Analysis and comparison of a proposed mutation operator and its effects on the performance of genetic algorithm

Sami Ullah, Abdus Salam, Mohsin Masood
Faculty of Engineering and Computing, Abasyn University, Peshawar Campus, Pakistan

Article Info

Article history:

Received Jun 29, 2021
Revised Dec 8, 2021
Accepted Dec 22, 2021

Keywords:

Genetic algorithm
Inverse operator
Max one problem
M-mutation operator
Mutation operators
Shuffle operator
Swap operator

ABSTRACT

Genetic algorithms (GAs) are dependent on various operators and parameters. The most common evolutionary operators are parent selection, crossover, and mutation. Each operator has broad implementations with its pros and cons. A successful GA is highly dependent on genetic diversity which is the main driving force that steers a GA towards an optimal solution. Mutation operator implements the idea of exploration to search for uncharted areas and introduces diversity in a population. Thus, increasing the probability of GA to converge to a globally optimum solution. In this paper, a new variant of mutation operator is proposed, and its functions are studied and compared with the existing operators. The proposed mutation operator as well as others such as m-mutation, shuffle, swap, and inverse are tested for their ability to introduce diversity in population and hence, their effects on the performance of GA. All these operators are applied to Max one problem. The results concluded that the proposed variant is far more superior to the existing operators in terms of introducing diversity and hence early convergence to an optimum solution.

Corresponding Author:

Sami Ullah
Faculty of Engineering and Computing, Abasyn University
Peshawar Campus, Ring Road, Peshawar, Pakistan
Email: samiullahpmp@gmail.com

1. INTRODUCTION

Genetic algorithm (GA) is a metaheuristic approach [1] to solve optimization problems [2]. Some papers have suggested its applications in NP-complete problems [3], [4]. The concept of GA was first introduced by Holland [5], influenced by Darwin's theory of evolution. GA is a good option to solve problems with a wide search space that would otherwise take an unreasonably long time by using traditional computing methods. GA can parse through a wide range of possible solutions by narrowing down the search space, guided by various operators. GAs are based on the Darwinian concept of survival of the fittest [6]. Good genetic makeup encourages better adaptation capabilities, which increases the chances of these genetically superior individuals to live longer and hence reproduce more offspring. GA might not necessarily provide a perfect solution; however, it can provide near optimum solutions in polynomial time.

GA is population-based [7] where each individual of the population represents a solution to a given problem. All these individuals have virtual chromosomes or DNA that define their attributes. The length and encoding of DNA are dependent on the type of problem that needs to be solved. The process involves applying crossover to produce new off springs which shuffle genes and then applying random changes to these genes through a mutation operator. Selection criteria drive the evolution by increasing the probability of better members in a population to reproduce more and hence increasing the chance of retaining good genes in the gene pool. The potential of GA has become apparent recently. NASA has used genetic algorithms for the creation of satellite antennas as described in the book [8] and for project cost estimation [9]. Airlines also use GAs to help create flight schedules [10], and crew management. Numerous other wireless applications of GAs have been explored by researchers in papers [11], [12] moreover, GA has promising results in routing and spectrum assignments (RSA) [13]-[15]. Researchers have even recently discovered a potential use of GA in cryptanalysis [16], recruitment systems [17], course scheduling [18], and various other fields [19], [20]. The pseudo code of GA is presented in Figure 1.

```
Require: set Population Size, Max Iteration, Max Fitness
        Generate initial random population
while $i \leq MaxIteration \wedge fitness \leq MaxFitness$ do
        Calculate Fitness
        Selection Crossover
        Mutate
        Replace the old population with a newly created one
end while
return Individual with Maximum fitness
```

Figure 1. Pseudo code for genetic algorithm

There are various operators and parameters in GA that influence its performance. The commonly used operators are parent selection, crossover, and mutation. The parent selection operator selects individuals with better fitness from a population for reproduction. Selection operators exploit already existing good genes of better individuals in the population and ensure that there is a higher chance for these fit individuals to pass on their superior genes to future generations. The efficiency of common selection operators is discussed in papers [21]-[24]. Hussain and Muhammad [25] propose a new split ranked selection operator and examines its effects on GA to converge to a solution for travelling salesman problem (TSP). Various researchers have forged new selection operators for different problems like researcher Kaya suggests a new operator [26] and studies its effects on the production cost of beams. However, different variants of genetic algorithms have been suggested by various researchers, some mix other heuristic algorithms to create a hybrid algorithm GA [27], while others have modified some operations within the GA [28].

The crossover operator combines genes of two or more parents to produce new offsprings. There are numerous variants of this operator, and it is discussed in depth in papers [29], [30]. Crossover operator ensures incorporation of good genes from parents to off-springs. Manzoni *et al.* [31] propose new variants of this operator for use in cryptography and combinational designs. Lim *et al.* [32] discusses both exploitation and exploration zones for crossover operators and their effects on GA's performance. Masrom *et al.* [33] discusses various parametric settings and concludes that a higher crossover rate provides better results.

