ISSN: 2093-5374 / E-ISSN: 2093-5382

Premature Avoidance in Genetic Algorithm using Dynamic Mutation Probability

Rawan Nassri Abulail[1*]

[1]*Associate Professor, Computer Science Department, Philadelphia University, Amman, Jordan.
rabulail@philadelphia.edu.jo, https://orcid.org/0009-0002-8168-2455

Received: November 13, 2024; Revised: December 23, 2024; Accepted: February 14, 2025; Published: March 31, 2025

Abstract

Evolutionary algorithms are optimization techniques based on biological and natural evolution mechanisms. These algorithms are a subset of evolutionary computation and fall under unsupervised learning. The Genetic Algorithm (GA) is one of the most common types of evolutionary algorithms. It begins with an initial set of candidate solutions and starts the evolutionary process by applying certain operators to generate new solutions. The newly produced solutions are expected to outperform the previous ones. Premature convergence is a problem encountered by most evolutionary algorithms, particularly genetic algorithms. It occurs when parental solutions fail to generate better offspring or children with superior traits. Self-adaptive mutations and Panmictic populations are the main factors contributing to premature convergence. Several approaches can be applied to avoid premature convergence and sustain population diversity, including the crowding method, incest prevention algorithm, scheduled sharing approach, cooperation-based approach, syntactic analysis of convergence, random offspring generation, selective mutation, and dynamic reproduction operators. The lack of population diversity leads directly to convergence, forcing the evolutionary algorithm to stop evolving and return the dominant value as the candidate solution. In most cases, this is not an optimal solution. One approach to sustaining population diversity is applying dynamic reproduction genetic operators. The main objective of this research is to propose an enhancement to the standard genetic algorithm to overcome premature convergence. A dynamic reproduction mutation operator is proposed to vary the probability of mutation based on the fitness value in each iteration. The methodology employed by the researcher involves conducting experiments to demonstrate the results achieved after applying the enhanced genetic algorithm (Rowe, 2008). Three different experiments with varying population sizes and mutation probability values were carried out to identify the best solution for an optimization problem. A total of 100 generations were produced by applying 10,000 iterations, and a binary genetic algorithm was used for running iterations with 16-bit chromosome lengths to represent candidate solutions. The results show that improvements in fitness scores were achieved, which enhanced the performance of the genetic algorithm for the produced generations (offspring). Moreover, population diversity was maintained.

Keywords: Evolutionary Algorithms, Genetic Algorithms, Premature Convergence, Dynamic Mutation Probability.

1 Introduction

Evolutionary algorithms are optimization algorithms based on biological and natural evolution mechanisms (Muralidharan, 2024). These algorithms are a subset of evolutionary computation and are

Journal of Wireless Mobile Networks, Ubiquitous Computing, and Dependable Applications (JoWUA), volume: 16, number: 1 (March), pp. 528-542. DOI: *10.58346/JOWUA.2025.I1.031*

*Corresponding author: Assistant Professor, Business Information Technology Department, Liwa College, Abu Dhabi, UAE.

categorized under unsupervised learning (Vikhar, 2016). The main characteristic of evolutionary algorithms is that they start with many candidate solutions (population) instead of one solution and go through many natural operations in order to produce better candidate solutions (offspring) for the optimization problem (LaTorre & Molina, 2020; Sadulla, 2024). The Genetic Algorithm (GA) is one of the most common types of evolutionary algorithms, which starts from an initial candidate number of solutions and begins the evolutionary process by applying certain operators to produce new solutions (Mansour, 2024). The newly produced solutions are expected to be better than the previous ones. The initial population in GA is represented as chromosomes, and each chromosome consists of a number of genes, which carry 1 or 0 values in binary-coded genetic algorithms (Majid et al., 2024). The initial population evolves by applying natural operators like selection, crossover, and mutation to produce new offspring in each iteration. The produced offspring must be evaluated by measuring the fitness function and are supposed to be better than the initial population (Guo et al., 2010). GA will keep running until it reaches one of the stopping conditions of the algorithm, such as reaching population convergence. In some cases, the algorithm stops because of convergence with a suboptimal solution, referring to the premature convergence occurrence problem (LaTorre & Molina, 2020).

Premature convergence is a problem that faces most evolutionary algorithms, especially GA. It means that convergence has occurred in the optimization problem population too early as a consequence of the inability of the parents to generate better offspring or children carrying better traits, which refers to a failure in the genetic operators applied to the parents (Lei & Liu, 2020; Prasath, 2024). In simple words, premature convergence happens when the algorithm returns a solution, but not the optimal one, because it has gotten stuck in a single solution as a result of population convergence and loss of diversity (Lin et al., 2010). Factors leading to premature convergence can be summarized in two main reasons: the first one is self-adaptive mutations, and the second one is panmictic (unstructured) populations (Cui et al., 2018). These two factors are considered the main reasons that affect population diversity and lead to the loss of diversity. There are many premature convergence avoidance approaches that can be applied to evolutionary algorithms to sustain population diversity, such as the crowding method, incest prevention algorithm, scheduled sharing approach, cooperation-based approach, syntactic analysis of convergence, random offspring generation approach, selective mutation approach, and dynamic reproduction operators (Bi et al., 2021; Loganathan & Poonkodi, 2018).

Problem Statement: Premature convergence in evolutionary algorithms, particularly in GA, can be defined as reaching the convergence of the population (chromosomes) for a specific optimization problem too early, leading to a suboptimal solution (Carvalho & Fernandes, 2018). Premature convergence is an obvious problem facing evolutionary algorithms, especially genetic algorithms, where the candidate solution starts from the initial population (chromosomes) and evolves by applying the known genetic crossover and mutation operators. The aim of applying the genetic algorithm operators is to produce offspring in each iteration that are supposed to be better than their parents. However, if premature convergence happens, it will lead to producing offspring carrying the same features and traits as their parents without any improvements to their fitness value. The decision of reaching premature convergence can be measured if 95% or more of the population carry the same value for a specific gene in binary-coded GA (Lan, 2023). Convergence can be simply defined as the absence of population diversity. The absence of population diversity leads directly to convergence, which will force the evolutionary algorithm to stop evolving and return the dominant value as the candidate solution. This solution, if convergence happens too early, leads to a suboptimal solution or not the global best solution. In this case, the performance evaluation of the evolutionary algorithm will be low, and the main goal of applying evolutionary algorithms will not be achieved. One of the approaches that can be used to sustain population diversity and avoid premature convergence is dynamic reproduction operators.

