سیزدهمین کنفرانس مهندسی برق ایران
۲۲-۲۰ اردیبهشت ۱۳۸۴

A Modified Genetic Algorithm Comparable to Quantum GA

Tahereh Kahookar Toosi
Ferdowsi University of Mashhad
t_k_toosi@wali.um.ac.ir

Habib Rajabi Mashhadi
Ferdowsi University of Mashhad
h_rajabi@ferdowsi.um.ac.ir

Abstract: *Recently, researchers proposed an evolutionary algorithm, QGA based on quantum concepts .It have shown that QGA's convergence and global search ability are superior to conventional GA's. This paper proposed a modified genetic algorithm that uses genetic operators effectively to have more advantages than QGA without employing any quantum feature.*

Keywords: Quantum Genetic Algorithm, A Hybrid GA, Knapsack Problem

1 Introduction

Recently, researchers have applied genetic algorithms (GAs) to address some problems in quantum computation. Also, there has been some works in the designing of genetic algorithms based on quantum theoretical concepts and techniques. The so-called Quantum Evolutionary Programming has two major sub-areas: Quantum Inspired Genetic Algorithms (QIGAs) and Quantum Genetic Algorithms (QGAs). The former adopts qubit chromosomes as representations and employs quantum gates for the search of the best solution. The later tries to solve a key question in this field: what GAs will look like as an implementation on quantum hardware? There is not a complete answer for this question. An important point for QGAs is to build a quantum algorithm that takes advantage of both the GA and quantum computing parallelism as well as true randomness provided by quantum computers.

There are two significant differences between a classical computer and a quantum computer. The first is in storing information, classical bits versus *quantum bits*. The second is the quantum mechanical feature known as entanglement, which allows a measurement on some quantum bits to affect the values of other quantum bits. The present QGAs use only the first feature. In fact, they are GAs that is using some quantum concepts. So, an improved GA (or a kind of hybrid GA) can be introduced which works more efficiently while doesn't use any quantum concepts.

This paper is organized as follows. Section 2 describes the QGA and its structural analysis. Section 3 introduces the HGA and its structure. Section 4 contains a description of the experiment with conventional GA, QGA, and HGA and the experimental results. Concluding remarks follow in Section 5.

2 Quantum Genetic Algorithm

QGA is based on the concepts of qubits and superposition of states of quantum mechanics.

2.1 Qubit

The smallest unit of information stored in a quantum computer is called a quantum bit or qubit [5].A qubit may be in the '1' state, in the '0' state or in any superposition of the two. The state of a qubit can be represented as

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle \qquad (1)$$

$|\alpha|^2$ gives the probability that the qubit will found in '0' state and $|\beta|^2$ gives the probability that the qubit will found in the '1' state.

If there is a system of m-qubits (quantum register) the system can represent 2^m states at the same time.

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \dots & \alpha_m \\ \beta_1 & \beta_2 & \dots & \beta_m \end{bmatrix}, \qquad (2)$$

is a m-qubit qregister. When the qregister is observed (i.e. measured) only one state can be seen. This is the "collapse" of the superposition [1] [2].

QGA uses the qegisters as chromosomes. This enables the chromosome to store exponentially more data than a classical chromosome of the same size.

2.2 Analysis of the QGA

The major advantage for a QGA is the increased diversity of a quantum population. A quantum population can be exponentially larger than a classical population of the same size because each quantum individual is a superposition of multiple classical individuals.

Convergence can be obtained with the quantum representation. As $|\alpha|^2$ or $|\beta|^2$ approaches to 1 or 0,the quantum chromosome converges to a single state and the property of diversity disappears gradually. That is, the quantum representation is able to possess the two characteristics of exploration and exploitation, simultaneously.

As QGA has diversity caused by the quantum representation, there is no need to use the genetic operators. However, some genetic operators can be applied, such as mutation that creates new individuals by a small change in a single individual, and crossover that creates new individuals by combining parts from tow or more individuals. Mutation and crossover can make the probability of linear superposition of states change. Thus, if the probabilities of mutation and crossover are high, the performance of QGA can be decreased notably.

