Victor Franke and Laurenz Ohnemüller
Alard Roebroeck and Michael Capalbo
4. December 2018

Genetic Algorithm Report

Practical Session 1

Table of Content:

I. **Introduction**

Imagine a monkey would receive a typewriter on which he would hit random keys for an infinitive length of time, then he would almost certainly type all books of the British Library. This phenomena is called the *Infinite-Monkey-Theorem*.
In this experiment, the monkey can be seen as an abstract devise that produces a continuous sequence of random letters and symbols. Moreover, the term "almost certainly" has a mathematical meaning implying the probability of 1.

In our research, we are applying this phenomena with the theory of evolution. Hence, the "monkey", which we from now on relate to as individual, will not be typing any book or poem in an infinitive amount of time, but type the famous String "HELLO WORLD".
By this means, we are focusing on the question if the time under the added value of evolutionary pressure can be reduced?
To find the solution, we can design a "Genetic Algorithm" reflecting the process of natural selection where the fittest individual (referring to Charles Darwins evolutionary theory - "*Survival of the Fittest*") are selected for reproduction in order to produce an offspring of the next generation.

Let us start by describing and explaining our used notations.
Our individual is represented by its genes or rather exactly 11 characters, which consist of the alphabet and the space character. Genes are joined into a string to form a chromosome. A collection of individuals is called population.
The reproduction starts with a selection method and is achieved by two operators. Firstly, the *Cross-Over*, where you combine two of the selected individuals and a new one arises. Secondly, the *Mutation*, where one gene of a chromosome is changed. Furthermore, the selection can be performed in different ways, which will be discussed later on. The most famous are the *Elitist Selection*, the *Roulette Wheel Selection* and the *Tournament Selection*.
After the reproduction, the new generation will have some properties of its parents and perhaps some new individual ones. The fitness of each individual can now be determined by these properties. Hereby, a higher fitness correlates with a higher the chance of surviving. After all, this process keeps on iterating until the fittest individual is found, which is in our case, the individual with the string chromosome "HELLO WORLD".

These Genetic Algorithms are more frequently used as a powerful tool to solve various optimization problems. For that reason, they have also found a widespread use in scheduling-problems, financing, engineering, chemistry and also self-driving cars, which will be discussed later on.

II. Methods

A Genetic Algorithm consist of five important and essential modules and methods.
Let us start by describing the origin population, which in our experiment represent an amount of one 100 individuals. Through a specific fitness function each individual is assigned to one fitness value. This function gives the particular individual a score-point, if a gene character is on the intended position. Hereby, the maximum score that can be reached is eleven points showing an individual with the string chromosome “HELLO WORLD”.

Afterwards, a selection method has to be implemented. The basic idea is to select individuals and let them pass their genes onto the next generation. By this means, a high fitness score can enhance the chance to be selected.
In the following, we will describe three of the most used selection methods.

Firstly, we will describe the one that we used in our implementation, the *Elitist Selection*. In this selection model, the top 10/20/… individuals with the highest fitness scores are getting selected for reproduction. In our experiment, we chose to use the 20 individuals with the highest score. This is presumably the easiest and fastest method to implement. Moreover, it is really efficient, since on every individual lies a constant selection pressure. After all, this method still has some disadvantages.

The most significant downside would be the immense loss of genetic material and information, considering that the weaker individuals have no chance of surviving even though they could have crucial and relevant features. Additionally, the *Elitist Selection* is less sophisticated in comparison to the following methods like the *Roulette Wheel Selection*.

In this method, a proportion of the wheel is assigned to each of the possible fitness values, whereby a higher value is assigned to a greater proportion. Then a random selection is made through the “spin” of the wheel. This reduces the loss of genetic material and, in addition, weaker individuals can survive. Nevertheless, if the fitness values of the individuals are getting more unequal, same problem as with the *Elitist Selection* model can arise.
Lastly, we want to present the *Tournament Selection*. Hereby, a random number k of individuals from a population is getting selected. These are going to compete in a tournament, where the winners, so the ones with the highest fitness score, will be selected for reproduction.

This method is very straightforward, is efficient to code as well as implement and does not share the same problems as the above-named models. Furthermore, there is as well a constant adjustable selection pressure on each and every individual.
However, a large k could boil down the random selection leading to reduced chance for weaker individuals to survive.

After the selection is made, the reproduction begins through the *Cross-Over*, where an offspring is produced through exchanging genes of the parents.
In our specific case, we paired the 20 selected ones five times to again receive a population of hundred individuals again.