Research Objectives: The main objective of this research is proposing a method or algorithm to overcome the problem of premature convergence. The researcher will use the dynamic reproduction operator approach and propose a modification of the standard genetic algorithm by forcing the probability of mutation to change depending on the fitness value in each iteration. Experiments will be done to show the result of the modified GA algorithm, and the results will be displayed and discussed.

Research Structure: The research consists of five sections. The first section will give a brief introduction about evolutionary algorithms, premature convergence, the research problem statement, and the research objectives. The second section will cover the background and literature review of evolutionary algorithms, genetic algorithms, premature convergence, factors that lead to premature convergence, and finally the approaches that can be applied to avoid premature convergence occurrence. The third section will display the research methodology, while the fourth section will show and discuss the results of the research. Finally, the fifth section will summarize the results as a conclusion of the research work.

2 Background and Literature Review

This section will display the background and literature review of evolutionary algorithms and their importance and applications, genetic algorithms and their evolving life cycle, followed by the main factors that affect genetic algorithms, the definition of premature convergence, and how it affects the performance of genetic algorithms, factors that lead to premature convergence, and finally the approaches that can be applied to avoid premature convergence occurrence (Jiménez-Carrión et al., 2023).

Evolutionary Algorithms

Evolutionary algorithms are optimization algorithms and techniques based on biological and natural evolution mechanisms. These algorithms are a subset of evolutionary computation and are categorized under unsupervised learning (Zhao et al., 2024). The main characteristic of evolutionary algorithms is that they start with many candidate solutions (population) instead of one solution and go through many natural operations in order to produce better candidate solutions (offspring) for the optimization problem ((Yu et al., 2009) l., 2016; Lofandri et al., 2024). Evolutionary algorithms consist of many algorithms and applications, such as mimetic algorithms, genetic algorithms, artificial immune systems, evolution strategies, genetic programming, evolutionary programming, metaheuristics, ant colony optimization, simulated annealing, and swarm optimization (Cheng et al., 2015).

Genetic Algorithm

GA is a nature-inspired algorithm used for searching and optimization problems. Genetics and natural selection are the basic ideas for its evolution process. Genetic algorithm is categorized as one of the evolutionary algorithms, which starts from an initial candidate number of solutions and begins the evolutionary process by applying certain operators to produce new solutions. The newly produced solution is expected to be better than the previous one. Many applications in different domains have employed GAs successfully, such as economics, natural language processing, optimization, bioinformatics, image processing, grammar inference, and pattern recognition. Holland was the first developer of GA's basic principles. Then many researchers worked on those principles and provided detailed overviews and descriptions to implement GAs in many domains and fields. GAs can be summarized as follows: an initial population of candidate solutions for a specific optimization problem is represented using encoding methods. In the implementation process, each candidate solution in the initial population is evaluated by the value obtained from applying the objective function, which is being

optimized. The obtained value is called the fitness value in GAs. The fitness value is the main measure used in GAs for the reproduction process, starting with selecting the candidate parents to mate and recombine to produce new offspring after applying the crossover and mutation operators. The new offspring will carry most of their features from the candidate parents (Oh et al., 2004). The new offspring is called the new population or the new generation, which means the initial population after modifications. The quality of the new generation is evaluated by a measure called the fitness function. This fitness function allocates a fitness value for each individual in the population. While the evolution process is for searching or optimization, these fitness values will change as a consequence of the reproduction operators. The GAs keeps running and producing new generations if there are improvements in the fitness values. After a certain number of iterations or generations, strong individuals will dominate the reproduction process because of their high fitness values, leading to a decrease in population diversity. That low population diversity implies population convergence (Guo et al., 2010; Moses et al., 2022). If GAs reach this convergence in the population, it means the reproduction process will get stuck with fitness values, and there will be no improvements in the new offspring, which means stopping the reproduction process and returning the last population as the optimal solution. The remarkable issue in this case is that, in some cases, convergence happens very early, and the returned value is actually suboptimal. This is called premature convergence, which is the problem that will be discussed in detail in this research.

The fitness values of individuals in any population are the guidance for iterative search or optimization in GAs. In each iteration, the individuals are evaluated by their fitness values, and the best are selected as candidate parents. These obtained parents from the selection process will mate and combine to produce offspring. It's obvious that the parent features are partially preserved, and some of their features will be lost in the new generations. This refers to the effect of reproduction operators (crossover and mutation). The loss of some of the parent's features leads to difficulties in managing the offspring features. These difficulties result from many factors. The first one is that each individual in the new generation appears to have been checked and seen before. The second reason is the competition between individuals to survive, which leads some in the wrong direction. The third reason, originating from the second one, is the consequence of leaving a large portion of the initial population unexplored for the optimal solutions (Sun et al., 2021). GAs have been applied in many domains, as mentioned before in this research, but they were not as adaptive as expected. This refers to the premature convergence problem (Lin et al., 2010).

Genetic Algorithm Lifecycle

Figure (1) displays the life cycle of the standard genetic algorithm, starting from a random population encoded by binary 0 or 1 values. After the encoding process is finished, each candidate solution is evaluated by applying its value in the optimization problem to calculate the fitness value. The fitness value will determine the probability of selecting that solution. After selecting the same number of the initial population, the newly selected population will be divided into pairs to apply the reproduction operators, crossover and mutation. After applying the two reproduction operators, the new offspring will be produced. The new offspring will be evaluated by calculating its fitness values, and then the algorithm will check if it has reached convergence or not. If convergence is reached, the algorithm will stop and return the calculated solution; otherwise, it will repeat the iteration starting from the selection process (Rowe, 2008). The genetic crossover and mutation operators play the most important role in the standard genetic algorithm's performance.

