Available online at www.worldscientificnews.com

World Scientific News

WSN 11 (2015) 138-150 EISSN 2392-2192

Use Genetic Algorithm in Optimization Function For Solving Queens Problem

Lubna Zaghlul Bashir[a], Nada Mahdi[b]

Building and Construction Department, University of Technology, Baghdad, Iraq

[a,b]E-mail address: Lubna_Zaghlul@yahoo.com , Nadaimage@yahoo.com

ABSTRACT

Genetic algorithms are stochastic search techniques that guide a population of solutions towards an optimum using the principles of evolution and natural genetics. This paper shows the way that genetic algorithms can be used to solve 4-Queen problem. The NQP is a classical artificial intelligence problem. The N-Queens problem can be defined as follows: place N queens on an N x N chessboard, each queen on a square, so that no queen could capture any of the others, that is, a configuration in which there exists at most one queen on a given row, column or diagonal. Experimentally results shows that the genetic algorithm have the ability to find optimal solution or find solutions nearby optimal solutions. And The Genetic Algorithm is well suited to has been extensively applied to solve complex design optimization problems because it can handle both discrete and continuous variables, and nonlinear objective and constrain functions without requiring gradient information.

***Keywords*:** genetic algorithm; evolutionary algorithms; optimization; queen problem

استخدام الخوارزمية الجينية للحصول على افضل حل لمشكلة التيجان

لبنى زغلول بشير ندى مهدي

الجامعة التكنولوجية

الملخص-الخوارزمية الجينية تقنية بحث عشوائية التي تتضمن مجموعة من الحلول لايجاد الحل الامثل باستخدام قواعد الجينات الطبيعية.في هذا البحث نوضح طريقة الخوارزمية الجينية لحل مشكلة التيجان.مشكلة التيجان احد مشاكل الذكاء الاصطناعي تعرف كالاتي:يوضع عدد من التجان على لوحة ابعادها4x4 ،كل تاج في مربع،بحيث لا يتجاور اي تاج مع اخر في نفس الصف او العمود او قطريا.التجارب بينت قابلية الخوارزمية الجينية لايجاد افضل حل. والخوارزمية الجينية ملائمة لحل مشاكل امثلية معقدة.

1. INTRODUCTION

In modern compilers and optimizers, the set of possible optimizations is usually very large. In general, for a given application and workload, optimization options do not accrue toward ultimate performance. To avoid selection complexity, users tend to use standard combination options, in general, however, these standard options are not the optimal set for a specific application executing its representative workload [1,2].

Modern compilers present a large number of optimization options covering the many alternatives to achieving high performance for different kinds of applications and workloads. Selecting the optimal set of optimization options for a given application and workload becomes a real issue since optimization options do not necessarily improve performance when combined with other options [2].

Genetic algorithms (GA) [3,4] present an attractive solution to this problem of selecting an optimal set of options. The problem is easily mapped to the original problem of gene optimization. The extended time required to reach to a preferred solution is justified by the much longer life of the optimized program [2].

In the computer science field of artificial intelligence, a genetic algorithm (GA) is a search heuristic that mimics the process of natural selection. This heuristic (also sometimes called a meta heuristic) is routinely used to generate useful solutions to optimization and search problems. Genetic algorithms belong to the larger class of evolutionary algorithms (EA), which generate solutions to optimization problems using techniques inspired by natural evolution, such as inheritance, mutation, selection, and crossover [5].

2. GENETIC ALGORITHMS FOR OPTIMIZATION

Optimization is a process that finds a best ,or optimal solutions for a problem. the optimization problems are catered around three factors:

1. An objective function: which is to be minimize or maximize.

2. A set of unknowns or variables: that effect the objective function.

3. A set of constraints: that allow the unknowns to take on certain values.

An optimization problem is defined as finding values of the variables that minimize or maximize the objective function while satisfying the constraints but exclude others [6].

The Genetic Algorithms are direct, stochastic method for optimization. Since they use populations with allowed solutions (individuals), they count in the group of parallel algorithms. Due to the stochastic was of searching, in most cases, it is necessary to set limits at least for the values of the optimized parameters [7].

A genetic algorithm developed by J.H. Holland, 1975 [8]. Genetic algorithms are stochastic search techniques that guide a population of solutions towards an optimum using the principles of evolution and natural genetics [1].

The aim of genetic algorithms is to use simple representations to encode complex structures and simple operations to improve these structures. Genetic algorithms therefore are characterized by their representation and operators [1].

