2010 Second International Conference on Intelligent Human-Machine Systems and Cybernetics

A Novel Quantum-inspired Genetic Algorithm with Expanded Solution Space

Renjie Liao
School of Automation Science and Electrical Engineering
Beihang University
Beijing 100191
Email: lrjconan@gmail.com

Xueyao Wang
School of Automation Science and Electrical Engineering
Beihang University
Beijing 100191
Email: daphenne@sina.com

Zengchang Qin
School of Automation Science and Electrical Engineering
Beihang University
Beijing 100191
Email: zcqin@buaa.edu.cn

***Abstract*—In this paper, we present a novel quantum-inspired genetic algorithm with expanded solution space. Based on the double chains quantum genetic algorithm (DCQGA), we have expanded the solution space by increasing the number of solution space transformation functions. And we propose a novel method for quantum rotation gate's update by using the sign function and the gradient of objective function. With this method we can automatically determine the direction of quantum rotation gate and adaptively adjust the magnitude of quantum rotation gate. Through experimenting on 2 benchmark problem in the optimization literature: Rosenbrock function and Schaffer's F6 function, we demonstrate that our expanded solution space quantum genentic algorithm (ESSQGA) has achieved more satisfactory results than DCQGA and common genetic algorithm.**

I. Introduction

The evolutionary algorithm, such as the genetic algorithm (GA) [1], artificial immune algorithm (AIA) [2], particle swarm optimization (PSO) [3], ant colony algorithm (AC) [4], is essentially a process of imitation of biological system, populations' adaption to the environment and their interaction. It is thought as a significant guidance to researches of algorithms' fusion. Considering the quantum mechanism which is well known as accelerating calculation and enhancing global search capability[5], many researchers have tried to combine the evolutionary algorithm with quantum mechanism, thus approximately realizing the advantages of quantum mechanism on non-quantum computers. At present, the fusion of quantum mechanism and evolutionary algorithm is focus on the population coding and evolution strategy [6].

II. Expanded Solution Space Quantum Genetic Algorithm

For a variety of quantum evolutionary algorithm, the population coding is based on the binary coding of qubits' measurement and the evolution is realized through changes in the phase qubit. However, the frequent encoding and decoding process of binary code will increase the computation undoubtedly in numerical optimization problem. And the common method of changing phase qubit is based on lookup table which involves many conditional judgements. To overcome the shortcomings forementioned, we propose a novel quantum-inspired genetic algorithm which is improved from the double chains quantum genetic algorithm (DCQGA) [7].

A. Double Chain Coding

In the DCQGA, the probability amplitude of quantum bits is used for coding and the double chain coding is proposed. Considering the randomness of initial population and the constraints the quantum probability amplitude should satisfy, we take the double chain coding mechanism in our algorithm. The chromosome described in double chain coding mechanism is shown below:

$$p_i = \left[\left| \begin{matrix} \cos(t_{i1}) \\ \sin(t_{i1}) \end{matrix} \right| \begin{matrix} \cos(t_{i2}) \\ \sin(t_{i2}) \end{matrix} \left| \ldots \right| \begin{matrix} \cos(t_{in}) \\ \sin(t_{in}) \end{matrix} \right] \quad (1)$$

In this equation, $t_{ij} = 2\pi \times r$, r is a random number between 0 and 1; $i = 1, 2, \cdots, m; j = 1, 2, \cdots, n$; m represents the population size, while n represents quantum bits. In double chain coding mechanism, every quantum bit's probability amplitude is thought to be two genes up and down side by side, and every chromosome contains two parallel gene chains. Besides, every gene chain represents a optimal solution. Thus, every chromosome represents two optimal solutions of the search space in the same time. It can be shown in Equation (2) and Equation (3).

$$p_{is} = [\sin(t_{i1}), \sin(t_{i2}), ..., \sin(t_{in})] \quad (2)$$

$$p_{ic} = [\cos(t_{i1}), \cos(t_{i2}), ..., \cos(t_{in})] \quad (3)$$

In these two equations, $i = 1, 2, \cdots, m$. p_{is} is the sine solution and p_{ic} is the cosine solution. This can avoid the randomness in measurement. Moreover, during each iteration, these two solutions update in the same time. Therefore, with the same population size, we can enhance the ergodic ability to the search space and accelerate the process of optimization.

B. Solution Space Transformation

To make the algorithm applicable to every optimization problem, a process of solution space transformation is necessary. Through the transformation, we can obtain the final solution of the objective function from gene chains. In DCQGA, the solution space transformation is only limited to two dimension. It means that we can get two solutions from

978-0-7695-4151-8/10 $26.00 © 2010 IEEE
DOI 10.1109/IHMSC.2010.148

two gene chains. To improve the algorithm, we construct a transformation which can give four solutions from two gene chains, thus expanding the number of candidate solutions and increasing the probability of obtaining the global optimal solution.