2.3 The Structure of QGA

Procedure QGA
begin
t←0
initialize Q(t)
make P(t) by observing Q(t) states
evaluate P(t)
store the best solution among P(t)
while (**not** termination-condition) **do**
begin
t ← t+1
make P(t) by observing Q(t-1) states
evaluate P(t)
update Q(t) using quantum gatesU(t)
store the best solution among P(t)
end
end

QGA maintains a population of qubit chromosomes, $Q(t) = \{\mathbf{q}_1^t, \mathbf{q}_2^t, \ldots, \mathbf{q}_n^t\}$ at generation *t*, where n is the size of population, and $\mathbf{q}_i^t$ is a qubit chromosome defined as

$$\begin{bmatrix} \alpha^t_1 & \alpha^t_2 & \cdots & \alpha^t_m \\ \beta^t_1 & \beta^t_2 & \cdots & \beta^t_m \end{bmatrix}, \qquad (3)$$

where m is the number of qubits, i.e., the string length of the qubit chromosome, and *j=1,2,…,n*.

In 'initialize Q (t)' step we initialize all probability amplitudes of all chromosomes with $1/\sqrt{2}$ to have the linear superposition of all possible states with the same probability:

$$\left|\psi_{q_j^0}\right\rangle = \sum_{k=1}^{2^m} \frac{1}{\sqrt{2^m}} \left|S_k\right\rangle \qquad (4)$$

where S_k is the *k*-th state represented by the binary string $\mathbf{x}=(x_1 \ldots x_m)$,where x_i ,*i=1,2,…,m*, is either 0 or 1.

In the step of 'make P(t)' , a set of binary solutions, $P(t)= \{\mathbf{x}_1^t, \mathbf{x}_2^t, \ldots, \mathbf{x}_n^t\}$, is made by observing Q (t). states. $\mathbf{x}_j^t$ is a binary solution string of length m ,and is formed by selecting each bit using the probability of qubit, either $|\alpha_i|^2$ or $|\beta_i|^2$ of $\mathbf{q}_j^t$ .Then each solution $\mathbf{x}_j^t$ is evaluated to give some measure of its fitness. The initial best solution is then selected and stored among the binary solutions, P(t).

The step of 'update Q (t)' is added in the **while** loop to have fitter states of the qubit chromosome. In this step, a set of qubit chromosome Q(t) is updated by applying some appropriate quantum gate U(t) ,which is formed by using the binary solutions P(t) and the stored best solution. The appropriate quantum gates can be designed in compliance with practical problems. Some Q-gates are NOT gate, controlled NOT gate, or Hadamard gate and Rotation gate. NOT gate changes the probability of the 1(or0) state to that of the 0 (or1) state .It can be used to escape a local optimum. In controlled NOT gate, one of the two bits should be a control bit. If the control bit is 1, the NOT operation is applied to the other bit. It can be used for the problems that have a large dependency of two bits. The Hadamard gate is suitable for the algorithms which use the phase information of Q-bit, as well as the amplitude information [4] [2]. The Rotation gate is used as a Q-gate in QGA such as

$$U(\Delta\theta_i) = \begin{bmatrix} \cos(\Delta\theta_i) & -\sin(\Delta\theta_i) \\ \sin(\Delta\theta_i) & \cos(\Delta\theta_i) \end{bmatrix} \qquad (5)$$

where $\Delta\theta_i$, $i=1,2,...,m$, is a rotation angle of each Q-bit toward either 0 or 1 state depending on its sign as shown in figure 1.

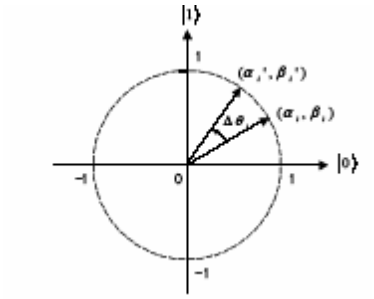

Fig. 1.Polar plot of the rotation gate

3 Hybrid Genetic Algorithm

The global search performs well in QGA due to its probabilistic characteristics. Also, this algorithm uses the rotation gate to update the population in order to search solutions near the optimum (local search).

A kind of "Hybrid Genetic Algorithm" can be developed to have these two benefits without using quantum concepts. Instead of stochastic selection, the fittest individual, the HGA, shares its genetic information with all others using simple GA operators. For this purpose, a crossover operation between the individuals of a generation and the best individuals of the previous generations is applied. So, the new individuals converge to fitter individuals, just like the rotation gate operation in update procedure of the QGA.

