Volume 6, Issue 8, August 2016 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Apply Genetic Algorithm for Pseudo Random Number Generator

Fadheela Sabri Abu-Almash
Scholarships and Cultural Relations Directorate,
Iraq- Baghdad

Abstract*— *A random number generator is a standard computational tool can use it to create a sequence of apparently unrelated numbers, which are often used in statistics and other computations. The genetic algorithm is one of the search methods to find the optimal solution and often used in cryptography. In this paper, we proposed a new approach to generates a binary sequence by applying genetic algorithm, There are Five statistical tests applied in this experiment for each chromosome, which are: (Frequency test, Serial test, Poker test, Runs test, Autocorrelation test). The main goal of this research paper is applying genetic algorithm (GA) to generates a binary sequence of chromosomes. The results of this experiment support the effectiveness of this idea, we used the standard terms such as accuracy and robustness evaluating the performance of proposed approach.

Keywords*— *Genetic, pseudorandom numbers, cryptosystems, optimal solution.

I. INTRODUCTION

In practice, pseudo-random numbers are important for simulations and for certain algorithms, e.g., Monte Carlo methods. A good generator produces numbers that are not distinguishable from truly random numbers in a limited computation time, if the seed is not known. It is sometimes tolerated, that the numbers are distinguishable from truly random numbers using functions, which cannot appear as a part of the application domain, where the generator is used [1]. Random numbers are needed in a variety of scientific, mathematical, engineering, and industrial applications including cryptography [2].

The need for random and pseudorandom numbers arises in many cryptographic applications. For example, common cryptosystems employ keys that must be generated in a random fashion. Many cryptographic protocols also require random or pseudorandom inputs at various points, e.g., for auxiliary quantities used in generating digital signatures, or for generating challenges in authentication protocols. The purpose of this paper is to obtain optimal solution in population and then use them as the building of the next generations. This goal has been obtained through a selection and crossover and mutation operation.

A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications that may be used for many purposes including cryptographic, modelling, and simulation applications.

Evolutionary algorithms often perform good optimum solutions to any problem. The GA try to find the optimal solution of a problem in the form of strings binary numbers, the best performance are usually those that reflect something about the problem being solved, by applying operators such as crossover and mutation.

The performance of a genetic algorithm is dependent on the genetic operators such as crossover and mutation, these two operators are work together to seek and explore the search space by creating new variants gens in the chromosomes.

II. STATISTICAL TESTS

This section displaying of some tests designed to measure the quality of a generator purported to be a random bit generator (A random bit generator is a procedure or algorithm which outputs a string of statistically autonomous and unbiased binary digits)[2].

This is accomplished by taking a sample output sequence of the generator and subjecting it to various statistical tests. Each statistical test determines whether the sequence possesses a certain attribute that a truly random sequence would be likely to exhibit. An example of such an attribute is that the sequence should have roughly the same number of 0's as 1's [2].

Let s = s0, s1, s2… sn_{-1} be a binary sequence of length n.

This paper offer five statistical tests that are usually used for setting whether the binary sequence s possesses some specific characteristics that a truly random sequence would be likely to display [3].

If sequences succeed in all five tests, there is no guarantee that it was indeed produced by a random bit generator. The used statistical tests are:

(i) Frequency test (mono bit test)

For a random string of length n the number n1 of ones and the number n0 of zeros should be about the same.. The statistic used is in equation (1) [3]:

$$x_1 = \frac{(n_0 - n_1)^2}{n} \quad \ldots\ldots (1)$$

(ii) Serial test (two-bit test

The purpose of this test is to determine whether the number of redundancy of 00, 01, 10, and 11 are approximately the same. Let, n00, n01, n10, and n11 denote the number of occurrences of "00", "01", "10", and "11" in S, respectively, . The statistic calculated by equation (2) :

$$x_2 = \frac{4}{n-1}\left(n_{00}^2 + n_{01}^2 + n_{10}^2 + n_{11}^2\right) - \frac{2}{n}\left(n_0^2 + n_1^2\right) + 1 \quad \ldots\ldots (2)$$

(iii) Poker test

The poker test determines whether the sequences of length *m* each appear approximately the same number of times in s , as would be expected for a random sequence[2,3,4]. The statistic as given in equation (3)

$$x_3 = \frac{2^m}{k}\left(\sum_{i=1}^{2^m} n_i^2\right) - k \quad \ldots\ldots (3)$$

