Comparison of Genetic Algorithm and Quantum Genetic Algorithm

Zakaria Laboudi and Salim Chikhi
SCAL Group of the MISC Laboratory, University Mentouri, Algeria

Abstract: *Evolving solutions rather than computing them certainly represents a promising programming approach. Evolutionary computation has already been known in computer science since more than 4 decades. More recently, another alternative of evolutionary algorithms was invented: Quantum Genetic Algorithms (QGA). In this paper, we outline the approach of QGA by giving a comparison with Conventional Genetic Algorithm (CGA). Our results have shown that QGA can be a very promising tool for exploring search spaces.*

Keywords: *Genetic algorithm, knapsack problem, quantum genetic algorithm, quantum computing.*

Received October 18, 2009; accepted May 20, 2010

1. Introduction

Genetic Algorithms (GA) are a representative example of a set of methods known as evolutionary algorithms. This approach started in the 1970s by John Holland, and knew for a decade strong growth. A GA is an iterative algorithm based on the notion of generation, but it is also inherently highly parallel in that it simulates the evolution of a range of solutions.

Quantum computation is a newly emerging interdisciplinary science of information science and quantum science. The first quantum algorithm was proposed by Shor [10], for number factorization. Grover [2] also proposed a quantum algorithm for random search in databases, the complexity of its algorithm was reduced to be of the order of (N^½). More recently, quantum computation has attracted a wide attention, and it becomes a very interest research field.

QGA is a combination of GA and quantum computing. There were some efforts to use QGA for exploring search spaces; we quote for example [3] where a QGA was used to solve the knapsack problem [1], where a quantum-inspired differential evolution algorithm was proposed to solve the N-queens problem and [11] who proposed a parallel version of QGA. In [7], a QGA was also used to solve the binary decision diagram ordering problem. More recently, QGA where used to evolve Cellular Automata rules (CA) [5, 6] to solve the density classification problem.

In this work, we propose to make a comparison between GA and QGA to extract some computational abilities of QGA to perform processing in an effective and an efficient manner. We have considered the classic 0/1 knapsack problem. Indeed, it existed such work in the literature [4, 11]. But in our case, the 0/1 knapsack problem was tackled from several sides and with more details.

This paper is organized as follows. Section 2 describes some conventional GA principles. A description of the basic concept of quantum computing and QGA principles is presented in section 3. Section 4 tackles the 0/1 knapsack problem and some conventional GA solving methods. In section 5 we summarize and analyze the experimental results. We finish our paper by concluding remarks follow and some perspectives in section 6.

2. Conventional Genetic Algorithm (CGA)

GA is an exploration algorithm based on genetic evolution and natural selection. It manipulates a population of individuals called chromosomes. At each time step a new generation is constructed by applying genetic operators between some selected chromosomes. The structure of a CGA is illustrated in Figure 1. The simplest way for coding chromosomes is to represent them by binary strings. The initial population has to start with random chromosomes uniformly distributed over the entire search space. The next step is the evaluation operation. Its role is to mark the individuals of the population. After that, the individuals will be sorted according to their marks. The selection operation has as goal to elect some number of individuals to enable reproduction. The cross-over operation can be performed by exchanging some parts of selected individuals in random positions which leads to create a new set of chromosomes replacing the old ones. Before repeating the process, it is recommended to perform a mutation to correct stochastic errors to avoid a genetic drift and to ensure a genetic diversity in the population. It consists of changing some random

positions of the individuals according to a small probability (typically between 1% and 0.1%).

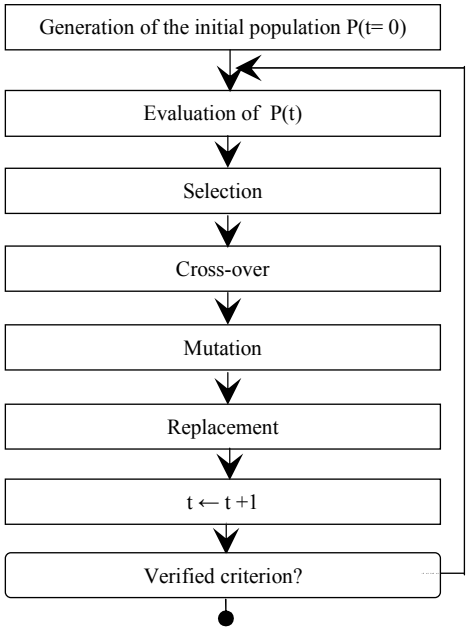

Figure 1. CGA structure.

