Bulletin of Electrical Engineering and Informatics
Vol. 9, No. 5, October 2020, pp. 2125~2133
ISSN: 2302-9285, DOI: 10.11591/eei.v9i5.2233

Improvement of genetic algorithm using artificial bee colony

Ali Abdul Kadhim Taher, Suhad Malallah Kadhim
Computer Science department, University of Technology, Baghdad, Iraq

Article Info

Article history:

Received Nov 13, 2019
Revised Jan 4, 2020
Accepted Feb 8, 2020

Keywords:

Artificial bee colony
Genetic algorithm
Random number generation
Travelling salesman problem

ABSTRACT

Genetic algorithm (GA) is a part of evolutionary computing that simulates the theory of evolution and natural selection, where this technique depends on a heuristic random search. This algorithm reflects the operation of natural selection, where the fittest individuals are chosen for reproduction so that they produce offspring of the next generation. This paper proposes a method to improve GA using artificial bee colony (GABC). This proposed algorithm was applied to random number generation (RNG), and travelling salesman problem (TSP). The proposed method used to generate initial populations for GA rather than the random generation that used in traditional GA. The results of testing on RNG show that the proposed GABC was better than traditional GA in the mean iteration and the execution time. The results of testing TSP show the superiority of GABC on the traditional GA. The superiority of the GABC is clear in terms of the percentage of error rate, the average length route, and obtaining the shortest route. The programming language Python3 was used in programming the proposed methods.

Corresponding Author:

Ali Abdul Kadhim Taher,
Computer Science Department,
University of Technology, Baghdad, Iraq.
Email: 0110114@student.uotechnology.edu.iq

1. INTRODUCTION

Artificial intelligence (AI) has been shown great performance in several tasks [1-4]. Genetic algorithm (GA) is an AI algorithm that is a metaheuristic method invented in 1975 by John Holland [5]. GA has a good global search capability through its operation strategy but has a slow convergence speed [6]. GA shows high performance in complex areas because it works repetitively as the research is likely to focus on typical representative area areas found to produce good behavior [7]. Artificial bee colony (ABC) algorithm was proposed by Karaboga in 2005, which is simple in concept and requires very few initialization parameters to adjust. It has properties of fast convergence speed, good quality of solutions and good robustness, etc., but it also has the disadvantages of early convergence and ease of falling into the local optimum [8].

Important problems that test the efficiency of the algorithm are RNG and TSP, which will be adopted in this thesis as a case study, which will be explained later. Random numbers are numbers generated by an operation that unpredictable it's an outcome and cannot be sequentially reliably reproduced. Random numbers are necessary for a variety of applications. For instance, a popular cryptosystem used keys that should be produced in a random mode [9]. TSP is one of the problems known as NP-hard and is a well-known problem in the field of computer science and mathematics. This problem has been used in many fields such as semiconductor manufacturing, logistics, and transportation [10].

The rest of the paper is organized as follows: section 2 presented some related work. In section 3, background theories were displayed. In section 4, described the design of the proposed methods. In section 5, experimental results were evaluated, and in the last section, a brief conclusion was presented.

2. RELATED WORK

There are many pieces of research focusing on improving GA. In Abu-Almash [9] genetic algorithm used to propose a new approach to generate keys. The Five basic tests applied as a fitness function in this experiment for each chromosome. The results proved that the competence of a genetic algorithm for key generation. The optimal representation for the genetic algorithm may be seen by the initial population, size of population, crossover, and mutation. The large population size, little mutation probability, and a high crossover probability.

Alkafaween [11] has presented a new mutation called "IRGIBNNM", that was a hybrid of two mutations which were a random-based mutation and a knowledge-based mutation. The proposed mutation was used to improve the select best mutation (SBM) strategy. The proposed mutation performance was compared with three mutations, additionally to the SBM. The performance of the proposed mutation and the SBM strategy was evaluated on several experiments on twelve TSP instances were conducted. Those instances were taken from the TSPLIB. The experiential results showed the efficiency of the "IRGIBNNM" mutation and the SBM strength where both methods have benefited from the randomization and knowledge provided by the nearest neighbor approach.