A mutation operator is used to introduce diversity in a population. It randomly changes the genes of an individual to reflect natural mutation which might cause an organism to thrive and to pass on the good genes. The Mutation operator's function is to steer the GA into unexplored search areas, increasing the diversity of a population [34], [35], and the probability to find a globally optimum solution. For genetic algorithms to be effective there should be a balance between their exploration and exploitation capabilities which is discussed in detail [36] and device a new greedy mutation operator for TSP. Rares [37] explores the effects of mutation operator and diversity on the convergence rate of GA to find an optimum solution. A higher mutation rate has been shown to counteract the exploitation process of selection and crossover operators, which ensures the accumulation of good alleles in future generations [38]. On the other hand, a very low mutation rate will reduce the exploration effects and the genes that might be useful will never be tested by the GA. Milton *et al.* [38] also suggest the use of information theory to identify individuals with low information density and apply mutation to those individuals, which produces better results. Cervantes and Stephens [39] examines the optimum mutation rate concerning different parameters and finds the optimum rate to be $1/N$, where N is the length of the chromosome. However, Ochoa [40] show many factors influence the optimal mutation rate and each problem warrants different parametric settings of mutation operation to be effective. Haupt [41] and Zhang *et al.* [42] investigate the correlation between population size and mutation rate. These studies suggest that a low population size benefits from a high mutation rate and vice versa. Seeding population diversity influences the convergence performance of GA. Papers [43]-[45] examine seeding population techniques for TSP, however Hassanat [46] suggests the regression seeding technique as more efficient.

Bit flip is a type of mutation operator. As described in paper [47], bit flip selects one or more random genes and flips the value of that gene. If the gene's value is 1 it flips to 0 and vice versa. It is commonly used in binary encoded genomes such as Max one problem. The shuffle operator is another commonly used mutation operator which is also used in permutation problems [48]. In shuffle mutation, a subset of genes is picked, nd their positions are scrambled or shuffled randomly Figure 2 shows pseudo code of shuffle operator. A new variant of shuffle mutation operator is m-Mutation [49], which has shown promising results in TSP. Its function is based on three cut points on chromosomes hence dividing the sequence into multiple segments. The sequence of the middle two segments is reversed. These two segments are then rearranged in the original chromosome i.e., the first one attached in the front of the chromosome and the second one appended at the chromosomes, thus forming mutation, its functionality can be seen in Figure 3. Another example of a mutation operator is Inverse mutation. It works by selecting a section of genes in chromosomes and reversing the order of those genes, its pseudo code is displayed in Figure 4. Swap mutation as shown in Figure 5, is also widely used. In swap mutation, two genes on chromosomes are selected and their positions are swapped [50]. Fitness function is one of the critical parameters, which helps to decide the fate of an individual in a population as described in the paper [51]. Each problem warrants a different fitness function for it to run its processes. It gives each individual a fitness score that depicts the individual's suitability or superior genetic makeup. Selection operators, select parents for crossover, are highly dependent on the fitness function. Hence, the fitness function helps to ensure those fitter individuals get a higher chance to reproduce and pass on their good genes to the next generations. Harun and Ibrahim [52] and Xiaoqun [53] show how fitness function can vary to address different problems.

```
Require: Mutation rate
 n = Random number between 0 and 1
 Select two random positions A & B
 if n < Mutation Rate then
 for i = B ;i > A; i − − do
         Generate random position between A & i
         Switch position of A & i
 end for
 end if
return Mutated Genes
```

Figure 2. Shuffle mutation

```
Require: Mutation rate
 if n < Mutation Rate then
         Swap elements in Position 1 & 2 with 3 & 4 Inverse
         position 1 & 2
         for i = 6; i ≥ 4; i − − do
                 Copy to New Array
         end for
         Shift 3 positions of genes from 7 to n, to a new position
         7 - 3 to n-3;
         Append the values of the New Array to the last position
         in the chromosome;
 end if
 return Mutated Gene
```

Figure 3. M-mutation

```
Require: Mutation rate
while i < chromosome lenght do
         n = Random number between 0 and 1;
if n < Mutation Rate then
         Mutate current gene
         i + +
else
         i + +
 end if
 end while
 return Mutated Genes
```

Figure 4. Inverse mutation

```
Require: Mutation rate
 n = Random number between 0 and 1;
 if n < Mutation Rate then
     Select random gene A
     Select random gene B
     Swap values of A and B
 end if
 return Mutated Genes
```

Figure 5. Swap mutation

2. PROPOSED METHOD

A new mutation operator is proposed in this paper. This proposed variant of mutation operator works by applying mutation to every single gene on the entire length of a chromosome. It is based on the mutation rate provided at the initialization of GA. This surges the overall chances of mutated genes in a chromosome. Hence the mechanism of the proposed operator is based on the length of a chromosome, the longer the length, the higher the probability to have an increased number of mutated genes as opposed to other operators which are independent of length. Theoretically in the proposed operator, any gene in the chromosome might be selected for mutation regardless of its position. This increases the diversity which in turn elevates the likelihood of GA to venture into unexplored areas, hence increasing the chance to find a globally optimal solution.

