Computer Science and Engineering

Second Semester / Course 2024/2025

Artificial Intelligence 2
Practical Work IV

Done by

Christopher Cobo Piekenbrock

Rodrigo Sáez Escobar

Gonzalo Eusa Silvela

Jaime López de Heredia Delgado

Juan Fernández Cerezo

Teacher

Moisés Martínez Muñoz

Abstract

In this practical work, we explored how genetic algorithms can be used to evolve a population of strings towards a set target sentence. Our implementation used genetic operators like selection, mutation and crossover and we tested how elitism impacted the performance. The goal was to understand how different configurations affected the algorithm's ability to reach the solution. We observed how the population evolved over time while measuring metrics like fitness, match accuracy and the population diversity. These experiments helped us view the behavior of the algorithm and how some small changes in the parameters could affect the convergence.

Table of Contents

Table of Figures

Index of Tables

Index of Equations

1. Introduction

Genetic algorithms are a way to solve problems by simulating the process of natural evolution. In this practical work, we built a simple genetic algorithm with the goal of evolving a random population of strings into a set target sentence. Each individual is made up of characters that represent genes, and through multiple generations, the algorithm tries to improve the population using selection, mutation and elitism.

Our main objective is to understand how different genetic operators affect the algorithm's ability to find the target sentence. We ran tests using elitism and without and tracked how the population changed over time. The idea was not only to reach the target sentence but also to learn how different parameters could affect the speed and quality of convergence. This research document reflects what we did and what we learned.

All experiments done are influenced by the learnings obtained in class and with the slides provided by our teacher from Unit 8 – Genetic Algorithms [1]

2. Exercise 1

2.1. Description of the exercise and their goals.

In this exercise, we have implemented a genetic algorithm to evolve a population of individuals. As we have seen in the theory classes, each individual has a chromosome, which is a sequence of characters. The length of the chromosome is the same as the target sentence we want to find.

The idea is to improve the population over time so that individuals get closer to the target sentence. This happens by using some genetic operations learned like selection, crossover, and mutation. The algorithm measures the fitness of each individual by counting how many characters match the target sentence perfectly in characters and position, which helps guide the evolution.

Some important configurations for the algorithm can be the target sentence, which for this practical work is a fixed sentence, how big the population is, how many generations it runs for, the mutation rate, and how often the progress of the evolution is reported, or samples of the population are taken.

The main objective is to see how the population evolves to reach the target sentence, check how different genetic operations like using elitism or not affects the evolution, and analyze how the algorithm does and converges using different configurations.

2.2. Description of the Dataset

In this practical work, we're working with a single target sentence: "ANIMULA VAGULA BLANDULA". This sentence is the end goal for our genetic algorithm, which tries to create solutions/individuals that match the sentence exactly. Each individual is a string of characters that's the same length as the target phrase.

The characters come from a set that includes all the uppercase letters in the Spanish alphabet like specified in the instructions, also, in this set, we also include the space character, having a total of 28 possible characters for each position in the string.

Because of this, our dataset isn't a regular table of numbers or values like in the other practical works, instead, it's just this symbolic target that guides the whole evolution process. The main objective of the genetic algorithm is to evolve strings that get closer and closer to the target sentence mentioned above, and the way we measure success is by counting how many characters in the individual match the target exactly in the same spots.

2.3. Data Preparation Methods

Since the dataset in this project is just one target sentence, the data preparation was simple and focused on preparing that sentence for the genetic algorithm. We did the following things:

Alphabet Setup
As mentioned before, we defined the Spanish alphabet to include all uppercase letters and the space character so the algorithm could handle every character that appears in the target string.

Encoding
Each character in the target sentence was given a number based on where it appears in the alphabet. This let us represent each individual in the population as a list of numbers, which made it easier to apply genetic operations like crossover and mutation.

Preparation for evaluation
To make the comparison individuals faster, the target sentence was also converted into its number sequence so the algorithm could quickly check how many genes match the target during the fitness calculation.