For each chromosome, it contains $2n$ probability amplitude of quantum bits in one population. The $2n$ amplitude can be mapped to the solution space Ω from the n-dimensional unit space. Every probability amplitude corresponds to an optimized variable in the solution space. Setting $[\alpha_i^j, \beta_i^j]^T$ as the ith quantum bit in chromosome p_j, we can get the solution space variables based on the transformation: Equation (4) - Equation (7).

$$X_{i1}^j = \frac{1}{2}\left[b_i\left(1+\alpha_i^j\right) + a_i\left(1-\alpha_i^j\right)\right] \tag{4}$$

$$X_{i2}^j = \frac{1}{2}\left[b_i\left(1+\beta_i^j\right) + a_i\left(1-\beta_i^j\right)\right] \tag{5}$$

$$X_{i3}^j = \frac{1}{2}\left[b_i\left(1+\alpha_i^j\right) + a_i\left(1-\beta_i^j\right)\right] \tag{6}$$

$$X_{i4}^j = \frac{1}{2}\left[b_i\left(1+\beta_i^j\right) + a_i\left(1-\alpha_i^j\right)\right] \tag{7}$$

where b_i is the ith solution's lower limit and a_i is the ith solution's upper limit. As a result, we can get four solutions to the optimization problem from only one chromosome.

C. Crossover

In quantum-inspired genetic algorithm, the crossover operation is realized by rotating the phase qubit. We use the quantum rotation gate to update the phase qubit. It is shown in Equation (8).

$$U(\Delta\theta) = \begin{bmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) \\ \sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} \tag{8}$$

The update process is as below:

$$\begin{bmatrix} \cos(\Delta\theta) & -\sin(\Delta\theta) \\ \sin(\Delta\theta) & \cos(\Delta\theta) \end{bmatrix} \begin{bmatrix} \cos(t) \\ \sin(t) \end{bmatrix} = \begin{bmatrix} \cos(t+\Delta\theta) \\ \sin(t+\Delta\theta) \end{bmatrix} \tag{9}$$

where $\Delta\theta$ is the magnitude of the phase of the quantum rotation gate. From Equation (9), it is clear that the quantum rotation gate can change the phase instead of the length of quantum bits.

It is known that the magnitude and direction of phase $\Delta\theta$ can affect the convergence speed and efficiency of the algorithm[7]. The detailed implementation is as below:

1) The Direction of Quantum Rotation Gate: Assume α_0, β_0 as the probability amplitude of the global optimal solution's one qubit and α_1, β_1 as the probability amplitude of the current solution's one qubit.

Here, we assume matrix A as

$$A = \begin{vmatrix} \alpha_0 & \alpha_1 \\ \beta_0 & \beta_1 \end{vmatrix} \tag{10}$$

Thus, the direction of quantum rotation angle $\Delta\theta$ can be decided by the following equation:

$$\mathrm{sgn}(\Delta\theta) = \begin{cases} -\mathrm{sgn}(A) & A \neq 0 \\ -1 \ or + 1 & A = 0 \end{cases} \tag{11}$$

where $\mathrm{sgn}(x)$ is the sign function.

2) The Magnitude of the Phase of the Quantum Rotation Gate: Most quantum genetic algorithms take the lookup table method to determine the magnitude of the phase of the quantum rotation gate. However, its computation grows large since the conditional judgments in the table increase. To overcome the shortcoming, we propose a method based on the gradient of the objective function as below:

$$\Delta\theta_{k+1} = -\mathrm{sgn}(A) \times \left[\eta_k \Delta\theta_k + \lambda(1-\eta_k)\frac{\partial \Delta f_k}{\partial \theta_k}\right] \tag{12}$$

where $\Delta\theta_k$ is the kth magnitude of the phase of the quantum rotation gate, f is the objective function to be optimized, λ is the momentum factor and η_k is the kth rotation coefficient and is determined by the following function:

$$\eta_{k+1} = \begin{cases} \mu_{inc}\eta_k & \Delta f_{k+1} < \Delta f_k \\ \mu_{dec}\eta_k & \Delta f_{k+1} > \Delta f_k \end{cases} \tag{13}$$

In Equation (13), μ_{inc} is the rotation coefficient increase factor, and μ_{dec} represents the rotation coefficient decrease factor.

Considering the discrete optimization, we use the first order difference replacing the derivation as Equation (14) shows to us.

$$\Delta f_k = f_k - f_{k-1} \tag{14}$$

D. Mutation

For mutation procedure, we choose quantum Non-gate as the key operation. To realize it, we first choose one chromosome randomly. Then several quantum bits are chosen to do quantum Non-gate transformation, which can realize the exchange of two probability amplitude of quantum bits. Finally, two gene chains can variate in the same time. This kind of variation is actually realized by rotating the quantum bits' phases. For example, if one quantum bit's phase is t, after this variation, its phase becomes $\pi/2 - t$, that is, its phase rotate $\pi/2 - 2t$ forward. Every rotation is forward without comparing with the current best chromosome and thus can help to increase the diversity of population and lower the probability of premature convergence.

