BioComputation Coursework

University of the West of England

Name: Juan Camilo Rodriguez
Student ID: 16020551

Module: BioComputation
Module Code: UFCFY3-15-3

Word Count: 3746

1. INTRODUCTION

This report focuses on finding global minimums and maximums of functions, also known as optimisation. Genetic Algorithms (GA) was the approach used to be able to solve such problem. There were many challenges when creating a GA, such as it accepts both binary and real-values, GA operations that can handle both types of data, and data manipulation throughout the program. Once a GA was achieved, the following tasks was to use different parameters for each function. Parameters such as population, generations, mutation rate, and crossover rate. The idea was to experiment with each variable to personally explore each function. This led to interesting observations that will be discussed later in the report.

2. BACKGROUND RESEARCH

Bajpai, P. and Kumar, M. (2010) defines **optimisation** as "*finding decisions that satisfy given constraints, and meet a specific goal at its optimal value".* This is a task that is faced many times throughout a daily basis. However, finding the most optimum solution to problems is not always feasible, especially when time is the factor (Hartmann, A.K. and Rieger, H., 2002). Due to this impracticality, many disciplines have developed ways in which to get almost optimum solutions.

That being said, when having to solve an optimisation problem, there are factors that needs consideration which help identify the optimisation type or technique to use. Here are the considerations from Stanford (2017) and Wisconsin (2019) Universities:

- **Constrained vs unconstrained –** This is when variables are restricted to a range
- **Discrete vs Continuous –**Discrete variables means that the values can only be certain values (example: a set of integers). While Continuous data refers to data that can be of any value (example: weight, temperature, real numbers)
- **Static vs dynamic –** This refers to either the data is static or changing over time.
- **Deterministic vs Stochastic –** Stochastic input is at random. While with deterministic the values given are known.

Genetic Algorithms are a branch of Evolutionary Algorithms (EA) (Weise, T., 2011) that attempt to get as near as possible to optimum solutions. Bajpai, P. and Kumar, M. (2010) define **Genetic Algorithms** as "*adaptive heuristic search algorithm premised on the Darwin's evolutionary ideas of natural selection and genetic*". In Chapter 3 goes more in-depth about GA's

The world is filled with many variables that can act in all sorts of ways. Therefore, EA's are used for this purpose. For instance, there has been research conducted using Genetic Algorithms in order to investigate the famous disappearance of Malaysian flight MH370. The way in which GA was used for this paper was to be able to predict the likelihood of where the plane crashed. Variables used for this problem include: wave speed, direction, depth, and more. The GA has two ways in which it can be used. First, if a debris is found then could backtrack the GA solution that could give a better indication of where the plane crashed. Second, if there was an unidentified object in the ocean, then the GA could predict if such object is likely to have belong from the plane or if it belonged to another source (Marghany, M. *et al*, 2016).

3. EXPERIMENTATION

3.1. Genetic Algorithm Implementation

As mentioned in the GA definition, this algorithm is inspired by evolution process. Therefore, the stages that it undertakes are also based upon evolutionary aspects. Here are some concepts and the stages for GA development:

Genes, Chromosomes & Representations

As suggested by the name of the technique, genes are a feature used. Genes can be encoded in many ways, ranging from letters, binary and/or real numbers. Each *gene* represents a **variable** of the problem and the collection/list of these genes are the **solution** for the problem, also known as *chromosome* (Bajpai, P. and Kumar, M., 2010).

Examples of both Binary and real-valued chromosomes:

$$\boldsymbol{Binary\ Chromosome\ Encoding}:$$
$$[\,0,0,1,1,0,1,0,0\,]$$

$$\boldsymbol{Real\ Value\ Chromosome\ Encoding}:$$
$$[2.12,-1.11,0.32,-2.12,0.12]$$

Initial Population

Typically, a random population of chromosomes are generated. Normally, would want a large set of solutions as that would promote diversity in the population, which avoids being stagnated in some variables or local maximums/minimums (Bajpai, P. and Kumar, M., 2010).

Fitness Function

The chromosome is then fed to the fitness function to calculate the fitness of an induvial based upon the circumstances and rules. This will state how good of a solution is for a given function.