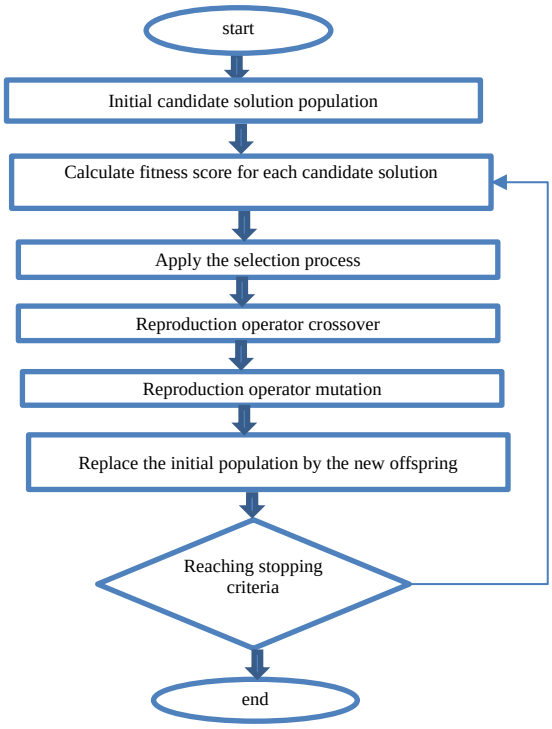

Figure 1: Standard Genetic Algorithm Life Cycle

Factors Affecting Standard Genetic Algorithm Performance

There are a number of factors that must be taken into account when evaluating the performance of the standard GA. The most important factors can be listed as follows: initial candidate solutions, candidate solutions diversity, fitness function, search space, pressure of the selection process, difficulty of the problem to be optimized, and size of the population. The mentioned factors affect the performance of the standard GA directly, and the most important factor is maintaining the population diversity to guarantee that the algorithm will find the global optimal solution and avoid premature convergence (Dimitrov & Baumann, 2011).

Premature Convergence

Premature convergence can be defined as a problem facing most evolutionary algorithms, especially GA. It means that the convergence of a specific population is reached very early, giving the final solution for an optimization problem, which forces the algorithm to stop and return the results. The results obtained in this case are suboptimal; this is referred to as the failure of parents (chromosomes) to produce better offspring carrying better traits than their parents. In other words, reproduction of the new generations is useless, and the genetic reproduction operators are not adding any improvements to the fitness values of the new offspring. Some researchers define the occurrence of convergence if 95% or more of the offspring carry the same traits for a specific gene (Tinós et al., 2023).

Population diversity must be preserved during the evolution phases to avoid premature convergence. It is not easy to determine the occurrence of premature convergence, and it is also hard to predict whether it will occur in the future. There are many measurements that can be used to evaluate the improvement of population fitness in each phase. One of them is the difference between the fitness average of the previous population and the fitness average of the current population, and the second is calculating the difference between the max fitness in the previous population and the maximum fitness in the current population. In both cases, if there is a difference, it means that we haven't reached convergence. But we

must ensure that the differences lead to better fitness; in other words, improvement of the population's feature quality (Yu et al., 2009). These differences, in some algorithms, are used to change the probability of crossover and mutation that are applied in the reproduction process in a dynamic way to produce new offspring after parent mating and exchanging their features. Another kind of measurement is population diversity, which is commonly used to measure the occurrence of premature convergence. The population diversity measurement depends on the idea that losing the population diversity leads directly to premature convergence. However, there are many concerns related to the conditions that must be applied to maintain population diversity, especially since it may affect the robustness of the population (Kawulok & Kawulok, 2022).

What Factors Lead to Premature Convergence?

From the previous studies (Cheng et al., 2015) of genetic algorithms and premature convergence, the researcher summarized the main causes that lead to premature convergence, which will be listed as follows:

- **Self-Adaptive Mutations**

Self-adaptive mutations were proposed by Rechenberg. Its main concept is using evolution strategies for self-adaptation in the mutation distributions. In other words, the parameters used to control the mutation distributions use self-adaptation evolution instead of using predetermined parameters (Gliesch et al., 2017). This new technique is called the "1/5-success rule of evolution strategies (1 + 1)-ES." The control parameter for the mutation step size will be increased based on some factors related to the ratio of positive mutations to the total number of mutations (Yu et al., 2009). If the ratio is greater than 1/5 in a specific period of time, then the control parameter for the step size will be increased, and if the ratio is less than 1/5, then the control parameter will be decreased (Kawulok & Kawulok, 2022). Self-adaptive mutation is considered one of the main causes for the occurrence of premature convergence.

- **Panmictic (unstructured) Populations**

Most evolutionary algorithms, especially genetic algorithms, start with an initial panmictic or unstructured population of chromosomes, where every chromosome is eligible to be selected as a candidate parent to mate and recombine to produce offspring based on their fitness value (Gliesch et al., 2017). This means that the chromosomes with high fitness values can spread their features to the produced populations in many generations. The produced generations will not be better than the previous generations. This leads to a loss in population diversity, especially in small populations, and to premature convergence as a consequence (Cui et al., 2018). One of the countermeasures that could be used to preserve population diversity is switching to new models of populations, which means introducing substructures into the initial population. This technique can preserve population diversity over a longer time and a higher number of generations (Kawulok & Kawulok, 2022).

- **Premature Convergence Occurrence Avoidance Approaches**

Previous studies showed that many approaches have been proposed to regain population variation. Each strategy targeted one of the GA phases; some of them are related to parents who will be mated and recombined (mating strategy), and some researchers call it incest prevention. Another kind of approach is related to the crossover operator, and it is called uniform crossover. There are also some strategies dealing with the population itself before selection, and it depends on the idea of sorting the population by their similarity and making replacements of the individuals. This strategy is called (crowding or preselection). Some approaches use segmentation based on fitness values in the population by grouping

individuals having similar fitness values into one segment. This strategy is called (fitness sharing). While there are approaches targeting the population size to avoid premature convergence by increasing the size of the population in order to maintain high diversity in the population features (Segura et al., 2013). In addition to the mentioned approaches, premature convergence risk can be minimized by using structured populations instead of (unstructured) panmictic populations (Bi et al., 2022). Finally, there are approaches called dynamic reproduction operators that target the genetic algorithm crossover and mutation operators by forcing their values to be dynamic, not fixed, depending on the offspring fitness values. This research will modify the standard binary genetic algorithm by applying a modification to the mutation operator using a dynamic value for the mutation probability. The mutation phase can play an important role in preventing premature convergence by regaining genetic variation through applying the process in a highly random way (Dadgar et al., 2017).