3. Quantum Genetic Algorithm (QGA)

3.1. Quantum Computing

In quantum computing, the smallest unit of information storage is the quantum bit (qubit) [3]. A qubit can be in the state 1, in the state 0 or in a superposition of both. The state of a qubit can be represented as [3]:

$$|\Psi> = \alpha\,|0> + \beta\,|1> \quad (1)$$

Where $|0>$ and $|1>$ represent the values of classical bits *0* and *1* respectively, α and β are complex numbers satisfying:

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

$|\alpha|^2$ is the probability where a qubit is in state *0* and $|\beta|^2$ represents the probability where a qubit is in state 1. A quantum register of m qubits can represent 2^m values simultaneously. However, when the 'measure' is taken, the superposition is destroyed and only one of the values becomes available for use. That's why we think that quantum computers can be used mainly in applications involving a choice among a multitude of alternatives.

In general, a quantum algorithm has less complexity than its classic equivalent algorithm through the concept of quantum superposition. Among the most famous quantum algorithms we quote Shore's algorithm for number factorization [10] and Grover's algorithm for research in a non sorted database [2]. Both algorithms have solved problems which their complexity was reduced.

3.2. QGA Principles

QGAs are a combination between GA and quantum computing. They are mainly based on qubits and states superposition of quantum mechanics. Unlike the classical representation of chromosomes (binary string for instance), here they are represented by vectors of qubits (quantum register). Thus, a chromosome can represent the superposition of all possible states. The structure of a QGA is illustrated in Figure 2 [3]:

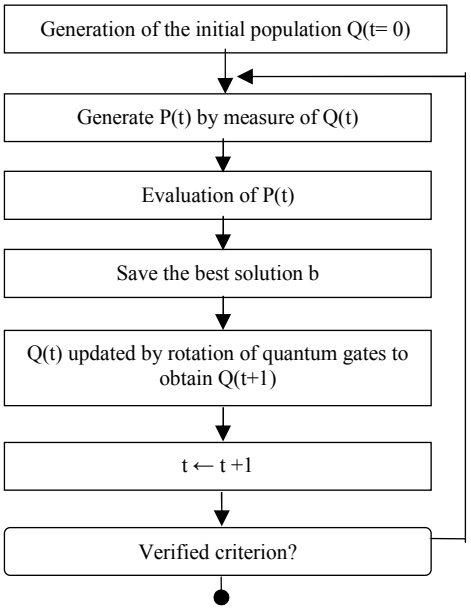

Figure 2. QGA structure.

3.2.1. Structure of Quantum Chromosomes

A chromosome is simply a string of m qubits that forms a quantum register. Figure 3 shows the structure of a quantum chromosome.

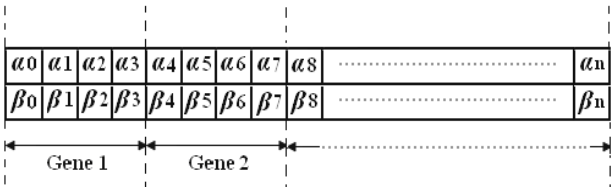

Figure 3. Quantum chromosome structure.

3.2.2. Initializing the Population

The easiest way to create the initial population is to initialize all the amplitudes of qubits by the value *1/(2^ ½)* [3]. This means that a chromosome represents all quantum superposition states with equal probability.

3.2.3. Evaluation of Individuals

The role of this phase is quantifying the quality of each quantum chromosome in the population to make a reproduction. The evaluation is based on an objective function that corresponds to each individual, after measuring, an adaptation value. It permits to mark individuals in the population.