III. Experiments

A. Benchmark Function

To test the performance of ESSQGA and make a comparison with DCQGA and GA, two standard benchmark functions commonly found in the GA literature are employed. They are shown as follow:

- Rosenbrock Function:

$$f_1(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - {x_i}^2)^2 + (x_i - 1)^2] \qquad (15)$$

To make the Rosenbrock function visualized, we only draw the image of input vector which contains two dimensions.The image of Rosenbrock function is shown in Fig.1.

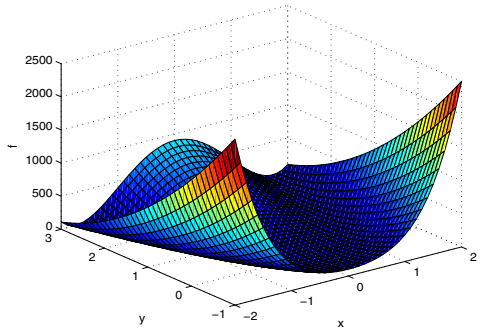

Fig. 1. The image of 2 dimension Rosenbrock function.

- Schaffer's F6 Function:

$$f_2(x) = 0.5 + \frac{(\sin\sqrt{x^2 + y^2})^2 - 0.5}{(1.0 + 0.001(x^2 + y^2))^2} \qquad (16)$$

The image of Schaffer's F6 function is shown in Fig.2.

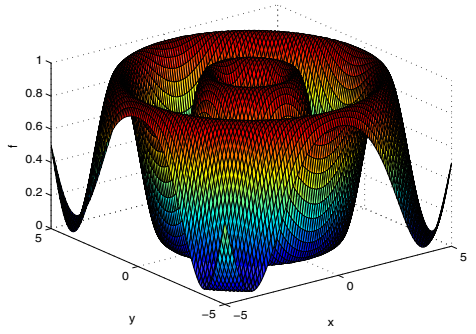

Fig. 2. The image of Schaffer$_F$6 function.

B. Detailed Implementation

To eliminate the parameter settings' effect on algorithm's performance as much as possible, we take the shared parameters fixed. Such as, the population size was set to 100, the probability of crossover was set to 0.8 and the probability of mutation was set to 0.1. The simulations were stopped when a predetermined stopping criterion was reached or after a maximum number of generations were run. The maximum number of generations for implementation was fixed at 500. For the specific parameters in ESSQGA, we set them as below:

the momentum factor λ equaled 0.9, the initial rotation coefficient η_0 equaled 0.1, the increase factor of rotation coefficient μ_{inc} equaled 1.25 and the decrease factor of rotation coefficient μ_{dec} equaled 0.8.

For Rosenbrock Function, it has a global minimum value 0 at the coordinate (1,1). For simplification, we took vector of two dimensions as the input of simulation and limited each dimension of the input vector in the range of -100 to 100. And we multiplied the Rosenbrock function by -1, thus the problem to seek the global minimum value had been turned into seeking the global maximum value. The predetermined stopping criterion of simulation was that the optimized result during one specific optimization was greater than -25. If the criterion was met, we took the optimized result as one convergence and regarded it as the final result of one specific simulation. Otherwise, we took the best optimized result in the population as the final result of one specific simulation.

For Schaffer's F6 function, it has a global maximum value 1 at the coordinate (0,0). We also limited each dimension of the input vector in the range of -100 to 100. The predetermined stopping criterion of simulation was that the optimized result during one specific optimization was greater than 0.995. We use the principles forementioned to analyze the result.

C. Experimental Results and Discuss

We conducted 100 simulations on each function above for each algorithm (GA,DCQGA,ESSQGA). To establish a framework of comparison, we used some indicators of evaluation listed in the tables.

1) The comparison of one specific simulation: To see the comparison result of these three algorithms in one specific simulation, we painted three optimization result in the same figure.

For Rosenbrock function, the results of the 3 optimization algorithms in one specific simulation are shown in Fig.3 and Fig.4.

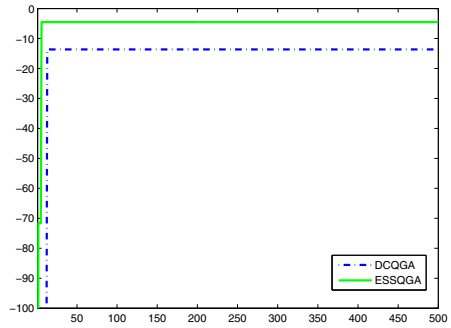

Fig. 3. The result of DCQGA and ESSQGA for Rosenbrock function.