The Proposed Dynamic Mutation Operator

In this section, the researcher will explore the enhancements that were proposed to the mutation genetic operator in the standard genetic algorithm. Let's start by writing down the steps that are applied in the standard GA process:

- Population initialization: Initialize a number of candidate solutions presented as chromosomes. 2- Set stopping criteria: Setting the stopping criteria means defining the criteria that must be checked at the end of each iteration. Some of these criteria are the maximum number of iterations (generations), reaching the predefined goal value, or reaching population convergence. Population convergence means that there are no improvements in the offspring (newly produced generation) fitness value.
- Set crossover values: There are two main values that must be set in this step: the first one is the probability for selection, and the second value is related to the crossover points, which clarify at which points the traits (value of genes) will be exchanged if the crossover happens.
- Set mutation value: This step is very important because it has the main role of changing the gene values and, in some cases, adding new values not inherited from parents. The mutation value can be defined as the probability of mutation. The mutation operator will be applied to each gene in the chromosome. For each gene, there will be a comparison between a random number and the mutation probability. If the random number is less than the probability of mutation, the mutation will occur; otherwise, the gene value will not be changed. To simplify the meaning of mutation in binary GA: if the mutation happens, the value of 1 will be changed to 0, and if the value is 0, it will be changed to 1.
- Fitness value (score) calculation: This step is done by applying the chromosome value in the function being optimized and finding out the fitness value for each chromosome.
- Calculate the selection probability: This can be done in different ways, but the most common way is to divide the fitness score for each chromosome by the summation of fitness scores in the population.
- Apply the selection operator to the population and select parental solutions.
- Apply the crossover operator to produce offspring.
- Apply the mutation value to produce the modified offspring if any of their genes are affected by the mutation probability.
- Offspring fitness value: After producing the new generation, the fitness scores must be calculated again and compared with the previous fitness scores for the parental candidate solutions. 11- Checking the predefined stopping conditions: If any of the predefined stopping conditions are met, the algorithm will stop and return the final value.

- 12- If not reaching any of the stopping conditions, go back to step five and apply the genetic operators to the newly produced offspring to produce a new generation. The process of the standard GA is displayed in Figure 2.

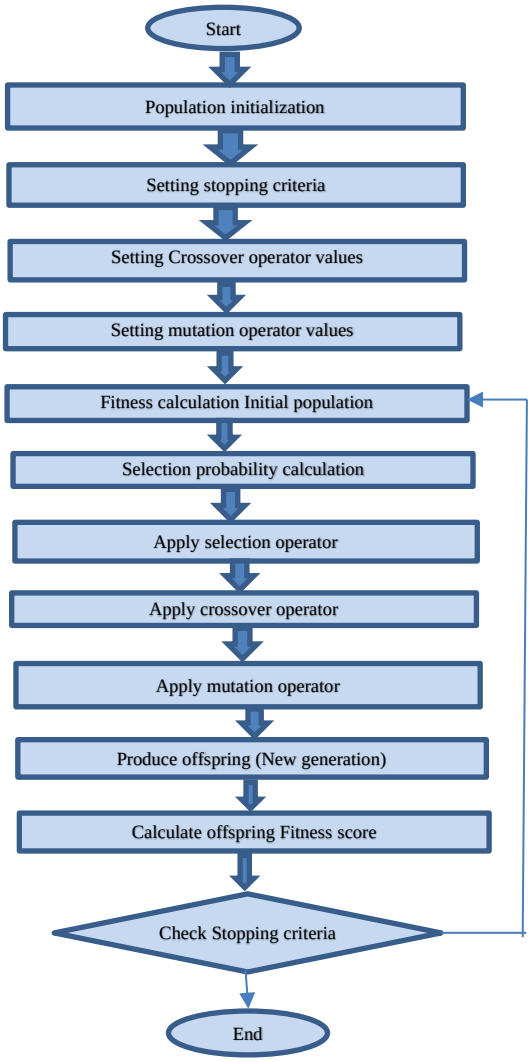

Figure 2: Standard GA process

After displaying the standard GA process and as the flowchart shows, the algorithm will stop running after checking the stopping criteria for each iteration. All the stopping criteria are checked in one step. In simple words, if any of the stopping criteria (conditions) are met, the algorithm will stop and return the solution. As mentioned before, the stopping conditions in standard GA are the maximum number of iterations (generations) predefined by the user, reaching the predefined goal value, or reaching population convergence. Stopping the algorithm as a consequence of reaching the maximum number of iterations or reaching the goal will return a satisfied solution for the user. But the main problem is stopping the algorithm after reaching population convergence. In this case, there will be two scenarios: the first happy scenario is that the convergence happened after reaching the optimal solution; the second unhappy scenario is that the algorithm reached population convergence very early and returned a solution that is not good enough. In this case, the occurrence of premature convergence happened in the algorithm.

