Journal of mathematics and computer science 14 (2015), 183-192

Contents list available at JMCS

Journal of Mathematics and Computer Science

Journal Homepage: www.tjmcs.com

Designing and Comparing Classic Versus Quantum Artificial Bee Colony Algorithm

Kooroush Manochehri[1,*], Amir Alizadegan[1,+]

[1]Department of IT and Computer Engineering, Islamic Azad University (Parand Branch), Parand, Tehran, Iran.

[*]*kmk1381@aut.ac.ir*

[+]*amiralizadegan@yahoo.com*

Article history:
Received November 2014
Accepted December 2014
Available online December 2014

Abstract

Artificial Bee Colony (ABC) algorithm is based on natural behavior of honey bees and has earned good success in optimization area. In this paper a new quantum inspired algorithm that is called Quantum Artificial Bee Colony (QABC) is presented. QABC is a general method and in this work it is adapted to be applied on Knapsack 0-1 problem. In the experiments QABC is compared with classic ABC and the results present robustness of QABC.

Key words:
Optimization- Artificial Bee Colony Algorithm- Quantum Computing- Knapsack 0-1 problem

1. Introduction

Natural inspired algorithms that have been created for optimizing (minimizing or maximizing) problems, obtain their idea from theory of Darvean: “Existence of the stronger and better animals in the nature and so, extinction of weakers”.

Evolutionary Algorithms (EAs) are a set of algorithms that use such a biological evolution theory. The Genetic Algorithm (GA) [1-3], Evolutionary Programming (EP) [4], Evolution strategy (ES) [5], and Genetic programming (GP) [6] are well known and useful methods of EAs.

Later, some populated behavior of animals were concentrated by researchers and caused introducing a new set of optimization algorithms that are known as Swarm Intelligence methods. Ant Colony Optimization [7], Particle Swarm Optimization [8], Differential Evolution [9], Artificial Bee Colony [10-12], and Harmony Search [13] are the most renowned Swarm Intelligence algorithms. These algorithms are based on population of solutions and by means of some techniques like stochastic and mathematic formulas, produce new solutions and maybe based on some conditions (or may not) can be replaced in place of a current solution in population. Because these algorithms are robust, easy to use, and effective in producing good results, they have been used a lot in optimizing various real problems.

Quantum algorithms are known as those that must be implemented on real quantum computers and because of its difficulties, little quantum algorithms have been introduced.

Shor's quantum factoring algorithm [14], and Grover's database search algorithm [١٥], are famous quantum algorithms. But ordinary evolutionary and swarm intelligence algorithms can be inspired by quantum computing concepts and some merging and hybrid approaches have been proposed during last decades [16-19].

In this work, an Artificial Bee Colony (ABC) algorithm that is affected by some principles of quantum concepts, like qubits and superposition of states is presented that is called QABC. For presenting the affectivity, an adapted QABC for knapsack problem is suggested too, and some experiments is done on the knapsack problem. In the following, a brief description of Artificial Bee Colony algorithm, and quantum concepts is come in sections 1.1 and 1.2, respectively. Section 2 presents new Quantum Artificial Bee Colony algorithm and introduce knapsack problem and QABC algorithm for knapsack. Finally the results of various experiments are shown in sections 3.

1.1. Artificial Bee Colony

Artificial Bee Colony (ABC) algorithm was proposed by Karaboga for optimizing numerical problems in 2005 [10]. The algorithm simulates the intelligent foraging behavior of honeybee swarms. It is a very simple, robust and population-based stochastic optimization algorithm. Karaboga and Basturk have compared the performance of the ABC algorithm with those of other well-known modern heuristic algorithms such as Genetic Algorithm (GA), Differential Evolution (DE), Particle Swarm Optimization (PSO) on unconstrained problems [11,12].

Following is the detailed pseudo-code of the ABC algorithm:

1: Initialize the population of solutions x_i, $i = 1..SN$

2: Evaluate the population

3: cycle = 1

4: **repeat**

5: Produce new solutions v_i for the employed bees by using (2) and evaluate them

6: Apply the greedy selection process

7: Calculate the probability values P_i for the solutions x_i by (1)
8: Produce the new solutions v_i for the onlookers from the solutions x_i selected depending on P_i and evaluate them
9: Apply the greedy selection process
10: Determine the abandoned solution for the scout, if exists, and replace it with a new randomly produced solution x_i by (3)
11: Memorize the best solution achieved so far
12: cycle = cycle+1
13: **until** cycle = MCN