3. RESEARCH METHOD

In this research, a proposed mutation operator is tested against several known operators such as m-mutation, swap, shuffle, inverse, and with no applied mutation in the Max One problem. The applied methodology is to test how each of them increases the diversity of a population and its effects on GA to converge to an optimal solution in the Max One problem. Diversity is the measure of variety between the objects compared. The diversity of a population can be represented by the following (1).

$$Diversity = Fitness\ of\ best\ individual - fitness\ of\ worst\ individual \qquad (1)$$

As the goal of max one problem is to have all 1s in a chromosome, so the fitness of an individual is based on the number of 1s in its chromosomes. Fitness function for individuals in a population is determined by the (2). With this function, the individuals who have more 1s tend to get higher scores. The exponentiation of the fitness function is added in the formula, which increases the fitness score with a higher ratio than the linear function. Hence, the addition of a positive one allele increases the fitness exponentially as compared to a linear function. This increases the probability of selection of individuals with a higher number of 1s for crossover operation. This in turn, leads to early arrival to an optimum solution.

$$Fitness\ = \left(\frac{Number\ of\ 1's}{Total\ lenght\ of\ chromosome}\right)^{10} \qquad (2)$$

Seeding population has substantial effects on convergence to a solution [10]. For consistency of results, the seeding population is the same for all the tests. The seeding and subsequent populations have only ten members, each has a binary value string with a chromosome length of 20 characters. The individuals have the same chromosome as tested in paper [47]. Following are the chromosomes of 10 individuals which are tested in this paper.

P1 = [0 1 0 0 0 1 0 1 0 1 0 1 0 0 1 0 1 0 1 0]	P6 = [0 1 0 0 1 0 1 0 0 1 0 0 1 0 1 0 0 0 1 1]
P2 = [1 0 1 0 1 1 0 1 0 1 0 0 0 1 0 1 1 1 0 1]	P7 = [0 1 0 0 1 0 1 0 0 1 1 1 1 1 0 1 0 1 0 0]
P3 = [1 1 1 0 1 0 0 1 0 0 1 0 1 0 0 1 0 1 0 0]	P8 = [0 1 0 0 1 1 0 1 1 0 1 0 1 0 0 1 1 1 1 0]
P4 = [0 1 0 1 0 0 1 0 0 1 1 1 0 1 0 1 0 0 1 0]	P9 = [0 1 0 0 1 0 1 0 0 1 0 1 0 0 0 1 1 1 1 0]
P5 = [0 1 0 0 1 0 1 0 0 1 0 1 0 1 0 0 1 0 0 0]	P10 = [0 1 0 0 1 1 0 1 0 0 0 0 1 0 1 0 0 1 0 1]

As small population and a higher mutation rate have positive effects on the performance of GA, the proposed mutation variant introduces more mutation at lower probability settings [41], [42]. This mutation operator works by parsing through all the genes in a chromosome and applies mutation based on the given probability on each gene. This increases the probability of at least one mutation per chromosome which can be presented by (3). Where n is the length of the chromosome and P is the mutation rate. Moreover, every single gene has the probability to mutate regardless of its position in a chromosome. This makes the proposed operator fractionally slower for a very long n, with a time complexity of $O(n)$. The Shuffle and Inverse mutation operators that are tested, select a particular segment in a chromosome for mutation. Which in turn narrows the range of the set of mutated genes. However, Shuffle and inverse mutation operators still select random segments based on the length of the chromosome, so their worst-case scenario may be the mutation of an entire chromosome that is still $O(n)$. In contrast swap and m-Mutation are independent of chromosome size and select a fixed number of genes for mutation, making them fractionally faster than the rest of the tested operators in time complexity. Swap mutation mutates only two genes and m-Mutation reshuffles 5 genes regardless of the length of the chromosome, having a time complexity of $O(1)$.

$$\sum_{i=1}^{n} P \tag{3}$$

The (3) shows, the proposed mutation operator can incorporate an increased number of mutated genes as compared to other mutation operators at same the mutation rate. Other parameters of the proposed system are detailed in Table 1. Each operator is tested for hundred times and the results are ranked based on the parameters such as minimum generations to converge, maximum generations to converge, total failures, average, and median. Median is a good measure of central value as it is not influenced by outlying data points. For benchmark purposes, a hundred tests are also conducted without any mutation operator. The pseudo-code for the proposed mutation operator is described in Figure 6, while the pseudo codes for various mutation operators are given below.

