A New Approach to Solve Quadratic Equation Using Genetic Algorithm

Bibhas Roy Chowdhury[1], Md. Sabir Hossain[1*], Alve Ahmad[1], Mohammad Hasan[2], Md. Al-Hasan[2]

[1]Chittagong University of Engineering & Technology, Chattogram, Bangladesh.
[2]Bangladesh Army University of Science & Technology, Saidpur, Bangladesh.
sabir.cse@cuet.ac.bd

Abstract. Solving quadratic equation efficiently is a real-world challenge nowadays, due to its wide applications in the task of determining a product's profit, calculating areas or formulating the speed of an object. The general approach of finding the roots of a quadratic equation is not enough efficient due to the requirement of high computation time. Because of the Genetic Algorithm's stochastic characteristics and efficiency in solving problems it can be used to find roots of quadratic equation precisely. In modern athletics reducing the computation time of solving the quadratic equation has been so inevitable where using a genetic algorithm can find a quick solution that doesn't violate any of the constraints and with high precision also. Optimization has been done in the Crossover and Mutation process which has reduced the number of iterations for solving the equation. It reduces the time complexity of the existing approach of solving the quadratic equation and reaches towards the goal efficiently.

Keywords: Genetic Algorithms, Crossover, Mutation, Chromosomal Fitness, Population, Quadratic equation.

1 Introduction

The approach we are going to follow to solve 2nd order linear equation is the Genetic Algorithm [2]. The Genetic Algorithm [2] is an efficient way to solve both unconstrained & constrained optimization problems that are based on natural selection. The Genetic Algorithm [2] repeatedly modifies a population of individual solutions. The basic features on which the Genetic Algorithm [2] stands are:

i) Competition in individual populations for resources and mates.
ii) Successful individuals are allowed to create more offspring.
iii) The propagation of the gene flows from fittest parent to generation.
iv) Survival of the fittest.

The basic outcome of the Genetic Algorithm [2] is a Chromosome and the collection of the chromosomes is known as a population. The process of the fitness function is applied to chromosomes to check their stability and select them for going to the next

stage. With the participation of selected chromosomes in the crossover stage, they produce offspring (child chromosomes) combining with the parent's gene. In the later stage, few chromosomes will undergo the mutation process. After that selected chromosomes which will shift to the next generation for a repeated procedure which will be determined from fitness value that indicates Darwin's theory of evolution [16]. After completing several generations following the above steps, the best and precise value of the mathematical equality problem can be gained. The whole process can be summarized by the following steps:

1. Initialization of random value to each chromosome
2. Evaluation of objective function
3. Selection based on fitness probability
4. Crossover the chromosomes
5. Mutation

We will go through these repeatedly until we get the best value of the chromosome. Hence, we find the best solution for each variable of the equation by finding the best fittest chromosome using the Genetic Algorithm [2].

There exists a local solution of generating two equations from the given quadratic equations containing the two roots and solving those which require severely increased computation time given a large Datasets and requires further data training. Another approach can be found the roots using all the coefficients of variables and constants of the equation and which is also inefficient when a root becomes imaginary. So using the genetic algorithm can be the best approach to overcome these limitations.

The sequential arrangement of the following part of this paper is as follows. Section 2 of the paper contains the background of related works. The proposed Genetic Algorithm model is described in Section 3. Section 4 contains implementation details. And section 5 holds our experimental result. Section 6 gives the conclusion and lastly Section 7 finishes with all possible future works. Lastly, in section 8 we included all the references we used.

2 Related Works

We adapted the idea of solving a quadratic equation in Nayak [1] using the Genetic Algorithm [2]. In this paper, they have used the generalized Schur form of genetic algorithm. To solve the 2nd order linear equation, they were limited to real-valued arithmetic & real-valued variables. So in their solutions, the probability to get a correct answer is high. By using the Hybridized Genetic Algorithm [3], their solution is not the most efficient one.

