




Boosting GA Performance: A Fuzzy Approach to Uncertainty Issues Involving Parameters in Genetic Algorithms

João Victor Ribeiro Ferro¹^a, José Rubens da Silva Brito¹^b, Robério José Rogério dos Santos²^c,
Roberta Vilhena Vieira Lopes¹ and Evandro de Barros Costa¹

¹Computing Institute, Federal University of Alagoas, Av. Lourival Melo Mota, Maceio, Brazil

²Eixo das Tecnologias, Campus do Sertão, Federal University of Alagoas, Delmiro Gouveia, Brazil

Keywords: Fuzzy Logic, Genetic Algorithms, Uncertainty, Optimization.

Abstract: This article addresses issues involving two sources of uncertainty in the stochastic search problem based on a genetic algorithm approach. We improve the mutation rate parameter by fuzzifying the population diversity and the individual adaptation value. A relevant aspect of this investment is related to the fact that this parameter, which presents uncertainty of the possibilistic type, directly interferes with the uncertainty of the probabilistic type of the genetic algorithm and also in the convergence and quality of the solution found by the genetic algorithm. Moreover, in parallel, we improve the understanding behavior of selection and replacement methods. Experiments were carried out on the case study with the classic OneMax problem to evaluate the performance of the proposed solution, analyzing aspects such as the convergence time, the quality of the solution, and the diversity of the population. The results obtained through the treatment of uncertainty and its impacts are presented in this article, showing relevant performance for the proposed algorithm, with the respective treatment of uncertainties.

1 INTRODUCTION

One of the problems faced by stochastic optimization models based on genetic algorithms (GA) is that the speed of solution convergence is quite sensitive to the choice of parameters such as mutation rate and crossover rate. In addition, the process of defining these parameters is also associated with the uncertainties of which values to choose to improve the performance of the algorithmic solution. For this reason, genetic algorithms, by definition, have two sources of uncertainty: inherent uncertainty in the adopted stochastic model and the uncertainty of choosing the best parameters for a good quality of the stochastic search performance process.


In this sense, uncertainty is present in the GA through the genetic operators of crossover, selection of individuals, mutation, and the definition of the stop state since the choice of parameterization of each method is a process done in a non-automatic way and based only on the programmer's experience levels,


seen as a decision-making agent. In addition, we have uncertainty linked to the very definition of the genetic algorithm because it is a non-deterministic solution; that is, each interaction can bring a different solution but close to the optimal global solution.


In this way, we observe the need to reduce uncertainty in the definition and selection of its key parameters because, on the one hand, the genetic mutation operator conducts Exploration to identify yet unexplored solution subspaces i.e., those subspaces that visit entirely new regions of a search space. On the other hand, Exploitation is performed by crossover, which operates on neighboring solution subspaces, i.e., within the vicinity of previously visited points, to find optimal local solutions (Črepinšek et al., 2013).

When doing an Exploration, genetic algorithms employ the mutation rate, responsible for determining the number of new search subspaces to be considered for construction in the next iteration of the algorithm. However, determining the optimal value for the mutation rate is a challenge, given that different problems may require different values.

Programmers determine the mutation rate choice, which can be used as a default value by trial and error. However, these ways of choosing the mutation rate

^a <https://orcid.org/0000-0001-5806-2798>

^b <https://orcid.org/0000-0002-2291-0668>

^c <https://orcid.org/0000-0001-6304-0083>

may generate or not an efficient Exploration search due to all sources of uncertainty in the decision process of selecting the proper parameters.

In this article, we investigate the impact of uncertainty sources of a genetic algorithm in the combination of selection and substitution methods on the GA convergence time and population diversity, evaluating the behavior of two types of uncertainty: the uncertainty inherent to the stochasticity of the algorithm itself (probabilistic uncertainty) and the possibilistic uncertainty present in the definition of population diversity, as well as in the definition of chromosomal fitness as the main parameters, due to the inherent vagueness of these concepts, in the view of the decision-making agent (the programmer of the genetic algorithm).