(iv) Runs test

The purpose of this test is to determine whether the number either zeros or ones of various lengths string in the sequence S is as expected for a random sequence: [2,3,4]. The statistic as given in equation (4)

$$x_4 = \sum_{i=1}^{k} \frac{(B_i - e_i)^2}{e_i} + \sum_{i=1}^{k} \frac{(G_i - e_i)^2}{e_i} \quad \ldots\ldots (4)$$

(v) Autocorrelation test

The purpose of this test is to check for relationships between the series s and shifted versions of it. in 1996 A. Menezes, P. van Oorschot, and S. Vanstone wrote [2]

" Let *d* be a fixed integer, $1 \le d \le n/2$. The number of bits in *s* not equal to their d-shifts is $A(d) = \sum_{i=0}^{n-d-1} s_i \oplus s_{i+d}$, where $\oplus$ denote the *XOR* operator .which approximately follows an N(0, 1) distribution if $n - d \ge 10$. [2,3,4]. Since small values of *A(d)* are as unexpected as large values of *A(d)*, a two-sided test should be used" . The statistic as given in equation (5)

$$x_5 = 2\left(A(d) - \frac{n-d}{2}\right) / \sqrt{n-d} \quad \ldots\ldots (5)$$

III. GENETIC ALGORITHMS

Genetic algorithms are adaptive algorithms, based on the evolutionary ideas of natural selection. The powerful of this technique, that employ concepts of evolutionary biology to evolve optimal solutions to a given problem. Genetic algorithm works with a population of individuals represented by chromosomes. Each chromosome is evaluated by its fitness value as computed by the objective function of the problem. The population undergoes transformation using three primary genetic operators – selection, crossover and mutation which form new generation of population. This process continues to achieve the optimal solution. Basic algorithm of genetic algorithm is:

The work of the simple genetic algorithms can be showing in Fig (1) represent the main steps that should be performed to produce the required solution.

IV. EXPERIMENTAL SETUP

To implement this experiment, we used Delphi language to apply simple genetic algorithm to generate the data. This algorithm is shown below:

```
Algorithm simple GAs:
  Initialization new [population];
    Evaluation each chromosome [population];
    Generation:=0;
      DO
        Selected.parents:= selection[population];
        Created.offspring:=recombination
           [selected.parents];
           Mutation [created.offspring];
           Population:=created.offspring        for        each
chromosome;
           Evaluation each chromosome in[population];
           Generation:=geaeration+1;
      UNTIL stop-criterion;
```

Figure (1): The main steps of a Simple Genetic Algorithm

For generator the key we are implemented five test to fined and select the best solution. from these tests we can evaluate the word. These tests are:

The random number generator is one of the important components of evolutionary algorithms. Therefore, when we try to solve function optimization problems using the evolutionary algorithms, we must carefully choose a good pseudo-random number generator (key).

In the evolutionary algorithms, the random number generator is often used for creating uniformly distributed individuals. In this study, as the low-discrepancy sequences allow us to create individuals more uniformly than the random number sequences, we apply the low-discrepancy sequence generator, instead of the pseudo-random number generator, to the evolutionary algorithms. Since it was difficult for some evolutionary algorithms, such as binary-coded genetic algorithms, to utilize the uniformity of the sequences, the low-discrepancy sequence generator was applied to binary-coded genetic algorithms.

The numerical experiments show that the low-discrepancy sequence generator improves the search performances of genetic algorithms.

A. initial Population (encoding)

A random population of chromosomes encoded (Chromosomes representation by generation binary string 0's and 1's), the length of chromosome depended on the problem) is generated. The population represent individuals; these individuals are representing the start population in the search space for the simple Genetic Algorithms [5]. The number of bits in each chromosome is equal to the number of elements key (i.e. 11) .This is demonstrated in Fig (2).

1	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*
1	1	1	1	0	0	1	0	1	0	1

Figure (2): an example of Sample of Chromosome

B. Fitness Evaluation

In this work used fitness from Statistical tests that used in section 3, the value of the fitness function should be in the range of 5 to 50. If the value is equal to 5 then it have a lower fitness value of chromosome, If the value is equal to 50 then it will produce a high fitness value and produce feasible solutions. Feasible solutions have a greater chance of being followed by the algorithm.