Table 1. Parameters of experiment

Parameter	Value
Number of generations	500
Population size	10
Mutation probability	0.01%
Chromosome length	20 characters
Crossover probability	100%
Crossover method	Variable single point
Selection method	Ranked roulette wheel

```
Require: Mutation rate
while i < chromosome lenght do
        n = Random number between 0 and 1;
if n < Mutation Rate then
        Mutate current gene
        i + +
else
        i + +
end if
end while
return Mutated Genes
```

Figure 6. Proposed mutation pseudo code

4. RESULTS AND DISCUSSION

Overall, 600 tests were conducted and for each mutation operator, 100 tests were performed. Each test ran for 500 generations. If the result was not provided within 500 generations, it was marked as a failure. Table 2 displays the performance of GA. The proposed mutation operator, m- mutation, and shuffle provided answers in every instance within 500 generations. Among these, the proposed operator outperformed the m-mutation and shuffle operators. The GA converged in the lowest recorded generations using the proposed mutation operator in just 19 generations, and its worst convergence rate is 93, the best among all the tests. The median of the proposed operator was the lowest recorded among others, at 40.5. Figure 7 shows the proposed operator introduced diversity earlier in the generations than the rest of the operators. The diversity reached 40 within 12 generations and peaked at 80 in 19 generations, at this point, it provided the solution. This increase in diversity can be attributed to the fact that all genes have an equal probability of mutation. This certainly increases the diversity at a higher rate as compared to other mutation operators.

GA with shuffle operator worked slower than the proposed mutation operator. The shuffle operator provided its best result in 21 generations. However, the median was high at 79 generations as compared to the proposed operator. Its performance is attributed to the random reshuffle of a random subset of chromosomes, which promotes better alleles to emerge with the help of a crossover operator.

Table 2 shows m-Mutation operator performed better as compared to inverse and swap operators, with minimum generation at 24 however, its worst result is significantly higher than the proposed and shuffle operator at 346 generations, with the second-highest median of 108.5. The increase in diversity of the m-mutation operator was relatively slow as compared to the proposed and shuffle operators, which is apparent in Figure 1. However, the diversity steadily increased with each generation. Two dips were at 14th generation and 18th generation. This indicates loss of good alleles which can be the result of how this mutation operator performs. The m-mutation operator shifts gene values to the start and at the end of the chromosome with a narrow range of only 5 gene swaps. This effect will not be noticeable in small-length chromosomes, but as the length of the chromosome increases the rate of incorporation of diversity will get slower and it will significantly affect the GA's capabilities to provide answers.

Swap and inverse failed 60 and 15 times, respectively. Inverse performed better than swap in both minimum and maximum generations and its median at 265 is lower than swaps median of 378. As compared to swap, inverse started to significantly increase diversity around 35 generations and continued to add diversity and provided an answer in 54 generations with a diversity of 40. Swap, however, could not significantly add diversity within 55 generations, the highest it attained was 5. Since the swap mutation operator selects only two genes to mutate, the accumulation of good alleles is very slow. Due to this, its median was the worst recorded among the others at 378.

The tests without any mutation operator performed worst of all and did not return results in any instance. This can be attributed to low population size and loss of better alleles. Parent selection is used for exploitation, which is to select individuals with better fitness and produce new offspring using a crossover operator. GAs with no mutation cannot introduce new genes with only a crossover operator. So, no new search areas are explored. Such GAs is highly dependent on existing diversity in the population, which can only happen when the population size is very large.

Table 2. Summary of GA convergence

	Suggested	Inverse	m-Mutation	Shuffle	Swap	No mutation
Failure	0	15	0	0	60	100
Minimum	19	54	24	21	230	501
Maximum	93	499	346	248	491	501
Average	44.63	245.7	117.4	91.6	377	501
Median	40.5	265	108.5	79.5	378	501

