conditions defined as:

$$vb'_{k,i} \in \{[x_i^l \ x_i^l], [x_i^l \ x_i^u], [x_i^u \ x_i^l], [x_i^u \ x_i^u]\} \quad (7)$$

So, a whole variable-boundary coded chromosome may be, for example, is in follow form:

$$vb'_k = [x_1^l \ x_1^l | x_2^l \ x_2^l | \dots | x_i^l \ x_i^l | \dots | x_N^u \ x_N^u] \quad (8)$$

and then the variable-boundary coded chromosome population in generation t can be described as:

$$VB^t = \{vb'_k \mid k = 1, 2, \dots, s\} \quad (9)$$

Rules of decoding: As described in Eq. (7), the substring of a variable-boundary coded chromosome with respect to variable x_i can be one of the four conditions. The four conditions correspond to four value regions (namely I, II, III and IV) of x_i , which are gotten by equally dividing the first quadrant, illustrated in Fig. 1. Let $\Delta x_i = x_i^u - x_i^l$ then every region represents a value span of $\Delta x_i / 4$. The decoding rules of variable-boundary coded chromosome are given in Table 1.

In Table 1, r is a random number between 0 and 1. If $vb'_{k,i} = [x_i^l \ x_i^l]$, x_i will take a small value inclining to the lower bound, the corresponding value region is Region I. If $vb'_{k,i} = [x_i^l \ x_i^u]$ and $vb'_{k,i} = [x_i^u \ x_i^l]$, x_i will take a intermediate value, the corresponding value region are Region II and Region III respectively. Then if $vb'_{k,i} = [x_i^u \ x_i^u]$, x_i will take a big value inclining to the upper bound, the corresponding value region is Region IV.

Adaptive quantum rotation gate strategy: In many kinds of quantum-inspired algorithms, a primary updating operator for chromosomes is quantum rotation gate, which is defined as following as (Han, 2000):

$$U(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

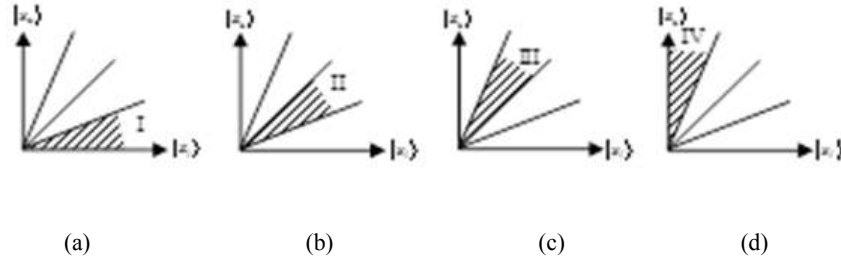


Fig. 1: Four value regions of xi corresponding to vbki

$vb_{k,i}^t$	x_t	Illustrations
$[x_t^l, x_t^u]$	$x_t = x_t^l + r \cdot \frac{\Delta x_t}{4}$	Region I in Fig. 1a
$[x_t^l, x_t^u]$	$x_t = x_t^l + (1+r) \cdot \frac{\Delta x_t}{4}$	Region II in Fig 1b
$[x_t^u, x_t^l]$	$x_t = x_t^u - (1+r) \cdot \frac{\Delta x_t}{4}$	Region III in Fig 1c
$[x_t^u, x_t^l]$	$x_t = x_t^u - r \cdot \frac{\Delta x_t}{4}$	Region IV in Fig 1d

where, θ is rotation angle, which is generally looked up from a table. In our algorithm, quantum rotation gate for the substring of a qubit chromosome with respect to variable x_i is represented as:

$$U_{k,ij}^t(\theta) = \begin{bmatrix} \cos(\theta_{k,ij}^t) & -\sin(\theta_{k,ij}^t) \\ \sin(\theta_{k,ij}^t) & \cos(\theta_{k,ij}^t) \end{bmatrix}, j = 1,2 \quad (10)$$