C. Genetic Operators

Genetic algorithm consist of two types of operators:, crossover (two point), and mutation. Performance of GA very depends on them. Type and implementation of operators depends on encoding and also on a problem.

1. Selection Operator

In this work we used roulette wheel selection (RWS), where the chromosome being selected is depending to its fitness value [6]

2 - Reproduction (Crossover operation)

Practically, we used crossover operation, which are two chromosomes are used as parents and new individuals (parents) are formed by changing a sub-sequence between the two strings as in fig (3).

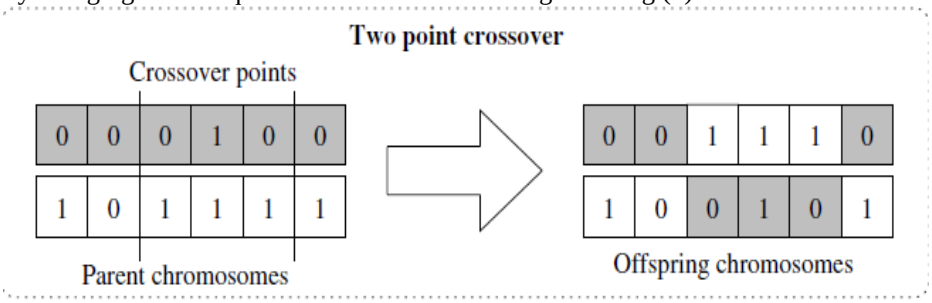

Figure (3): two-point crossover

3. Mutation operation

In this operation two mutation types were used:

1- The mutation process moves between two random points. The mutation probability has to be small. The mutation helps to prevent the algorithm from being stuck in a local optimal point.

2- Inversion**.** This is known as the inversion operator [5]. An inversion is select two positions randomly from chromosome and the portion of a chromosome detaches from the rest of the chromosome, then changes direction and recombines with the chromosome. This is demonstrated in Figure 3.

Chromosome
Before Inversion: 010 **<u>10100101101010</u>** 100
After Inversion: 010 **<u>01010110100101</u>** 100

Figure (3): The Inversion Operator

4. Stop Criterion

This work will stop when best solution has been made .Genetic algorithm will stop when the get a solution which represent a sequence passes all five tests.

V. EXPERIMENTAL RESULTS

The genetic algorithm used in this work for generating keys and obtaining for best solution (key) through many generations. Currently, Genetic Algorithms (GA) are widely used in different optimization problems. One of the problems with GAs is key generation in cryptography.

In this research, different parameters have been used each chromosome as a whole solution for a specific problem. The performance of this algorithm has been shown on the key generation. Many statistical and comparative are used .All test for genetic algorithms were generated with 6 runs per data point using 'Delphi' language.

- **Test I – Effect of Population size**

Population is defined as an array of individual solutions.Thus the Size of population in important parameter that is to be considered for the success of the GA.

If the population size is too small, then would not yield in the best possible solution.

If the population size is too large, the algorithm becomes inefficient as more number of tests is performed than necessary for each generation.

Genetic algorithm run 6 times with varied population size from 25 to 150, constant crossover probability is 0.8 , mutation probability is 0.2 and length of chromosome is 2000 then results are obtained. This is demonstrated in Table1.

The effect of size of population is inversely proportional to the number of generations

In other words if we are increasing the size of population then the number of generations are decreased.

- **Test 2- Effect of Crossover Rate**

In order to see the effect of crossover rate on the number of generations and percentage of search space 6 tests were performed.

Genetic algorithm run 6 times with varied crossover rate from (0.55) to (0.85), constant population size (25) , mutation probability is 0.2 and length of chromosome is1000 and number of generations is 50 then results are obtained. This is demonstrated in Table number (2) .

When the crossover rate is set to low values (0.6) , the percentage of search space is higher in the initial stage but it eventually settles down the at low percentage search space . When the crossover rate is set to high value (0.85) , the genetic algorithm settles down in a lower percentage of search space , indicating that too high or too crossover rate will not yield optimal result .

- **Test 3- Effect of Chromosome length**

In order to see the effect of length of chromosome on the number of generations and percentage of search space 6 tests were performed.