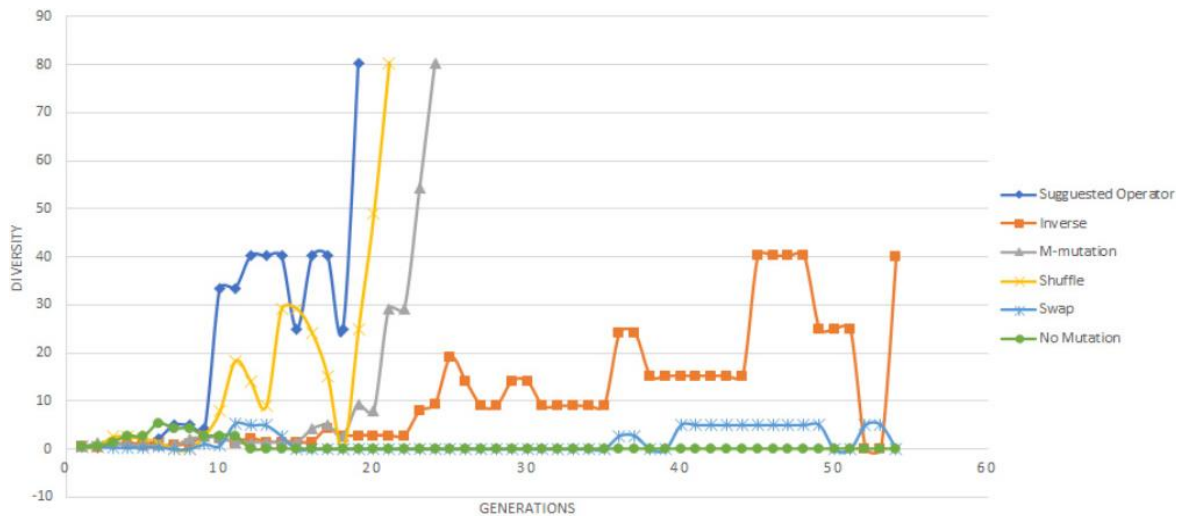

Figure 1. Diversity performance of mutation operators

5. CONCLUSION

The success of GAs is highly dependent on the choice of mutation operator for a particular problem and its ability to introduce diversity in a population. The m-mutation operator only shuffles 5 genes in chromosomes as compared to shuffle mutation and the proposed mutation operator in this research. The shuffle and proposed operators increase the overall likelihood of mutation by flipping more genes at a lower mutation rate than the rest of the operators. Thus, shuffle and proposed variant performed better by converging the GA at fewer generations with better incorporation of diversity.

The lower diversity assimilation of m-mutation can be attributed to the fact that it only mutates fixed 5 genes. Its effects may not be noticeable in a small chromosome, but as the size of a chromosome increases, the performance of fixed length gene mutation operators will suffer adversely as opposed to the proposed operator which incorporates diversity more efficiently throughout the length of a chromosome. Similarly, swap mutation only mutates two genes in the chromosome which explains its even slower diversity rise.

The probability of mutation for the proposed mutation operator is dependent on the length of a chromosome. In the proposed mutation operator, there is a higher chance for a chromosome to have more mutated genes. This in turn increases the diversity in a population. While in the other tested mutation operators, the chances of genetic mutation are independent of the chromosomal length. As the mutation probability of genes is fixed at the initialization of GA, these operators are not able to incorporate diversity more efficiently as compared to the proposed mutation operator. The better performance of the proposed mutation operator can also be attributed to the fact that every single gene has a chance to be mutated instead of a selected gene segment of a chromosome, as in the case of other operators. Hence it is concluded that the proposed mutation operator is far more superior than the tested operators in terms of zero failure rate, the best overall recorded average, and median. The proposed mutation operator outperformed the rest in both the minimum and the maximum number of generations to find a solution. This led to early convergence to a globally optimal solution hence, Increasing the efficiency of GA to find better solutions in less time. The proposed operator might be beneficial in permutation problems such as scheduling, fleet allocations, assignment allocation, RSA et cetera. Further research may be conducted to check its efficiency in real life problems and proposed mutation operator's performance with different population sizes.

REFERENCES

[1] S. Mirjalili, "Genetic Algorithm," *Studies in Computational Intelligence*. Springer International Publishing, pp. 43–55, 2018, doi: 10.1007/978-3-319-93025-1_4.

[2] S. S. Juneja, P. Saraswat, K. Singh, J. Sharma, R. Majumdar, and S. Chowdhary, "Travelling Salesman Problem Optimization Using Genetic Algorithm," *2019 Amity International Conference on Artificial Intelligence (AICAI)*. IEEE, 2019, doi: 10.1109/aicai.2019.8701246.

[3] G. Panchal and D. Panchal, "Solving NP hard problems using genetic algorithm," *International Journal of Computer Science and Information Technologies*, vol. 6, no. 2, pp. 1824–1827, 2015.

[4] B. H. Arabi, "Solving NP-complete Problems Using Genetic Algorithms," *2016 UKSim-AMSS 18th International Conference on Computer Modelling and Simulation (UKSim)*. IEEE, 2016, doi: 10.1109/uksim.2016.65.

[5] J. H. Holland, "Adaptation in Natural and Artificial Systems." The MIT Press, 1992, doi: 10.7551/mitpress/1090.001.0001.

[6] G. J. Balady, "Survival of the Fittest — More Evidence," *New England Journal of Medicine*, vol. 346, no. 11, pp. 852–854, 2002, doi: 10.1056/nejm200203143461111.