Hassanat et al. [12], studied GA was improved by dividing the large-scale TSP problem into small subsets based on regression. This was achieved by using the regression line and its perpendicular line, where cities were grouped into four sub-problems and frequently, each site locates the group in which the city belongs. This process was repeated until the size of the problem became very small or reached a size that cannot be divided. Some famous TSP instances that are derived from TSPLIB were used in experiments. The experiments showed that the proposed method outperformed the other methods using population seeding techniques such as the nearest neighbor based techniques and random techniques regarding mean convergence and error rate.

Alkharji et al. [13] describe a method for using GA to generate random keys for a fully homomorphic encryption scheme (FHE) and check its efficiency. The five statistical tests used in this experiment. This paper explains that the powerful, high-quality, non-repetitive random keys generated by GA will increase the security of FEH schemes, and making it difficult for analysts to break the data. Akter et al. [14] have presented a new crossover operator for solving the TSP. The new crossover consisted of choosing two crossover points and creating new offspring by comparing the cost. The proposed crossover was performed with two traditional crossover operators a sequential crossover operator (SCX) and a triple crossover operator (TCO), and used eight benchmark instances taken from the TSPLIB. The experimental results among the proposed crossover operator, SCX, and TCO showed that the proposed operator has outperformed other operators via providing a solution in less iteration that significantly reducing time and memory.

Gupta et al. [10] have focused on the development of a heuristic technique by combining two common optimization methods: particle swarm (PSO) and GA for TSP. The main inspiration was used to improve a hybrid GA-PSO algorithm to take advantage of PSO such as the self-improvement of individuals and high convergence rate, which compensated the weakness of GA. In the GA-PSO hybrid algorithm, new individuals were generated through the PSO mechanism as well as crossover and mutation operators. The performance of the GA-PSO hybrid algorithm against GA and PSO was used ten TSP standards in terms of finding optimal and execution time. Superior GA-PSO mixed performance between GA and PSO for TSP with concern to the standard, i.e. error rate and average computational time.

3. PRELIMINARIES

3.1. Genetic algorithm

GA is a non-linear, discrete random process that does not require mathematical formulation. Optima are evolving from generation to another [15]. It has grown as important in solving large complex problems that require a long time according to traditional programming techniques, compared to GA, which has a huge amount of alternative solutions, where the solution is often optimal or near to optimal, over time [16].

At first, GA selects parents from an initial population of chromosomes, and then generates offspring using crossover and mutation. It evaluates all chromosomes to determine their fitness using a fitness function. Then, fitness values were used to determine if chromosomes are retained or disposed of. The least adapted chromosomes are eliminated and the most adapted chromosomes are retained in generating new populations according to the principle of survival of the fittest. The new population replaces the old. The processes are repeated while a specific termination condition satisfied [17]. A simple GA is illustrated in Figure 1.

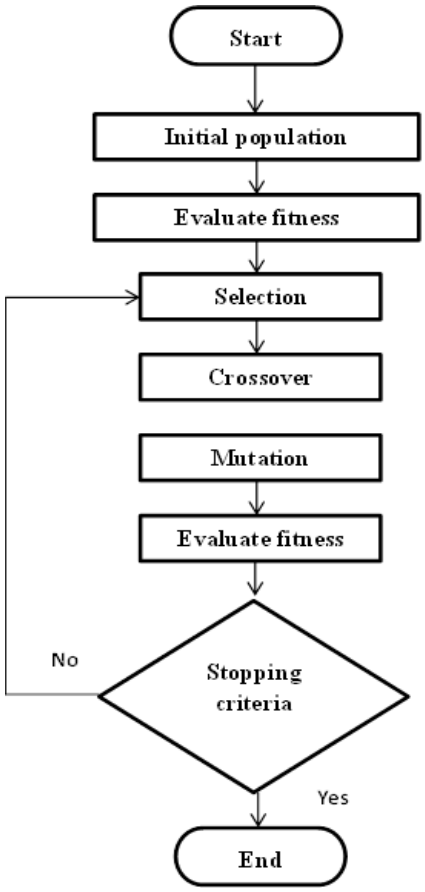

Figure 1. Flow chart of a typical GA

3.2. Artificial bee colony (ABC)

ABC is a method of improving heuristic population-based. The populations in the algorithm represent bees. Every bee searches for the best solution (food resource). It is assumed that each solution is a food resource and the amount of nectar for each resource represents the quality of each solution [18]. ABC is a population-based improvement algorithm that aims to obtain a global minimum [19].