In ABC algorithm, the position of a food source represents a possible solution to the optimization problem and the nectar amount of a food source corresponds to the quality (fitness) of the associated solution. The number of the employed bees or the onlooker bees is equal to the number of solutions in the population. At the first step, the ABC generates a randomly distributed initial population $P(G=0)$ of SN solutions (food source positions), where SN denotes the size of the population. Each solution $x_i(i=1,2,...,SN)$ is a D-dimensional vector. Here, D is the number of optimization parameters. After initialization, the population of the positions (solutions) is subjected to repeated cycles, $C=1,2,...,MCN$, of the search processes of the employed bees, the onlooker bees and scout bees. An employed bee produces a modification on the position (solution) in her memory depending on the local information (visual information) and tests the nectar amount (fitness value) of the new source (new solution). Providing the nectar amount of the new one is higher than that of the previous one, the bee memorizes the new position and forgets the old one. Otherwise, she keeps the position of the previous one in her memory. When all employed bees complete the search process, they share the nectar information of the food sources and their position information with the onlooker bees on the dance area. An onlooker bee evaluates the nectar information taken from all employed bees and chooses a food source with a probability related to its nectar amount. As in the case of the employed bee, she produces a modification on the position in her memory and examines the nectar amount of the candidate source. Providing that its nectar is more than the previous one, the bee memorizes the new position and forgets the old one.

An artificial onlooker bee chooses a food source depending on the probability value associated with that food source, P_i, calculated by the following expression (1):

$$P_i = \frac{fit_i}{\sum_{n=1}^{SN} fit_n} \tag{1}$$

where fit_i is the fitness value of the solution i which is proportional to the nectar amount of the food source in the position i, and SN is the number of food sources which is equal to the number of employed bees.

In order to produce a candidate food position from the old one in memory, the ABC uses the following expression (2):

$$v_{ij} = x_{ij} + \Phi_{ij}\left(x_{ij} - x_{kj}\right) \tag{2}$$

where $k \in \{1,2,...,SN\}$ and $j \in \{1,2,...,D\}$ are randomly chosen indices. Although k is determined randomly, it has to be different from i. Φ_{ij} is a random number between [-1, 1]. It controls the production of neighbor food sources around x_{ij} and represents the visual comparison of two food positions by a bee. As can be seen from (2), as the difference between the parameters x_{ij} and x_{kj} decreases, the perturbation on the position x_{ij} decreases, too. Thus, as the search approaches to the optimum solution in the search space, the step length is reduced adaptively.

The food source of which the nectar is abandoned by the bees is replaced with a new food source by the scouts. In ABC, this is simulated by random production of a position and replacing it with the abandoned one. In ABC, providing that a position can`t be improved further through a predetermined number of cycles, then that food source is assumed to be abandoned. The value of predetermined number of cycles is an important control parameter of the ABC algorithm, which is called "limit" for abandonment. Assume that the abandoned source is x_i and $j \in \{1,2,...,D\}$, then the scout discovers a new food source to be replaced with x_i. This operation can be defined as (3):

$$x_i^j = x_{\min}^j + rand(0,1)\left(x_{\max}^j - x_{\min}^j\right) \tag{3}$$

When each candidate source position v_{ij} is produced and evaluated by the artificial bee, its performance is compared with that of its old one. If the new food source has equal or better nectar than the old source, it is replaced with the old one in the memory. Otherwise, the old one is retained in the memory. In other words, a greedy selection mechanism is employed as the selection operation between the old and the candidate one. There are three control parameters in the ABC: The number of food sources which equals to the number of employed or onlooker bees (SN), the value of limit, and the maximum cycle number (MCN).

In a robust search process, exploration and exploitation processes must be carried out together. In the ABC algorithm, while onlookers and employed bees carry out the exploitation process in the search space, the scouts control the exploration process.

1.2. Quantum Computers

The lack of abilities of classic computers for dealing with quantum concepts in physics, caused some extensions in presenting quantum concepts and so, subsequent activities in quantum world. Here some important notions about quantum computers that are important for our work are explained briefly.

The smallest data unit in quantum computer is called quantum bit or qubit. Against classical bit that only can possess either ‘0’or ‘1’, any qubit can be in ‘0’, ‘1’, or any superposition of the two.