[7] S. Mirjalili, J. Song Dong, A. S. Sadiq, and H. Faris, "Genetic Algorithm: Theory, Literature Review, and Application in Image Reconstruction," *Nature-Inspired Optimizers*. Springer International Publishing, pp. 69–85, 2019, doi: 10.1007/978-3-030-12127-3_5.

[8] J. D. Lohn, G. S. Hornby, and D. S. Linden, "An Evolved Antenna for Deployment on Nasa's Space Technology 5 Mission," *Genetic Programming Theory and Practice II*. Springer-Verlag, pp. 301–315, doi: 10.1007/0-387-23254-0_18.

[9] A. F. Sheta, "Estimation of the COCOMO Model Parameters Using Genetic Algorithms for NASA Software Projects," *Journal of Computer Science*, vol. 2, no. 2, pp. 118–123, 2006, doi: 10.3844/jcssp.2006.118.123.

[10] N. F. Mohamed, N. A. Mohamed, N. H. Mohamed, and N. Subani, "Comparison of two hybrid algorithms on incorporated aircraft routing and crew pairing problems," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 18, no. 3, p. 1665, 2020, doi: 10.11591/ijeecs.v18.i3.pp1665-1672.

[11] U. Mehboob, J. Qadir, S. Ali, and A. Vasilakos, "Genetic algorithms in wireless networking: techniques, applications, and issues," *Soft Computing*, vol. 20, no. 6, pp. 2467–2501, 2016, doi: 10.1007/s00500-016-2070-9.

[12] M. Hamid Hassan *et al.*, "A general framework of genetic multi-agent routing protocol for improving the performance of MANET environment," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 2, p. 310, 2020, doi: 10.11591/ijai.v9.i2.pp310-316.

[13] D. T. Hai, M. Morvan, and P. Gravey, "Combining heuristic and exact approaches for solving the routing and spectrum assignment problem," *IET Optoelectronics*, vol. 12, no. 2, pp. 65–72, 2018, doi: 10.1049/iet-opt.2017.0013.

[14] D. T. Hai, "Multi-objective genetic algorithm for solving routing and spectrum assignment problem," *2017 Seventh International Conference on Information Science and Technology (ICIST)*. IEEE, 2017, doi: 10.1109/icist.2017.7926753.

[15] D. T. Hai and K. M. Hoang, "An efficient genetic algorithm approach for solving routing and spectrum assignment problem," *2017 International Conference on Recent Advances in Signal Processing, Telecommunications & Computing (SigTelCom)*. IEEE, 2017, doi: 10.1109/sigtelcom.2017.7849820.

[16] M. S. A. Forhad, M. S. Hossain, M. O. Rahman, M. M. Rahaman, M. M. Haque, and M. K. H. Patwary, "An improved fitness function for automated cryptanalysis using genetic algorithm," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 13, no. 2, p. 643, 2019, doi: 10.11591/ijeecs.v13.i2.pp643-648.

[17] S. Tkatek, S. Bahti, O. Abdoun, and J. Abouchabaka, "Intelligent system for recruitment decision making using an alternative parallel-sequential genetic algorithm," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 22, no. 1, p. 385, 2021, doi: 10.11591/ijeecs.v22.i1.pp385-395.

[18] W. Wen-jing, "Improved Adaptive Genetic Algorithm for Course Scheduling in Colleges and Universities," *International Journal of Emerging Technologies in Learning (iJET)*, vol. 13, no. 06, p. 29, 2018, doi: 10.3991/ijet.v13i06.8442.

[19] O. Kramer, “Genetic Algorithms,” *Genetic Algorithm Essentials*. Springer International Publishing, pp. 11–19, 2017, doi: 10.1007/978-3-319-52156-5_2.

[20] M. Moza and S. Kumar, “Routing in Networks using Genetic Algorithm,” *Bulletin of Electrical Engineering and Informatics*, vol. 6, no. 1, pp. 88–98, 2017, doi: 10.11591/eei.v6i1.574.

[21] J. Zhong, X. Hu, J. Zhang, and M. Gu, “Comparison of Performance between Different Selection Strategies on Simple Genetic Algorithms,” *International Conference on Computational Intelligence for Modelling, Control and Automation and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC’06)*. IEEE, doi: 10.1109/cimca.2005.1631619.

[22] N. M. Razali and J. Geraghty, “Genetic Algorithm Performance with Different Selection Strategies in Solving TSP,” 2011.

[23] N. Saini, “Review of selection methods in genetic algorithms,” *International Journal of Engineering and Computer Science*, vol. 6, no. 12, pp. 22261–22263, 2017.

[24] K. Jebari, M. Madiafi, and A. Elmoujahid, “Parent selection operators for genetic algorithms,” *International Journal of Engineering Research & Technology*, vol. 2, no. 11, pp. 1141–1154, 2013.