Example of fitness function:

$$f(x) \rightarrow x\text{^}2$$

$$Binary\ Encoding\ Chromosome: \quad [0,0,1,1,0,1,0,0]$$
$$Binary\ to\ actual\ number: \quad 44$$
$$Fitness\ for\ function: \quad 1936$$

Parent Selection

This selection is in charge of selecting individuals from the population to further "evolve" them. Typically, it is desired to have to some degree fit individuals in the parent pool. Parent selection is not about selecting the most fit or else the population will be narrowed very quickly to a particular chromosome, also known as **premature convergence** (Abuiziah, I. and Shakareneh, N., 2013). Therefore, it creates a random selection, but these selection methods have a way so that individuals that are fitter have an advantage of being picked over the rest.

What this step achieves to accomplish is to narrow the search scope of the population towards the fitter individuals, while at the same time keeping diversity present to avoid narrowing the scope search too quickly.

The Parent Selection technique used throughout this coursework was *Tournament Selection.* Which involved selecting k number of individuals from the population, typically between 3 -5. From this group the individual that had the highest/lowest fitness

would be selected as a parent. Tournament selection is done a certain number of times on the population, but candidates that are selected from the population to be parents are not reconsidered in the generation.

Crossover

Since the above process has a good chance of obtaining above average fit individuals. Then crossover role here is to exploit this chance by using two candidates to produce a better result. This is accomplished typically by exchanging a select number of genes from the two parents. What this accomplishes is to get an offspring that is different but not completely different to its parents.

This stage is in hope that two fit individuals are chosen and that they are able to produce an even fitter offspring than the parents (Abuiziah, I. and Shakareneh, N., 2013).

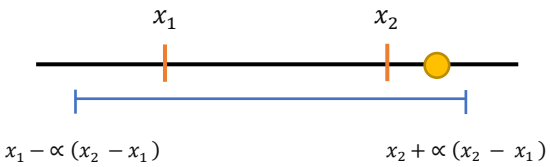
The Crossover technique that was used for binary encoding was *One Point Crossover.* What this method does is grab two individuals from the parent pool and select a random point to split the parents' chromosome. The tail part of the chromosome split is then swapped between the two.

For real-value encoding, *Blend Crossover* was used. There are two parents, x_1 and x_2, a random number is selected to determine what gene will be used from both parents. The likelihood of the two genes being the same is very unlikely, therefore, these two number are used to create a range, with some range outwards as well. The range is set by using the following equations:

$$Range: [\, x_1 - \propto (x_2 - x_1)\ ,\ x_2 + \propto (x_2 - x_1)\,]$$

Where $\propto$ is normally 0.5

Equations visually would represent this:



The randomly selected number then replaces the genes that were taken from both parents.

Mutation

There are two problems that emerge with Parent Selection and Crossover:

1) These two operations start narrowing the search space, which can lead to being stuck in a local minimum/maximum.

2) There is an off chance, as the GA goes through many generations that it is near the most optimum solution, however, none of the prior parents had a gene that would allow it to evolve to the most optimum solution.

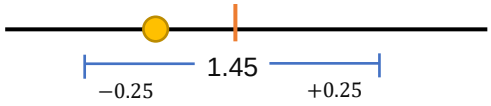
Therefore, the role of mutation is to create small alterations to individuals' chromosomes, which could eventually lead to it discovering a new gene, which would deliver an even better solution.

The Mutation technique that was used for binary encoding was *Bit Flip Mutation.* Individuals gene is flipped, meaning that if the gene value is 0 then it is flipped to 1, and vice versa.

The Mutation technique for real-value encoding chromosomes is different. *Random Mutation* was used, which means that once again a random gene of the chromosome is chosen. There is a set value, example: ± 0.25, and from the current value that range is applied to both sides. This sets a range,

and a random number within the range is selected and replaces the gene value.

Illustrative example as to how this technique works:



Survivor Selection

This is the final stage of the GA, at this point there are new solutions that are hoped to bring better and more fitter individuals to the population and following generations.

The new solutions typically replaces individuals from the population. They can either replace them based on age in the population, meaning that the older a solution has been in the population then the more likely they are to be replaced. Another way to replace is based on fitness, meaning that candidates from the population that have a low fitness based on what is being search are then replaced with the new solutions. The latter technique was used for this development

Generally speaking, Genetic Algorithm operators are in essence about “selection” and “variation”. *Selection* being pathing the algorithm as to where it seeks to reach. *Variation* is about introducing similar but not completely different values to the pool of individuals, which helps find alternative/possible solutions (Eiben, A.E. and Smith, J.E., 2016).