The state of a qubit can be represented as

$$|\Psi\rangle = \alpha|0\rangle + \beta|1\rangle \tag{4}$$

Where α and β are complex numbers that specify the probability amplitudes of the corresponding states. $|\alpha|^2$ Gives the probability that the qubit will be found in '0' state and $|\beta|^2$ gives the probability that qubit will be found in the '1' state. Normalization of the state to unity guarantees

$$|\alpha|^2 + |\beta|^2 = 1 \tag{5}$$

One qubit is defined with a pair of complex numbers, α, β , as

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix},$$

Which is characterized by (4) and (5).

An m-qubit representation is defined as

$$\begin{bmatrix} \alpha_1 & \alpha_2 & \cdots & \alpha_m \\ \beta_1 & \beta_2 & \cdots & \beta_m \end{bmatrix} \tag{6}$$

Where $|\alpha_i|^2 + |\beta_i|^2 = 1$, $i = 1, 2, ..., m$ .This representation has the advantage that it is able to represent any superposition of states. If there is, for instance, a three-qubits system with three pairs of amplitudes such as

$$\begin{bmatrix} \frac{1}{\sqrt{2}} & \frac{1}{2} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & \frac{\sqrt{3}}{2} & \frac{-1}{\sqrt{2}} \end{bmatrix} \tag{7}$$

Then the states of the system can be represented as

$$\frac{1}{4}|000\rangle - \frac{1}{4}|001\rangle + \frac{\sqrt{3}}{4}|010\rangle - \frac{\sqrt{3}}{4}|011\rangle + \frac{1}{4}|100\rangle - \frac{1}{4}|101\rangle + \frac{\sqrt{3}}{4}|110\rangle - \frac{\sqrt{3}}{4}|111\rangle \tag{8}$$

This result means that the probabilities to represent the states $|000\rangle, |001\rangle, |010\rangle, |011\rangle, |100\rangle, |101\rangle, |110\rangle, |111\rangle$ are $1/16, 1/16, 3/16, 3/16, 1/16, 1/16, 3/16, 3/16$ respectively. By consequence, the three qubits system of (7) contains the information of eight states.

Swarm intelligence and evolutionary computing with qubit representation has a better characteristic of population diversity than classical approaches, since it can represent superposition of states. Only one qubit system such as (7) is enough to represent eight states, but in binary representation at least eight string, (000), (001), (010), (011), (100), (101), (110), (111) are needed.

2. Quantum Artificial Bee Colony (QABC) Algorithm

2.1. General QABC Algorithm

The QABC algorithm is a general method that can be appropriated for various continuous and discrete problems. In the next subsection, we will show the proper QABC for knapsack problem. The pseudo code of the general QABC algorithm is shown below:

begin
$t = 0$
Initialize
produce $Foods\ t$ by observing $Q\ t$ states
evaluate $Foods\ t$
store the Best solution among $Foods\ t$
while (**not** termination-condition) **do**
begin
$t = t + 1$
For each solution in $Foods$
begin
Produce a new solution by observing $Foods\ t$ and $Q\ t-1$ states by Employed bees
Evaluate the new solution
if new solution is better than the current solution in $Foods$
then replace new solution
end
Calculate probability for each solution`s fitness
For each Onlooker bee in QABC
begin
Choose a solution based on its fitness probability
Produce a new solution by observing $Foods\ t$ and $Q\ t-1$ states by Onlooker bees
Evaluate the new solution
if new solution is better than the current solution in $Foods$
then replace new solution
end
update $Q\ t$ using quantum gates $U\ t$
Store the Best solution among $Foods\ t$
end
Until (t<maxcycle)
end

In *initialize* step, the problem to be optimized, and also some control parameters of QABC is defined and initiated. The most important part of this step is defining and launching a new structure, we call Q in this algorithm.

At iteration t, we have $Q\ t$ like as below:

$$Q\ t\ =\begin{bmatrix} \left\langle \alpha_{1,1}^{t} \middle| \beta_{1,1}^{t} \right\rangle & \left\langle \alpha_{1,2}^{t} \middle| \beta_{1,2}^{t} \right\rangle & \cdots & \left\langle \alpha_{1,D}^{t} \middle| \beta_{1,D}^{t} \right\rangle \\ \left\langle \alpha_{2,1}^{t} \middle| \beta_{2,1}^{t} \right\rangle & \left\langle \alpha_{2,2}^{t} \middle| \beta_{2,2}^{t} \right\rangle & \cdots & \left\langle \alpha_{2,D}^{t} \middle| \beta_{2,D}^{t} \right\rangle \\ \vdots & \vdots & \vdots & \vdots \\ \left\langle \alpha_{n,1}^{t} \middle| \beta_{n,1}^{t} \right\rangle & \left\langle \alpha_{n,2}^{t} \middle| \beta_{n,2}^{t} \right\rangle & \cdots & \left\langle \alpha_{n,D}^{t} \middle| \beta_{n,D}^{t} \right\rangle \end{bmatrix} \tag{9}$$