There are two types of bees in the ABC algorithms that are employed and unemployed bees that be consisting of the onlooker bees and the scout bees. The population of the colony usually composed two parts employed that form the first half and the rest include onlooker bees. At first, employed bees exploit the sources of nectar that have been explored, and then provide information to the onlooker bees in the beehive about the location of the food sources they exploit.

After completing the search process by the employed bees, they share information about food sources and their location with the onlooker bees in the dance space. The onlooker bees evaluate information from employed bees and choose the source of food with the probability of nectar. Scouts randomly examine the environment to identify a new food source based on potential external evidence or internal motivation. Algorithm 1 represents the main algorithm of ABC.

```
Algorithm (1): Formal Model of ABC
INPUT: Initial population
OUTPUT: the Best solution
Begin
Initialize phase
Repeat
Employed Bees Phase
Onlooker Bees Phase
Scouts Bees  Phase
          Memorize the best solutions attained till now
Until (Cycle= Maximum cycle number or  a Maximize run time )
End
```

3.3. Travelling salesman problem

TSP is considered an optimization problem [20]. It is a problem that is an easy description tricky solution and classified as a problem that has not been solved in polynomial time. This problem belongs to NP-hard [21]. The major purpose of this problem is to detect the shortest route through a group of cities (starting from city N and ending in the same city) so that each city is visited only once [22].

3.4. Random number generators

Random numbers are a wide field with a strong theoretical and empirical basis. There are many methods used to generate random numbers such as published random number tables and throwing dice. The expansion of digital computers has opened up a new area of inquiry, such as measurement of physical parameters, random numbers of real generators, etc [23]. A valid RNG will generate a sequential series of distributed numbers, at the interval [0, 1]. Using any type of computer algorithm, if we know the initial state of the algorithm which be also called the seed, the sequence of numbers can be determined [24].

3.5. Statistical tests

This section presents several tests aimed at measuring the quality of a generator called a random bit generator. The process is done by taking a sampling of the sequence and presenting it to several statistical tests. Every one of the five tests determined whether a particular attribute sequence is likely to actually display a random sequence [25]. Let $s=s_0, s_1, s_2 \ldots s_{n-1}$ is a binary sequence of lengths n. The statistical tests that are used in this paper are [26]:

- The frequency (mono-bit) test

For a random series of length n, the number 0's should be approximately equal to the number of 1's. The statistic test used in (1):

$$X1 = \frac{(s_0 - s_1)^2}{s} \quad (1)$$

where: s_0, s_1 indicates the numbers of 0's and 1's in s, respectively.

- Serial (two-bit) test

This test determined if the frequency count is 00, 10, 01 and 11 are roughly the same. A statistic test used in (2):

$$X2 = \frac{4}{s-1}(s00^2 + s10^2 + s01^2 + s11^2) - \frac{2}{s}(s0^2 + s1^2) + 1 \quad (2)$$

where s00, s10, s01 and s11 indicate the number of "00", "01", "10", and "11" in s respectively.

- Poker test

Suppose p is a positive integer where $[s/p] \geq 5$, (2^p), and suppose $D=[s/p]$. Divide the sequence s into non-overlapping segments D whose length is p, and assume that s_i is the number of sequence-type appearances of length p, $1 \leq I \leq 2p$. In this test, we determine whether the length p sequence shows approximately the same number of times in seconds, as expected for the random sequence. The statistic test used in (3):

$$X3 = \frac{2^p}{D}\left[\sum_{i-1}^{2^p} s_i^2\right] - D \quad (3)$$

- Run test

The purpose of this test is to determine whether the number is either zeros or numbers from a series of different lengths in the S sequence as expected in a random sequence. The expected number of gaps (or blocks) of length i in a random sequence of length s is $e_i = (s - i + 3)/2i + 2$. Suppose L is equals to the maximum integer i was $e_i \geq 5$. Suppose B_i and G_i are the numbers of block and gap respectively, of length i in n for each i, $1 \leq i \leq L$. The statistic test is calculated by (4):

$$X4 = \sum_{i=1}^{L} \frac{(B_i - e_i)^2}{e_i} + \sum_{i=1}^{L} \frac{(G_i - e_i)^2}{e_i} \quad (4)$$