Parameters

Furthermore, the GA can also be altered greatly with what is known as parameters. Examples of parameters in Genetic Algorithms are the number of individuals in the population, the length of a chromosome, or even the mutation/crossover rate. All of these are factors that contribute to how to the Genetic Algorithm performs.

In many occasions once completed the initial GA, there is for sure more room for improvement. Much of it comes from tuning the parameters as the problem can be heavily be affected by such values. Especially when it comes to population, mutation rate and crossover rate

3.1. Function 1

The function to solve was:

$$\boldsymbol{f(x)} \rightarrow x^2$$

$$with, 0 \leq x \leq 255$$

The task was to find the **global maximum** using **binary encoding**.

Taking this into consideration the way to represent x’s large domain was by using the binary encoding list index number and applying the following equation when there was a 1 present in given index:

$$2^{index\_num}$$

Example as to how this was implemented:

$$[\,0, 0, 1, 1, 0, 1, 0, 0\,]$$

$$0 + 0 + 2^2 + 2^3 + 0 + 2^5 + 0 + 0 = 44$$

Followed by the fitness function would give:

$$44^2 = 1936$$

After the initial population and fitness calculation, the individuals would go through Tournament Selection, One Point Crossover, Bit Flip Mutation and Fitness Based Selection.

Parameter Tuning

General Results

Here are some results obtained and some observations made by using different parameters to solve the problem

Parameters for following tests:

Population: 100 individuals
Parent Selection: 25 individuals
Crossover Rate: 50%
Mutation Rate: 50%
Survivor Selection: 15 individuals

It is believed that these parameters are average and fair. These will be used unless stated otherwise.

The following three diagrams are 10 random attempts using the “fair” parameters:

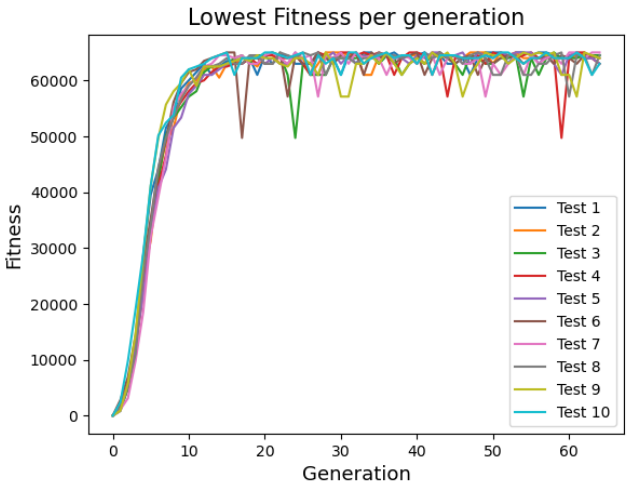


Figure 1 - The lowest fitness for each generation throughout the whole run- for all 10 attempts

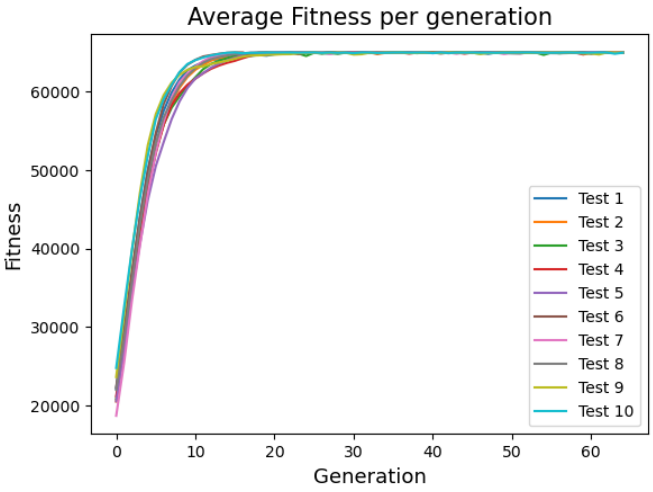


Figure 2 - The average population fitness for each generation throughout the whole run - for all 10 attempts