S.D. Bapon et al. [4] have shown improvement from the existing method using a new algorithm about solving a 1st linear equation using a genetic algorithm. In their algorithm, they encoded solutions as chromosomes. It was presented by Roulette Wheel [5] in the selection after the process of evaluation. They also worked with the mutation rate to get the optimal solution more quickly.

Solving a linear equation using the evolutionary algorithm was also discussed in [6]. This paper also solved the equation but not as efficient as the previous method. The

even structural improvement in the genetic algorithm was discussed in [7]. These improvements assist in reducing complexity. A fixed point is also used in [12].
In the product recommendation system [8], U. Janjarassuk and S. Puengrusme provide a method to get the best guess for the crossover to increase efficiency. In this paper [9] they solved non-linear equations using a genetic algorithm. Their proposed technique is applied to the benchmark problem adopted from Grosan [10]. They have made a comparative analysis to substantiate the effectiveness and reliability of the proposed method in handling nonlinear systems which involved transcendental functions. Sensitivity analysis was also made to validate the selection of parameters of GA. We have also studied some optimization techniques using a genetic algorithm in [13-15].

3 Proposed Methodology

In most of the cases, we get optimal solutions from the genetic algorithm. It's because of some exclusive features of the genetic algorithm like adaptive characteristics. Mutation, crossover, and selection method are behind this algorithm's character.

3.1 Initial population

The process begins with the Population which is nothing but a set of an individual. Individuals are a solution to the problem we want to solve. An individual is characterized by a set of parameters (variables) known as Genes. Genes are joined into a string to form a Chromosome (solution). In a genetic algorithm, the individual's set of genes is represented, in terms of an alphabet, using a string. Most of the time binary values are used (a string of 1s and 0s). First, we encode the gene in a chromosome then we decode it before evaluating their fitness. Only the fittest chromosomes move to the next generation.

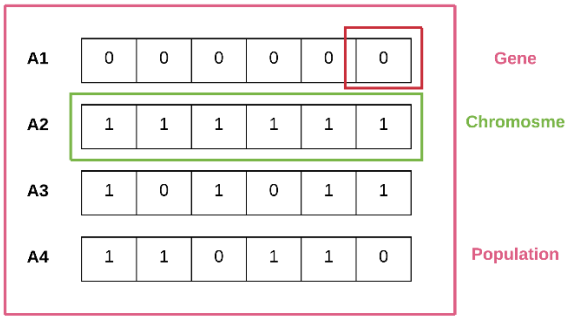

Fig. 1. Initial Population.

3.2 Fitness Evaluation

Evaluation Function also is known as the Fitness Function finds how close is a given solution compared to the optimum solution of the problem. How much fit a solution is stated by it. Each of the solutions is generally represented by a chromosome as a string of binary numbers in the Genetic Algorithm. We have to test these chromosomes and come up with the best solution to solve a given problem. Each of the problems has its fitness function. The fitness function is used depends on the given problem. In our problem the fitness function that we have considered is:

$$\text{Func\_fit} = \frac{1}{1+\text{objective function}}$$

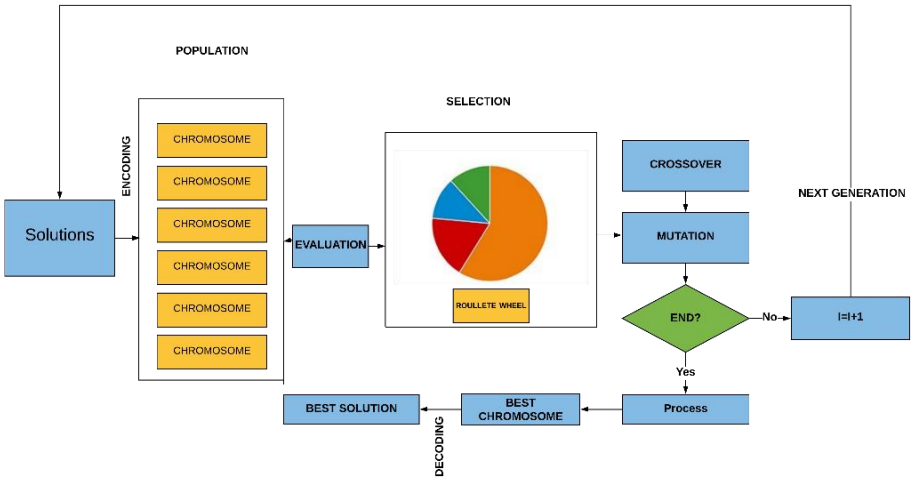

Fig. 2. The methodology of our proposed system.

3.3 Cross-over

Here "par_chromosome" chromosome will be selected as parent chromosome. Here, the set of randomized numbers Random [par_chromosome] < ρc will be selected. Now, as the crossover rate is set to 35% so randomly taken chromosomes which are less than 0.35 are selected for crossover.

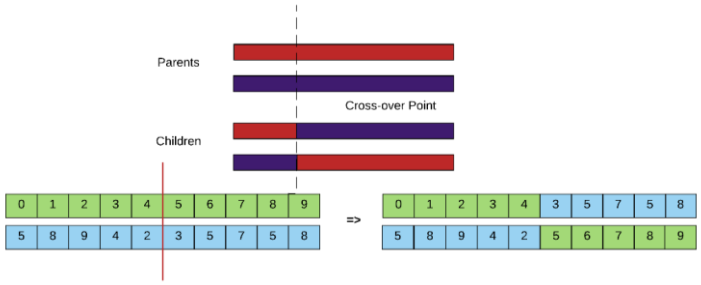

Fig. 3. Cross-over [11].

3.4 Mutation

In the mutation process, a significant change in the chromosome has been done. Here, Total gene = number of genes in Chromosome × the number of populations. The mutation is a process of change and so if the mutation rate or changing rate is kept low (in a range of 0.01 to 0.1) than it provides the fittest result. The mutation rate 12 is defined by ρm. The basic purpose of mutation in GAs is preserving and introducing diversity. The mutation is done during evolution based on a user-definable mutation probability. The probability has to be set low. If it is too high, the search would turn into a primitive random search.

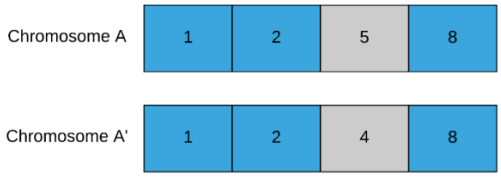

Fig. 4. Mutation.

The Pseudocode for Genetic Algorithm

The parameters of our genetic algorithm are the initial population, Max-Iteration, Best fitness, Max-fitness.

1: Generation =0
2: Initialize Population
3: While Generation < Max Generation
4: Evaluate fitness of population members
5: **for** i to 1 to elitist
5.1: Select best individual
5.2: **end for**
6: **for** i from elites to population_ size
6.1.1: **for** j to 1 from tournament size
6.1.2: select best parents
6.1.3: **end for**
6.1.4: **end for**
6.2.1: **for** k from elites to population_size * (1-mutation rate)
6.2.2: crossover parents → child (Just add this tick mark)
6.3.1: **for** k from population_size*(1-mutation rate) to population_size
6.3.2: mutant parents → child
6.3.3: **end for**
6.2: insert child for next generation's population
6.3: **end for**
7: update current population
8: generation ++
9: **end while**

4 Implementation Details

Let, there is a quadratic equation to be like $x^2 - 8x + 15 = 0$. Here, the Genetic algorithm is used to find the root of the equation x1 and x2. five steps will be needed such as initialization, evaluation, selection, crossing over and mutation to execute the whole process.

4.1 Initialization

At first, we take the total number of 6 chromosomes as the initial population. Then, we initialize the random values of gene x1 in each chromosome.