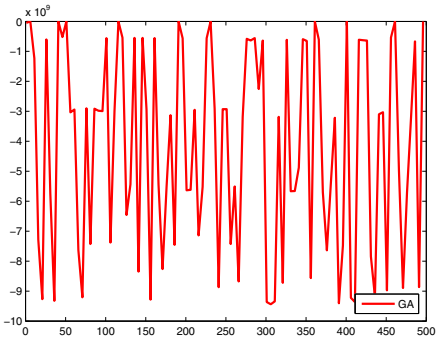

Fig. 4. The result of GA for Rosenbrock function.

We can see from the figure that the result of the GA is oscillated and DCQGA's and ESSQGA's results are more stable. In this figure, the best result GA got is -80.1467 which is smaller than the DCQGA's -13.6553 and the ESSQGA's -4.5293.

For Schaffer's F6 function, the results of the 3 optimization algorithm in one specific simulation are shown in Fig.5.

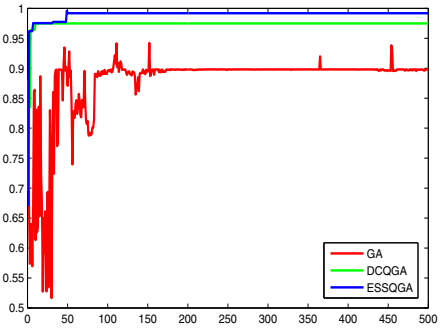

Fig. 5. The result of GA, DCQGA and ESSQGA for Schaffer's F6 function.

Through the figure, we can clearly see that genetic algorithm with quantum mechanism performs more stably and tends to get better results than common genetic algorithm.

2) The comparison of 100 simulation: We use indicators forementioned to evaluate the average performance of these 3 algorithms. The results of 100 simulation on Rosenbrock function are shown in Table1.

The results of 100 simulation on Schaffer's F6 function are shown in Table2.

Through the results in the tables, we can see that, for Rosenbrock function, though our algorithm's CPU time spend is higher than GA, it achieved the best average value and the best maximum value of optimized results. And for Schaffer's F6 function, our algorithm's CPU time spend is least, average value of optimized results is second and maximum value of optimized results is best.

The kind of algorithm	GA	DCQGA	ESSQGA
The time of convergence	1	97	100
The average spend of compute time	2.3541	5.8115	4.0593
The average value of optimized results	-2.2545e+009	-5.0946	-1.3208
The maximum value of optimized results	-20.0697	-0.0152	-0.0020

TABLE I
THE RESULTS OF SIMULATION ON ROSENBROCK FUNCTION

The kind of algorithm	GA	DCQGA	ESSQGA
The time of convergence	25	38	51
The average spend of compute time	5.69	6.47	1.28
The average value of optimized results	0.9420	0.9827	0.9772
The maximum value of optimized results	0.9992	0.9975	0.9998

TABLE II
THE RESULTS OF SIMULATION ON SCHAFFER'S F6 FUNCTION

IV. CONCLUSION AND FUTURE WORK

Through a great deal of simulation, our algorithm is proved better than common genetic algorithm and the DCQGA in most indicators. However, as many other evolutionary algorithms, the principles of setting parameters are not well-founded. Thus, it also needs empirical fine tuning. In the future, we will commit ourselves to giving rigorous mathematics analysis of our algorithm's advantages, and give some reliable principles of setting algorithm's parameters.

ACKNOWLEDGMENT

This work is partially funded by the NCET from the Ministry of Education of China.

REFERENCES

[1] Holland, J.: Adaptation in Natural and Artificial Systems. The University of Michigan Press, Ann Arbor, 1975.
[2] Dsgupta D. Artificial Immune Systems and Their Applications[M]. Berlin Heidelberg: Springer-Verlag, 1999.
[3] J. Kennedy, R. Eberhart.: Particle Swarm Optimization. Proc. IEEE Conf. On Neural Network (1995), pp.1942-1948.
[4] Dorigo, M., Maniezzo, V. and Colorni, A. (1996). The Ant System: Optimization by a Colony of Cooperating Agents. IEEE Trans. Syst. Man Cybern. B 26, 29-41.
[5] Grover L. K. A fast quantum mechanical algorithm for database search[C]. Proc. of the 28th annual ACM Symp. on Theory of Computing. New York, USA: ACM Press, 1996.6:212-219.
[6] Dero J and Siarry P. An ant colony algorithm aimed at dynamic continuous optimization[J].Applied Mathematics and Computation, 2006, 181(1): 457-467.
[7] LI P C and LI S Y. Quantum-inspired evolutionary algorithm for continuous spaces optimization[J]. Chinese Journal of Electronics, 2008, 17(1): 80-84.