3. BIOLOGICAL GENETIC ALGORITHM CONCEPT

- **Gene -** a single encoding of part of the solution space, i.e. either single bits or short blocks of adjacent bits that encode an element of the candidate solution. Figure 1 shows gene representation.

Figure 1. Gene representation.

- **Chromosome** - a string of genes that represents a solution. Figure 2 shows chromosome representation.

Figure 2. Chromosome representation.

- **Population -** the number of chromosomes available to test. Figure 3 shows population representation [9]

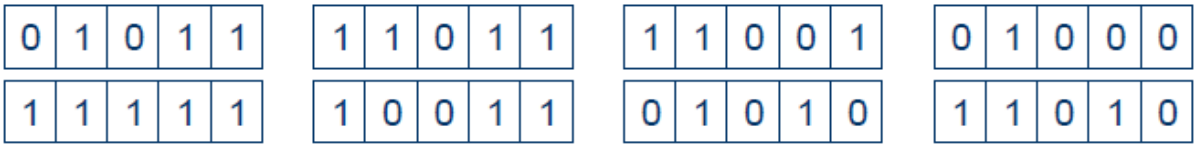


Figure 3. Population representation.

- **Reproduction or selection** - From the population, the chromosomes are selected to be parents to crossover and produce offspring. select these chromosomes According to Charles Darwin's evolution theory "survival of the fittest" - the best ones should survive and create new offspring.

The Fitness function quantifies the optimality of a solution (chromosome) so that a particular solution may be ranked against all the other solutions. The function depicts the closeness of a given 'solution' to the desired result [8,10].

- **Rolette wheel selection -** the chance of an individual's being selected is proportional to its fitness, greater or less than its competitors' fitness. The Probability of selection of i [th] individual is:

$$P_i = \frac{f_i}{\sum_{j=1}^{n} f_j}$$

where fi :fitness of i [th] individual, N : number of individuals. figure.4 illustrate Rolette wheel selection. [10]

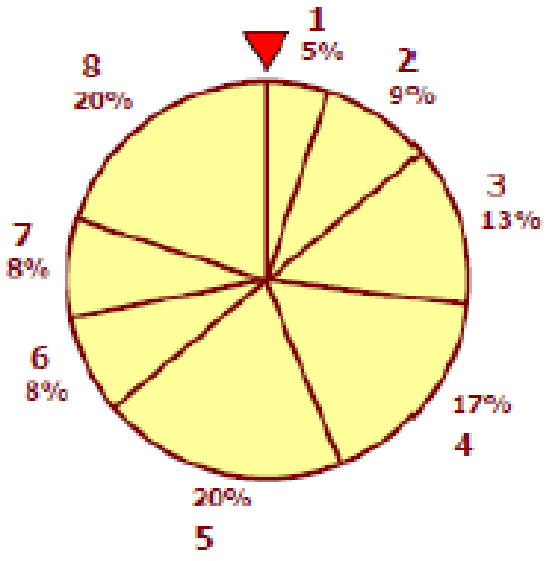
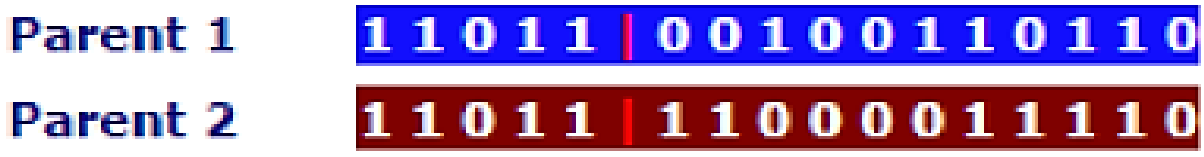


Figure 4. Rolette wheel selection.

- **Crossover -** Selects genes from parent chromosomes, combines them and creates a new offspring. New chromosome may be better than both of the parents if it takes the best characteristics from each of them, Consider the two parents selected for crossover. Figure 5 illustrate crossover operation [10].



Interchange the parents chromosomes after crossover points. The offspring produced are:

Figure 5. Crossover operation

- **Mutation-** Mutation alters one or more gene values in a chromosome from its initial state. The mutation operator simply inverts the value of the chosen gene i.e. 0 goes to 1 and 1 goes to 0. Consider the two original offspring's selected for mutation figure.6 illustrate mutation operation [8,10].

The Mutated Offspring produced are:

Figure 6. Mutation operation.