Thus, our proposal focused on the development of a self-adaptive genetic algorithm, investing in the mutation rate with variation in the combination of selection and substitution methods, but also in the use of fuzzy logic to deal with the conceptual vagueness present in the definition's parameters of population diversity and chromosomal fitness.

In summary, the main contribution of this article is to expose how uncertainty affects the genetic algorithm due to its stochastic nature, but also how to take advantage of the correct choice of evolutionary methods responsible for the search for the optimal individual and the convergence time of the genetic algorithm, dealing directly with conceptual vagueness, with fuzzy sets, associated with population variability and chromosomal aptitude of the best-adapted individuals.

2 SOURCES OF UNCERTAINTY IN THE GENETIC ALGORITHM

In the evaluation of a genetic algorithm, we find two sources of uncertainty:

- 1) uncertainty inherent to the stochastic GA model
- 2) uncertainty inherent in random choices of parameterization by the programmer

In GA, uncertainty is based on these two points. On the one hand, we have to model through the genetic operators of mutation, crossover, algorithm stopping point, and method of choosing the most adapted individuals from one generation to another; on the other hand, we have the programmer (decision-making agent) with his prior knowledge on how to properly define these parameters, which often need

to be modified and tested during application development, being adjusted empirically and often based on vague definitions about quality of the parameters to choose.

The GA itself uses the uncertainty of the probabilistic type in the process of search and self-adaptation of individuals better adapted to the solution environment, being carried out through distributions of random variables that underlie the process of stochastic search and directly influence the convergence itself, still being related to GA parameterization.

On the other hand, the possibilistic uncertainty is due to the vagueness in the precision of the information regarding the choice of parameters that best fit a stochastic search problem and is associated with the programmer's decision (and his degree of knowledge) about the most appropriate values for the parameters from GA.

These two types of uncertainty significantly impact the results and performance of a genetic algorithm, as some of the uncertainty factors can affect the quality of the result, leading to inaccurate results or a longer computation time than necessary (Majumder et al., 2018). In addition, uncertainty can make it difficult to compare results between different cycles of executions of the genetic algorithm.

Therefore, it is essential to consider the types of uncertainty and their impacts when developing and evaluating genetic algorithms and finding strategies to mitigate such expected impacts.

3 RELATED WORK

In this section, we presented current studies considering these types of uncertainty and their impacts on GA's overall performance.

Some studies evaluate the behavior of genetic algorithms by varying the crossover and mutation rate as pointed out by (Hassanat et al., 2019), (Sun and Lu, 2019). In addition to these studies, some use Fuzzy Logic such as (Fadel et al., 2021), which shows its use in reducing GA convergence time. However, in most of these works, there is no comparison between the developed algorithm, the impact of selection and substitution methods, and the characterization of uncertainty in the genetic algorithm.

The work of (Hassanat et al., 2019) shows the importance of varying the mutation rate and crossover population size to solve various problems, demonstrating each method's impact in finding the optimal solution. However, selection and substitution methods also show how they can help or hinder the search

for the optimal individual.

The work of (Sun and Lu, 2019) depicts that the premature convergence of GA unexpectedly affects the algorithm's performance. The main reason is that the evolution of exceptional individuals, which multiply rapidly, will lead to premature loss of population diversity. To solve the problem, we constructed a method to qualify the diversity of the population and the similarity between adjacent generations.

The experiment results show that it can search for the optimal solution for almost all reference functions and effectively maintain the diversity of the population. Where (Fadel et al., 2021) uses the integrated Fuzzy Logic and GA method and uses the growth rate function to calculate the change of maximum fitness and average fitness between two successive generations to adapt the crossover and mutation parameters of GA dynamically, the results showed that the proposed technique gives a high and more accurate performance in terms of maximum fitness and average fitness. Despite this, no investigation shows the impact of choosing the combination of selection and replacement methods on the algorithm's performance and also observes how the algorithm would behave if we chose diversity population as a parameter in fuzzy Logic.