In order to make the qubit chromosomes effectively converge to the fitter states, we put forward an adaptive rotation angles computing method, which is defined as:

$$\theta_{k,ij}^t = \text{sign} \left\{ \alpha_{k,ij}^t \cdot \beta_{k,ij}^t \cdot [f(\mathbf{b}^{t-1}) - f(\mathbf{X}_k^t)] \right\} \cdot \frac{b_i^{t-1} - x_{k,i}^t}{x_i^u - x_i^l} \times 0.05\pi \quad (11)$$

where, X_k^t is the solution k in generation t and b^{t-1} is best solution in generation $t-1$, $x_{k,i}^t$ and b_i^{t-1} are the value of their i th variable respectively, $f(X_k^t)$ and $f(b^{t-1})$ are their fitness respectively. $\text{sign}(\bullet)$ is a sign function which described as:

$$\text{sign}(\bullet) = \begin{cases} +1 & \text{if: } \alpha_{k,ij}^t \cdot \beta_{k,ij}^t \cdot [f(\mathbf{b}^{t-1}) - f(\mathbf{X}_k^t)] > 0 \\ -1 & \text{if: } \alpha_{k,ij}^t \cdot \beta_{k,ij}^t \cdot [f(\mathbf{b}^{t-1}) - f(\mathbf{X}_k^t)] < 0 \\ \pm 1 & \text{if: } \alpha_{k,ij}^t = 0 \text{ and } (b_i^{t-1} - x_{k,i}^t) \cdot [f(\mathbf{b}^{t-1}) - f(\mathbf{X}_k^t)] < 0 \\ \pm 1 & \text{if: } \beta_{k,ij}^t = 0 \text{ and } (b_i^{t-1} - x_{k,i}^t) \cdot [f(\mathbf{b}^{t-1}) - f(\mathbf{X}_k^t)] > 0 \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

By analyzing Eq. (11, 12) we can get: Rotation angles are adaptive selected according to the difference values between $x_{k,i}^t$ and b_i^{t-1} . The bigger the difference values are, the bigger the absolute value of rotation angles are also.

The rotation directions, which can be gotten by (12), of quantum gate can make the solution converge to the fitter states. For example, if $f(\mathbf{b}^{t-1}) - f(\mathbf{X}_k^t) > 0$ (i.e., solution \mathbf{b}^{t-1} is better than \mathbf{X}_k^t) and $b_i^{t-1} > x_{k,i}^t$, then we should increase the $\beta_{k,ij}^t$ so as to augment the probability of 'x^u' state in the variable-boundary-coded chromosome. Hence, if $\alpha_{k,ij}^t \cdot \beta_{k,ij}^t > 0$ (i.e., in the first quadrant), the quantum gate should rotate in anticlockwise direction and the rotation angle should be positive. This just agrees with the result we can get from Eq. (11) and (12). Other conditions can be analyzed in the same method.

Procedure of vbQGA: The algorithm of vbQGA can be implemented as follows:

```

procedure vbQGA
begin
    t ← 0
    initialize Qt
    make VBt by observing Qt states
    decode VBt into Xt and evaluate them
    store the best solution, bt, among Xt
    while (not termination-condition) do
        begin
            t ← t+1
            make VBt by observing Qt states
            decode VBt into Xt and evaluate them
            compare with Xt and bt-1, and update Qt
        end
    end
using quantum gates
    store the best solution, bt, among Xt
end
end
    
```

Experimental evaluation of VbQGA:

Test functions: For the experimental evaluation of the algorithm presented in above section eight typical test functions is chosen.

De Jong function: De Jong function is defined as:

$$F_1 = 100(x_1^2 - x_2)^2 + (1 - x_1)^2, \quad -2.048 \leq x_i \leq 2.048, \quad i=1,2 \quad (13)$$

Although being mono-peak, DeJong function is ill-conditioned and intractable to search the global minimal solution:

$$f(1,1) = 0$$

Coldstein Price function: Coldstein Price function is described as:

$$F_2 = [1 + (x_1 + x_2 + 1)^2 \cdot (19 - 14x_1 + 3x_1^2 - 14x_2 + 6x_1x_2 + 3x_2^2)] \cdot [30 + (2x_1 - 3x_2)^2 \cdot (18 - 32x_1 + 12x_1^2 + 48x_2 - 36x_1x_2 + 27x_2^2)], \quad -2 \leq x_i \leq 2, \quad i=1,2 \quad (14)$$

This function has only one global minimal solution:

$$f(0,-1) = 3$$

Schaffer function: Schaffer function is given by:

$$F_3 = 0.5 + \frac{\sin^2 \sqrt{x_1^2 + x_2^2} - 0.5}{[1.0 + 0.001(x_1^2 + x_2^2)]^2}, \quad -100 \leq x_i \leq 100, \quad i=1,2 \quad (15)$$

This function has only one global minimal solution:

$$f(0,0) = 0$$

Mono-pole and six-peak camelback function: Mono-pole and six-peak camelback function is formulated as:

$$F_4 = 10 + \frac{\sin(1/x)}{0.1 + (x - 0.16)^2}, \quad 0 \leq x \leq 1 \quad (16)$$