4. CODE OF GENETICS ALGORITHM

- Choose the initial population of individuals
- Evaluate the fitness of each individual in population
- Repeat until termination condition satisfied:
- Selection: Select the individuals with greater fitness for reproduction
- Crossover: Breed new individuals through crossover
- Mutation: Apply probabilistic mutation on new individuals
- Form a new population with these offsprings.
- Terminate

Figure 7. Shows the basic flow diagram of a genetic algorithm [8,10].

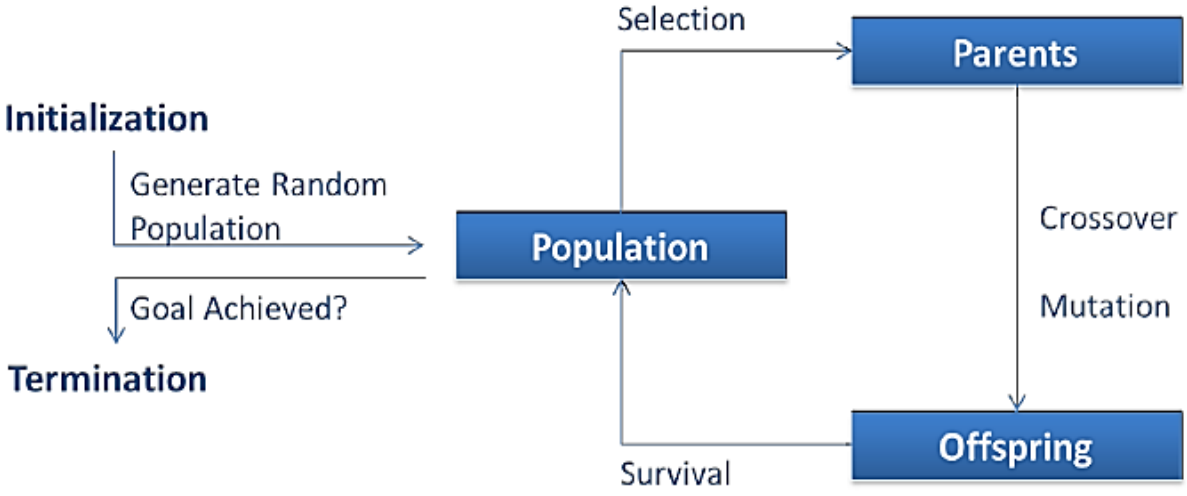


Figure 7. Genetic algorithm diagram

5. THE QUEENS PROBLEM

There are four queens as shown in Figure 8, we want to find a way to place them on a board divided into 16 grid units so that no two queens attach each others. In this work we use Genetic Algorithms to determine the best location of four queens.

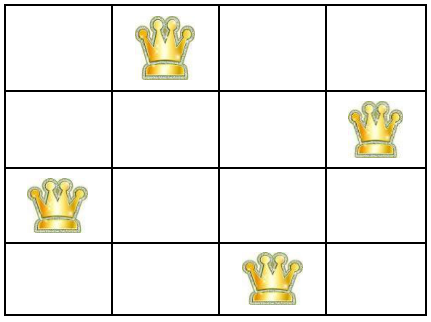


Figure 8. The Queens Problem

5. 1. Problem Representation (how to define a facility location)

Since each queen must be on a different row and column, we can assume that queen *i* is placed in *i*-th column. Position of a number in the tuple represents queen's column position, while its value represents queen's row position (counting from the bottom) Using this representation, the solution space where two of the constraints (row and column conflicts) are already satisfied should be searched in order to eliminate the diagonal conflicts. Figure 9 illustrates 4-tuples for the 4-queen problem [11,12].

			Q4
Q1			
		Q3	
	Q2		

Figure 9. 4-tuples for the 4-queen problem.

5. 2. Chromosome Structure

The variables in the problem are the locations of four queens. Then, the chromosome structure are as shows in Figure 10. Note that the genes of a chromosome are the problem variables.

Q1	**Q2**	**Q3**	**Q4**
3	**1**	**2**	**4**

4 genes (values from 1 to 4)

Figure 10. Chromosome Structure.

5. 3. Generate Population (50 to 100 is reasonable diversity & processing time)

(P1)

Q1	Q2	Q3	Q4
3	**1**	**4**	**2**

		Q3	
Q1			
			Q4
	Q2		

(P2)

Q1	Q2	Q3	Q4
3	**1**	**2**	**4**

			Q4
Q1			
		Q3	
	Q2		

(P3)

Q1	Q2	Q3	Q4
3	**4**	**2**	**1**

	Q2		
Q1			
		Q3	
			Q4

Figure 11. Shows sample of population.