In (Bajaj and Sangwan, 2019), work investigates the effectiveness of test case prioritization based on genetic algorithms. The work aims to find the best parameter settings and operator roles to increase the effectiveness of test case prioritization, which can save time and resources. A strength of this work is the investigation of how parameter configuration and operator roles affect the effectiveness of test case prioritization, providing a basis for improving the effectiveness of test case prioritization. However, a weakness is that the work is specific to genetic algorithm-based test case prioritization, which means that its findings do not necessarily apply to other optimization problems. Nevertheless, fuzzy Logic considerably improves the convergence time of the genetic algorithm compared to simple GA.

In addition, it shows how easy it is to implement and how it helps programmers who will use this technique to solve problems. However, one question remained: How can the choice of types of combination of selection and substitution methods impact the convergence time and uncertainty reduction using the algorithm optimized by Fuzzy Logic? We pointed out the ease of implementing these procedures of self-adaptation of mutation rate within the pseudo-code of a Genetic Algorithm (see algorithm 1) and the help that adopting these methods delivers to novice programmers in evolutionary computing.

4 METHODOLOGY

This section was divided into steps to show the treatment of uncertainties presented in GA. The first shows the environment in which we ran the algorithm, the second describes a chosen use case, the third presents the selected evaluation metrics to compare the algorithms to design and measure how good they were, and finally, it shows how a simple GA, in operation and a GA that used the treatment of uncertainty with fuzzy vagueness that was implemented in the GA, as well as the combination of selection and substitution parameters chosen in this work.

4.1 Test Environment

The test environment used to make the comparisons was Google Colab (Google Computer Engine) which has the following settings:

- RAM: 12GB
- HD: 108GB

We used Python version 3.7.13, and some libraries were also used, such as:

- ipython-autotime version 0.3.1 to determine the execution time
- matplotlib version 3.2.2 to display the graphs
- scikit fuzzy version 0.4.2 to build the fuzzy system

4.2 Case Study

The case study chosen in this work was OneMax, which consists of counting the one (1) bit each chromosome has and also represents the individual's fitness. Thus, the optimal binary is the individual who has all bits in one (1). Although the solution space or domain of the OneMax problem depends on the length of the chromosome, an essential feature of the OneMax domain is that all bits are unrelated (Giguere and Goldberg, 1998). The simplicity of the OneMax problem makes it an excellent candidate for studying uncertainty reduction by evaluating the performance of the simple genetic algorithm and the self-adaptive algorithm on mutation rate and verifying selection and replacement methods.

4.3 Evaluation Metrics

The metrics chosen in this work to perform the comparison of the Simple GA and the GA with the application of Fuzzy Logic were the number of generations to find the solution to the problem, the diversity

of the population, the execution time in seconds required for the genetic algorithm to find the optimal solution. Thus, shows, in a broad way, the performance obtained in the choice of each mutation method, selection, and replacement method.

In addition, We adopted the Kruskal-Wallis hypothesis test, the non-parametric test used when comparing three or more independent samples. It tells us if there is a difference between the GAs that used the combinations with substitution and selection. For comparison between the GAs, the Dwass-Steel-Crichtlow-Fligner (DSCF test) was used.

4.4 Algorithm Specification

This section is divided into each procedure presented in implementing the Genetic Algorithms adopted that are **AGEE**, **AGETV**, **AGRE**, **AGRTV**, **AGFEE**, **AGFETV**, **AGFRE**, **AGFRTV**. To show in detail, the choices adopted were divided into three topics, the first showed the basis of the algorithms, that is, what they have in common, in second showed the types of simple GA and the nomenclature used and in the third the types of proposed GA, the nomenclature used and the application of fuzzy Logic.

4.4.1 Basis of Both Algorithms

It shows how the basis of the two algorithms is defined, being:

- The representation defined was binary.
- Adopted chromosome size was twelve (12).
- Population size was one hundred (100) individuals.
- The crossover, the rate was 90%, so as not to interfere with the final result.
- Crossover was chosen with a cutoff point.