This is argued to be the most significant module of the genetic algorithm. Various forms of *Cross-Over* exist, but the simplest and most basic approach is the *Single-Point Cross-Over.* A random point within the chromosome is chosen, on which the cross-over is made. However, the critical loss in genetic material is one consequential problem of the Cross-Over model, since with every single crossing the offspring only takes some genes from the mother and some genes form the father, other properties are getting lost. To prevent this, we take advantage of *Mutation*.

Mutation is used to maintain genetic diversity by randomly changing one gene from the chromosome to another possible gene. However, the mutation rate should be set relatively low, otherwise if for example every individual would be mutated, the search would turn into a primitive random search.
In our algorithm only 5% of new individuals are getting mutated. This new generated population of offspring will now be our current generation.

To sum this part up, we want to repeat these five modules of initializing the population, assigning the fitness values, selecting individuals, crossing and finally mutating these until a generation with the individual with the highest possible fitness score is found.

III. Applications

A genetic algorithm affects a broad field and plays an important role in real-world applications. Since it is such a leading instrument in solving optimization problems, we quickly want to address it in this report.

One of these applications are computer games. The goal of the video game should be to match the players ability to the challenge presented. Hence, the genetic algorithm is supposed to develop customized "enemies" for each player. A famous example for the usage of a genetic algorithm is the game "Darwin's demons". In this game, the player is supposed to defend the galaxy against a horde of evolving enemies. Here, only the fittest enemies are reproduced, causing them to adapt to the players game strategy and making it continuously more challenging.

Another application lies in the field of robotics. The human designers and engineers intend different functions for their robots, therefore the genetic algorithm can be used to search for a range of optimal designs and components for each specific use. Additionally, it can return results for entirely new types of robots, which can perform multiple tasks and have more general applications through for example the idea of "Flexible Muscle-Based Locomotion for Bipedal Creatures" (by Thomas Geijtenbeek, Michiel van de Panne and A. Frank van der Stappen).

A further application is in self-driving cars and deep learning cars. These cars learn to maneuver through a course by themselves, using a neural network and a genetic algorithm (by Samuel Arzt).

In economics we can also find the usage of genetic algorithms in financial forecasting or even marketing. In addition, it can be used to characterize various economic models like asset pricing.

Lastly, we want to address the sector of machine learning, the so called "Genetics Based Machine Learning". This application is related and connected with most of the others, since a genetic algorithm is basically a way, how the machine learns and solves a specific optimization problem.

Since genetic algorithms are widely applicable, easy to implement and easy to parallelize, you can find them in much more areas like in scheduling problems, in engineering, chemistry and more.

IV. Results & Analysis

To gain further understanding of the problem we changed several inputs and analyzed the results. The questions that will be answered with these experiments are the following:

I. Does leaving out the crossover operator or the mutation affect the result?
II. What happens when the size of the population changes?
III. What happens when you adjust the mutation rate?

How will the mentioned modifications alter the amount of generations needed to acquire the string “HELLO WORLD”?

IV.I. Importance of Crossover & Mutation

Before looking into the effects of changing the population size and the mutation rate we will explain why the crossover and the mutation rate are necessary.

The crossover operator is the heart of this experiment. We are supposed to take the best 20 individuals and reproduce them so that the best genes are given on to the next generation. However, without the crossover operator the individuals will not combine their genes and the learning effect passed on to the next generation gets lost and we are stuck with the same individuals in every population. The mutation rate will be the only thing that changes the genes, but this would mean we have pure mutation. Henceforth, we are left with an endless seeming mutation until the result is finally achieved.

The mutation rate is used so that when the gene material of a population is stuck. This means it is simply not possible to achieve writing the string with the gene material the population has in it. At this point the mutation rate is the saving factor. It randomly chooses a character and puts it in a random position of the string of an individual, leading to new gene material. Without our code does not achieve a result.

IV.II. Population

The problem is set off with a standard population size of 100 individuals. To test how big the effect of increasing or decreasing the size of the population is, we fixed the mutation rate to five percent and the elitist selection to the top 20 individuals for all trial runs. Additionally, to the standard population size we probed two populations smaller and two greater than the 100. Each population was tested 1000 times, of which the average was taken for comparison. Therefore, the population size is our independent variable and the average generations the dependent variable. The results can be viewed in the table below.

Population Size	Average Generations
40	770,88
60	247,05
100	115,26
500	18,72
1000	11,91

Table 1: Population size compared to the average generations needed.

Clearly, the average generations needed for an individual to achieve the targeted string decreases as the population size increases. A more straightforward depiction is displayed by the following Figure 1.