- Autocorrelation test

In this test, we investigate the relationships between the sequence and its transmitted versions. Suppose d is a constant number, $1 \leq d \leq s/2$. The number of bits in s not equals to their d-shifts is A (d)=$\sum_{i=0}^{n-d-1} s_i + s_{i+d}$, where $\oplus$ indicates the XOR operator. Which follows roughly the distribution

of N (0, 1) if n-d≥10. Since the small values of A (d) are not as predictable as the large values of A (d), we should use a test on both sides. The statistic test is calculated by (5):

$$X5 = 2\left[A(d) - \frac{n-d}{2}\right]/\sqrt{n-d} \quad (5)$$

4. THE PROPOSED METHOD

The proposed method has employed an ABC algorithm to improve the population of GA. TSP and RNG are used as a case study. For RNG problem the five statistical tests were used to compute the fitness (each time any test was failed the counter of fitness incremented), such that the random numbers with minimum fitness was token, and the termination criterion of the proposed algorithm has reached the maximum cycle number or all the population is random numbers passed the tests. In TSP, the fitness function in this work was the minimum distance between cities. Total distance for each TSP route for cities n was calculated according to the coordinates of the cities as follows:

If the coordinates of any two cities i and k are (xi, yi) and (xk, yk), the distance between them was calculated according to the Euclidean distance equation:

$$D_{ik} = \sqrt{(x_i - x_k)^2 + (y_i - y_k)^2} \quad (6)$$

Total tour length f can be expressed as follows:

$$f = \sum_{i=1}^{n-1} D_{ik} + D_{n1} \quad (7)$$

where *n* is the total number of cities. The termination criterion of the proposed algorithm has reached a maximum cycle number.

4.1. Improving GA population using ABC for RNG

This method consists of two stages where the first stage focuses on generating random numbers using ABC while the second stage used GA on the random numbers that are generated in the first stage. Previous hybridization work was improved by generating an initial population by ABC then GA was applied to the initial population. At last, the population produced from GA was compared with the initial population and selects the best population (replaces the original parental population) as the new population to the next generation. Reproduction imitated natural selection. ABC generated numbers randomly within the boundaries of the parameters by using the following equation:

$$X_{ij} = X_j^{min} + rand(0,1)(X_j^{max} - X_j^{min}) \quad (8)$$

where; i=1. . .SN, j=1. . .D. Where SN is the solution number and D is the number of optimization parameters The created number was converted to binary. After this, the fitness function in the proposed algorithm was that the five statistical tests might be assigned to the X_{ij} solution by (9), in Algorithm 2.

$$Fitness_i = \sum f_i \quad (9)$$

where if it the counter value of the solution X_{ij}.

```
Algorithm 2: GABC for RNG
INPUT: key's length, population size
OUTPUT: binary keys (k)
Process:
Step1: Generate random numbers using Eq. 8 to be the initial population (I). /* using
ABC*/
Step2: Evaluate (I) using the five statistical tests to compute their fitness using Eq.9.
Repeat
Step3: Call algorithm (3) that takes I and produce a new solution (S) using GA.
Step4: Evaluate S using the five statistical tests to compute their fitness using Eq. 9.
Step5: Select the best populations between S and I to be a new population (K) according
to their fitness. /*Reproduction operation*/
Step6: Until the termination condition is met. /* reached the maximum cycle number or all
the population is random numbers passed the tests*/
Step7: Return K.
End.
```

Based on the initial population that was generated from the ABC stage GA performed genetic operators to generate new offspring. The genetic process was performed iteratively. The new population was sorted according to its fitness value; the roulette wheel operation was used in the proposed algorithm. An ordered crossover and uniform mutation were used in Algorithm 3.

```
Algorithm 3: GA stage for RNG
INPUT: initial population (I)
OUTPUT: New solutions(S)
Process:
Step1: Sort I according to their fitness value. /* ascending order */
Step2: Select from I the best solutions depending on their fitness value (BS). /* Using
roulette wheel selection method*/
Step3: Perform crossover operation between BS to produce new solutions (NS). /* ordered
crossover is used, with a crossover rate of 0.8*/
Step 4: If there is a mutation rate, perform mutation operation and update NS. /* uniform
mutation was used, with a mutation rate of 0.1*/
Step5: Return NS.
End.
```