The algorithm has the following format, as seen in (Jong, 2009):

4.4.2 Simple Genetic Algorithm, Without Uncertainty Treatment

In the Simple Genetic Algorithm, variations are found with the selection and substitution method, but the mutation rate is fixed.

- Adopted mutation rate 50

The nomenclature shown in Table 1 combines the selection and substitution methods.

Input: Typical Parameters

Output: Final Solution Population

- 1: *INITIALIZES* population with random candidate solutions
- 2: *EVALUATE* each candidate
- 3: **while** condition **do**
- 4: *SELECT* parents
- 5: *RECOMBINE* pairs of parents
- 6: *MUTATION* the resulting descendants
- 7: *EVALUATES* new candidates
- 8: *REPLACES* the individuals for the new generation
- 9: **end while**

Algorithm 1: Pseudocode of a Typical GA.

Table 1: Nomenclature of Simple GA Types.

Selection	Substitution	
	Elitist	Life Time
Elitist	AGEE	AGETV
Roulette Method	AGRE	AGRTV

4.4.3 Genetic Algorithm with Fuzzy Logic Approach

The GA combined with fuzzy Logic implies modifications in the methods of selection and substitution, in addition to the definition of the implementation of Fuzzy Logic.

Table 2 shows the terminology used in the algorithms implemented with combinations of selection and substitution methods:

Table 2: Nomenclature of the Proposed GA Types.

Selection	Substitution	
	Elitist	Life Time
Elitist	AGFEE	AGFETV
Roulette Method	AGFRE	AGFRTV

4.4.4 Application of Fuzzy Logic

The population diversity is assessed by calculating the percentage of unique individuals within the current population, i.e., those with non-repeating genetic material, as depicted in Equation 1. This computation reflects the diversity within the current population. This calculation is performed globally at each algorithm generation to determine the population diversity rate. Additionally, the individual's score within the population is computed to be used as input in Fuzzy rules, thereby determining the mutation percentage (MP).

$$PD = \frac{UniqueIndividuals}{Population} \times 100 \quad (1)$$

Twenty-five rules for the OneMax problem were considered in the application using the triangular membership function representation. The rules have the following antecedent values of individual's adaptation and population diversity (PD), both with the fuzzy set 'verybad', 'bad', 'medium', 'good', and 'verygood' as represented in Figure 1 and as a consequence the individual's MP, which is represented by the fuzzy set 'verylow', 'low', 'medium', 'high', 'veryhigh' which is depicted in Figure 2 [0,100].

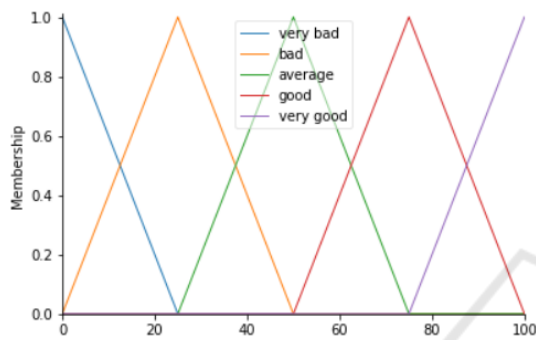


Figure 1: Population Quality and Diversity.

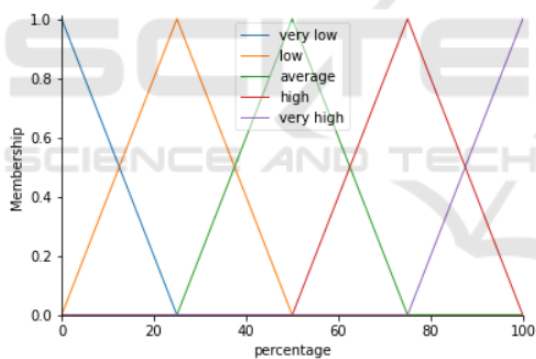


Figure 2: Mutation Percentage Graph.