Chromo [1] = [x1] = [00000010]
Chromo [2] = [x1] = [00000100]
Chromo [3] = [x1] = [00000110]
Chromo [4] = [x1] = [00000111]
Chromo [5] = [x1] = [00001001]
Chromo [6] = [x1] = [00001000]

It can be decoded as follows:

Chromo [1] = [x1] = [2]
Chromo [2] = [x1] = [4]
Chromo [3] = [x1] = [6]
Chromo [4] = [x1] = [7]
Chromo [5] = [x1] = [9]
Chromo [6] = [x1] = [8]

4.2 Evaluation

In this step of evaluation, the value of the objective function for each chromosome is computed.

Func_Objective[1]= Abs($2^2 - 2 * 8 + 15$)=3
Func_Objective[2]= Abs($4^2 - 4 * 8 + 15$)=1
Func_Objective[3]= Abs($6^2 - 6 * 8 + 15$)=3
Func_Objective[4]= Abs($7^2 - 7 * 8 + 15$)=8
Func_Objective[5]= Abs($9^2 - 9 * 8 + 15$)=24
Func_Objective[6]= Abs($8^2 - 8 * 8 + 15$)=15

In the next generation, the fittest chromosomes with higher probability will get selected. To determine fitness probability at first, we determine the fitness function of each chromosome. Then, 1 will be divided by (the objective function of each chromosome+1) to get the fitness probability.

Func_Fit [1] = 1 / (1+**Func_ objective** [1]) = 1/4 = .25
Func_Fit [2] = 1 / (1+**Func_ objective** [2]) = 1/2 = .5
Func_Fit [3] = 1 / (1+**Func_ objective** [3]) = 1/4 = .25
Func_Fit [4] = 1 / (1+**Func_ objective** [4]) = 1/9 = .11

Func_Fit [5] = 1 / (1+**Func_ objective** [5]) = 1/25 = .04

Func_Fit [6] = 1 / (1+**Func_ objective** [6]) = 1/16 = .0625

Total = 1.212

The rule for determining probability of each chromosome is:

Probability[i] = Func_Fit[i] / Total

Probability [1] = .25 / 1.212 = .206	**Probability** [4] = .11/ 1.212 = .090
Probability [2] = .5/ 1.212 = .412	**Probability** [5] = .04 / 1.212 = .033
Probability [3] = .25 / 1.212= .206	**Probability** [6] = 0.0625 / 1.212 = .0515

Seeing the above probabilities, it is seen that Chromo [2] has the highest probability of going to the next generation. A roulette wheel is used for this selection procedure. For the computation in the roulette wheel, we should compute the values of cumulative probability.

C_Probability [1] = .206

C_Probability [2] = 0.206+.412 = .618

C_Probability [3] = 0.206+.412 +.206= .824

C_Probability[4] 0.206+.412 +.206+.090 = .914

C_Probability[5]= 0.206+.412 +.206+.090 +.033 = .947

C_Probability[6]= 0.206+.412 +.206+.090 +.033 +.0515 = .9985

By calculating the cumulative probability of selection step using roulette wheel can be done to generate random number Random which range is in between 0-1 as given below-

Random [1] = .822	**Random** [4] = 0.943
Random [2] = 0.912	**Random** [5] = 0.201
Random [3] = 0.823	**Random** [6] = 0.610

If for example generated a random number is greater than C_Probability [1] and smaller than C_Probability [3] then select Chromo [3] as a chromosome for next generation in the existing population.

New_Chromo[1] = **Chromo**[3]	**New_Chromo**[4] = **Chromo**[5]
New_Chromo[2] = **Chromo**[4]	**New_Chromo**[5] = **Chromo**[1]
New_Chromo[3] = **Chromo**[3]	**New_Chromo**[6] = **Chromo**[2]

Now, the existing chromosomes in the population look like given below:

Chromo [1] = [6]	**Chromo** [4] = [9]
Chromo [2] = [7]	**Chromo** [5] = [2]
Chromo [3] = [6]	**Chromo** [6] = [4]

4.3 Cross-over

The crossover process generally helps to cut a chromosome by selecting a cutting point randomly and joining another chromosome at that point. This process is restrained by using a parameter called crossover-rate which is expressed by ρc.