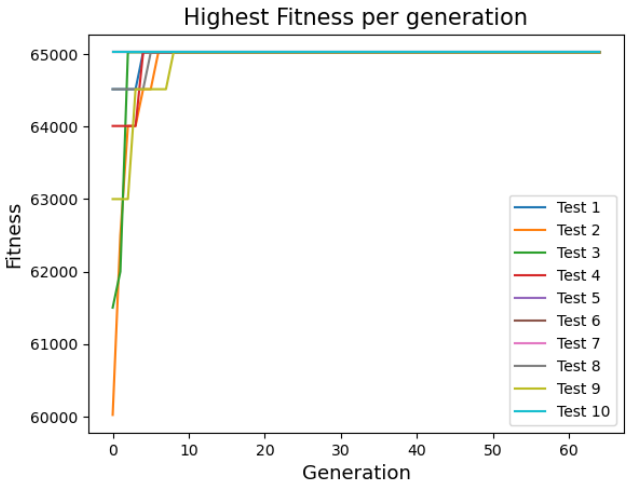


Figure 3 -The highest fitness for each generation throughout the whole run - for all 10 attempts

General observations:

- Figure 1 shows the most inconsistent results between figures 1-3. Regardless its test does follow a trend of improvement. With the occasional drop in value, most likely due to mutation.
- Figure 2 seems to have very consistent results throughout all attempts. It very quickly gets to the optimum results.
- Figure 3 seems shows that all attempts have at least a value that reaches the optimum solution at a very rapid pace.

Population Results

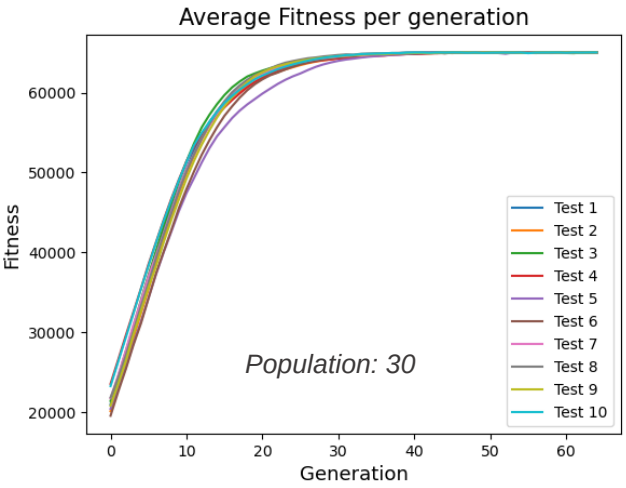


Figure 4 - High Population: Average Fitness

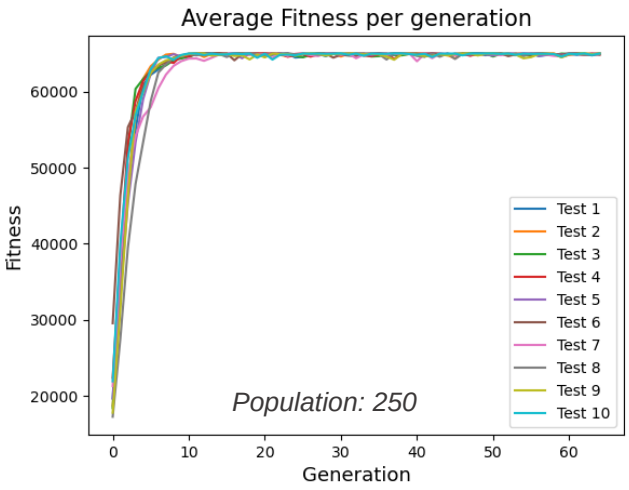


Figure 5 – Low Population: Average Fitness

Observations from Figures 4 and 5:

- With lower population seems to be quicker to get near the optimum solution. However, this is mainly due to the low count of individuals and it not being too difficult reaching the optimum solution, as it is an exponential curve.
- With higher population climbing towards the optimum solution is much slower. However, the results are more consistent across all attempts, which is represented by the smoothness of the line.