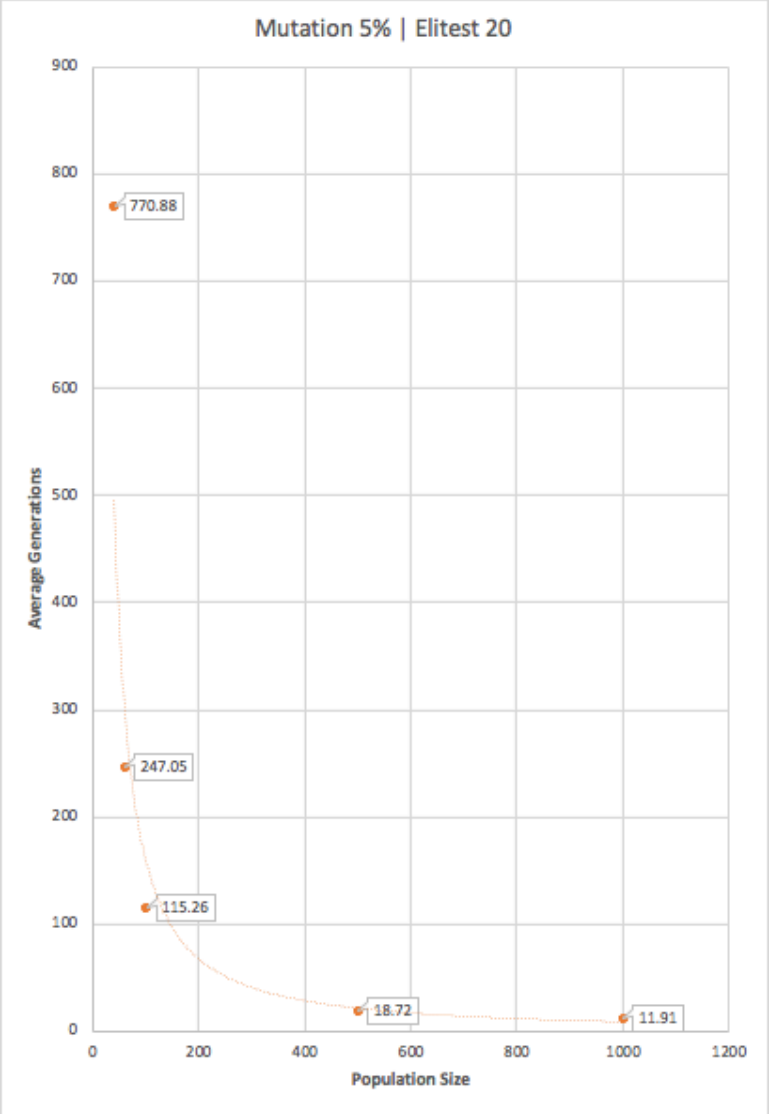

Figure 1: Population size plotted against average generations needed.

Again we observe a decrease in generations needed correlating to an increase in the population size. Furthermore, the trend line implies that an exponential relationship exists. There are two non-excluding explanations for this, of which one concerns the randomly created starting population and the other the crossover of individuals.

A higher starting population will most likely have more individuals with a high fitness value that are chosen to be the 20 elite reproducing themselves. For example, a trial with the population size of 100 might fill its first elite with three individuals with a fitness value of three a couple of two ranked individuals and the rest valued at one. Opposing, we have a trial with a population size of 1000, which can completely fill up its elite with individuals assigned a fitness value of three and occasionally some ranked even higher. This means the

first generation of a trial with a high population size starts closer to the solution than the first generation of a trial with a lower population size.

Furthermore, we can explain this observation by looking at the crossover of a larger populated trial. Since the elite are set to be the top 20 individuals in every case, each of the top 20 will reproduce itself a different amount of times depending on the population size. Take a trial with a population size of 100 and each elite will reproduce itself five times such that the following generation has a size of 100 individuals again. When comparing this to a trial with 1000 individuals, one has to note that each of the top 20 will reproduce itself 50 times to end up with 1000 individuals in the second generation. The higher rate of reproduction means that the elite will pass on more of their gene material, thus fewer characters that might be important get lost in the crossover process.

IV.III. Mutation Rate

Secondly, we will look at the effects of altering the mutation rate. After every single crossover between two individuals, there is a set chance that one random character in the new individual gets replaced by a random character of the alphabet or the space character. How high this chance is, is decided by the mutation rate. The mutation rate is used so that a population does not get stuck with certain material. Therefore, the independent variable here is the mutation rate. To generate conclusive dependent variables, the average generations needed, we fixed the other variables; 100 individuals as population size, 20 elitist individuals. The results are depicted in the graph below.