3.2.4. Quantum Genetic Operations

1. *Measuring Chromosomes*: In order to exploit effectively superposed states of qubits, we have to observe each qubit. This leads us to extract a classic chromosome as illustrated in Figure 4. The aim is to enable the evaluation of each quantum chromosome.

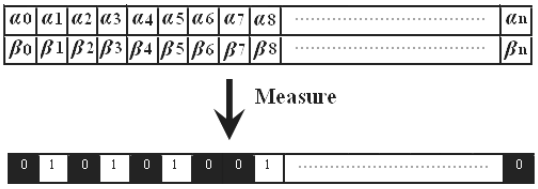

Figure 4. Measured chromosome.

A simple way to implement this function is given by the following pseudo code:

```
Function measure ()
begin
        r := get r in [0,1] ;
        if (r > α²)
                return 1 ;
          else
        return 0 ;
        end if
end
```

2. *Interference*: This operation allows modifying the amplitudes of individuals in order to improve performance. It mainly consists of moving the state of each qubit in the sense of the value of the best solution. This is useful for intensifying the search around the best solution. It can be performed using a unit transformation that allows a rotation whose angle is a function of the amplitudes (ai, bi) and the value of the corresponding bit in the reference solution. The value of the rotation angle δθ has to be chosen so that to avoid premature convergence. It is often empirically determined and its direction is determined as a function of the values of ai, bi and the value of the qubit located at the position i in the individual being modified [7].
3. *Qubit Rotation Gates Strategy*: The rotation of individual's amplitudes is performed by quantum gates. Quantum gates can also be designed in accordance with the present problem. The population Q(t) is updated with a quantum gates rotation of qubits constituting individuals. The rotation strategy adopted is given by the following equation:

$$\begin{bmatrix} \alpha_i^{t+1} \\ \beta_i^{t+1} \end{bmatrix} = \begin{bmatrix} cos(\Delta\theta_i) & -sin(\Delta\theta_i) \\ sin(\Delta\theta_i) & cos(\Delta\theta_i) \end{bmatrix} \begin{bmatrix} \alpha_i^{t} \\ \beta_i^{t} \end{bmatrix} \quad (3)$$

Where $\Delta\theta i$ is the rotation angle of qubit quantum gate i of each quantum chromosome. It is often obtained from a lookup table to ensure convergence.

4. Knapsack Problem

The knapsack problem is one the most combinatorial algorithms. The knapsack problem can be described as selecting from among various items those which are most profitable, given that the knapsack has a limited capacity. There are many types of knapsack problem, so the simplest one is called 0/1 knapsack problem. It 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) [3] as shown in equation 4:

$$f(x) = \sum_{i=1}^{m} p_i x_i \quad (4)$$

Subject to:

$$f(x) = \sum_{i=1}^{m} w_i x_i \le C \quad (5)$$

Where $x = (x_1, x_2, ..., x_m)$, x_i is 0 or 1, p_i and w_i are the profit and the weight of the i^{th} item. C is the capacity of the knapsack.

4.1. Choosing Parameters Values

Since it was found that the difficulty of such problem is greatly affected by the correlation between profits and weights [8], three randomly generated sets of data are considered [8]:

1. *Uncorrelated:*
   w_i = (uniformly) random([1..v])
   p_i = (uniformly) random([1..v])
2. *Weakly correlated:*
   w_i = (uniformly) random([1..v])
   p_i = w_i + (uniformly) random([-r..+r])
3. *Strongly correlated:*
   w_i = (uniformly) random([1..v])
   p_i = $w_i + r$

Higher correlation implies smaller value of the difference:

$$max_{i=1..m}\{p_i / w_i\} - min_{i=1..m}\{p_i / w_i\};$$

As reported in [8], higher correlation problems have higher expected difficulty. The knapsack capacity can be set according two types (again, following a suggestion from [8]):

1. Restrictive knapsack capacity (C1): The knapsack capacity $C=2v$. In this case, the optimal solution contains very few items [8].
2. Average knapsack capacity (C2): The knapsack capacity is determined as shown in equation 6:

$$f(x) = \frac{1}{2} \sum_{i=1}^{m} w_i \quad (6)$$

In this case, the optimal solution contains about half of the items [8].