[25] A. Hussain and Y. S. Muhammad, “Trade-off between exploration and exploitation with genetic algorithm using a novel selection operator,” *Complex & Intelligent Systems*, vol. 6, no. 1, pp. 1–14, 2019, doi: 10.1007/s40747-019-0102-7.

[26] M. Kaya, “The effects of a new selection operator on the performance of a genetic algorithm,” *Applied Mathematics and Computation*, vol. 217, no. 19, pp. 7669–7678, 2011, doi: 10.1016/j.amc.2011.02.070.

[27] A. Fadhile Jasim Al-Gburi, S. Naim, and A. Nasser Boraik, “Hybridization of Bat and Genetic Algorithm to Solve N-Queens Problem,” *Bulletin of Electrical Engineering and Informatics*, vol. 7, no. 4, pp. 626–632, 2018, doi: 10.11591/eei.v7i4.1351.

[28] K. Kamil, K. H. Chong, H. Hashim, and S. A. Shaaya, “A Multiple Mitosis Genetic Algorithm,” *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 8, no. 3, p. 252, 2019, doi: 10.11591/ijai.v8.i3.pp252-258.

[29] P. Kora and P. Yadlapalli, “Crossover Operators in Genetic Algorithms: A Review,” *International Journal of Computer Applications*, vol. 162, no. 10, pp. 34–36, 2017, doi: 10.5120/ijca2017913370.

[30] A. J. Umbarkar and P. D. Sheth “Crossover Operators In Genetic Algorithms: A Review,” *ICTACT Journal on Soft Computing*, vol. 06, no. 01, pp. 1083–1092, 2015, doi: 10.21917/ijsc.2015.0150.

[31] L. Manzoni, L. Mariot, and E. Tuba, “Balanced crossover operators in Genetic Algorithms,” *Swarm and Evolutionary Computation*, vol. 54, p. 100646, 2020, doi: 10.1016/j.swevo.2020.100646.

[32] S. M. Lim, A. B. M. Sultan, M. N. Sulaiman, A. Mustapha, and K. Y. Leong, “Crossover and Mutation Operators of Genetic Algorithms,” *International Journal of Machine Learning and Computing*, vol. 7, no. 1, pp. 9–12, 2017, doi: 10.18178/ijmlc.2017.7.1.611.

[33] S. Masrom, M. Mohamad, S. Mohamed Hatim, N. Baharun, N. Omar, and A. S. Abd. Rahman, “Different mutation and crossover set of genetic programming in an automated machine learning,” *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 3, p. 402, 2020, doi: 10.11591/ijai.v9.i3.pp402-408.

[34] D. Gupta and S. Ghafir, “An overview of methods maintaining diversity in genetic algorithms,” *International Journal of Emerging Technology and Advanced Engineering*, vol. 2, no. 5, pp. 56–60, 2012.

[35] P. A. Diaz-Gomez and D. F. Hougen, “Empirical Study: Initial Population Diversity and Genetic Algorithm Performance,” 2007.

[36] G. Zhao, W. Luo, H. Nie, and C. Li, “A Genetic Algorithm Balancing Exploration and Exploitation for the Travelling Salesman Problem,” *2008 Fourth International Conference on Natural Computation*. IEEE, 2008, doi: 10.1109/icnc.2008.421.

[37] M. Rares, “Adaptive mutation in genetic algorithms for shortest path routing problem,” *2015 7th International Conference on Electronics, Computers and Artificial Intelligence (ECAI)*. IEEE, 2015, doi: 10.1109/ecai.2015.7301163.

[38] J. Milton, P. Kennedy, and H. Mitchell, “The Effect of Mutation on the Accumulation of Information in a Genetic Algorithm,” *AI 2005: Advances in Artificial Intelligence*. Springer Berlin Heidelberg, pp. 360–368, 2005, doi: 10.1007/11589990_38.

[39] J. Cervantes and C. R. Stephens, “‘Optimal’ mutation rates for genetic search,” *Proceedings of the 8th annual conference on Genetic and evolutionary computation - GECCO ’06*. ACM Press, 2006, doi: 10.1145/1143997.1144201.

[40] G. Ochoa, “Error Thresholds in Genetic Algorithms,” *Evolutionary Computation*, vol. 14, no. 2, pp. 157–182, 2006, doi: 10.1162/evco.2006.14.2.157.

[41] R. L. Haupt, “Optimum population size and mutation rate for a simple real genetic algorithm that optimizes array factors,” *IEEE Antennas and Propagation Society International Symposium. Transmitting Waves of Progress to the Next Millennium. 2000 Digest. Held in conjunction with: USNC/URSI National Radio Science Meeting (Cat. No.00CH37118)*. IEEE, doi: 10.1109/aps.2000.875398.

[42] Y. Zhang, M. Sakamoto, and H. Furutani, “Effects of Population Size and Mutation Rate on Results of Genetic Algorithm,” *2008 Fourth International Conference on Natural Computation*. IEEE, 2008, doi: 10.1109/icnc.2008.345.