Genetic algorithm run 6 times with varied chromosome length from 100 to 6000 bits, constant population size 10 , mutation probability is (0.1) crossover rate is 0.8 and number of generations is 50 then results are obtained. This is demonstrated in Table number (3).we can see that number of generations is decreased whenever the chromosome length increase.

Table I Test I – Effect of Population size

Runs and GA parameters	Best solution. in generation 1	Worst solution generation 1	Average solution	Best solution in generation	Worst solution generation	Average solution
Population Size = 25 Generation number = 25 rossOver Prob. = 0.8 Mutation Prob. = 0.2 len Numbersof chrosome = 2000	40 Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	35 Frequency test=10 Serial test=10 Poker test=5 Runs test=5 Autocorrelation test=5	37.5	**50 in generation 20** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=10	**40 in generation 20** Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	45
Population Size = 50 Generation number = 25 rossOver Prob. = 0.8 Mutation Prob. = 0.2 len Numbersof chrosome = 2000	45 Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	40 Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	42.5	**50 in generation 4** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=10	**40 in generation 4** Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	45
Population Size = 75 Generation number = 25 rossOver Prob. = 0.8 Mutation Prob. = 0.2 len Numbersof chrosome = 2000	45 Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	40 Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	42.5	**50 in generation 3** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=10	**40 in generation 3** Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	45
Population Size = 100 Generation number = 25 CrossOver Prob. = 0.8 Mutation Prob. = 0.2 len Numbersof chrosome = 2000	40 Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	40 Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	40	**50 in generation 2** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=10	**40 in generation 2** Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	45
Population Size = 125	45	40	42.5	**50 in generation 2** Frequency	**40 in generation 2**	45

Generation number = 25 CrossOver Prob. = 0.8 Mutation Prob. = 0.2 len Numbersof chrosome = 2000	Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5		test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=10	Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	
Population Size = 150 Generation number = 25 CrossOver Prob. = 0.8 Mutation Prob. = 0.2 len Numbersof chrosome = 2000	45 Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	40 Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	42.5	**50 in generation 2** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=10	**40 in generation 2** Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	45

Table III Test 2- Effect of Crossover Rate

Runs and GA parameters	**Best solution. in generation 1**	**Worst solution generation 1**	**Average solution**	**Best solution in generation**	**Worst solution generation**	**Average solution**
Population Size = 25 Generation number = 50 CrossOver Prob. = 0.55 Mutation Prob. = 0.2 len Numbersof chrosome = 1000	45 Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	40 Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	42.5	**45 in generation 25** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	**40 in generation 25** Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	42.5
Population Size = 25 Generation number = 50 CrossOver Prob. = 0.6 Mutation Prob. = 0.2 len Numbersof chrosome = 1000	45 Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	40 Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	42.5	**50 in generation 19** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=10	**40 in generation 19** Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	45
Population Size = 25 Generation number = 50 CrossOver Prob. = 0.7 Mutation Prob. = 0.2 len Numbersof chrosome = 1000	45 Frequency test=10 Serial test=10 Poker test=10 Runs test=10	40 Frequency test=10 Serial test=10 Poker test=10 Runs test=5	42.5	**50 in generation 27** Frequency test=10 Serial test=10 Poker test=10 Runs test=10	**45 in generation 27** Frequency test=10 Serial test=10 Poker test=10 Runs test=10	47.5

	Autocorrelation test=5	Autocorrelation test=5		Autocorrelation test=10	Autocorrelation test=5	
Population Size = 25 Generation number = 50 CrossOver Prob. = 0.80 Mutation Prob. = 0.2 len Numbersof chrosome = 1000	45 Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	40 Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	42.5	**50 in generation 17** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=10	**45 in generation 17** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	47.5
Population Size = 25 Generation number = 50 CrossOver Prob. = 0.80 Mutation Prob. = 0.2 len Numbersof chrosome = 1000	45 Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	40 Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	42.5	**50 in generation 30** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=10	**45 in generation 30** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	47.5
Population Size = 25 Generation number = 50 CrossOver Prob. = 0.85 Mutation Prob. = 0.2 len Numbersof chrosome = 1000	45 Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	40 Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	42.5	50 in generation 32 Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=10	45 in generation 32 Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	47.5