2.4. Experimentation and Evaluation

2.4.1. Experimentation phase

In this experimentation phase, we ran the genetic algorithm using two different configurations to see how elitism affects the algorithm finds the target sentence.

We tried the following configurations:

With Elitism

In this configuration, 5 % of the best individuals(elites) were selected in each generation. These elites were copied, then mutated, and finally placed back into the population, replacing the worst individuals. This way, the best solutions are not lost, but they still change a bit because of the mutation, which allows them to explore new options.

Without Elitism

In this case, no individuals are protected between generations. All individuals, including the best ones, are processed normally using selection, crossover, and mutation.
This means that good solutions can be lost in the next generation.

For both configurations, the algorithm was run using the following final parameters, chosen after multiple tests and adjustments based on the graphs.

Population size: 1000

Generations: 10000

Mutation probability (Pm): 0.8 (For the mutation probability we got some misunderstandings because we thought that 0.8 would mutate 80% of the individuals, but finally we saw that without formula it´s the opposite. It´s 1 – Pm.)

nres: 2 generations

nsam: every 100 generations

For each generation, we recorded the measurements specified in the practical work instructions:

BESTf: The best fitness value in the population

BESTn: How many individuals had the best fitness

BEST%: Percentage of individuals with the best fitness

MatchMax: The highest number of matching characters with the target sentence

MatchAvg: The average number of matching characters in the whole population

Every 100 generations, we also took a random sample of 20% of the population to check the diversity and how close individuals were to the target sentence.

After running both configurations, we made multiple graphs showing how the metrics mentioned above changed over time. These graphs helped us compare the effect of elitism on how quickly and well the population converged on the target phrase.

2.4.2. Evaluation

To test how well our genetic algorithm works, we ran it twice: once using elitism and once without it. In both runs, the goal was to evolve a population until one matched the target sentence. During each execution, we measured values like how many characters matched (MatchMax), how many matched on average (MatchAvg), the best fitness (BESTf), and how many individuals had that best fitness (BESTn and BEST%).

What we observed

With elitism, the results improved dramatically. The population progressed faster, MatchMax and MatchAvg reached high values (up to 23 and 21 respectively), and BESTf converged to 1.0, meaning we successfully found the perfect solution.

Without elitism, the performance was much lower. The algorithm rarely exceeded 16-17 in MatchMax, the population stayed more random, and we never got close to the optimal fitness value. The best fitness without elitism was just 0.0183 compared to 1.0 with elitism.

Using the graphs

The difference between the two runs was very clear when we looked at the graphs. Elitism helped preserve good individuals, which pushed the population forward. Without elitism, good solutions were often lost, and the algorithm had to "start over" again and again.

2.5. Questions

2.5.1. What is the maximum value that the fitness can reach? Why?

The maximum fitness that an individual can reach is 1.0.
This happens when all the characters match exactly the target sentence, both the correct letters and in the correct positions.

$$F_i = e^{(nMatchi - targetLength)} - e^{(-targetLength)}$$

Equation 1 - Fitness Function

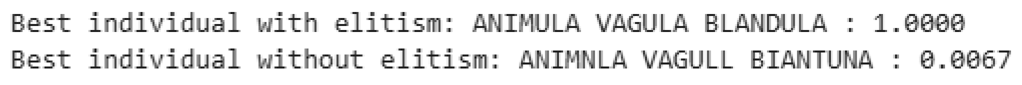
This is because, as shown in when the number of matches equals the target length of 23, the fitness function becomes:

$$f(x) = e^{23-23} - e^{-23} = e^{1} - e^{-23} = 1 - e^{-23}$$

Equation 2 - Fitness Function at nMatch = 23 - Self-made

Since e^{-23} is a very small number close to 0, the result is practically 1. This is the highest possible fitness value, as it means the individual is a perfect match to the target sentence.

In our case, as you can see in figure 1, the best individual we got with elitism has a fitness of 1, which means it got all the correct characters in the correct order. In the case without elitism, the best individual had a fitness of 0.0067, which means it got some correct characters but was still far from the target.



```
Best individual with elitism: ANIMULA VAGULA BLANDULA : 1.0000
Best individual without elitism: ANIMNLA VAGULL BIANTUNA : 0.0067
```

Figure 1 - Best individual with and without elitism – Self-made

These results show that elitism is essential to preserve the best solutions and help the algorithm converge efficiently.