4.2. Solving Knapsack Problem with Conventional GA (CGA)

There are three types of conventional GA algorithms [8]: algorithms based on penalty functions, algorithms based on repair methods and algorithms based on decoders.

In algorithms based on penalty functions, each solution is coded as a binary string of the length *m* representing a chromosome *x* to the problem. The profit *f(x)* of each chromosome is computed as shown in equation 7:

$$f(x) = \sum_{i=1}^{m} p_i x_i - pen(x) \quad (7)$$

Where *Pen(x)* is a penalty function. We consider here the three types of penalties discussed in [8]: logarithmic penalty, linear penalty, and quadratic penalty:

$$\begin{aligned} pen\ 1(x) &= \log_2\left(1 + \rho\left(\sum_{i=1}^{m} p_i x_i - C\right)\right) \\ pen\ 2(x) &= \rho\left(\sum_{i=1}^{m} p_i x_i - C\right) \\ pen\ 3(x) &= \left(1 + \rho\left(\sum_{i=1}^{m} p_i x_i - C\right)\right)^2 \end{aligned} \quad (8)$$

Where ρ is $max_{i=1..m}\{p_i / w_i\}$. The penalty function *Pen(x)* is zero for all feasible solutions *x*, i.e., solutions that:

$$f(x) = \sum_{i=1}^{m} p_i x_i \quad (9)$$

And greater than zero otherwise. In algorithms based on repair methods, the profit *f(x)* of each chromosome is computed as shown in equation 10:

$$f(x) = \sum_{i=1}^{m} p_i x' \quad (10)$$

Where x' is a repaired vector of the original vector x. Original chromosomes are replaced with a 5% probability in the experiment (it has been proved that that 5% is the most appropriate replacement percentage). We have used the two repair algorithms mentioned in [8]. The repair algorithms differ only in selection procedure, which chooses an item for removal from the knapsack:

1. *Random Repair (Rep1)*: In this case, a random element is removed from the knapsack.
2. *Greedy Repair (Rep2)*: All items in the knapsack are sorted in the decreasing order of their profit to weight ratios. The last item is always chosen for deletion.

We will not tackle here the third method (decoder based algorithms), because the chromosome representation is based on integers while quantum chromosomes can use only qubit representation.

5. Experimental Results and Discussion

In all our experiments we have coded solutions as binary strings of the length m for CGA and as qubit strings of the length m for QGA. The length of both strings is the same as the number of items. For CGA the i^{th} item is added to the knapsack if and only if the i^{th} element in the binary string is 1. Similarly for QGA, the i^{th} item is added to the knapsack with a probability of $|\beta_i|^2$ where $|\beta_i|$ is the amplitude of the i^{th} qubit in the qubit string. Before presenting our experimental results, we will announce the parameters of both algorithms and knapsack problem parameters.

5.1. CGA Parameters

The parameter values were chosen according to the most values found in the literature as mentioned in Table 1 (for instance Mitchell and al [9] have used the same parameter values to evolve cellular automata rules by CGA).

Table 1. List of CGA parameters.

Parameter	Value
Cross-over probability	50 %
Mutation probability	1 %
Population size	100

5.2. QGA Parameters

The population size was fixed to be 100. The amplitudes of the individuals are updated by a rotation of quantum gates according to the look-up Table 2.

Table 2. Look-up table for quantum gates rotation.

xi bi	f (x) > f(b)	Δθi	s (ai bi)			
			ai.bi > 0	ai.bi < 0	ai= 0	bi= 0
0 0	0	0.001π	-	+	±	±
0 0	1	0.001π	-	+	±	±
0 1	0	0.08π	-	+	±	±
0 1	1	0.001π	-	+	±	±
1 0	0	0.08π	+	-	±	±
1 0	1	0.001π	+	-	±	±
1 1	0	0.001π	+	-	±	±
1 1	1	0.001π	+	-	±	±