Each pair of $\left\langle \alpha_{i,j}^{t} \middle| \beta_{i,j}^{t} \right\rangle$; $1 \le i \le n$, $1 \le j \le D$ and $1 \le t \le maxcycle$, are characterized by (4) and (5). We suppose that the target problem is D-dimensional and the value of n is equal to the number of solutions. All of $\alpha_{i,j}$s and $\beta_{i,j}$s are initialized with $1/\sqrt{2}$. This means that each pair of $\alpha_{i,j}$ and $\beta_{i,j}$ have a same probability to be chosen.

In the next step, *produce*, we create a new solution based on the characteristic of optimization problem. To make *Foods*, we use *Q*, and fill it with legal values depending on defined optimization problem. After evaluating solutions, the best one is memorized.

Then in a loop, same as ABC, each of employed and onlooker bees is responsible to produce a new solution for a corresponding solution in *Foods*. The way of producing a new solution depends on the target problem and in this work, details of producing a new solution for Knapsack will be explained in 2.3. After producing a new solution, it should be evaluated. If the new solution is better than the current corresponding solution, the new solution will be replaced with the current solution.

In order to create various new solutions in each iteration that are affected by the values of α s and β s in *Q*, we make some modifications in *Q*. This is done in *update* step by means of quantum gates, *G(T)* in each repetition. More details of different steps in QABC will be shown in the next subsection.

2.2. Knapsack problem

Suppose that we have m items, each has its own weight and profit, w_i and p_i, $1 \le i \le m$, respectively, and there exists a knapsack with the capacity C. A selection of items can be shown with string $x = (x_1, x_2, ..., x_m)$, each x_i is 1 or 0, that means item x_i is selected or not, respectively.

The aim of solving knapsack problem is to select some items from {item$_1$,item$_2$, …,item$_m$} so that:

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

be maximized, subject to

$$\sum_{i=1}^{m} w_i x_i \le C.$$

2.3. QABC for Knapsack 0-1 problem (KQABC)

As mentioned before, QABC algorithm can be adapted for various real continuous and discrete problems. In this work, for showing the potency of new proposed QABC algorithm, we have altered QABC a little for knapsack problem and introduce new KQABC algorithm.

In this work, we want to realize the effect of quantizing each of employed and onlooker bees. Four states are supposed:

1. Classic ABC (KCABC).
2. Quantized Employed - Classic Onlooker Bees (KQABC#1).
3. Classic Employed - Quantized Onlooker Bees (KQABC#2).
4. Both Quantized Employed & Onlooker Bees (KQABC#3).

In KCABC, for producing a new solution by employed or onlooker bees, a dimension is selected randomly and its value is changed without interfering $Q\ t$.

In KQABC#1 for producing a new solution by employed bees, $Q\ t$ is used and onlooker bees produce new solutions classically. However, in KQABC#2 the story goes against the KQABC#1; employed and onlooker bees produce a new solution classically and by $Q\ t$, respectively. Also, in KQABC#3 both of employed and onlooker bees produce a new solution by using $Q\ t$.

Quantized employed and onlooker bees produce a new solution in this manner: a dimension from current solution is selected randomly. Also, a random value in [0,1] is selected and its value is compared with the α^2 portion of corresponding entry and dimension`s value from $Q\ t$ . If the random value is smaller and the dimension`s value in current solution is equal to 1, or if the random value is bigger and the dimension`s value in current solution is equal to 0, the dimension`s value will be changed. Else, its value will be unchanged and this process should be repeated again for another random dimension.

3. Experiments, Setting and Results

Any item in knapsack problem should have two quantities: weight and profit, which are selected as below:

$$W_i = \text{random } [1,10],$$

$$P_i = W_i + 5.$$

Also, the capacity of knapsack is set as:

$$C = \frac{1}{2}\sum_{i=1}^{m} W_i$$

Experiments are implemented for 100, 250, and 500 items on 25 independent runs with different random seed. The results of KQABC are shown in Table1 for best (b), mean (m), and worst (w) among 25 results.