3.1 The Structure of HGA

Procedure HGA
begin
t←*0*
initialize P(t)
evaluate P(t)
store the best solution among P(t)
while (**not** termination-condition) **do**
begin
t ← *t+1*
evaluate P(t)
select an individual randomly ch_i
update P(t) using operations
store the best solution among P(t)
end
end

where

Procedure update (P)
begin
uniform **crossover** between the selected individual (ch_i) and the best individual of the previous generations
mutation
end

4 Experiment

The knapsack problem is considered to compare three types of genetic algorithm described above. It can be described as selecting from among various items those items, which are most profitable, given that the knapsack has limited capacity. The 0-1 knapsack problem is described as: given a set of m items and a knapsack, select a subset of the items so as to maximize the profit f(x):

$$f(x) = \sum_{i=1}^{m} p_i x_i , \qquad (6)$$

subject to

$$\sum_{i=1}^{m} w_i x_i \le C \qquad (7)$$

where $\mathbf{x}=(x_1 \ldots x_m)$, x_i is 0 or 1,p_i is the profit of item i ,w_i is the weight of item i, and C is the capacity of the knapsack.

4.1 Conventional GA for the knapsack problem

Three types of conventional algorithms (algorithms based on decoders, algorithms based on penalty functions, and algorithms based on repair methods) are described and tested in [2] [10]. The most successful method was a combination of the algorithm based on linear penalty function with the algorithm based on random repair method [2]. So, we use this combined algorithm for the experiment.

4.2 QGA for the knapsack problem

The algorithm of QGA for the knapsack problem is based on the structure of QGA proposed in Sec2.3 and it contains a *repair* procedure that performs after each make procedure in the QGA procedure.
The make, repair, and update procedures are as follows:

Procedure make ($x_i^{\,t}$)
begin
i ← *0*
while (*i* < *m*) do
begin
i ←*i+1*
if *random [0,1]* > $|\alpha_i|2$
then $x_i \leftarrow 1$
else $x_i \leftarrow 0$

end
end

The binary string $\mathbf{x}_i^t$,$j=1,2,...,n$,of P(t) represents a *j*-th solution to the problem. For every bit in the binary string, we generate a random number r from the rang [0, 1]; if r > $|\alpha_i|^2$, we set the bit of the binary string. The *i*-th item is selected for the knapsack iff x_i =1,where x_i is the *i*-th bit of $x_j(t)$.

Procedure repair (x)
begin
knapsack-overfilled ← false
if $\sum_{i=1}^{m} w_i x_i > C$
then knapsack-overfilled← true
while (knapsack-overfilled) **do**
begin
select an i-th item from the knapsack
$x_i \leftarrow 0$
if $\sum_{i=1}^{m} w_i x_i \le C$
then knapsack-overfilled← false
end
while (not knapsack-overfilled) **do**
begin
select an *j*-th item from the knapsack
$x_j \leftarrow 1$
if $\sum_{i=1}^{m} w_i x_i > C$
then knapsack-overfilled ← true
end
$x_j \leftarrow 0$
end

The **repair** procedure is performed to assure that the capacity constraint is satisfied.
Procedure update (q)
begin
$i \leftarrow 0$
while ($i<m$) do
begin
$i \leftarrow i+1$
determine θ_i with the LUT
obtain (α'_i, β'_i) as:
$[\alpha'_i \ \beta'_i]^T = U(\theta_i)[\alpha_i \ \beta_i]^T$
end
$q \leftarrow q'$
end
In this procedure the *i*-th qubit value (α_i ,β_i) is updated as

$$\begin{bmatrix} \alpha_i' \\ \beta_i' \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta_i) & -\sin(\Delta\theta_i) \\ \sin(\Delta\theta_i) & \cos(\Delta\theta_i) \end{bmatrix} \begin{bmatrix} \alpha_i \\ \beta_i \end{bmatrix} \qquad (8)$$

In the knapsack problem θ_i is given as
sign(α_i β_i)*$\Delta\theta_i$ .The parameters used are shown in Table 1[2] [4]. The magnitude of $\Delta\theta_i$ has an effect on the speed of convergence, but if it is too big the solutions may diverge or converge prematurely to a local optimum. Thus the lookup table can be used as a strategy for convergence. The sign of $\Delta\theta_i$ determines the direction of convergence.