Throughout the experiment, we tested different values for population size, mutation probabilities, and generations, always comparing the results carefully. The combination we presented in this report, with elitism and the selected parameters, was the best and most stable we found after several tests.

2.5.2. For the target phrase *ANIMULA VAGULA BLANDULA*, analyze what happens to BESTn, BEST%, matchMax, and matchAvg. Explain the resulting graph. Also, examine how the consensus individual evolves under these conditions.

To evaluate the performance of the genetic algorithm under the target sentence "*ANIMULA VAGULA BLANDULA"*, we analyzed the evolution of four metrics across 10,000 generations with and without elitism, BESTn, BEST%, MatchMax, and MatchAvg. Additionally, we used BESTf as a reference to observe the overall fitness evolution.

We iteratively adjusted parameters such as mutation rate, elitism, and selection pressure. At each iteration, we generated visual graphs for all key metrics and observed their trends. Based on those observations, we made improvements to the configuration to reach the final performance shown in the figures below.

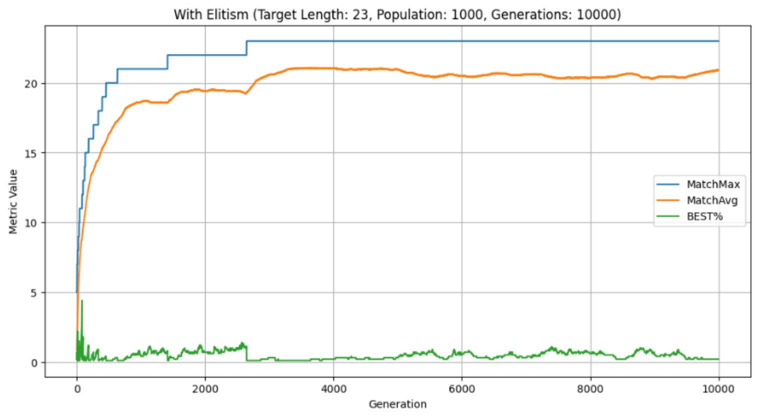


Figure 2 - Metrics with elitism – Self-made

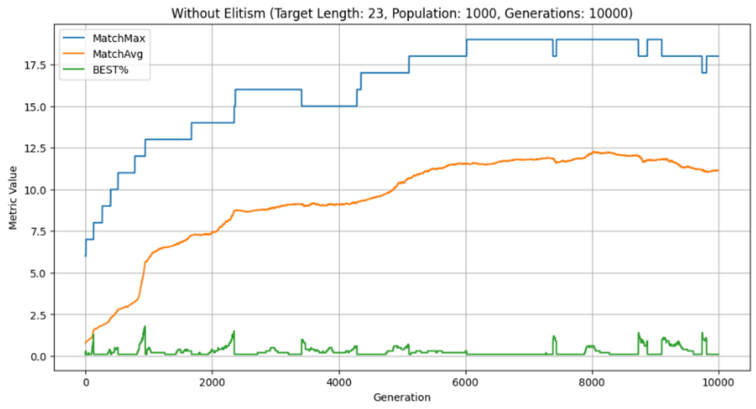


Figure 3 - Metrics without elitism – Self-made

Metric	With Elitism (Mean)	Without Elitism (Mean)
BESTf	0.7924	0.0069
BESTn	4.52	2.55
BEST%	0.45%	0.25%
MatchMax	22.31	16.31
MatchAvg	19.84	9.46

Table 1 - Summary of metrics with and without elitism – Self-made

We adjusted parameters such as mutation rate, elitism, and selection pressure. At each iteration, we generated visual graphs for all key metrics and observed their trends. Based on those observations, we made improvements to the configuration to reach the final performance shown here.

These metrics clearly show that elitism significantly improves performance in all areas.