Here “par_chromosome” chromosome will be selected as parent chromosome. Here, the set of randomized numbers Random [par_chromosome] < ρc will be selected.

Now, the crossover rate will be set to 35%. Now the process will be initialized as follows.
At first,a random number Random is generated as the number of population

Random [1] = 0.069
Random [2] = 0.172
Random [3] = 0.679
Random [4] = 0.437
Random [5] = 0.312
Random [6] = 0.826

Now it is clear that, as Random [1], Random [2], Random [5] have the values less than ρc. So, Chromo [1], Chromo [2] and Chromo [5] are selected for crossing over process.

Chromo [1] >< **Chromo** [2]
Chromo [2] >< **Chromo** [5]
Chromo [5] >< **Chromo** [1]

Now three crossover constants will be selected randomly for three cutting point of three chromosomes. So,
Cut_point [1] = 0
Cut_point [2] = 0
Cut_point [3] = 0
Then for crossover, crossover, parent's gens will be cut at gen number 1, e.g.

Chromo [1] >< **Chromo** [2] = [6] >< [7] = [7]
Chromo [2] >< **Chromo** [5] = [7] >< [2] = [2]
Chromo [5] >< **Chromo** [1] = [2] >< [6] = [6]

After crossover process, the chromosomes are,

Chromo [1] = [7]
Chromo [2] = [2]
Chromo [3] = [6]
Chromo [4] = [9]
Chromo [5] = [6]
Chromo [6] = [4]

4.4 Mutation

In the mutation process, a significant change in the chromosome has been done.
Total gene = number of genes in Chromosome * the number of populations = 1*6 = 6
The mutation is a process of change and so if the mutation rate or changing rate is kept low (in a range of 0.01 to 0.1) than it provides the fittest result. The mutation rate is defined by ρm. Here mutation rate is defined 10% (0.1). So, 10% of the total gen will be mutated. So, number of mutations will be = 0.10 * 6 = .6 ≈1
So, after mutation in the 3rd chromosome which was randomly chosen, the chromosomes will look like this,

Chromo [1] = [7] **Chromo** [4] = [9]
Chromo [2] = [2] **Chromo** [5] = [6]
Chromo [3] = **[1]** **Chromo** [6] = [4]

After mutation, the objective function will be again evaluated by following,

For **Chromo** [1] = [7],
Func_ objective [1] = Abs $(7^2 - 8 * 7 + 15)$ = 8
For **Chromo** [2] = [2]
Func_ objective [2] = Abs $(2^2 - 8 * 2 + 15)$ = 3
For **Chromo** [3] = [1]
Func_ objective [3] = Abs $(1^2 - 8 * 1 + 15)$ = 8
For **Chromo** [4] = [9]
Func_ objective [4] = Abs $(9^2 - 8 * 9 + 15)$ = 24
For **Chromo** [5] = [6]
Func_ objective [6] = Abs $(6^2 - 8 * 6 + 15)$ = 3
For **Chromo** [6] = [4]
Func_ objective [5] = Abs $(4^2 - 8 * 4 + 15)$ = 1

From this evaluation is clear that the objective functions are decreasing in some cases and the lowest value of an objective function will be considered much fitter.
Hereafter 1st iteration fitter Chromosome is: **Chromo [**6**]** = [4]
And this fitter chromosome will undergo the same process of this algorithm. After the next iteration, the value of fitness function will be decreased and after running 6 generations, the fittest chromosome will be obtained for which related objective function becomes 0. So fittest chromosome is: **Chromo** = [5]
After decoding the answer, the result will be transformed like as follows

Chromo= [00000101]

It is expressed that x1=5
Now if the value of x is put in the equation
x1+x2= -b/a (Where a & b are coefficients of x^2 & x)
or, 5 + x2 = 8 (as b= -8 and a =1)
or, x2=3
Now, (x1, x2)=(5,3)
For justification F(x)= $x^2 - 8x + 15$
Then F(5)= $5^2 - 8 * 5 + 15$=0
F(3)= $3^2 - 8 * 3 + 15$=0
So it is clear that the value of these variables generated by GA satisfies the mathematical equality.