The rules chosen in the fuzzy system were expressed as follows:

5 RESULTS AND DISCUSSION

To compare uncertainty issues and their treatment performance, we executed each algorithm a hundred (100) times; that is, each execution means that the algorithm was started and just stopped when it found the solution to the problem. When found, the data is stored, and the system is restarted. This looping happens until the 100 runs are completed, and then, the extraction of the results is performed. Another critical factor was the division into two sections, one with the

Table 3: Fuzzy Rules.

Rule	If	Quality	And	PD	Then	MP
1	If	very bad	And	very bad	Then	very high
2	If	very bad	And	bad	Then	very high
3	If	very bad	And	average	Then	very high
4	If	very bad	And	good	Then	very high
5	If	very bad	And	very good	Then	very high
6	If	bad	And	very bad	Then	very high
7	If	bad	And	bad	Then	high
8	If	bad	And	average	Then	average
9	If	bad	And	good	Then	low
10	If	bad	And	very good	Then	high
11	If	average	And	very bad	Then	high
12	If	average	And	bad	Then	average
13	If	average	And	average	Then	average
14	If	average	And	good	Then	low
15	If	average	And	very good	Then	low
16	If	good	And	very bad	Then	low
17	If	good	And	bad	Then	low
18	If	good	And	average	Then	low
19	If	good	And	good	Then	low
20	If	good	And	very good	Then	very low
21	If	very good	And	very bad	Then	very low
22	If	very good	And	bad	Then	very low
23	If	very good	And	average	Then	very low
24	If	very good	And	good	Then	very low
25	If	very good	And	very good	Then	very low

results of the Simple GA and another with the results of the Proposed GA.

5.1 Results Obtained in Simple GA, Without Uncertainty Treatment

First, simple genetic algorithms' convergence time and their variations were analyzed.

Figure 3 shows the time for each of the 100 runs that each algorithm obtained, that is, the time each run took to find the best solution to the OneMax problem. Given the above, it is remarkable the difference that exists in the execution time of the algorithms.

The AGRE algorithm, on average, obtained the worst performance, about 22.80 seconds (s), compared to the other simple genetic algorithms, which obtained AGEE 21.60 s, AGETV 0.30 s, and AGRTV 0.28 s.

One of the reasons analyzed for this difference in time of the simple GA that chose the Elitist substitution about the simple GA with the lifetime substitution was the delay of convergence of the population as seen in (Kim and Han, 2000) due to the size of the population that directly impacts on its convergence time, that is, the substitution by Life Time maintains

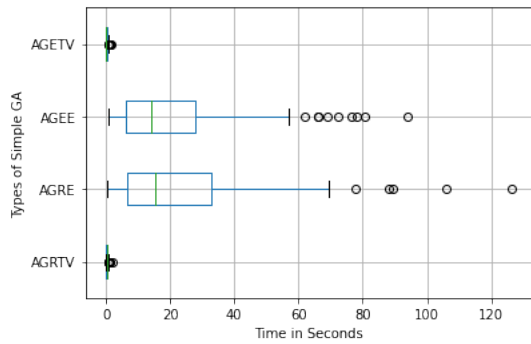


Figure 3: Execution Time Simple GA.

a more significant amount of living individuals per time interval, unlike the Elitist that in each iteration eliminates the individuals with a lower score, impairing the emergence of the optimal solution. Another critical point is that the roulette wheel method allows the crossover to individuals with different configurations of their chromosomes, bringing them good or bad, which provides a larger sweep in the Exploitation search. At the same time, the Electoral selection can be stuck at the local optimum.

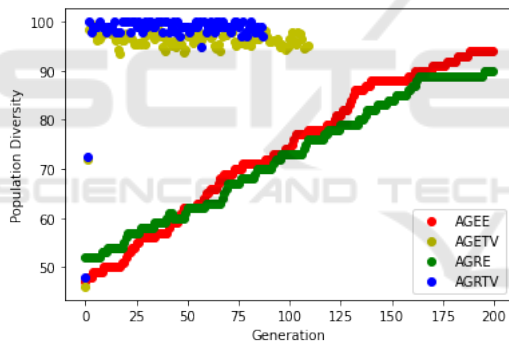


Figure 4: Population Diversity by Generation Simple GA.