x_i	b_i	$f(\mathbf{x}) \ge f(\mathbf{b})$	$\Delta\theta_i$	$s(\alpha_i\beta_i)$			
				$\alpha_i\beta_i > 0$	$\alpha_i\beta_i < 0$	$\alpha_i = 0$	$\beta_i = 0$
0	0	*false*	0	0	0	0	0
0	0	*true*	0	0	0	0	0
0	1	*false*	0	0	0	0	0
0	1	*true*	0.05π	-1	$+1$	±1	0
1	0	*false*	0.01π	-1	$+1$	±1	0
1	0	*true*	0.025π	$+1$	-1	0	±1
1	1	*false*	0.005π	$+1$	-1	0	±1
1	1	*true*	0.025π	$+1$	-1	0	±1

Table 1: Lookup table of θ_i, where s (α_i β_i) is the sign of θi and b_i and x_i are the *i*-th bits of the best solution **b** and the binary solution **x**, respectively.

4.3 HGA for the knapsack problem

HGA for the knapsack problem can be obtain with the small changes in conventional GA for solving the knapsack problem. Instead of applying crossover between the two selected individuals of a generation, crossover is made between a selected individual of a generation and the best individual of the previous generations (the best individual over previous generations is saved by elitism).

Procedure HGA
begin
$t \leftarrow 0$
initialize P(t)
repair P(t)
evaluate P(t)
store the best solution among P(t)
while (**not** termination-condition) **do**
begin
$t \leftarrow t+1$
select an individual randomly ch_i
update P(t) using operations
repair P(t)
evaluate P(t)
store the best solution among P(t)
end
end

4.4 Experimental Results

As a performance measure of the HGA, we used the problem specifications were used in previous

researches and collected the best solution found within 500 generations over 25 runs. The data used in the experiments were as follows:
The weights of items :w_i=*uniformly random*[1,10)
The profits of the items: $p_i = w_i + 5$,
The average knapsack capacity:

$$C = \frac{1}{2}\sum_{i=1}^{m} w_i \qquad (9)$$

The population size of CGA and HGA was equal to 100.Probabilities of crossover and mutation were 0.65 and 0.005 respectively. The population size of QGA was 10.We wrote the program codes in Matlab™.
Table 2 shows the experimental results of the knapsack problems with 100,250 and 500 items. In all cases HGA yielded superior results as compared to all the other algorithms.

#of items	100	250	500
CGA			
best	**574.20**	**1324.7**	**2683.0**
mean	**549.49**	**1310.4**	**2644.3**
worst	**530.19**	**1297.8**	**2598.4**
QGA			
best	**567.93**	**1329.7**	**2698.0**
mean	**538.89**	**1315.5**	**2650.5**
worst	**515.36**	**1301.1**	**2600.1**
HGA			
best	**608.17**	**1465.2**	**2948.0**
mean	**594.17**	**1445.2**	**2930.3**
worst	**580.90**	**1424.3**	**2882.2**

5 Conclusions

This paper presented a hybrid genetic algorithm, HGA that uses genetic operations effectively, without using quantum features to solve combinatorial optimization problems. HGA employs crossover to simulate the update procedure in QGA. The knapsack problem, a kind of combinatorial optimization problems, is used to discuss the performance of HGA, compare to CGA and QGA. It was shown that HGA reaches higher amount of profit than the two other algorithms at the same generation.

References

[1] J.Foster, B.Rylander et al., Quantum Evolutionary Programming

[2] K.H.Han and J.H.Kim, "Genetic quantum algorithm and its application to combinatorial optimization problem,"inProc.2000CongressonEvolutionary Computation.Piscataway, NJ: IEEEPress, July2000, vol.2, pp.1354–1360.

[3] A.Narayanan and M.Moore, "Quantum-inspired genetic algorithms," inProc.1996 IEEE Int. Conf. Evolutionary Computation. Piscataway, NJ: IEEE Press, May1996, pp.61–66.

[4] K. -H. Han and J. -H. Kim, "Quantum-inspired Evolutionary Algorithm for a Class of Combinatorial Optimization," IEEE Transactions on Evolutionary Computation, Piscataway, NJ: IEEE Press, vol.6, no.6, pp.580-593, Dec. 2002.

[5] K. -H. Han and J. -H. Kim, Quantum-Inspired Evolutionary Algorithms with a New Termination Criterion, H_ Gate, and Two-Phase Scheme, IEEE Transactions on Evolutionary Computations, vol. 8, no. 2, April 2004