5. 4. Evaluate the Population

Objective function = Maximize site score = Maximum of Σ d · W

$$\textbf{Score} = \mathbf{d_{Q1Q2}} \cdot \mathbf{W_{Q1Q2}} + \mathbf{d_{Q1Q3}} \cdot \mathbf{W_{Q1Q3}} + \mathbf{d_{Q1Q4}} \cdot \mathbf{W_{Q1Q4}} + \mathbf{d_{Q2Q3}} \cdot \mathbf{W_{Q2Q3}} + \mathbf{d_{Q2Q4}} \cdot \mathbf{W_{Q2Q4}} + \mathbf{d_{Q3Q4}} \cdot \mathbf{W_{Q3Q4}} \quad (1)$$

Let's consider the closeness weights (W) as follows:
$\mathbf{W_{QiQj}}$ **= 10** (positive means Q_i & Q_j far from each other)
$\mathbf{W_{QiQj}}$ **= -10** (negative means Q_i & Q_j close to each other)

Let's also consider the distance (d) between two queens as the number of horizontal and vertical blocks between them.

P1 Score = 3x10 + 3x10 + 4x10 + 4x10 + 3x10 + 3x10 = 200
P2 Score = 3x10 + 3x10 + 4x10 + 2x-10 + 5x10 + 3x10 = 160
P3 Score = 2x-10 + 3x10 + 5x10 + 3x10 + 5x10 + 2x-10 = 120

5. 5. Calculate the Merits of Population Members

Merit of **P1** = (200 + 160 + 120) / 200 = **2.4 = 2**
Merit of **P2** = (200 + 160 + 120) / 160 = **3**
Merit of **P3** = (200 + 160 + 120) / 120 = **4**

the sum of merits **= 9**
the smaller score gives higher merit because we are interested in minimization. In case of maximization, we use the inverse of the merit calculation.

5. 6. Calculate the Relative Merits of Population Members

RM of **P1** = merit * 100 / Sum of merits = 2 . 100 / 9 = **22.22**
RM of **P2** = merit * 100 / Sum of merits = 3 . 100 / 9 = **33.33**
RM of **P3** = merit * 100 / Sum of merits = 5 . 100 / 9 = **55.55**

5. 7. Randomly Select Operator (Crossover or Mutation)

Crossover rate = 96% (marriage is the main avenue for evolution)
Mutation rate = 4% (genius people are very rare)
To select which operator to use in current cycle, we generate a random number (from 0 to 100). If the value is between 0 to 96, then crossover, otherwise, mutation

5. 8. Use the Selected Operator (Assume Crossover)

Randomly select two parents P1 and P2 according to their relative merits of Step 5. 6. Figure 12 shows the two parents selected.

(P1)

3	**1**	**4**	**2**

(P2)

3	**1**	**2**	**4**

Figure 12. Two Parents Selected

Let's apply crossover to generate an offspring.Figure.13 shows the two offspring's obtained.

Offspring 1

3	**1**	**2**	**4**

Offspring 2

3	**1**	**4**	**2**

Offspring 1

			Q4
Q1			
		Q3	
	Q2		

Offspring 2

		Q3	
Q1			
			Q4
	Q2		

Figure 13. Two Offspring Obtained

5. 9. Evaluate the Offspring

Offspring1 Score = 3x10 + 3x10 + 4x10 + 2x-10 + 5x10 + 3x10 = 160
Offspring2 Score = 3x10 + 3x10 + 4x10 + 4x10 + 3x10 + 3x10 = 200

5. 10. Compare the Offspring's with the Population (Evolve the Population)

Since the offspring 2 score = **200** is better than the worst population member (P3 has a score of 120), then the offspring 2 survives and P3 dies (will be replaced by the offspring 2). Figure 14 illustrates (P3) chromosome.

3	**1**	**4**	**2**

Figure 14. Illustrates P3 chromosome

5. 11. Experimental Result

Executing the simple genetic algorithm code (SGA) written in Pascal programming language, the primary data structure is the population of 40 string as shown in table.1and the following parameters:

SGA Parameters
Population size = **40**
Chromosome length = **4**
Maximum of generation = **10**
Crossover probability = **0.6**
Mutation probability = **0.03**

Table 1. Queen problem population.