[43] P. V. Paul, “Performance analyses on population seeding techniques for genetic algorithms,” *International Journal of Engineering and Technology*, vol. 5, no. 3, pp. 2993–3000, 2013.

[44] P. Victer Paul, N. Moganarangan, S. S. Kumar, R. Raju, T. Vengattaraman, and P. Dhavachelvan, “Performance analyses over population seeding techniques of the permutation-coded genetic algorithm: An empirical study based on traveling salesman problems,” *Applied Soft Computing*, vol. 32, pp. 383–402, 2015, doi: 10.1016/j.asoc.2015.03.038.

[45] M. Shanmugam, M. S. S. Basha, P. V. Paul, P. Dhavachelvan, and R. Baskaran, “Performance assessment over heuristic population seeding techniques of genetic algorithm: benchmark analysis on travelling salesman problems,” *International Journal of Applied Engineering Research*, vol. 8, no. 10, pp. 1171–1184, 2013.

[46] A. Hassanat, V. Prasath, M. Abbadi, S. Abu-Qdari, and H. Faris, “An Improved Genetic Algorithm with a New Initialization Mechanism Based on Regression Techniques,” *Information*, vol. 9, no. 7, p. 167, 2018, doi: 10.3390/info9070167.

[47] U. Khair, Y. D. Lestari, A. Perdana, D. Hidayat, and A. Budiman, “Genetic Algorithm Modification Analysis Of Mutation Operators In Max One Problem,” *2018 Third International Conference on Informatics and Computing (ICIC)*. IEEE, 2018, doi: 10.1109/iac.2018.8780463.

[48] D. Ghosh, “Comparing genetic algorithm crossover and mutation operators for the indexing problem,” *Indian Institute of Management Ahmedabad*, pp. 1–12, 2016.

[49] D. N. Mudaliar and N. K. Modi, “Applying m-Mutation Operator in Genetic Algorithm to Solve Permutation Problems,” *2019 IEEE International Conference on System, Computation, Automation and Networking (ICSCAN)*. IEEE, 2019, doi: 10.1109/icscan.2019.8878867.

[50] S. Sarmady, “An investigation on genetic algorithm parameters.” School of Computer Science, Universiti Sains Malaysia, 2007.

[51] L. Bottaci, “A genetic algorithm fitness function for mutation testing,” 2001.

[52] S. Harun and M. F. Ibrahim, “A genetic algorithm based task scheduling system for logistics service robots,” *Bulletin of Electrical Engineering and Informatics*, vol. 8, no. 1, pp. 206–213, 2019, doi: 10.11591/eei.v8i1.1437.

[53] X. Qin, "Image Segmentation Research Based on GA and Improved Otsu Algorithm," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 7, no. 2, p. 533, 2017, doi: 10.11591/ijeecs.v7.i2.pp533-541.

BIOGRAPHIES OF AUTHORS

Sami Ullah is a student of MS computer science at Abasyn University. He obtained master's and bachelor's degrees in IT from Imsciences, Peshawar, and is also a certified PMP. He started working in the technology field in 2005 as a freelance developer and his projects included warehouse management systems, library management systems, and web development. Since 2007 he has been working for Pakistan international airlines, in his current position he has managed Passenger Service Systems projects in SABRE and HITIT. He has a keen interest in understanding technology and applying modern tools to streamline business processes. His recent interest is applying Genetic algorithms to solve crew and aircraft scheduling to reduce the operational cost of route management for an airline. He can be contacted at email: samiullahpmp@gmail.com.

Dr. Abdus Salam is a professor and head of the department of computing at Abasyn University Peshawar, Pakistan has a Ph.D. in Computer Science from International Islamic University, Islamabad. He obtained his MSc Degree in Computer Science from Quaid-e-Azam University Islamabad. His research interests are in the fields of databases, data mining, and software engineering. He email is: dr.salam@abasyn.edu.pk.

Dr. Mohsin Masood received his Ph.D. degree from the University of Strathclyde, Glasgow, UK from the Department of Electronics and Electrical Engineering for his research in Artificial Intelligence based Meta-heuristic Techniques for the optimization in MPLS Networks. He did MSc in Mobile Communication from University of East London (UEL), UK. He has been involved both in industrial and academic based projects along with research awards granted that include Erasmus funding for Technical University of Ostrava, Czech Republic. His research intersects includes Artificial Intelligence (AI), Machine learning algorithms, artificial Neural networks, Meta-heuristic Optimization Techniques for Computer and communication networks, Transportation and Sensor Networks optimization by investigating and working on evolutionary or swarm inspired algorithms along with Machine Learning Hybrid Models are the key research areas of interest. He can be contacted at email: mohsin.masood@abasyn.edu.pk.