xi and *bi* are the *i-th* bits of *x* and *b* (the best solution), respectively. f is the fitness function and s (*ai bi*) is the sign of the rotation angle θi. According to the lookup table, one can easily remark that this strategy improves, for each individual, the amplitudes of qubits that are bad according to an angle $\delta\theta 1=0.08\pi$ while it decreases, those that are good according to an angle $\delta\theta 2=0.001\pi$. The amplitude values were empirically determined (following a suggestion from [7]). The modification of the amplitudes of qubits is done according to the signs of the amplitudes, the best solution and the solution extracted by the individual container. It is natural that $\delta\theta 1>>\delta\theta 2$ because decreasing amplitudes serves only to correct stochastic errors to avoid a genetic drift and to ensure a genetic diversity in the population.

For the used Knapsack problem, the parameters' values were set as shown in table 3.

Table 3. List of knapsack problem parameters.

Parameter	Value
V	10
R	5
Repairing replacement percentage	5%

5.3. Experimental Results

We have executed both algorithms (CGA and QGA) over 25 runs for all possible cases, at each one the concerned algorithm was iterated for a maximum number=500 generations. Table 4 shows the experimental results of the knapsack problem with 100, 250 and 500 items. We have found that executing both algorithms more than 25 runs doesn't make difference.

Table 4. Experimental results of the knapsack problem.

Correl	No. of Items	Capacity Type		Method									
				CGA					QGA				
				Pen1	Pen2	Pen3	Rep1	Rep2	Pen1	Pen2	Pen3	Rep1	Rep2
None	100	C1	m	-	*	*	70.9	71.6	-	*	*	70.5	71.7
			b	-	*	*	96.5	88.4	-	*	*	96.6	88.4
			w	-	*	*	54.3	59.4	-	*	*	53.9	60.5
		C2	m	-	401.9	403.2	401.9	408.2	-	403.6	404.6	402.6	408.1
			b	-	434.5	434.8	441.1	451.0	-	436.5	433.9	443.0	450.6
			w	-	364.5	361.4	347.9	371.6	-	367.6	404.6	376.5	371.2
	250	C1	m	-	*	*	-	-	-	*	*	-	-
			b	-	*	*	-	-	-	*	*	-	-
			w	-	*	*	-	-	-	*	*	-	-
		C2	m	-	968.5	971.0	987.8	1041.7	-	1017.0	1025.4	1025.1	1045.7
			b	-	1025.4	1008.8	1048.2	1090.8	-	1097.6	1068.6	1102.7	1098.5
			w	-	907.3	937.9	947.2	987.2	-	946.7	981.6	980.2	990.5
	500	C1	m	-	*	*	-	-	-	*	*	-	-
			b	-	*	*	-	-	-	*	*	-	-
			w	-	*	*	-	-	-	*	*	-	-
		C2	m	-	1835.9	1831.8	-	-	-	2000.6	2000.5	-	-
			b	-	1900.0	1889.0	-	-	-	2066.7	2073.9	-	-
			w	-	1775.5	1759.3	-	-	-	1895.2	1863.2	-	-
Weak	100	C1	m	-	*	*	43.4	42.9	-	*	*	42.3	43.0
			b	-	*	*	57.7	52.8	-	*	*	57.7	52.8
			w	-	*	*	35.8	35.0	-	*	*	35.4	35.0
		C2	m	-	397.4	393.7	399.6	396.2	-	398.2	396.4	399.0	396.3
			b	-	430.4	418.6	437.0	427.5	-	431.8	420.9	436.0	427.6
			w	-	341.7	374.7	370.8	367.2	-	342.6	375.7	396.3	367.3
	250	C1	m	-	*	*	40.8	67.1	-	*	*	41.0	67.1
			b	-	*	*	46.6	75.8	-	*	*	46.5	75.8
			w	-	*	*	36.1	52.9	-	*	*	35.9	52.9
		C2	m	-	943.5	918.3	950.9	999.4	-	997.7	978.2	987.1	999.5
			b	-	999.0	988.9	1005.1	1037.7	-	1056.0	1041.7	1042.8	1038.0
			w	-	866.2	873.1	916.1	944.6	-	919.0	928.0	948.5	944.6
	500	C1	m	-	*	*	-	-	-	*	*	-	-
			b	-	*	*	-	-	-	*	*	-	-
			w	-	*	*	-	-	-	*	*	-	-
		C2	m	-	1764.4	1740.7	1812.4	1957.9	-	1941.4	1928.3	1952.8	1985.5
			b	-	1847.0	1789.6	1880.3	2055.6	-	2008.8	1984.0	2032.1	2087.7
			w	-	1879.3	1692.2	1736.8	1878.4	-	1850.9	1875.7	1876.3	1900.8
Strong	100	C1	m	-	*	*	81.1	79.6	-	*	*	81.2	79.4
			b	-	*	*	89.9	95.0	-	*	*	90.0	95.0
			w	-	*	*	74.8	65.0	-	*	*	75.0	65.0
		C2	m	-	605.7	605.0	609.1	612.7	-	607.8	609.3	609.0	611.8
			b	-	623.4	622.6	625.2	628.7	-	623.4	623.0	625.2	628.7
			w	-	592.4	591.3	593.8	592.6	-	597.4	594.7	593.8	592.6
	250	C1	m	-	*	*	71.7	92.7	-	*	*	69.3	93.3
			b	-	*	*	75.0	95.0	-	*	*	74.9	100.0
			w	-	*	*	65.0	90.0	-	*	*	64.3	90.0
		C2	m	-	1471.0	1461.0	1493.8	1525.9	-	1523.2	1523.0	1527.6	1532.0
			b	-	1492.9	1488.5	1530.7	1545.1	-	1549.4	1544.4	1565.8	1550.2
			w	-	1452.4	1441.8	1453.9	1499.8	-	1504.2	1503.5	1491.1	1500.9
	500	C1	m	-	*	*	68.1	101.6	-	*	*	67.3	102.2
			b	-	*	*	74.9	105.0	-	*	*	74.2	105.0
			w	-	*	*	64.6	95.0	-	*	*	59.6	95.0
		C2	m	-	2862.8	2848.1	2920.9	3001.6	-	3004.7	3003.0	3029.9	3059.9
			b	-	2911.8	2880.8	2972.7	3031.5	-	3032.3	3036.5	3078.5	3091.5
			w	-	2833.1	2802.8	2875.8	2962.7	-	2981.0	2963.2	2988.0	3023.0