	Q1	**Q2**	**Q3**	**Q4**
1.	**1**	**1**	**1**	**1**

2.	**2**	**1**	**1**	**1**
3.	**3**	**1**	**1**	**1**
4.	**4**	**1**	**1**	**1**
5.	**1**	**2**	**1**	**1**
6.	**1**	**3**	**1**	**1**
7.	**1**	**4**	**1**	**1**
8.	**1**	**1**	**2**	**1**
9.	**1**	**1**	**3**	**1**
10.	**1**	**1**	**4**	**1**
11.	**1**	**1**	**1**	**2**
12.	**1**	**1**	**1**	**3**
13.	**1**	**1**	**1**	**4**
14.	**2**	**2**	**2**	**2**
15.	**1**	**2**	**2**	**2**
16.	**3**	**2**	**2**	**2**
17.	**4**	**2**	**2**	**2**
18.	**2**	**1**	**2**	**2**
19.	**2**	**3**	**2**	**2**
20.	**2**	**4**	**2**	**2**
21.	**2**	**2**	**1**	**2**
22.	**2**	**2**	**3**	**2**
23.	**2**	**2**	**4**	**2**
24.	**2**	**2**	**2**	**1**
25.	**2**	**2**	**2**	**3**
26.	**2**	**2**	**2**	**4**
27.	**3**	**3**	**3**	**3**
28.	**1**	**3**	**3**	**3**
29.	**2**	**3**	**3**	**3**
30.	**4**	**3**	**3**	**3**
31.	**3**	**1**	**3**	**3**
32.	**3**	**2**	**3**	**3**

33.	**3**	**4**	**3**	**3**
34.	**3**	**3**	**1**	**3**
35.	**3**	**3**	**2**	**3**
36.	**3**	**3**	**4**	**3**
37.	**3**	**3**	**3**	**1**
38.	**3**	**3**	**3**	**2**
39.	**3**	**3**	**3**	**4**
40.	**4**	**4**	**4**	**4**

At the end of the run repeating the process thousands of times until the best solution is determined. One of the top solutions shows in Figure.15.

		Q3	
Q1			
			Q4
	Q2		

Figure 15. The Optimal Solution For Queen Problem

6. CONCLUSIONS

- This paper showed that Queen problem can be successfully solved using genetic algorithm for optimization function.
- Experimentally results shows that the genetic algorithm have the ability to find optimal solution or find solutions nearby optimal solutions.
- The GA does find near optimal results quickly after searching a small portion of the search space.
- Results shows that the genetic algorithm have the ability to find optimal solution for solving queens problem.
- The GA is well suited to and has been extensively applied to solve complex design optimization problems because it can handle both discrete and continuous variables, and nonlinear objective and constrain functions without requiring gradient information.

References

[1] Ms. Dharmistha D. Vishwakarma" Genetic Algorithm based Weights Optimization of Artificial Neural Network", *International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering* 1(3) (2012).

[2] Bashkansky, Guy and Yaari Yaakov" Black Box Approach for Selecting Optimization Options Using Budget-Limited Genetic Algorithms", 2007.

[3] D. Whitley. A genetic algorithm tutorial. *Statistics and Computing,* 4(2) (1994) 65-85. 1994.

[4] E.B. Baum, D. Boneh, and C. Garrett. Where genetic algorithms excel. *Evolutionary Computation*, 9(1) (2001) 93-124.

[5] "Genetic algorithm" From Wikipedia, the free encyclopedia, 2013.

[6] R. C. Chakraborty "Fandemantaels of genetic algorithms": AI course lecture 39-40, notes, slides, 2010.

[7] Andrey Popov," Genetic Algorithms For Optimization", Hamburg, 2005.

[8] Goldberg, David E. "Genetic Algorithm in Search, Optimization, and Machine Learning", Addison Wesley Longmont, International Student Edition 1989.

[9] Olesya Peshko,"Global Optimization Genetic Algorithms", 2007.

[10] Saif Hasan, Sagar Chordia, Rahul Varshneya,"Genetic algorithm", February 6, 2012.

[11] Amer Draa, Souham Meshoul, Hichem Talbi, and Mohamed Batouche,,"A Quantum-Inspired Differential Evolution Algorithm for Solving the N-Queens Problem", Computer Science Department, 2008.

[12] Marko Božikovic, Marin Golub, Leo Budin, "Solving n-Queen problem using global parallel genetic algorithm", 2003.

(Received 19 May 2015; accepted 09 June 2015)