4.2. Improving GA population using ABC for TSP

This method consists of two stages in the first stage ABC generates a new route by using (8), and in the second stage, the population generated in the first stage will be using GA operators to find the best route. ABC generated random routes by using the index of cities in the default route within the boundaries of the indices by using (8). After this, (7) was used to calculate the distance for each route. The proposed method to find the shortest path is shown in Algorithm 4.

```
Algorithm 4: GABC for TSP
INPUT: Number of cities, default route(D)
OUTPUT: Best solution(NS)
Process:
Step1: Generate new routes (NR) using Equation (8) in ABC.
Step2: Evaluate the new routes using equation (7).
Repeat
Step3: Call algorithm (5) that takes NR and produces NS using GA.
Step 4: Evaluate NS using equation (7).
Step5: Until the termination condition is met. /* Maximum cycle number*/
Step6: Return NS.
End.
```

At the GA stage, the population produced from the ABC stage was sorted depending on their fitness value. Roulette wheel operation ordered crossover and swap mutation were used in the proposed algorithm to generate a new gene. The GA stage is shown in Algorithm 5.

```
Algorithm 5: GA stage for TSP
INPUT: Initial population (NR)
OUTPUT: New solutions(NS)
Process:
Step1: Sort the NR according to their fitness value. /* ascending order */
Step2: Select from NR the best solutions depending on their fitness value (BS). /* Using
roulette wheel selection method*/
Step3: Perform crossover operation between BS to produce new solutions (NS). /* ordered
Crossover is used, with a crossover rate of  0.85 */
Step4: If there is a mutation rate, perform the mutation operation/*swap mutation, with a
mutation rate of 0.01*/
Step5: Return NS.
```

5. EXPERIMENTAL RESULTS AND DISCUSSION

The proposed method was evaluated in this section. Two experiments were carried out where the first experiment aimed to solve RNG while the second problem aimed to solve TSP. Then the solutions were compared with the traditional GA. All computational experiments were conducted on Intel Core i7-4600U 2.70 GHz machine and coded using Python 3 programming language.

5.1. RNG experiment parameters

The proposed method tested on data comprises 64, 120,128,192,256 and 512 keys' length respectively. The process was executed 5 times with various recommended parameters for the three variations of data. The population size was 10, 20, and 30, respectively, and the maximum iteration number

(MIN) was 100. The rate of the crossover was 0.8 and the rate of mutation was 0.1. The goal was that all keys were random according to the five statistical tests. It is based on the number of iterations to achieve the best solution (7), and the average time execution.

$$\text{Mean}=(\sum_0^n \text{rate})/n \quad (10)$$

Where the rate is the number of iterations to achieve stop condition, and n is the number of run times.

The summary of the simulation is presented in Tables 1-3. Through the tables below, the proposed algorithm showed better results than the traditional GA, as generated random numbers in the least number of iterations and at the lowest execution time, which is what was required to get better solutions in the least time.

Table 1. Comparison of three methods on the main iteration (population size=10)

Key size	Mean by using GA	Time	Mean by using GABC	Time
64	5.6	0.0316	2	0.012
120	15.2	0.1328	6.2	0.0602
128	13	0.1322	10.2	0.0926
192	24.8	0.3172	15.6	0.214
256	31	0.5766	22.2	0.3784
512	56.6	1.8448	44.2	1.4486

Table 2. Comparison of three methods on the main iteration (population size=20)

Key size	Mean by using GA	Time	Mean by using GABC	Time
64	5	0.0424	2.4	0.0304
120	11	0.1612	7	0.1212
128	11.8	0.189	8.2	0.1424
192	23.2	0.5502	17	0.4386
256	29.8	0.9176	22.2	0.724
512	49.2	3.1378	39.2	2.5222

Table 3. Comparison of three methods on main iteration (population size=30)

Key size	Mean by using GA	Time	Mean by using GABC	Time
64	4.6	0.0654	2.6	0.047
120	12.8	0.3098	6.8	0.1826
128	10.6	0.2456	7.8	0.2034
192	19.8	0.6904	16	0.5942
256	33	1.5206	21.2	1.0122
512	53.8	5.115	38.2	3.6634

5.2. TSP experiment parameters

Experiments were performed to calculate the percentage of relative error, the best tour, and the average of the tour. The rate of crossover and rate of mutation was 0.85 and 0.01 respectively. The population size equaled the number of cities, and the number of generations was 5000. The simulation was performed 5 times with various recommended parameters. The percentage of relative error (%) was calculated using (11).

$$\text{Error}(\%) = \frac{\text{best solution}-\text{optimal solution}}{\text{optimal solution}} * 100 \quad (11)$$

These algorithms were tested using five real TSP problems taken from the TSPLIB which include eil51, st70, pr76, eil76, and rd100 (the numbers attached to the problem names represent the number of cities). The default tour is taken from the library of TSPLIB for each instance. Through a Table 4, the proposed algorithm showed better results than the traditional GA in all five instances, which was what is required to get a speed convergence to an optimal solution in the least time. When comparing the proposed algorithm with the GA-PSO algorithm [6] and IRGIBNNM algorithm [7] it is showed that the results obtained from the proposed algorithm were better than the previous two algorithms. Table 5 showed the comparison among the algorithms.