Crossover Results

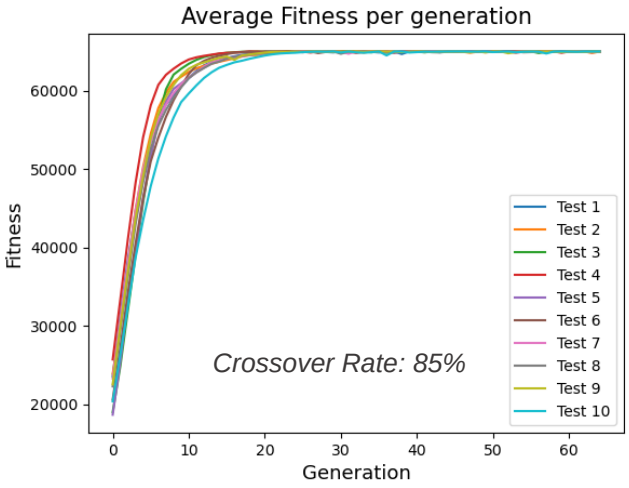


Figure 6 - High Crossover Rate: Average Fitness

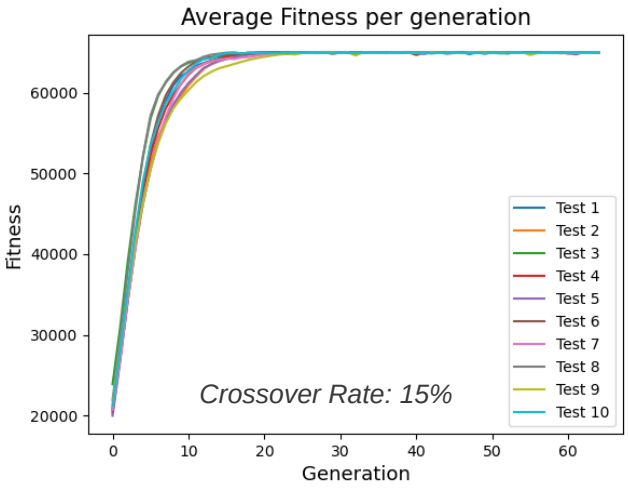


Figure 7 – Low Crossover Rate: Average Fitness

Observations from Figures 6 and 7:

- The general observation is that the crossover rate is not as impactful to this particular function. However, the low Crossover Rate does seem to deliver slightly better solutions. Again, this is probably since the function is not too difficult to solve and therefore it is easy with a few count of individuals to eventually get to the result without much help from the GA operations.

Mutation Results

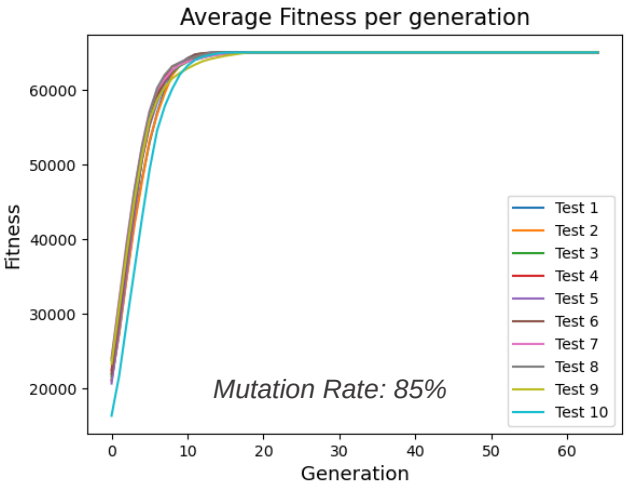


Figure 8 – High Mutation Rate: Average Fitness

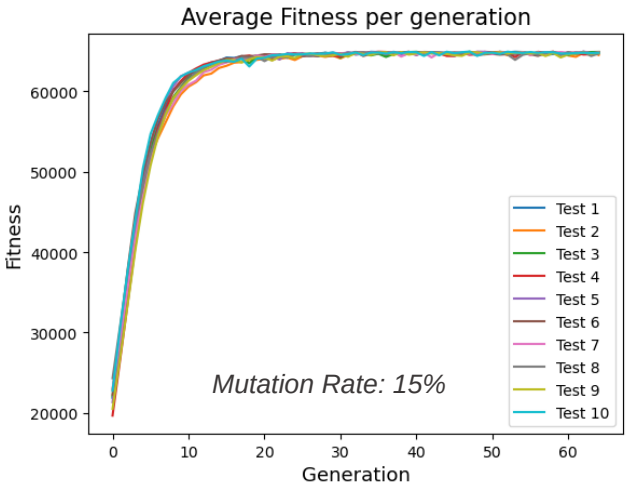


Figure 9 – Low Mutation Rate: Average Fitness

Observations from Figures 8 and 9:

- Mutation rate seems to even matter less than the crossover rate. This is mainly due to the Bit flip mutation technique being used, it is minimal compared to other mutation methods. While the operation just changes a gene, crossover can change up to 7 genes.
- However, if there is some sort of difference then high mutation rate delivers smoother results. This is not either good or bad, simply an observation.

3.2. Function 2

The function to solve was:

$$\boldsymbol{f(x,y)} \rightarrow 0.26 * (x^2 + y^2) - 0.48 * x * y$$

$$with, -15 \leq x, y \leq 15$$

The task was to find the **global minimum** using **binary encoding**.