The researcher has proposed a modification to the standard GA algorithm to overcome the premature occurrence, which happened as a consequence of the inability of the algorithm to generate a new population with higher fitness values or scores, with no improvements to the new offspring. To overcome this problem, two main modifications were added to the standard GA. The first one is to divide the checking points for stopping the algorithm into two separate points. The first checking point is to check the predefined maximum number of iterations and the predefined goal. If any of them is reached, the algorithm will return the solution. While the second checking point is to check population convergence, in this checking point, the algorithm must decide if the convergence occurred with a satisfied solution or if it happened very early. If the population converged very early, the algorithm reached premature convergence because the population lost its diversity and the genetic operators are unable to produce improved offspring. The second modification proposed by the researcher to overcome the problem of losing the diversity in the population is to force the algorithm to add new features and traits to the population using a dynamic mutation operator instead of using a fixed probability of mutation. The researcher's suggestion is to increase the probability of the preset value of mutation probability. To specify the optimal value of mutation probability, the researcher conducted a number of experiments with different population sizes and different mutation probabilities and found that the best modification for any preset mutation probability is to multiply the value by two. The increase in the mutation probability will ensure that more genes in the offspring will be affected by the mutation operator, leading to the addition of new features and traits to the population, especially in binary GA. The conducted experiments will be displayed in the next sections of this research. The proposed enhancement and modifications to the GA process will be presented in the following steps and flowchart:

- Population initialization: Initialize a number of candidate solutions presented as chromosomes.
- Set stopping criteria: Setting the stopping criteria means defining the criteria that must be checked at the end of each iteration. Some of these criteria are the maximum number of iterations (generations), reaching the predefined goal value, or reaching population convergence. Population convergence means that there are no improvements in the offspring (newly produced generation) fitness value.
- Set crossover values: There are two main values that must be set in this step: the first one is the probability for selection, and the second value is related to the crossover points, which clarify at which points the traits (value of genes) will be exchanged if the crossover happens.
- Set mutation value: This step is very important because it has the main role of changing the gene values and, in some cases, adding new values not inherited from the parent. The mutation value can be defined as the probability of mutation. The mutation operator will be applied to each gene in the chromosome. For each gene, there will be a comparison between a random number and the mutation probability. If the random number is less than the probability of mutation, the mutation will occur; otherwise, the gene value will not be changed. To simplify the meaning of mutation in binary GA: if the mutation happens, the value of 1 will be changed to 0, and if the value is 0, it will be changed to 1.
- Fitness value (score) calculation: This step is done by applying the chromosome value in the function being optimized and finding out the fitness value for each chromosome.
- Calculate the selection probability: This can be done in different ways, but the most common way is to divide the fitness score for each chromosome by the summation of fitness scores in the population.
- Apply selection operator to population and select parental solutions.
- Apply crossover operator to produce offspring.

- Apply mutation value to produce the modified offspring if any of their genes are affected by the mutation probability.
- Offspring fitness value: After producing the new generation, the fitness scores must be calculated again and compared with the previous fitness scores for the parental candidate solutions.
- Check population convergence: In this case, if the algorithm returns an acceptable solution, then stop; otherwise, go back to step 4 and modify the mutation value by multiplying it by two.
- Check the other two stopping conditions: the max number of iterations or reaching the predefined goal value. If any of them are met, the algorithm will stop and return the final value.
- If none of the stopping conditions are met, go back to step five and apply the genetic operators to the newly produced offspring to produce a new generation. The process of the standard GA is displayed in Figure 3.

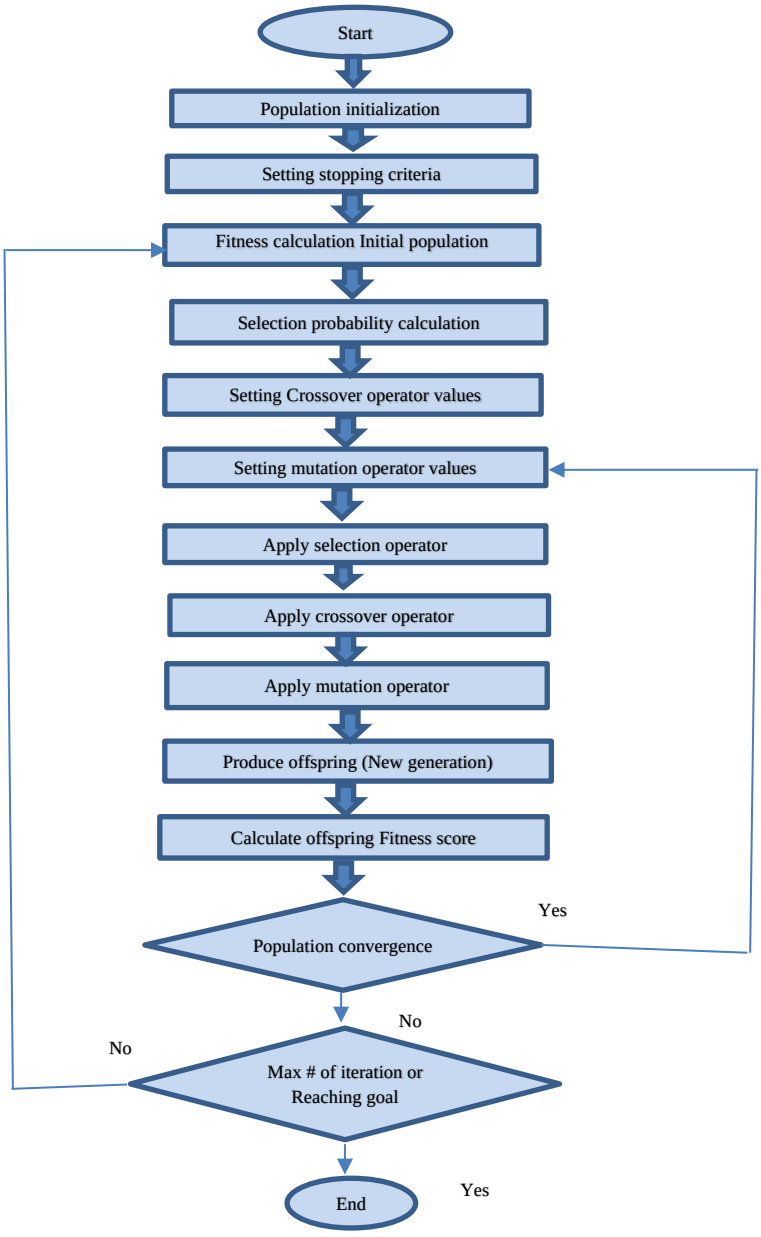

Figure 3: The Proposed Dynamic Mutation Operator GA Process

3 Research Methodology

Conducting experiments is the methodology which was applied by the researcher to show the results achieved after applying the enhanced standard genetic algorithm (modifying the mutation probability if the algorithm stuck because of convergence occurrence). Three different experiments were conducted to find the best solution for x2 function. The first experiment with n=20 for the population size, the second experiment with n=40, while the third one with n=100 for the population size. In each experiment five different mutation probabilities were applied to compare the results of the calculated fitness function to find out if there is an improvement of results for the x2 function being optimized. 100 generations were produced by applying 10,000 iterations in each experiment, binary genetic algorithm was used for running iterations using 16 bits to represent the candidate solution into 16-bit chromosomes lengths, the probability of cross over genetic operator was set to 0.3 while the probability of mutation genetic operator was set to 0.03, then multiply the mutation probability by 2, 4,6,8 and 10 respectively. Finally, the researcher uses random seed for the purpose of generating random number which used in the reproduction phase (selection, crossover and mutation). The experiments details are summarized in table (1).