Table 4. The summary of the simulation

Type	Optimal solution	No. of cities	algorithm	Best tour	Worst tour	Average of tour	Error%	Average time
eil51	426	51	GA	441.809	461.636	448.544	0.053	81.743
			GABC	427.050	441.311	435.394	0.002	80.509
st70	675	70	GA	697.434	742.412	718.210	0.033	158.934
			GABC	690.627	720.178	702.590	0.023	154.873
eil76	538	76	GA	578.734	593.785	588.499	0.076	199.709
			GABC	557.911	589.515	569.278	0.037	183.925
pr76	108,159	76	GA	114861.3	119446.8	118085.3	0.062	185.530
			GABC	109087.5	111842.6	110608	0.008	185.837
rd100	7910	100	GA	8922.557	9220.022	9070.083	0.128	287.97
			GABC	8171.62	8954.866	8284.771	0.033	281.483

Table 5. The summary of the simulation

Instance	GABC		GA-PSO		IRGIBNNM	
	Best tour	Average of tour	Best tour	Average of tour	Best tour	Average of tour
eil51	**427.050**	**435.394**	428	437	448	455.3
st70	690.627	**702.590**	**690**	703	733	753.4
eil76	557.911	569.278	----	----	----	----
pr76	**109087.5**	**110,068**	109,383	110,522	----	----
rd100	**8171.62**	**8284.771**	8238	8266	----	----

6. CONCLUSION

In this paper, GA was improved using ABC (GABC) by using ABC to generate an initial population rather than randomly generation that used in traditional GA. Through experimental results ,it was concluded that it using a hybrid between GA with ABC has improved GA in RNG by minimizing the number of iterations in less execution time. And using the five statistical tests has provided a good fitness function for the RNG problem. The final generated keys that were unique (not repeated), random and cryptographically strong (successfully passed the five statistical tests). The result of the proposed method in TSP was compared with traditional GA based on the percentage of relative error rate and the average of the tour's average. Results showed GABC was better than traditional GA with a lower error rate and high convergence rate.

REFERENCES

[1] L. Alzubaidi, M. A. Fadhel, S. R. Oleiwi, O. AlShamma, and J. Zhang, "DFU_QUTNet: Diabetic foot ulcer classification using novel deep convolutional neural network," *Multimedia Tools and Appications*, vol. 79, pp. 15655-15677, 2020.

[2] O. Al-Shamma, M. A. Fadhel, R. A. Hameed, L. Alzubaidi, and J. Zhang, "Boosting convolutional neural networks performance based on FPGA accelerator," *International Conference on Intelligent Systems Design and Applications*, vol. 940, pp. 509-517, 2018.

[3] L. Alzubaidi, O. Al-Shamma, M. A. Fadhel, L. Farhan, and J. Zhang, "Classification of red blood cells in sickle cell anemia using deep convolutional neural network," *International Conference on Intelligent Systems Design and Applications*, vol. 940, pp. 550-559, 2018.

[4] L. Al-Zubaidi, "Deep learning based nuclei detection for quantitative histopathology image analysis," Doctoral Dissertation, University of Missouri, Columbia, 2016.

[5] I. Chaari, A. Koubâa, S. Trigui, H. Bennaceur, A. Ammar, and K. Al-Shalfan, "Smartpath: An efficient hybrid ACO-GA algorithm for solving the global path planning problem of mobile robots," *Int. J. Adv. Robot. Syst.*, vol. 11, no. 7, pp. 1-15, 2014.

[6] C. Liu and L. H. Fan, "Conformal array pattern synthesis using a hybrid GA/ABC algorithm," *2015 International Conference on Artificial Intelligence and Industrial Engineering*, pp. 250-253, 2015.

[7] P. Vivekanandan and R. Nedunchezhian, "A new incremental genetic algorithm based classification model to mine data with concept drift," *J. Theoretical and Applied Inf. Technol.*, vol. 21, no. 1, pp. 36-42, 2010.