The representation for this function was different, although it did follow to some extent the same principles from the prior function. Instead of using 8 genes that would add to 255, here instead was used 4 genes to add to 15 and an additional gene to determine whether positive or negative sign.

Example:

$$[\,0, 0, 1, 1, 1]$$

$0 + 0 + 2^2 + 2^3$ and (negative sign) = - 12

This function had practically the same run as the prior one: initial population and fitness calculation, the individuals would go through Tournament Selection, One Point Crossover, Bit Flip Mutation and Fitness Based Selection.

However, this time the Tournament Selection and Survivor Selection were grabbing values with the least fitness as that would scope the individuals towards the global minimum, which was the task set for this problem.

General Results

Here are some results obtained and some observations made by using different parameters to solve the problem

Parameters for following tests:

Population: 100 individuals
Parent Selection: 25 individuals
Crossover Rate: 50%
Mutation Rate: 50%
Survivor Selection: 15 individuals

It is believed that these parameters are fair. These will be used unless stated otherwise.

The following three diagrams are 10 random attempts using the “fair” parameters:

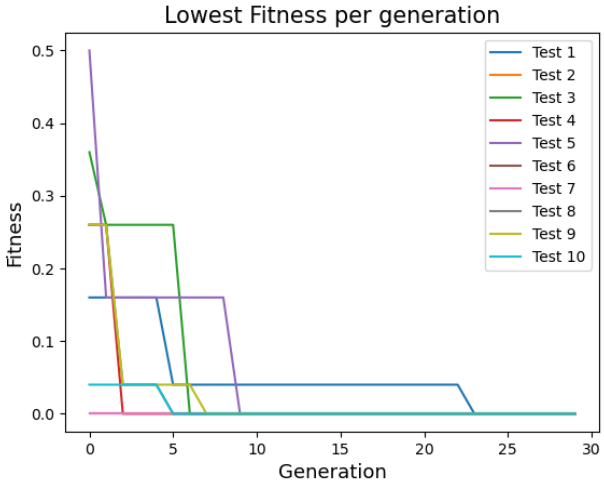


Figure 10 - The lowest fitness for each generation throughout the whole run- for all 10 attempts

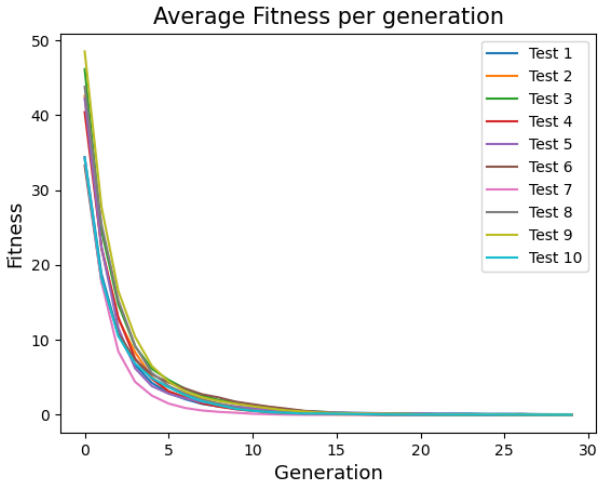


Figure 11 – The average fitness for each generation throughout the whole run- for all 10 attempts

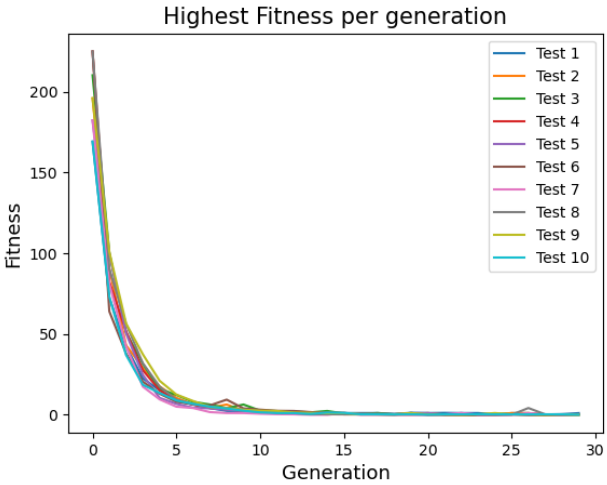


Figure 12 - The highest fitness for each generation throughout the whole run- for all 10 attempts

General observations:

- Between Figures 10, 11 and 12 all of them in the later generations seem to be able to find the most optimised minimum of the function. This function has an even quicker development than function 1.
- Figure 10 finds the minimum in most attempts very quickly, and none of them at the very end of their runs have different results.