Table 1: The conducted experiment details

Size of population	Number of applied iterations	Optimizing the maximization of x^2 function	Chromosomes length	Crossover probability	Mutation probability	Fitness
N=20	10000	x^2	16 bits	0.3	0.03 0.03*2 0.03*4 0.03*6 0.03*10	Fitness scores average
N=40	10000	x^2	16 bits	0.3	0.03 0.03*2 0.03*4 0.03*6 0.03*10	Fitness scores average
N=100	10000	x^2	16 bit	0.3	0.03 0.03*2 0.03*4 0.03*6 0.03*10	Fitness scores average

4 Results and Discussion

After conducting the experiments using different sizes of population with fixed value of mutation, the obtained solutions using standard genetic algorithm were kept to compare them later with the solutions which will be obtained using the modified genetic algorithm by applying the dynamic genetic mutation operators. After that the researcher had conducting the same experiments with same population sizes with different values of mutations probabilities, the different values of the mutation probability were obtained as a value of multiplying the mutation probability by 2, 4,6 and 10 respectively. The Genetic algorithm performance depending on the fitness scores were represented in figure 4, figure 5 and figure 6. The lines different colors in the figures represents the performance regarding value of mutation probability, where the orange line represent the performance using standard GA with fixed value of mutation probability, light blue with the value of mutation probability multiplied by 2, gray with the value of mutation probability multiplied by 4, yellow with the value of mutation probability multiplied by 6 and finally dark blue with the value of mutation probability multiplied by 10. The conducted experiments generated 100 generation after applying 10000 iterations

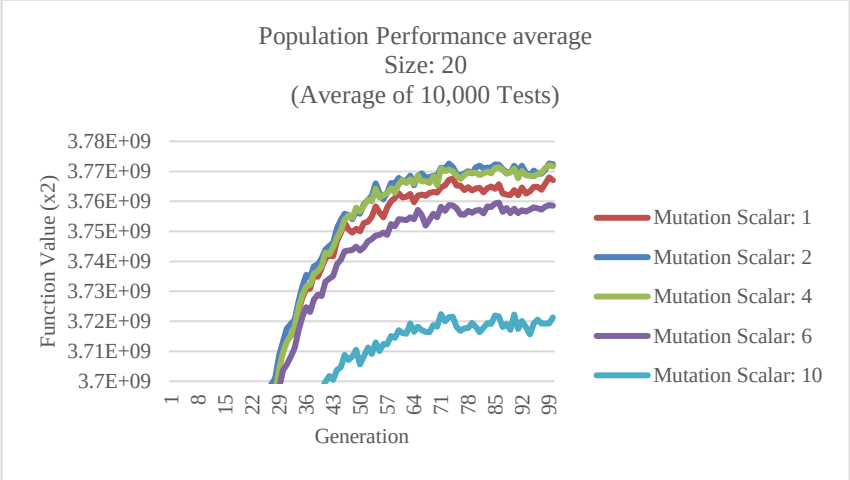

Figure 4: Performance Results for Population Size =20

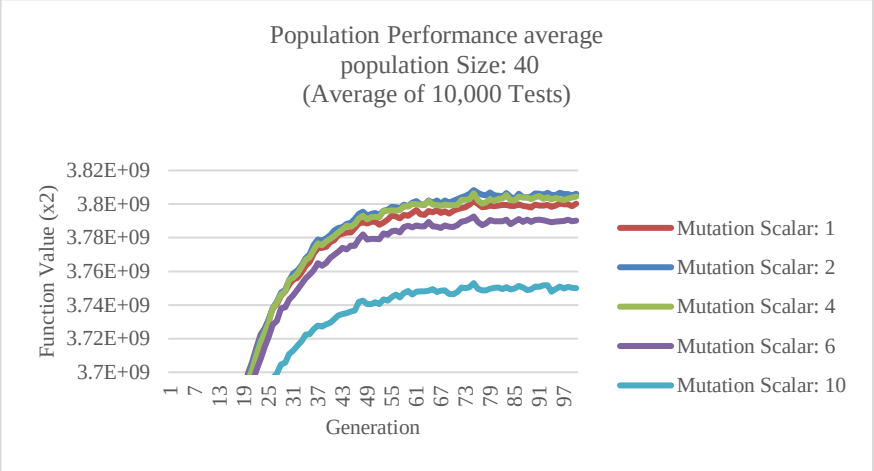

Figure 5: Performance Results for Population Size =40

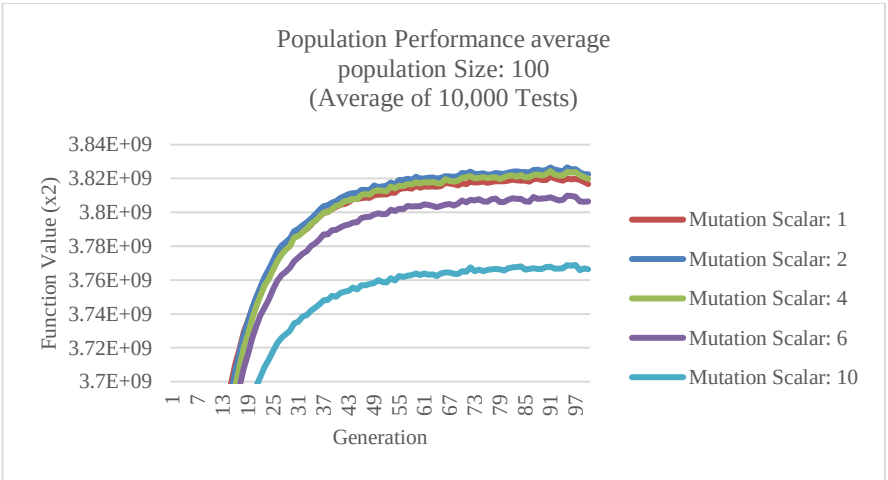

Figure 6: Performance Results for Population Size =100

As shown in figures 4,5 and 6, there were improvements in fitness scores which improved the performance of genetic algorithm for the produced generations(offspring). Applying a dynamic value of

genetic mutation operators played a major role in that performance improvements. The second main result which was concluded from the experiments shows that the best performance was after applying the value of mutation multiplied by two. The enhancement which was proposed by the researcher if the standard GA reaches the premature convergence is to set a new value for the predefined mutation probability by multiply it by two.