With the BESTf metric, which reflects how close the best individual is to the perfect match, we have seen that with elitism BESTf values are gradually increasing. You can see in the figure 4, around generation 3000, the fitness reached the maximum possible value of 1.

On the other hand, without elitism, the BESTf barely improves across generations. It remains low throughout the process, never getting to an optimal value.

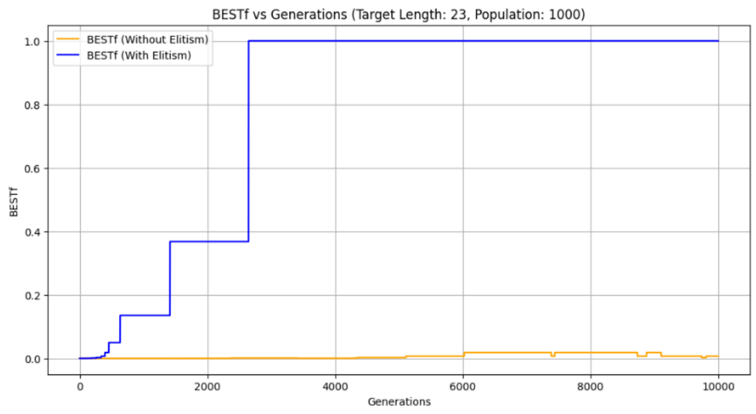


Figure 4 - BESTf with and without elitism – Self-made

Regarding the MatchMax metric we have been able to see that with elitism the population gets better quickly. *MatchMax* often reaches around 22, which means the best individual is very close to the target objective. On the other hand, without elitism, there's some progress at the start, but in a moment it gets stable, achieving an average of 16.31, showing that the algorithm struggles to consistently generate individuals close to the target.

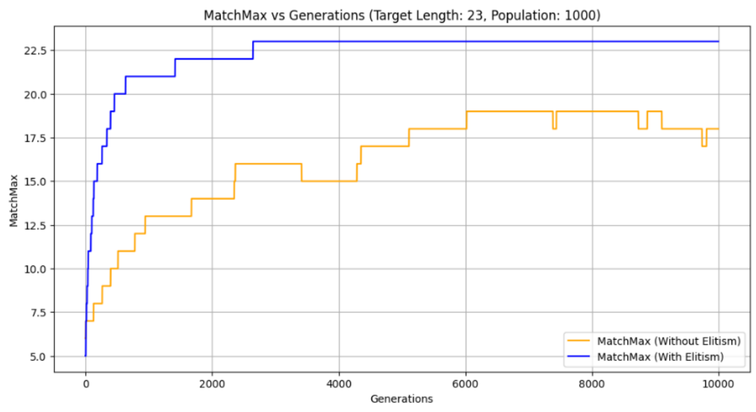


Figure 5 - MatchMax with and without elitism – Self-made

With the MatchAvg metric we have know how good the whole population is on average.

We have been able to see that with Elitism we achieve values that goes up to 20, which means most individuals were very close to the target sentence. Without elitism, the average stayed low (around 11), showing that only a few good individuals were found, and they were often lost.

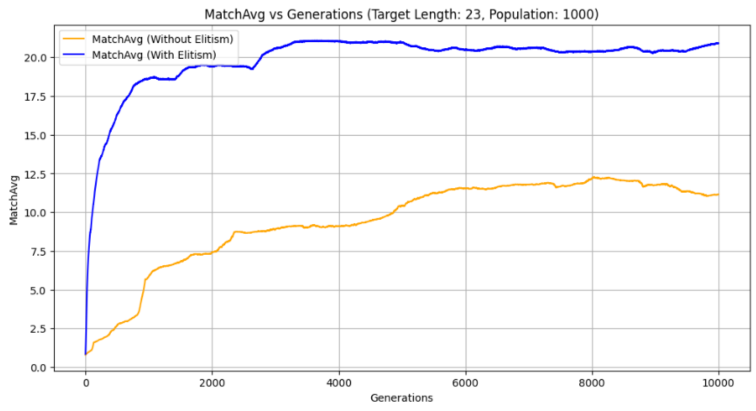


Figure 6 - MatchAvg with and without elitism – Self-made

BEST% shows how much of the population reaches the best fitness. With elitism, the average is about 0.45%, showing more stable preservation of top individuals. Without elitism, it stays low and unstable, around 0.25%, meaning that the best ones often disappear in the next generation. This can be observed in the figure 7.