[8] D. Karaboga, "An idea based on honey bee swarm for numerical optimization," *Technical Report-Tr06, Erciyes University, Engineering Faculty, Computer Engineering Department*, pp. 1-10, 2005.

[9] F. S. Abu-Almash, "Apply genetic algorithm for pseudo random number generator," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 6, no. 8, pp. 8-19, 2016.

[10] I. K. Gupta, S. Shakil, and S. Shakil, "A hybrid GA-PSO algorithm to solve traveling salesman problem," *Computational Intelligence: Theories, Applications and Future Directions-Volume I*, vol. 798, pp. 453-462, 2019.

[11] E. Alkafaween and A. B. A. Hassanat, "Improving TSP solutions using GA with a new hybrid mutation based on knowledge and randomness." *arXiv:1801.07233v1*, pp. 1-18, 2018.

[12] A. B. Hassanat, V. B. S. Prasath, M. A. Abbadi, S. A. Abu-Qdari, and H. Faris, "An improved genetic algorithm with a new initialization mechanism based on regression techniques," *Information*, vol. 9, no. 7, pp. 1-30, 2018.

[13] M. Alkharji, M. Al Hammoshi, C. Hu, and H. Liu, "Genetic algorithm based key generation for fully homomorphic encryption," *Proceedings of 16th Annual Security Conference*, pp. 1-11, 2017.

[14] S. Akter, N. Nahar, M. ShahadatHossain, and K. Andersson, "A new crossover technique to improve genetic algorithm and its application to TSP," *2019 International Conference on Electrical, Computer and Communication Engineering (ECCE)*, pp. 1-6, 2019.
[15] K. Sörensen and F. W. Glover, "Metaheuristics," *Encyclopedia of Operations Research and Management Science*, vol. 62, pp. 960-970, 2013.
[16] J. Holland, "Adaptation in natural and artificial systems: An introductory analysis with application to biology control, and artificial intelligence," University of Michigan, 1975
[17] K-F. Man, K-S. Tang, and S. Kwong, "Genetic algorithms: Concepts and applications [in engineering design]," *IEEE Trans. Ind. Electron.*, vol. 43, no. 5, pp. 519-534, 1996.
[18] J. Kennedy, "Particle swarm optimization," *Encycl. Mach. Learn.*, pp. 760-766, 2011.
[19] T. D. Seeley, "The wisdom of the hive Cambridge," *MA Belkn. Press Harvard Univ. Press, Google Scholar*, 1995.
[20] T. Rashid and S. Abdullah, "A hybrid of artificial bee colony, genetic algorithm, and neural network for diabetic mellitus diagnosing," *ARO-The Sci. J. Koya Univ.*, vol. 6, no. 1, pp. 55-64, 2018.
[21] D. S. Johnson and L. A. McGeoch, "The traveling salesman problem: A case study in local optimization," *Local search Comb. Optim.*, vol. 1, no. 1, pp. 215-310, 1997.
[22] A. Hassanat and E. Alkafaween, "On enhancing genetic algorithms using new crossovers," *International Journal of Computer Applications in Technology*, vol. 55, no. 3, pp. 202-212, 2017.
[23] M. Olteanu and N. Paraschiv, "The influence of random numbers generators upon genetic algorithms," *2013 IEEE INISTA*, pp. 1-5, 2013.
[24] D. Knuth, "The art of computer programming volume 2: Seminumerical algorithms," Addison-Wesley Publishing Company, 1981.
[25] A. J. Menezes, J. Katz, P. C. Oorschot, and S. A. Vanstone, "Handbook applied of cryptography," CRC Press, 1996.
[26] G. Carter, "Statistical tests for randomness," *Proccedings of the Workshop on Stream Ciphers. Report*, vol. 89, no. 1, 1989.

BIOGRAPHIES OF AUTHORS

Ali AbdulKadhim Taher received his B.E degree in computer science from University of Technology in 2010, received his M.S.C degree in computer science from University of Technology in 2020. He is currently working in the Ministry of Higher Education & Scientific Research, University of Wasit/ Iraq-Wasit

Suhad Malallah Kadhim received her B.E degree in computer science from University of Technology in 1994, received her M.S.C degree in computer science from University of Technology in 1997, and received her Ph.D. degree in computer science from University of Technology in 2003. She is currently working in University of Technology/Iraq-Baghdad.