5 Conclusion

In this research the researcher has presented the standard genetic algorithm as one type of evolutionary algorithms describing its importance and different applications, then the genetic algorithm process was explained in details showing the different phases and genetic operators which applied to initial population to produce offspring. The stopping conditions of GA was mentioned in this research. One of the stopping conditions for GA is reaching population convergence. An explanation of convergence and premature convergence were presented showing the causes that lead to premature convergence occurrence. The main cause for premature convergence is the absence of population diversity which means that all the chromosomes in the population almost the same and as a consequence the GA will be unable to produce better offspring or better generations. Then the researcher has presented the importance of mutation genetic operator and its ability to maintain the population diversity because it is the only genetic operator which can add new values or traits to the chromosomes by changing its gene values especially in binary GA. Mutation operator is vital in maintaining the population diversity, the researcher proposed an enhancement to standard GA regarding mutation operator. The modification which was proposed by researcher is forcing the standard GA to check population convergence after each iteration, if population convergence was reached the algorithm evaluates the result obtained, if the result is acceptable then stop and return back the result, but if the result is not acceptable then change the predefined mutation probability value by multiplying it by two then apply the new mutation operator to offspring. These enhancements were presented as steps and flow chart. Number of experiments was conducted by the researcher to find out the best way to apply the dynamic mutation probability value, the results showed that the best improvements for population and produced generations is changing mutation probability value by multiply it by two.

As a conclusion an adaptive mutation probability technique when applied to standard GA achieved improvements in its performance by increasing fitness values and scores in the new produced population and prevent the occurrence of premature population convergence by maintaining population diversity.

References

[1] Bi, J., Yuan, H., Zhai, J., Zhou, M., & Poor, H. V. (2022). Self-adaptive bat algorithm with genetic operations. *IEEE/CAA Journal of Automatica Sinica*, *9*(7), 1284-1294. https://doi.org/10.1109/JAS.2022.105695

[2] Bi, J., Zhai, J., & Yuan, H. (2021, December). An Adaptive Hybrid Bat Algorithm with Genetic Operations and Dynamic Inertia Weight. In *2021 IEEE International Conference on Networking, Sensing and Control (ICNSC)* (Vol. 1, pp. 1-6). IEEE. https://doi.org/10.1109/ICNSC52481.2021.9702210

[3] Carvalho, L. C. F., & Fernandes, M. A. (2018, July). Convergence analysis of evolutionary algorithms solving the Flexible Job Shop Problem. In *2018 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1-7). IEEE. https://doi.org/10.1109/CEC.2018.8477685

[4] Cheng, J., Yen, G. G., & Zhang, G. (2015). A many-objective evolutionary algorithm with enhanced mating and environmental selections. *IEEE Transactions on Evolutionary Computation*, *19*(4), 592-605. https://doi.org/10.1109/TEVC.2015.2424921

[5] Cui, L., Huang, Q., Li, G., Yang, S., Ming, Z., Wen, Z., ... & Lu, J. (2018). Differential evolution algorithm with tracking mechanism and backtracking mechanism. *IEEE Access*, *6*, 44252-44267.https://doi.org/10.1109/ACCESS.2018.2864324

[6] Dadgar, M., Couceiro, M. S., & Hamzeh, A. (2017, October). RDPSO diversity enhancement based on repulsion between similar ions for robotic target searching. In *2017 Artificial Intelligence and Signal Processing Conference (AISP)* (pp. 275-280). IEEE. https://doi.org/10.1109/AISP.2017.8324096

[7] Dimitrov, T., & Baumann, M. (2011, July). Genetic algorithm with genetic engineering technology for multi-objective dynamic job shop scheduling problems. In *Proceedings of the 13th annual conference companion on Genetic and evolutionary computation* (pp. 833-834). https://doi.org/10.1145/2001858.2002112

[8] Gliesch, A., Ritt, M., & Moreira, M. C. (2017, July). A genetic algorithm for fair land allocation. In *Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 793-800). https://doi.org/10.1145/3071178.3071313

[9] Guo, P., Wang, X., & Han, Y. (2010, October). The enhanced genetic algorithms for the optimization design. In *2010 3rd international conference on biomedical engineering and informatics* (Vol. 7, pp. 2990-2994). IEEE. https://doi.org/10.1109/BMEI.2010.5639829

[10] Jiménez-Carrión, M., Flores-Fernandez, G. A., & Jiménez-Panta, A. B. (2023). Efficient Transit Network Design, Frequency Adjustment, and Fleet Calculation Using Genetic Algorithms. *Journal of Internet Services and Information Security*, *13*(3), 26-49. https://doi.org/1010.58346/JISIS.2023.I4.003

[11] Kawulok, J., & Kawulok, M. (2022, July). A genetic algorithm for classifying metagenomic data. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion* (pp. 59-60).https://doi.org/10.1145/3520304.3533944

[12] Lan, Y. (2023, March). Binary-like Real Coding Genetic Algorithm. In *2023 International Conference on Pattern Recognition, Machine Vision and Intelligent Algorithms (PRMVIA)* (pp. 98-102). IEEE. https://doi.org/10.1109/PRMVIA58252.2023.00023

[13] LaTorre, A., & Molina, D. (2020, July). On The Role of Execution Order In Hybrid Evolutionary Algorithms. In *2020 IEEE Congress on Evolutionary Computation (CEC)* (pp. 1-8). IEEE. https://doi.org/10.1109/CEC48606.2020.9185676

[14] Lei, L., & Liu, N. (2020, October). Research on optimization performance of nonlinear function based on multigroup genetic algorithm. In *2020 IEEE 20th International Conference on Communication Technology (ICCT)* (pp. 1498-1502). IEEE. https://doi.org/10.1109/ICCT50939.2020.9295871

[15] Lin, F., Zhou, C., & Chang, K. (2010). Convergence rate analysis of allied genetic algorithm. *49th IEEE Conference on Decision and Control (CDC)*, 786–791.