Table IIIII Test 3- Effect of Chromosome length

Runs and GA parameters	**Best solution. in generation 1**	**Worst solution generation 1**	**Average solution**	**Best solution in generation**	**Worst solution generation**	**Average solution**
Population Size = 10 Generation number = 50 CrossOver Prob. = 0.8 Mutation Prob. = 0.1 len Numbersof chrosome = 100	35 Frequency test=10 Serial test=10 Poker test=5 Runs test=5 Autocorrelation test=5	30 Frequency test=10 Serial test=5 Poker test=5 Runs test=5 Autocorrelation test=5	32.5	**40 in generation 50** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=10	**45 in generation 50** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	42.5
Population Size = 10 Generation number = 50 CrossOver Prob. = 0.8 Mutation Prob. = 0.1 len Numbersof chrosome = 700	40 Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	35 Frequency test=10 Serial test=10 Poker test=5 Runs test=5 Autocorrelation test=5	37.5	**50 in generation 35** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=10	**40 in generation 35** Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	45

Population Size = 10 Generation number = 50 CrossOver Prob. = 0.8 Mutation Prob. = 0.1 len Numbersof chrosome = 1000	40 Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	40 Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	40	**50 in generation 33** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=10	**45 in generation 33** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	47.5
Population Size = 10 Generation number = 50 CrossOver Prob. = 0.8 Mutation Prob. = 0.1 len Numbersof chrosome = 2000	45 Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	40 Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	42.5	**50 in generation 29** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=10	**45 in generation 29** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	47.5
Population Size = 10 Generation number = 50 CrossOver Prob. = 0.8 Mutation Prob. = 0.1 len Numbersof chrosome = 5000	45 Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	40 Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	42.5	**50 in generation 25** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=10	**45 in generation 25** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	47.5
Population Size = 10 Generation number = 50 CrossOver Prob. = 0.8 Mutation Prob. = 0.1 len Numbersof chrosome = 6000	45 Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	40 Frequency test=10 Serial test=10 Poker test=10 Runs test=5 Autocorrelation test=5	42.5	**50 in generation 19** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=10	**45 in generation 19** Frequency test=10 Serial test=10 Poker test=10 Runs test=10 Autocorrelation test=5	47.5

VI. CONCLUSION

In this paper we have submitted a genetic algorithm For Pseudo random number generation the results of the experimental indicated in clearly that the genetic algorithm has powerful new tool For key generation. The results of this work proved the efficiency of genetic algorithm for pseudo random number generation , and it's efficiency can be enhanced by the number of parameters such as initial population mutation, crossover operation and size of population . The large size of population, elevation crossover and a tiny mutation probability are the optimal representation for Genetic Algorithm. Also we can see in results of this work the worse happens when the population size and crossover operator are decreases, and when the increase the coefficient mutation.

ACKNOWLEDGMENT

I would like to thank the Ministry of Higher Education and Scientific Research, Scholarships and Cultural Relations Directorate in Iraq, for their support always.

REFRENCES

[1] M. Matsumoto and T. Nishimura. Mersenne twister: A 623-dimensionally equidistributed uniform pseudorandom number generator. *ACM Trans. on Modeling and Computer Simulation*, 8(1):3–30, 1998.

[2] A. Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography, For further information, see www.cacr.math.uwaterloo.ca/hac, CRC Press, 1996.

[3] arter G., "Statistical Tests For Randomness", EISS, Karlsruhe, England, 1989.

[4] Alfred J. Menezes, Paul C. van Oorschot, and Scott A. Vanstone, Handbook of Applied Cryptography. CRC press, (1997).

[5] Firas Alabsi and Reyadh Naoum "Comparison of Selection Methods and Crossover Operations using Steady State Genetic Based Intrusion Detection System " Journal of Emerging Trends in Computing and Information Sciences, VOL. 3, NO.7, ISSN 2079-8407, All rights reserved. http://www.cisjournal.org, 1053, July 2012.

[6] S.N. Sivanandam and S. N. Deepa "Introduction to Genetic Algorithms" , Published by Springer, ISBN 10: 8132211057 / ISBN 13: 9788132211051, Book Condition: New , 1st Edition ,2013.

ABOUT AUTHOR

Fadheela Sabri Abu-AlMash received her B.E degree in computer science from Baghdad University, College of Education in 2001, and received her M.S.C degree in computer science from AL_Mustansiriyah University in 2006. She is currently working in the Ministry of Higher Education & Scientific Research, Scholarships and Cultural Relations Directorate/ Iraq – Baghdad. Her research interests include soft computing.

