Genetic algorithm - Assignment 1

Vlad-Cristian Stoica — Data Science and Artificial Intelligence, 1st year

Abstract.

This report explores the application of a Genetic Algorithm (denoted as GA) to evolve a population of candidate strings toward a predefined target: "HELLO WORLD". I have used the tournament selection procedure, single point crossover, and character wise mutation. The algorithm was tested in different mutation rates and population sizes. The results show that moderate mutation rates and sufficiently large populations significantly improve convergence speed.

1 Methodology and algorithm design.

This section will present the actual flow of the algorithm, together with the fitness function which was modified to receive the best results while still keeping everything clean and simple. The design of the actual code was made with respect to the Software Engineering principles which were, of course, applied during the actual process of fabricating the code.

1.1 Chromosome representation.

Each individual is a fixed-length string of characters. The gene pool consists of uppercase letters (from A to Z) and the space character, meaning a 26 uppercase characters, which provides a minamal yet sufficient alphabet letters in order to express the target.

1.2 Fitness function.

Fitness is computed as the proportion of characters that match the target string in both position and value:

$$\text{Fitness} = \frac{\text{Number of matching characters}}{\text{Total characters}}$$

This provides a smooth and intuitive gradient for selection pressure.
At first glance, I tried a more complicated fitness function, but I changed it to the current one since I wanted to keep it simple and effective.

1.3 Selection: tournament.

Parent selection is performed using tournament selection with a tournament size of k = 5. This method balances selectition of parents and population diversity by consistently favoring fitter individuals while still allowing weaker candidates a chance to be chosen. Compared to the roulette wheel selection method, tournament selection tends to reduce the risk of convergence.

1.3.1 Elitism.

In addition to the parent selection, the algorithm incorporates **elitism**: the best individual in the current generation is directly copied into the next one without any alteration. This guarantees that the highest quality solution of individuals is preserved, improving convergence speed and ensuring that valuable individual is not lost due to crossover or mutation.

1.4 Crossover.

Having 2 parent chromosomes, we pick a random position along the string (from 1 to 26). We split both parents at that position and combine them by taking the prefix from the first parent and the suffix from the other one. The result is a new string that has some genes from both.

1.5 Mutation.

Mutation is the probability of changing a letter in the String. As discussed in previous studies on Genetic algorithms [1], mutation plays a crucial role in maintaining diversity and avoiding premature convergence. So each gene has a fixed probability p of being replaced by a random character. The mutation rate is experimentally varied between 0.001 and 0.3 to assess impact as shown below.

1.6 Parameters.

The Genetic algorithm was tested using different parameters, in order to check which ones do the best job. Additionaly, I have to mention that while the population size might work best (hypothetically) with a value of 100 and a mutation rate of 0.001, I had to test all of the value of each parameter for a specific population size to get the best result.

- Population size: 20, 50, 100, 500
- Mutation rate: 0.001–0.3
- Tournament size: 5 (fixed)
- Max generations: 10,000 (fixed, performance wise works the best)

2 Experimental results.

The following tables have been created using only the most effective strategies of parameters, since this matters the most. More strategies were experimented but because of computational costs and efficiency I chose to display only the relevant data inside the tables.

2.1 Effect of mutation rate

Table 1: Mutation rate vs. generations to convergence

Mutation Rate	Generations	Best phrase
0.001	217	HELLO WORLD
0.01	68	HELLO WORLD
0.03	31	HELLO WORLD
0.07	21	HELLO WORLD
0.1	16	HELLO WORLD
0.3	6065	HELLO WORLD

*using a population of 100 samples.

2.2 Effect of population size.

Table 2: Population size vs. generations to convergence

Population size	generations	Best phrase
20	137	HELLO WORLD
50	41	HELLO WORLD
100	16	HELLO WORLD
500	10	HELLO WORLD

*using a mutation rate of 0.05.

2.3 Observations.

Lower mutation rates preserve information but can stagnate. Excessive mutation increases randomness. Very small populations are slower due to lack of diversity, and large populations converge faster but at higher computational cost.
A high number of generations will consume a lot of computational and time resources (since an experiment with 100.000.000 generations was conducted), which makes this value have no sense in advancing on. I have concluded that the maximum number of generations would be 10.000, just because it offers efficiency while still maintaining a great performance. Also, a very high number of generations will make the algorithm to converge and not be able to adress the target ("HELLO WORLD").
Of course, the parameters you choose to implement will independently depend on the context or usage of the GA.

3 Discussion.

The design choices were driven by simplicity, interpretability, and performance - the actual requirments of the assessment. Tournament selection avoided early convergence and maintained healthy variation. The single point crossover leveraged partial matches. Mutation acted as the key exploration tool, too much mutation destabilized convergence, while too little caused plateaus. The latest fitness function provided a clear gradient for selection.

4 Conclusion.

The GA successfully evolved a phrase to match the target, no matter what the input parameters where given (based on the data in the table). The best performance was achieved using a mutation rate of 0.1 and a population size of 500. The results highlight the sensitivity of convergence to parameter tuning and suggest broader applications of GAs in discrete search problems, such as finding a launch parameter for a probe in the space.

References

[1] A. Lambora, K. Gupta and K. Chopra, "Genetic Algorithm- A Literature Review," 2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon), Faridabad, India, 2019, pp. 380–384, doi: 10.1109/COMITCon.2019.8862255.