Table 1. Results of KCABC and three KQABC algorithms

		KCABC	KQABC#1	KQABC#2	KQABC#3
100	b	604.0	612.6	609.1	616.6
	m	599.0	608.8	605.4	610.6
	w	594.0	602.6	604.0	606.6
250	b	1497.1	1514.0	1504.5	1517.2
	m	1482.7	1499.6	1492.1	1508.9
	w	1472.1	1489.0	1484.5	1502.3
500	b	2956.5	2983.9	2965.6	2970.3
	m	2934.6	2963.8	2947.9	2951.3
	w	2920.4	2954.3	2925.5	2938.2

From Table1 it is obvious that all of three quantized ABC have better performance with respect to classic one. Also, KQABC#1, quantized just employed portion of ABC, has better results in comparison with KQABC#2, quantized just onlooker portion of ABC. Moreover, KQABC#3 produce the best results among three quantized ABC algorithms, except for number of items 500 that KQABC#1 produce the best results.

4. Conclusion

The new proposed optimization algorithm that is called QABC, was presented in this work. The quantum properties of QABC, makes it an effective approach for dealing with various discrete and continuous problems. QABC is a general algorithm. For showing the performance and abilities of QABC, an adapted version for knapsack problem was introduced. The results on this practical benchmark show the high performance of QABC with respect to classic ABC.

5. Acknowledgment

The authors would like to appreciate Parand Azad University for their funded and scientific supports of this study.

References

[1] A.S. Fraser, "*Simulation of genetic system by automatic digital computers. I.*", Introduction, Austral. J. Biol. Sci. 10 (1957) 484–491.
[2] J.H. Holland, "*Adaptation in Natural and Artificial Systems*", University of Michigan Press, Ann Arbor, (1975).
[3] D.E. Goldberg, "*Genetic Algorithms in Search, Optimization and Machine Learning*", Addison-Wesley, Reading, MA, (1989).
[4] L.J. Fogel, A.J. Owens, M.J. Walsh, "*Artificial Intelligence through Simulated Evolution*", Wiley, New York, (1966).

[5] H.P. Schwefel, "*Numerical Optimization of Computer Models*", Wiley, New York, (1977).
[6] J.R. Koza, "*Genetic Programming: On the Programming of Computers by Means of Natural Selection and Genetics*", MIT Press, Cambridge, MA, (1992).
[7] M. Dorigo, "*Optimization, learning and natural algorithms (in Italian)*", PhD Thesis, Politecnico di Milano, Italy, (1992).
[8] J. Kennedy, R.C. Eberhart, "*Particle swarm optimization*", in: Proc. IEEE Int. Conf. on Neural Networks, WA, Australia, (1995) 1942–1948.
[9] R. Storn, K. Price, "*Differential Evolution – A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces*", J GLOBAL OPTIM. 11 (1997) 341–359.
[10] D. Karaboga, B. Basturk, "*A powerful and efficient algorithm for numerical function optimization: Artificial Bee Colony (ABC) algorithm*", J GLOBAL OPTIM. 39 (2007) 459–471.
[11] D. Karaboga, B. Basturk, "*On the performance of Artificial Bee Colony (ABC) algorithm*", APPL SOFT COMPUT. 8 (2008) 687–697.
[12] D. Karaboga, B. Akay, "*A comparative study of Artificial Bee Colony algorithm*", APPL MATH COMPUT. 214 (2009) 108-132.
[13] K.S. Lee, Z.W. Geem, "*A new meta-heuristic algorithm for continuous engineering optimization: harmony search theory and practice*", Computer Methods in Applied Mechanics and Engineering, 194 (2004) 3902-3933.
[14] P.W. Shor, "*Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer*", SIAM Journal on Computing. 26 (1997) 1484–1509.
[15] L.K. Grover, "*A fast quantum mechanical algorithm for database search*", in: Proceedings of the 28th Annual ACM Symposium on the Theory of Computing, Philadelphia. (1996) 212–219.
[16] K. H. Han, J. H. Kim, "*Quantum-inspired evolutionary algorithm for a class of combinatorial optimization*", IEEE Transaction On Evolutionary Computation. 6 (2002) 580–593.
[17] G. X. Zhang, M. Gheorghe, C. Z. Wu, "*A Quantum-Inspired Evolutionary Algorithm Based on P systems for Knapsack Problem*", Fundamenta Informaticae. 87 (2008) 1-24.
[18] A. Layeb, "*A hybrid quantum inspired harmony search algorithm for 0–1 optimization problems*", Journal of Computational and Applied Mathematics. 253 (2012) 14–25.
[19] Y. Wang, X.Y. Feng, "*A novel quantum swarm evolutionary algorithm and its applications*", NEUROCOMPUTING. 70 (2007) 633-640.