In addition, the diversity of the population present in a run was verified, as shown in Figure 4. It was observed that the Simple Genetic Algorithms with Elitism in the present replacement diversity, in the beginning, were equal to the simple GA with replacement by lifetime, but over the generations, this diversity changes because the algorithms AGETV and AGRTV have a larger population, but the lifetime set in each generation is different, thus implying a more significant amount of living individuals, but when the maximum lifetime is reached this generation is destroyed, which helps in the high diversity. On the other hand, the AGEE and AGRE algorithms have a reduced population, but the amount of different individuals within it is not large by keeping optimal individuals, which sometimes have the same genetic material.

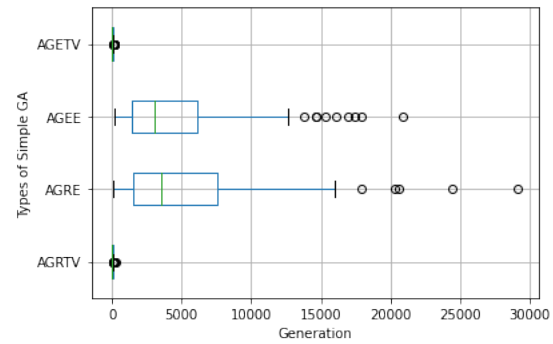


Figure 5: Result of the Generation by Run.

Finally, Figure 5 shows the number of generations needed to find the optimal individual in each run, which averaged AGRTV 41.48, AGRE 5253.03, AGEE 4798.50 and AGETV 41.42. Being highlighted, the Simple GA is implemented with the lifetime, which can find the optimal individual first and with a small number of iterations.

5.2 Results Obtained in the Proposed GA with Fuzzy Approach

The tests performed on the Simple Genetic Algorithm were also performed on the Genetic algorithm combined with Fuzzy Logic with its variations on the substitution and selection method to be able to make comparisons between them.

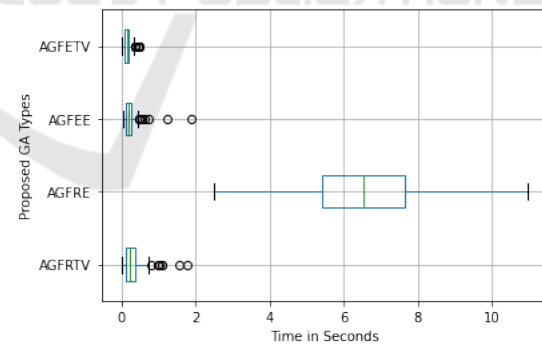


Figure 6: Proposed GA Runtime.

The AGFRE algorithm had the worst performance, averaging about 6.52 s compared to the other genetic algorithms combined with Fuzzy Logic, which had AGFETV 0.16s, AGFRTV 0.30s, and AGFEE 0.23s.

The self-adaptive genetic algorithms that used the method of Lifetime Replacement have an increased population, which impacts reducing the convergence time of the GA. In addition, the self-adaptive GA maintains a high search in the Exploration approach,

i.e., effectively sweeps a more significant number of search spaces; this occurs because it can infer an appropriate mutation rate for each chromosome in the population, taking into account the diversity of the population and the quality of the chromosome, which interferes with faster convergence.

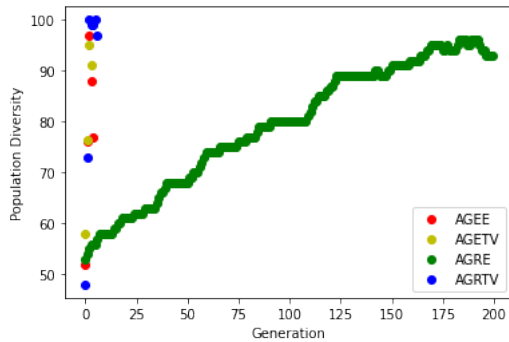


Figure 7: Population Diversity by Generation Proposed GA.