b, m, and w mean best, mean, and worst, respectively.
'-' means that an experiment did not made in this case.
'*' means that no valid solution has been found within given constraints.

5.4. Discussion

Table 2 shows the experimental results of the knapsack problem where the number of items was 100, 250 and 500. The hardware and software configuration was as follows: Intel Pentium 4 (3.4 GHz), 1 Go MB of memory, Windows XP OS, Java Programming language (JDK 1.5.0).

In the case of 100 items, both algorithms were equivalent for all problem instances. This because the number of items is so small (we have 2^100 possibility). However, augmenting the number of items (500 items where we have 2^500 possibility) leads QGA to behave better than CGA, and this in all problem solution variants.

By using repairing methods, Table 2 shows that for a small number of items the profits are approximately close, with some preference for QGA especially in the case of Rep2 where elements are selectively removed from the knapsack, in contrast to Rep1 where elements are randomly removed. In fact, using repairing methods influences directly the evolution process. For instance, we can easily compare the best results for each variant of the problem. By observing penalty methods, we can then judge that QGA's performance is higher than the one of CGA because we have only used an evolution process.

Moreover, quantum algorithms have generally the ability to minimize the complexity of equivalent algorithms that run on classic computers. We can make a simple comparison between the global complexity of QGA and the one of GA to estimate the reduction in complexity we can achieve. Starting with QGA, the global complexity is of the order of O(N), N is the size of the population (Evaluation + Interference).

For a standard GA, the global complexity is often of the order of $O(N^2)$ (Evaluation + Selection + Cross-over + Mutation). Therefore, we believe that this result is very interesting because the complexity here has been reduced to become linear. Indeed, one can imagine what happen if we consider a very large population of chromosomes; it will be very useful to use QGA instead of GA.