The only one global maximal solution is

$$f(0.1275) = 19.8949$$

Dual-pole and six-peak camelback function: Dual-pole and six-peak camelback function is defined as:

$$F_5 = (4 - 21x_1^2 + \frac{1}{3}x_1^4)x_1^2 + x_1x_2 + (-4 + 4x_2^2)x_2^2, \quad -3 \leq x_i \leq 3, \quad i=1,2 \quad (17)$$

This function has two global minimal solutions, i.e.,

$$f(-0.0898, 0.7126) = f(0.0898, -0.7126) = -1.031628$$

Multi-peak positive function: Multi-peak positive function is described as:

$$F_6 = e^{-0.001x} \cos^2(0.8x), \quad x \geq 0 \quad (18)$$

This function has two local optimal solutions and one global maximal solution:

$$f(0) = 1$$

Ackley function: Ackley function is given by:

$$F_7 = -20e^{-0.2\sqrt{\frac{1}{n}\sum_{i=1}^n x_i^2}} - e^{\frac{1}{n}\sum_{i=1}^n \cos(2\pi x_i)} + 22.71282, \quad -5 \leq x_i \leq 5 \quad i=1,2,\dots,n \quad (19)$$

This function has only one global minimal solution: $f(0,0,0,0,\dots,0,0) = 0$. In the experimental evaluation we will take into account two conditions, $n = 2$ and $n = 10$.

Rastrigin function: Rastrigin function is formulated as:

$$F_8 = 10n + \sum_{i=1}^n [x_i^2 - 10 \cos(2\pi x_i)], \quad -5.12 \leq x_i \leq 5.12, \quad i=1,2,\dots,n \quad (20)$$

The only one global minimal solution is $f(-420.9687, -420.9687, \dots, -420.9687) = 0$. We will take $n = 6$ for the experimental evaluation.

Optimization and results: In order to test and evaluate the effectiveness and performance of vbQGA, we will optimize the aforementioned eight functions with sGA, GQA and vbQGA.

In sGA, binary code, roulette wheel selection, one-point crossover and 0-1 mutation is adopted. The controlling parameters are: variable precision $p = 0.000001$, population size $s = 50$, crossover probability $pc = 0.8$, mutation probability $pm = 0.01$ and total generations of iteration $t = 500$. The algorithm of GQA we used here is the same as that mentioned in Han (2000). With GQA we will take controlling parameters as: $p = 0.000001$, $s = 10$ and $t = 500$, which are the same as those taken in vbQGA.

All the algorithms are integrated in a test system programmed by Java language. The test system is operated under the following environments: Microsoft windows XP 2002, Intel Pentium 1600 MHz and 504 M memory. For each algorithm 20 runs are performed with respect to the eight functions. The results are presented in Table 2.

In Table 2, f_{opt} denotes the function value of optimum, \bar{f} and sd are respectively average and standard deviation of function value over 20 runs, \bar{t} (*sec/run*) represents the average elapsed time per one run. $F_1 \sim F_8$ represent the corresponding functions described in the fore-subsection, e.g., F_1 represents De Jong function, F_2 represents Coldstein Price function etc. We should notice that F_7 represents two conditions' Ackley function, so there are two lines of results, the upper one corresponding to $n = 2$ and the lower one corresponding to $n = 10$.

For giving a much clearer view of the results, the data in Table 2 are illustrated by Fig. 2 to 5.

Table 2: Results of experimental evaluation

Functions	sGA				GQA				vbQGA			
	f_{opt}	\bar{f}	sd	\bar{t}	f_{opt}	\bar{f}	sd	\bar{t}	f_{opt}	\bar{f}	sd	\bar{t}
F_1	0.006	0.145	0.233	1.326	0.000	0.015	0.035	0.883	0.000	0.002	0.003	0.187
F_2	3.027	3.951	0.910	1.801	3.000	3.000	0.000	0.983	3.000	3.228	0.204	0.299
F_3	0.009	0.121	0.074	2.668	0.000	0.006	0.008	0.886	0.009	0.013	0.007	0.193
F_4	19.894	19.790	0.211	0.756	19.894	19.810	0.055	0.530	19.894	19.894	0.000	0.096
F_5	-1.032	-1.024	0.005	1.042	-1.032	-1.030	0.003	0.730	-1.032	-1.031	0.000	0.188
F_6	1.000	0.999	0.002	0.683	1.000	1.000	0.000	0.395	1.000	1.000	0.000	0.210
F_7	0.036	0.534	0.317	1.350	0.005	0.005	0.000	0.625	0.005	0.087	0.059	0.306
	1.437	2.468	1.220	3.602	0.016	0.149	0.704	4.602	0.040	0.206	0.087	0.862
F_8	3.231	8.417	2.956	2.278	0.995	2.773	2.721	1.434	0.995	2.912	0.536	0.522