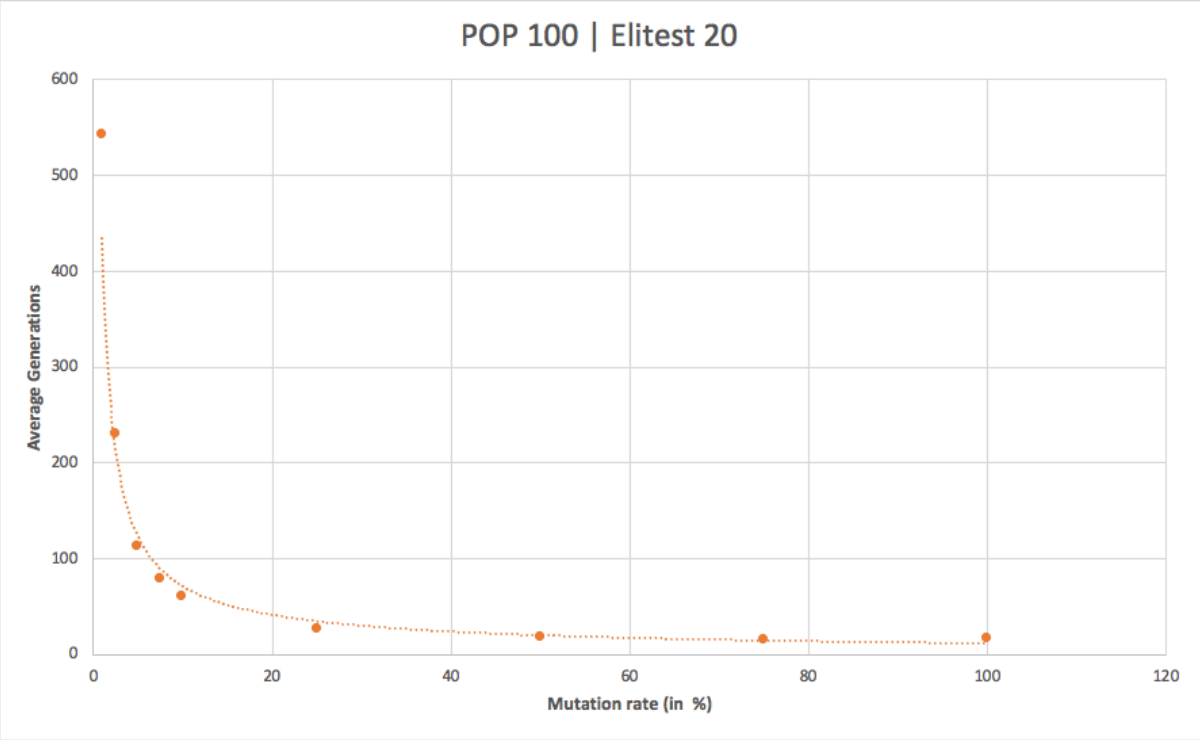

Figure 2: Mutation rate plotted against average generations needed.

Figure 2 also shows an exponential relationship between the two variables, where the average generations decrease the higher the mutation rate is. However, looking at the actual data shown in Table 2 below, we can observe that the average generations needed dip at 75 percent. This fall leads us to believe that the lowest average generation lies somewhere between 50 and 100 percent. An explanation for the ascending average generations from 75 percent to 100 percent could be that since the mutation rate is too high, characters that are valued correctly are replaced, disrupting the evolving string of characters.

Mutation rate (in %)	Average Generations
1	541,87
2,5	230,42
5	113,3
7,5	79,39
10	60,57
25	27,1
50	17,40
75	15,39
100	17,27

Table 2: Mutation rate compared to the average generations needed.

Nonetheless, since a too high mutation rate would defeat the purpose of reproduction, as mentioned in the method section we chose five percent as the final mutation rate for our algorithm.

IV. References

Thomas Geijtenbeek, Michiel van de Panne and A. Frank van der Stappen
"Flexible Muscle-Based Locomotion for Bipedal Creatures"
https://www.cs.ubc.ca/~van/papers/2013-TOG-MuscleBasedBipeds/2013-TOG-MuscleBasedBipeds.pdf

Samuel Arzt (2017)
"*Deep Learning Cars*"
https://arztsamuel.github.io/en/projects/unity/deepCars/deepCars.html