5 Experimental Result & Complexity Analysis

5.1 Execution Time Comparison

Results obtained by implementing our proposed algorithm are shown in Table 1. Table 1 shows variations in runtime when ran for three consecutive times.

Table 1. Data table for execution time.

Equations	Execution Time 1st run(second)	Execution Time 2nd run(second)	Execution Time 3rd run(second)
$x^2 - 8x + 15$	0.92	3.277	1.028
$x^2 - 7x + 12$	0.934	2.857	1.011
$x^2 - 2.5x + 1$	2.014	2.995	2.012
$x^2 - 3x + 2$	1.384	3.055	1.023
$x^2 - x - 2$	0.873	3.265	2.995

The total number of generations depends on the number of randomly taken chromosomes. In our implementation, the results we found are shown in Table 2. This table shows changes in generations which are depending on the number of random values.

Table 2. Data table for generations.

Number of random values	Generations
6	6
8	8
5	5
4	4
3	3

As the genetic algorithm moves towards the solution by consecutive crossovers and mutations among the chromosomes so it shows variable execution time for a fixed code, the execution time can be much lower or much higher when we run a single time. So as genetic algorithm code has no specific execution time therefore the efficiency of using the algorithm cannot be compared with other existing methods [Fig. 5].

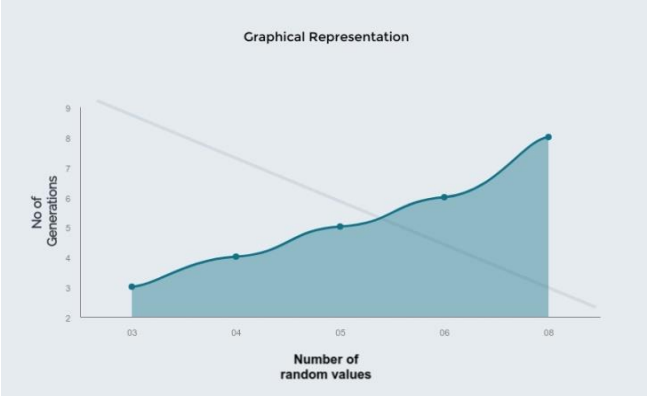

Fig. 5. Graphical representation for given data.

As total no of generation= number of genes in Chromosome * number of populations, a total number of generations and the number of initial population was found the same in our implementation.

5.2 Complexity Analysis

The time complexity of Genetic algorithm depends on three parameters-

1. Fitness function.
2. Selection operator.
3. Variation operator

Among the three parameters, it mainly depends on the fitness function. If the time complexity of GA is evaluated with Big O notation, then it can be O (NG), where N is the size of population and G stands for the number of iterations. On the other hand, if we assume N be the size of the population and L be the length of the genotypes, then for a simple Rastrigin function, the time complexity for the evaluation of the whole generation will be O(NL). If the selection is stochastic then it is needed to sort the population. In this case, time complexity will be O(NlogN), while for tournament selection it will be O(N). If the population is transformed with a crossover and mutation operator, then time complexity will be O(NL).

5.3 Best and Worst-Case Complexity

In Table 1, we have taken several equations and found their computation time. We can see for $x^2 - 8x + 15 = 0$ equation on 1st Run the computation time was least. The reason behind it was the requirement of a few steps of the crossover and mutation process to find out the required root.