There was a check on the diversity of the population present in a run, as shown in Figure 7. It was observed that the Self-Adaptive Genetic Algorithms with Electism in the substitution present diversity, in the beginning, equal to that of the Self-Adaptive GA with replacement by Life Time, but during the generations, this diversity diverges because the algorithms AGFETV and AGFRTV have a larger population and their diversity as well. On the other hand, the algorithms AGFEE and AGFRE have a reduced population, which impacts an increasing diversity throughout the generations; that is, there is a greater chance of having individuals with the same genetic material.

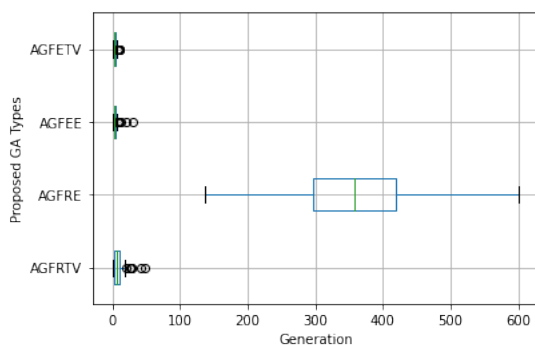


Figure 8: Generation of Each Proposed GA Run.

Figure 8 illustrates the number of generations needed to find the best-fit chromosome, which had an average of AGFRTV 7.92, AGFRE 356.84, AGFEE 3.76, AGFETV 3.40 generations. The algorithms that stood out were AGFEE and AGFETV, which had the Elitist selection method. It shows the importance of correctly defining the mutation rate and the choice of

selection and substitution methods.

5.3 Hypothesis Testing

The Kruskal-Wallis test is a non-parametric test used when comparing three or more independent samples. It indicates if there is a difference between at least two of them (Rumsey, 2015).

Therefore, the number of generations needed to find the optimal individual per run was selected for analysis. First, the following hypotheses were raised to check if there is a difference in the choice between the selection and replacement methods:

- H_0 : There is no statistically significant difference between the group means.
- H_1 : There is a difference between group means.

A significance level of 5% was chosen. The test result was less than 0.001 s; that is, the result is lower than the significance level, and there is a statistical difference. Table 4 shows the comparison between each of the Simple GA types.

Table 4: Kruskal-Wallis Test Simple GA.

		W	p
AGEE	AGETV	-17.261	0.001
AGEE	AGRE	0.734	0.955
AGEE	AGRTV	-17.257	0.001
AGETV	AGRE	17.225	0.001
AGETV	AGRTV	1.294	0.797
AGRE	AGRTV	-17.240	0.001

Table 4 shows no difference between the AGEE and AGRE methods, and the same is true for AGETV and AGRTV. However, the Life Time Replacement method causes a statistically relevant difference in the cases.

The same test was performed with the GAs that used Fuzzy Logic, adopting a significance level of 5%. Again, the result was less than 0.001 s; that is, there is a statistical difference between the groups. Table 5 shows the comparison between the groups.

Table 5: Kruskal-Wallis Test GA with Fuzzy Logic.

		W	p
AGFEE	AGFETV	-0.142	0.987
AGFEE	AGFRE	17.317	0.001
AGFEE	AGFRTV	6.721	0.001
AGFETV	AGFRE	17.308	0.001
AGFETV	AGFRTV	6.9787	0.001
AGFRE	AGFRTV	-17.283	0.001

In Table 5, there is no difference between the AGFEE and the AGFETV methods, whereas there is

a statistically significant difference between the other methods.