6. Conclusions

In this study we have made a comparison between two optimization techniques: QGA and CGA. The first one is based on quantum computing principles such as concepts of qubits and superposition of states. The second is based on based on Neo-Darwinism (genetic evolution and natural selection). Our experimental results have shown that QGA can be a very promising tool for exploring large search spaces while preserving the relation efficiency / performance. Our future work will focus on comparing different QGA strategies to study the effect of choosing rotation gate angles. Another perspective of this work is to study parallel QGA because QGA are highly parallelizable.

References

[1] Draa A., Meshoul S., Talbi H., and Batouche M., "A Quantum-Inspired Differential Evolution Algorithm for Solving the N-Queens Problem," *The International Arab Journal of Information Technology*, vol. 7, no. 1, pp. 21-27, 2010.

[2] Grover L., "A Fast Quantum Mechanical Algorithm for Database Search," *in Proceedings of 28th Annual ACM Symposium on the Theory of Computing*, USA, pp. 212-221, 1996.

[3] Han K., "Genetic Quantum Algorithm and Its Application to Combinatorial Optimization Problem," *in Proceedings of IEEE Congress on Evolutionary Computation*, USA, pp. 1354-1360, 2000.

[4] Han K., Park K., Lee C., and Kim J., "Parallel Quantum-Inspired Genetic Algorithm for Combinatorial Optimization Problem," *in Proceedings of IEEE Congress of Evolutionary Computation*, South Korea, pp. 1422-1429, 2001.

[5] Laboudi Z. and Chikhi S., "Evolution d'Automates Cellulaires par Algorithmes Génétiques Quantiques," *in Proceedings of Conférence Internationale sur l'Informatique et ses Applications*, Algérie, pp. 1-11, 2009.

[6] Laboudi Z. and Chikhi S., "Evolving Cellular Automata by Parallel Genetic Algorithm," *in Proceedings of IEEE Conference on Networked Digital Technologies*, Ostrava, pp. 309-314, 2009.

[7] Layeb A. and Saidouni D., "Quantum Genetic Algorithm for Binary Decision Diagram Ordering Problem," *International Journal of Computer Science and Network Security,* vol. 7 no. 9, pp. 130-135, 2007.

[8] Michalewicz Z., *Genetic Algorithms+Data Structures=Evolution Programs*, Springer-Verlag, 1999.

[9] Mitchell M., Hraber P., and Crutchfield J., "Evolving Cellular Automata to Perform Computation: Mechanisms and Impediments," *Journal of Physica D: Lonelier Phenomena*, vol. 75, no. 1-3, pp. 361-391, 1994.

[10] Shor P., "Algorithms for Quantum Computation: Discrete Logarithms and Factoring," *in Proceedings of the 35th Annual Symposium on the Foundation of Computer Sciences*, NM, pp. 20-22, 1994.

[11] Shuxia M. and Weidong J., "A New Parallel Quantum Genetic Algorithm with Probability-Gate and Its Probability Analysis," *in Proceedings of International Conference on Intelligent Systems and Knowledge Engineering*, pp. 1-5, 2007.

Zakaria Laboudi is a teacher researcher at Computer Science Department of Larbi Ben M'hidi University, Oum El-Bouaghi – Algeria. Currently, he is a PhD candidate in complex systems field at Mentouri University of Constantine – Algeria. He received his Master's degree in computer science in 2009 from Mentouri University of Constantine – Algeria. In 2010, he joined the SCAL group of the Laboratory of Complex Systems (MISC) as a member of its researcher team. His current research interests include Complex Systems, Artificial Life, Parallel and Distributed Computing, Combinatorial Optimization Problems and Meta-heuristics.

Salim Chikhi received his MSc degree in computer systems from University Mentouri – Constantine-Algeria in collaboration with Glasgow University, UK. He received his PhD degree in computer science from University Mentouri – Constantine – Algeria in 2005. Currently, he is an associate professor at the same University and leads the SCAL group within the MISC laboratory. His research areas include artificial life and soft computing techniques applied to complex systems.