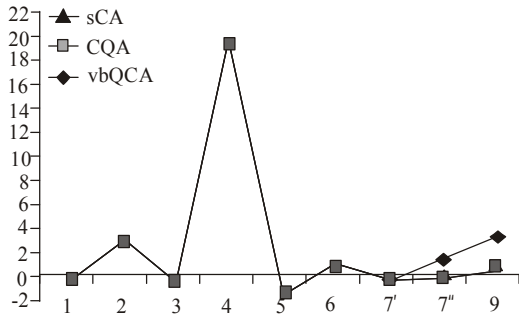


Fig. 2: f_{opt} of the eight functions

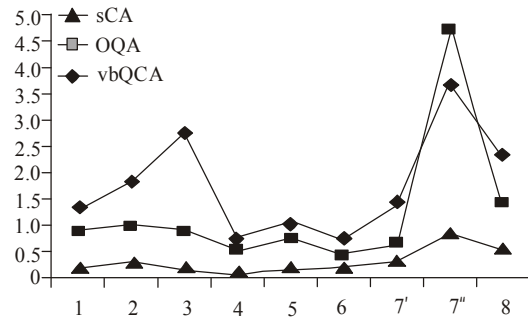


Fig. 5: \bar{f} of the eight functions over 20 runs

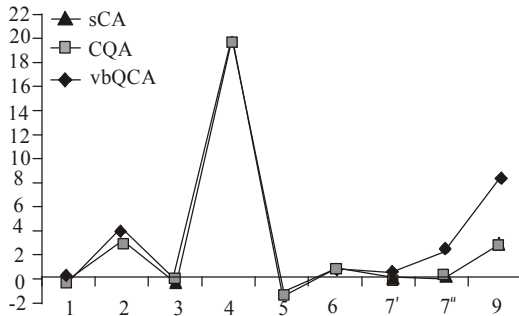


Fig. 3: \bar{f} of the eight functions over 20 runs

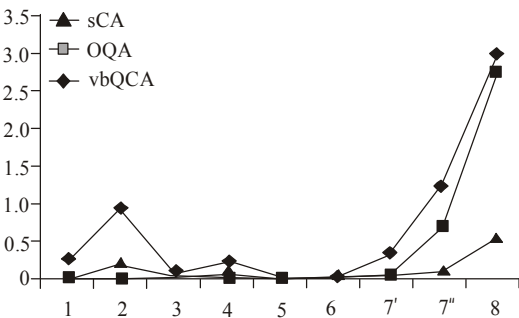


Fig. 4: sd of the eight functions over 20 runs

In Fig. 2 to 5, the numbers of x-coordinate represent the index of the corresponding functions, e.g., '1' represents F_1 (i.e., De Jong function), '2' represent F_2 (i.e., Coldstein Price function) etc.. We should also notice that '7'' and '7''' represent respectively Ackley function under condition $n = 2$ and $n = 10$.

It can be known from Fig. 2 that three algorithms under discussion can all get optimums with respect to $F_1 \sim F_6$. However, for F_7 and F_8 , No one of the three algorithms can get optimums under the given controlling parameters. Though, the solutions of GQA and vbQGA are still obviously better than that of sGA. From Fig. 6 and 7 we can find that F_7 and F_8 are very complicated and intractable for their exiting many local optimums. By taking population size as $s = 50$ and remaining other parameters unchanged, we carried out some test runs and the results show that GQA and vbQGA can exactly find out the optimums of F_7 and F_8 .

Fig. 3 tells us that sGA, GQA and vbQGA gain closely approximate averages of the function value of $F_1 \sim F_6$ over 20 runs. In contrast with this, for F_7 and F_8 , the averages obtained by GQA and vbQGA are very approximate and evidently superior to those by sGA.

Figure 4 shows that vbQGA gets the smallest standard deviations among the three algorithms and sGA gets the largest ones. It reveals that vbQGA is more robust than the two algorithms.