[16] Lofandri, W., Selvakumar, C., Sah, B., & Sangeetha, M. (2024). Design and Optimization of a High-Speed VLSI Architecture for Integrated FIR Filters in Advanced Digital Signal Processing Applications. *Journal of VLSI Circuits and Systems*, *6*(1), 70-77. https://doi.org/10.31838/jvcs/06.01.12

[17] Loganathan, P., & Poonkodi, M. (2018). A Dual CUK AC/DC Converter for DC Nano-Grid with Three Terminal Outputs. *International Journal of Advances in Engineering and Emerging Technology*, *9*(3), 119-129.

[18] Majid, U. M. A., Atan, N. A., Rukli, R., & Khan, A. (2024). Framework of Computer Science Learning Through Hybrid Service-Learning Oriented Visual Toward the Continuum of Visualization Thinking and Generic Skills. *Indian Journal of Information Sources and Services*, *14*(3), 192–206. https://doi.org/10.51983/ijiss-2024.14.3.25

[19] Mansour, M. M. (2024). Using a Genetic Algorithm to Determine the Best Factory Layout in Southern Iraq: Review. *International Academic Journal of Science and Engineering*, *11*(1), 362–373. https://doi.org/10.9756/IAJSE/V11I1/IAJSE1141

[20] Moses, M. B., Nithya, S. E., & Parameswari, M. (2022). Internet of things and geographical information system-based monitoring and mapping of real time water quality system. *International Journal of Environmental Sciences*, *8*(1), 27-36.
[21] Muralidharan, J. (2024). Optimization techniques for energy-efficient RF power amplifiers in wireless communication systems. *SCCTS Journal of Embedded Systems Design and Applications*, *1*(1), 1-5. https://doi.org/10.31838/ESA/01.01.01
[22] Oh, I. S., Lee, J. S., & Moon, B. R. (2004). Hybrid genetic algorithms for feature selection. *IEEE Transactions on pattern analysis and machine intelligence*, *26*(11), 1424-1437.https://doi.org/10.1109/TPAMI.2004.105
[23] Prasath, C. A. (2024). Optimization of FPGA architectures for real-time signal processing in medical devices. Journal of Integrated VLSI, Embedded and Computing Technologies, 1(1), 11-15. https://doi.org/10.31838/JIVCT/01.01.03
[24] Rowe, J. E. (2008, July). Genetic algorithm theory. In *Proceedings of the 10th annual conference companion on Genetic and evolutionary computation* (pp. 2535-2558). https://doi.org/10.1145/2330784.2330923
[25] Sadulla, S. (2024). Optimization of data aggregation techniques in IoT-based wireless sensor networks. *Journal of Wireless Sensor Networks and IoT*, *1*(1), 31-36. https://doi.org/10.31838/WSNIOT/01.01.05
[26] Segura, C., Coello, C. A. C., Segredo, E., Miranda, G., & León, C. (2013, June). Improving the diversity preservation of multi-objective approaches used for single-objective optimization. In *2013 IEEE congress on evolutionary computation* (pp. 3198-3205). IEEE. https://doi.org/10.1109/CEC.2013.6557961
[27] Sun, W., Yuan, H., Su, Q., & Chen, Y. (2021, September). Optimize performance analysis on Adaptive Genetic Algorithm with Linear Adjustment of Probabilities. In *2021 International Conference on Computer Information Science and Artificial Intelligence (CISAI)* (pp. 517-520). IEEE. https://doi.org/10.1109/CISAI54367.2021.00105
[28] Tinós, R., Przewozniczek, M., Whitley, D., & Chicano, F. (2023, July). Genetic algorithm with linkage learning. In *Proceedings of the Genetic and Evolutionary Computation Conference* (pp. 981-989). https://doi.org/10.1145/3583131.3590349
[29] Vikhar, P. A. (2016, December). Evolutionary algorithms: A critical review and its future prospects. In *2016 International conference on global trends in signal processing, information computing and communication (ICGTSPICC)* (pp. 261-265). IEEE. https://doi.org/10.1109/ICGTSPICC.2016.7955308
[30] Xue, B., Zhang, M., Browne, W. N., & Yao, X. (2015). A survey on evolutionary computation approaches to feature selection. *IEEE Transactions on evolutionary computation*, *20*(4), 606-626.https://doi.org/10.1109/TEVC.2015.2504420
[31] Yu, Y. M., Zhao, G. Q., & Liu, J. D. (2009). Hyperchaotic genetic algorithm theory and functions optimization. In *Proceedings of the first ACM/SIGEVO Summit on Genetic and Evolutionary Computation* (pp. 1041-1044). https://doi.org/10.1145/1543834.1544003
[32] Zhao, Y., Ding, Y., & Pei, Y. (2024). Adaptive Optimization in Evolutionary Reinforcement Learning Using Evolutionary Mutation Rates. *IEEE Access*. https://doi.org/10.1109/ACCESS.2024.3493198

Author Biography

Rawan Abulail is an Associate Professor in Philadelphia University and currently working as the head of computer science department in the information technology faculty. She received her B.Sc. degree in computer science and computer information systems from Philadelphia University, Amman, Jordan in 2002, B.Sc. degree in accounting from Philadelphia University, Amman, Jordan in 2004 M.Sc. and Ph.D. degrees from The Arab Academy for Banking and Financial Sciences, Amman, Jordan, in 2004, and 2009 respectively, all in Computer Information Systems.