6 CONCLUSION AND FUTURE WORK

In this article, we present an approach that addresses types of uncertainty inherent in the development and execution of genetic algorithms. Experimentally, we applied the proposed method with the fuzzy approach and different selection and substitution methods. We showed that the genetic algorithm combined with the conceptual vagueness treatment, using the elitist selection and the lifetime substitution method, presented the best results compared to the other combinations. In this sense, this work highlights the relevance of this approach to improve the performance of genetic algorithms and also shows the reduction of uncertainty found in genetic algorithms through implementing a self-adaptive algorithm that helps and facilitates the search for the global optimum. This contribution is a significant step towards improving performance and knowledge representation in GA.

In our approach, Fuzzy Logic was used in the proposed algorithm in a simple way to implement, which confirms the adoption of these modifications in its implementation, showing the importance of modification in the current population-sensitive interactive mutation rate.

In addition, it was also shown the importance of testing with different methods of substitution and selection because the reduction in convergence time is significant in the OneMax problem, this difference in most cases tested being greater than 90%, i.e., the methods that used Time of Life substitution had an advantage. On the other hand, the methods that used Elitism had an increasing diversity along the generations and a long time of convergence.

Using Fuzzy has a downside linked to the algorithm's convergence time; a larger population results in longer computational times. This study is limited by analyzing only a single test case, the Onemax problem, diminishing the generalization of diversity understanding for uncertainty reduction in the GA.

In future work, we intend to apply the proposed GA to other classes of problems and analyze how this algorithm behaves. In this way, it facilitates the developer in deciding which problems the adoption of the self-adaptive genetic algorithm is recommended. For example, in the Traveling Salesman Problem (TSP) function Optimization, the methodology adopted in this work focuses on the variation of the mutation rate and variation of the substitution and selection

method to corroborate the results obtained in this work. Linked to this, analyze in more detail in Fuzzy Logic the consequences of changing the mutation rate in different functions of representation of knowledge as the Trapezoidal, Gaussian.

Another thread for future work is to verify if the methodology applied here can be extended to all rates, such as crossover and population size variation. Thus, the importance of changing the parameters during the execution of the genetic algorithm iterative can be seen.

REFERENCES

- Bajaj, A. and Sangwan, O. P. (2019). Study the impact of parameter settings and operators role for genetic algorithm based test case prioritization. In *Proceedings of International Conference on Sustainable Computing in Science, Technology and Management (SUSCOM)*, Amity University Rajasthan, Jaipur-India.
- Črepinšek, M., Liu, S.-H., and Mernik, M. (2013). Exploration and exploitation in evolutionary algorithms: A survey. *ACM computing surveys (CSUR)*, 45(3):1–33.
- Fadel, I. A., Alsanabani, H., Öz, C., Kamal, T., Iskefiyeli, M., and Abdien, F. (2021). Hybrid fuzzy-genetic algorithm to automated discovery of prediction rules. *Journal of Intelligent & Fuzzy Systems*, 40(1):43–52.
- Giguere, P. and Goldberg, D. E. (1998). Population sizing for optimum sampling with genetic algorithms: A case study of the onemax problem. *Genetic Programming*, 98:496–503.
- Hassanat, A., Almohammadi, K., Alkafaween, E., Abunawas, E., Hammouri, A., and Prasath, V. S. (2019). Choosing mutation and crossover ratios for genetic algorithms—a review with a new dynamic approach. *Information*, 10(12):390.
- Jong, K. D. (2009). Evolutionary computation. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(1):52–56.
- Kim, K.-j. and Han, I. (2000). Genetic algorithms approach to feature discretization in artificial neural networks for the prediction of stock price index. *Expert systems with Applications*, 19(2):125–132.
- Majumder, S., Saha, B., Anand, P., Kar, S., and Pal, T. (2018). Uncertainty based genetic algorithm with varying population for random fuzzy maximum flow problem. *Expert Systems*, 35(4):e12264.
- Rumsey, D. J. (2015). *U Can: statistics for dummies*. John Wiley & Sons.
- Sun, N. and Lu, Y. (2019). A self-adaptive genetic algorithm with improved mutation mode based on measurement of population diversity. *Neural Computing and Applications*, 31(5):1435–1443.