Figure 5 illustrates the comparison of average of elapsed time per one run among the three algorithms. It can be seen that vbQGA takes the least run time. Let us sum up all the t of eight functions for sGA, GQA and vbQGA and we can get 15.5062 sec, 11.0675 sec and 2.8631 sec, respectively. Obviously, vbQGA takes much less run time than the other two algorithms. In addition, it can be also seen that as the dimension and complexity of a function increase, this advantage will get more distinct.

To sum up, vbQGA is superior to sGA in all respects. Comparing with GQA, vbQGA can get very

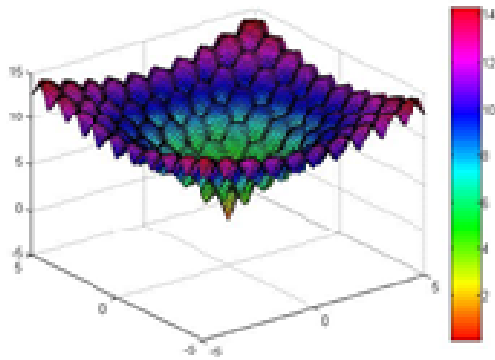


Fig. 6: Figure of function Ackley (n = 2)

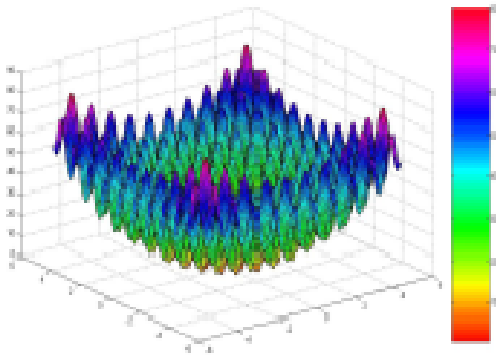


Fig. 7: Figure of function Rastrigin (n = 2)

approximate quality of solutions. However, the standard deviation and the average elapsed time per one run of vbQGA are, especially for the multidimensional and complicated functions, less than GQA. This indicates that vbQGA has better robustness and solving velocity.

CONCLUSION

In this study, a variable-boundary-coded quantum genetic algorithm was proposed, vbQGA, based on the core idea and principle of quantum computation. In this algorithm, qubit chromosomes are collapsed into variable-boundary-coded chromosomes instead of binary-coded chromosomes, a new adaptive selection strategy for angle parameters used for rotation gate is adopted. An experimental evaluation, in which eight typical functions are selected to optimize and sQA and GQA are selected as contrasts, has been conducted. Four statistical values have been used as measurements of performance to evaluate vbQGA. The results reveal

that vbQGA is significantly superior to sGA in all aspects and outperforms GQA in robustness and solving velocity, especially for multidimensional and complicated functions. These demonstrate effectiveness and good performance of vbQGA.

ACKNOWLEDGMENT

This research work has been supported by Hubei Province Department of Education in China (Grant No. D20121102) and Key Laboratory of Metallurgical Equipment, Control of Ministry of Education in Wuhan University of Science and Technology. The author is grateful to the anonymous reviewers for their valuable suggestions and comments.

REFERENCES

- Benioff, P., 1980. The computer as a physical system: A microscopic quantum mechanical Hamiltonian model of computers as represented by Turing Turing machine. *J. Stat. Phys.*, 22(8): 563-591.
- Feynman, R.P., 1982. Simulating physics with computers. *Int. J. Theor. Phys.*, 21(7): 467-488.
- Grover, L.K., 1996. A fast quantum mechanical algorithm for database search. *Proceeding of the 28th Annual ACM Symposium on the Theory of Computing*. Philadelphia, pp: 212-221.
- Han, K.H., 2000. Genetic quantum algorithm and its application to combinatorial optimization problem. *Proceeding of the 2000 Congress on Evolutionary Computation*. La Jolla, CA, 2: 1354-1360.
- Narayanan, A. and M. Moore, 1996. Quantum-inspired genetic algorithms. *Proceeding of the 3rd IEEE International Conference on Evolutionary Computation*. Nagoya, pp: 61-66.
- Shor, P.W., 1994. Algorithm for quantum computation: Discrete logarithms and factoring. *Proceeding of the 35th Annual Symposium on the Foundation of Computer Sciences*. IEEE Computer Society, Washington, DC, USA, pp: 124-134.
- Wang, L., H. Wu, F. Tang and D. Zheng, 2005. A hybrid quantum-inspired genetic algorithm for flow shop scheduling. *Lect. Notes Comput. Sc.*, 3645(1): 636-644.