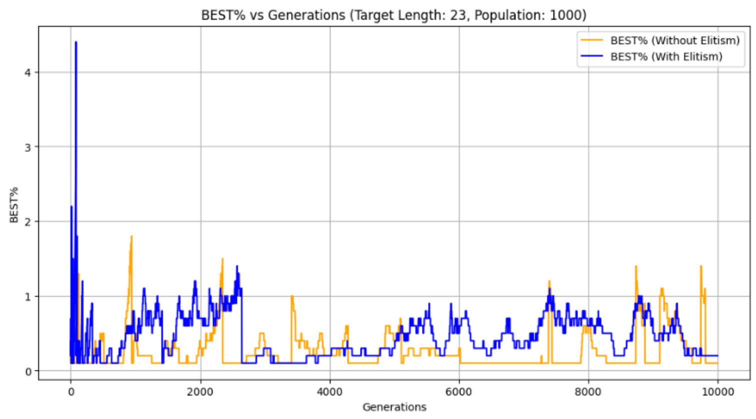


Figure 7 - BEST% with and without elitism – Self-made

BESTn shows how many individuals share the top fitness. With elitism, around 4 or 5 individuals per generation reached the best fitness. Sometimes we even got up to 44. This helped spread the best solutions. Without elitism, it was usually just 1 or 2, and they could disappear quickly, slowing down progress.

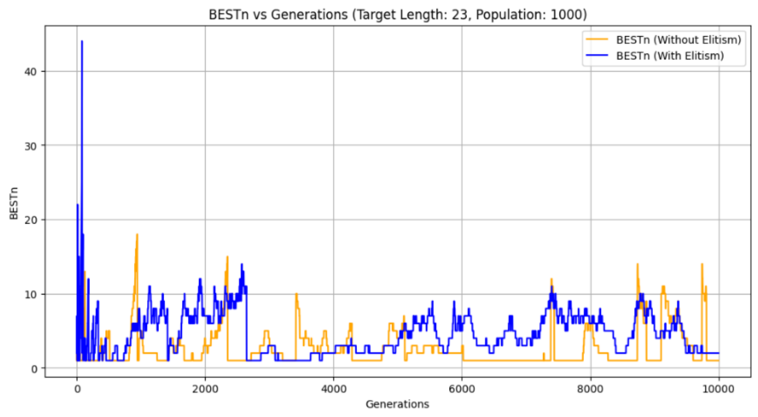


Figure 8 - BESTn with and without elitism – Self-made

As a conclusion, we can clearly see that elitism improves the algorithm a lot because the population becomes more stable, it evolves faster and it finds better solutions. Thanks to elitism, the final individual matched the exact target phrase “ANIMULA VAGULA BLANDULA”, while without elitism, it did not.

We tested different values and improved our settings step by step using the graphs. The final configuration as we mentioned before was:

Pm = 0.8, nres = 2, nsam = 100, population = 1000, generations = 10000.

This practical work shows that keeping the best individuals is very important when solving problems like this one.

3. Conclusions

To conclude all the work we’ve done, we learned a lot about how genetic algorithms behave and how important the right configuration is.

When we used elitism, the population improved more quickly and stayed on track. It was like slowly shaping the solution. Without elitism, the results were much more unstable. Sometimes things improved, but sometimes good individuals were lost and the population got worse before getting better.

Elitism clearly helped the algorithm to perform better. But we also realized that small changes, like adjusting the parameters, can make a big difference.

Overall, this project showed us how important it is to test, observe, and improve step by step. In the end, we saw how order can come from randomness when the right choices are made.

4. References

[1] M. M. Muñoz, “Unit 8. Genetic Algorithms”.