On the other hand, in Table 1, for the same equation's 2nd run, the computation time was highest. The only reason was it took a lot of steps in the crossover and mutation process to find the final roots. These were the best and worst time complexity for this implementation.

6 Conclusion

To find an efficient solution of a quadratic equation within acceptable time form was our primary focus of this work. We compared our proposed genetic algorithm approach with an existing method of solving equations. After numerous analysis, we have to come in the conclusion that GAs has an upper-hand for obtaining a solution. After further considering the obtained result and CPU times and comparing them with other based known existing solutions of solving a quadratic equation it can be stated that the GA based proposed approach performs well and much more efficiently.

7 Future Work

In the future, there is a scope for working on crossing over rate. We were not getting expected outcome when both of the roots are imaginary, negative or fractional. So the future recommendation is to implement the solving technique of second and higher-order equations using GA considering these things also.

8 References

1. Nayak, T.: Solution to Quadratic Equation Using Genetic Algorithm. In Proceedings of National Conference on AIRES, Andhra University (2012).
2. Holland, J. H.: Genetic algorithms. In Scientific American 267**,** 66–72 (1992).
3. Li, K., Jia, L., and Shi, X.: An Efficient Hybridized Genetic Algorithm. In Proceedings of IEEE International Conference of Safety Produce Informatization, IICSPI 2018, 118–121(2019).
4. Bapon, S.D, Hossain, M.S., Fahad, M.N.: Improvement of Solving First Order Linear Equations by Adopting Genetic Algorithm. Unpublished Undergraduate Thesis. In Chittagong University of Engineering & Technology, Chattorgram, Bangladesh,(February 2019).
5. Rodríguez, A. & Mendes, B.: Probability, Decisions and Games. Probability, Decisions and Games. In John Wiley & Sons, Inc., *(*2018).
6. Bashir, L. Z.: Solve Simple Linear Equation using Evolutionary Algorithm. In World Scientific News **19,** 148–167 (2015).
7. Chen, T. Y. & Chen, C. J.: Improvements of simple genetic algorithm in structural design. In International Journal for Numerical Methods in Engineering 40**,** 1323–1334 (1997).
8. Janjarassuk, U., Puengrusme, S.: Product recommendation based on genetic algorithm. In 5th International Conference on Engineering, Applied Sciences and Technology (ICEAST), Luang Prabang, Laos, 1-4 (2019).
9. Uddin, M., Mangla, C., Ahmad, M.: Solving System of Nonlinear Equations using Genetic Algorithm. In Journal of Computer and Mathematical Sciences10(4),877-886 (April, 2019)
10. Grosan, C. and Abraham, A.: A new approach for solving nonlinear equations systems. In IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans, 38(3), 698–714 (2008).
11. Riazi, A.: Genetic algorithm and a double-chromosome implementation to the traveling salesman problem. In SN Appl. Sci. 1**,** 1397 (2019).
12. Rovira, A., Valdés, M. & Casanova, J.: A new methodology to solve non-linear equation systems using genetic algorithms. In International Journal for Numerical Methods in Engineering 63, 1424–1435 (2005).
13. Zhang, Y., Jin, W., Hu, Z., Chan, C. W.: A Genetic-Algorithms-Based Approach for Programming Linear and Quadratic Optimization Problems with Uncertainty. In Journal of Mathematical Problems in Engineering, 12, 1024-123X, (2013).
14. Tsutsui, S. & Fujimoto, N.: Solving quadratic assignment problems by genetic algorithms with GPU computation: A case study. In Genetic and Evolutionary Computation Conference, GECCO-2009, 2523-2530(2009), 10.1145/1570256.1570355.
15. Sheta, A. & Turabieh, H.: A comparison between genetic algorithms and sequential quadratic programming in solving constrained optimization problems. In ICGST International Journal on Artificial Intelligence and Machine Learning (AIML), 6, 67-74 (2006).
16. Darwin, C.: On the Origin of the Species. In Darwin 5, 386 (1859).