- Figure 11 average attempts starts off decently dispersed but over time all the attempts get gradually closer.
- Figure 12 has quite similar results to what is seen in figure 11, once again the GA for this problem seems to work very well with finding the global minimum.

Population Results

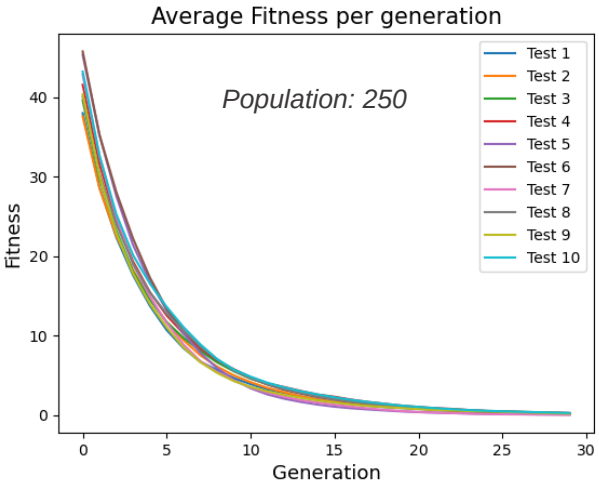


Figure 13 - High Population: Average Fitness

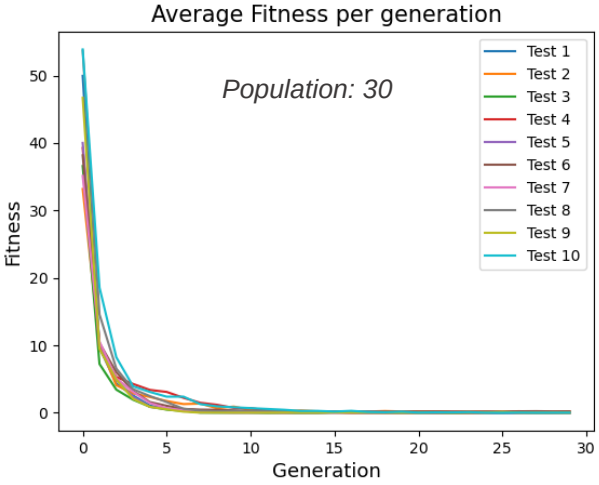


Figure 14 – Low Population: Average Fitness

Observations from Figures 13 and 14:

- For this function it seems that population has to some extent of an impact towards the function. Works similar to function 1, in the sense that with less individuals the minimum is found quicker but with more individuals the trajectory between attempts is much smoother and consistent.

Crossover Results

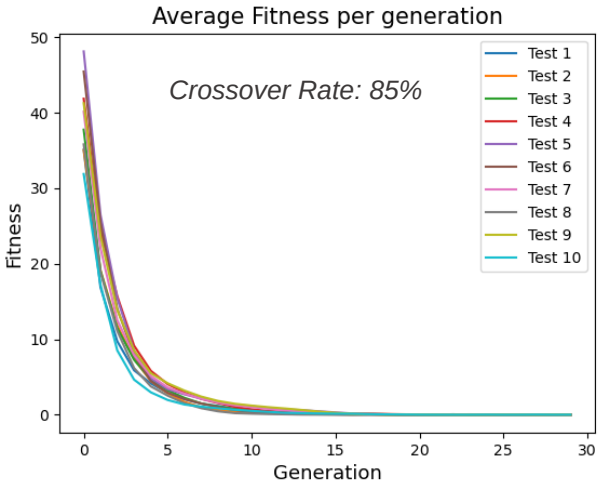


Figure 15 – High Crossover: Average Fitness

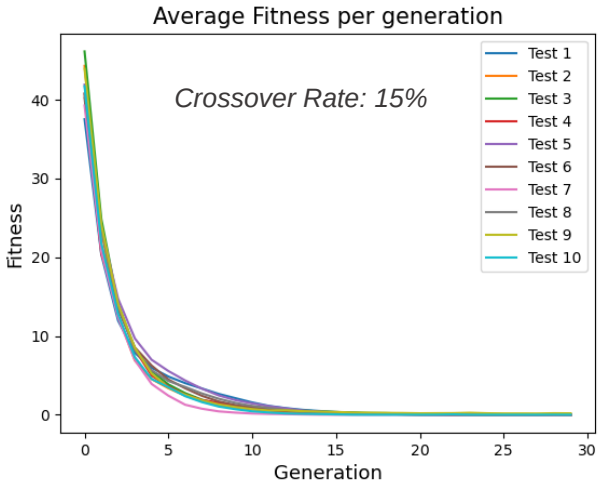


Figure 16 – Low Crossover: Average Fitness

Observations from Figures 15 and 16:

- Higher crossover for this function seems more beneficial. The results seem to work better as they get over time closer to the minimum
- Lower crossover seems that the GA struggles as it gets closer to the global

minimum. Perhaps this is because the GA needs more diversion but isn't getting lucky

Mutation Results

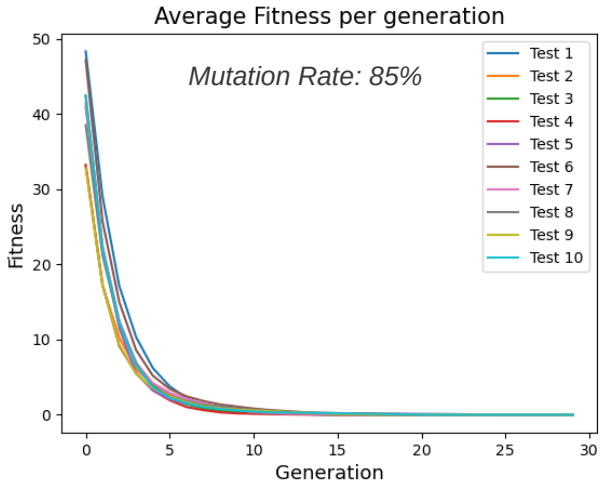


Figure 17 – High Mutation Rate: Average Fitness

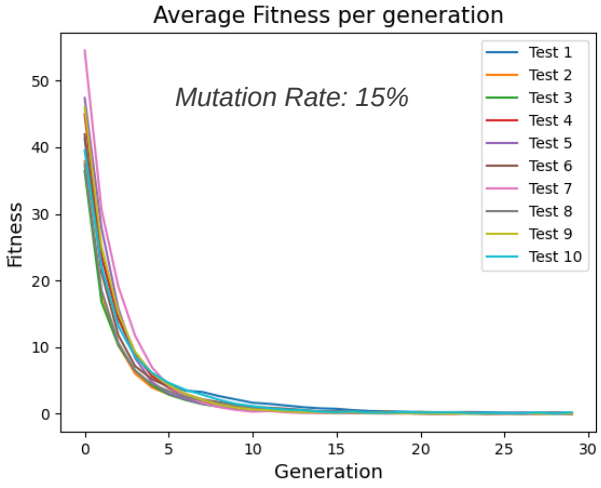


Figure 18 – Low Mutation Rate: Average Fitness

Observations from Figures 17 and 18:

- The results obtained from both figures are close and practically no difference whatsoever. Therefore, mutation rate for this particular function has null impact towards the solutions. It is believed that this is because naturally this function finds the minimum quickly, therefore, the mutation operation is not needed in this case.

3.3. Function 3

The function to solver was:

$$\boldsymbol{f(x)} \rightarrow 10n + \sum_{i=1}^{n} x_i - 10 * \cos(2\pi * x_i)$$

$$with, -5.12 \leq \boldsymbol{x_i} \leq 5.12$$

$$n = 10, 20$$

The task was to find the **global minimum** using **real-value encoding**.

It is better through an example explain as to how the list of real values worked to get the fitness function.

Example:

$$Individual\ Chromosome\colon\ [3.12, -1.12, 5.03]$$

$$n\colon 3$$

$$\rightarrow 10(3) + [\ 3.12 - 10 * \cos(2\pi * 3.12)\ ] +$$

$$[\ (-1.12) - 10 * \cos(2\pi * (-1.12)\ ] +$$

$$[\ 5.03 - 10 * \cos(2\pi * 5.03\ ]$$

The way in which both crossover and mutation worked for real-value encoding were covered in the early part of this chapter

General Results

Here are some results obtained and some observations made by using different parameters to solve the problem

Parameters for following tests:

Population: 100 individuals

Parent Selection: 25 individuals

Crossover Rate: 50%

ALPHA: 0.25

Mutation Rate: 50%

Mutation Range: ±0.5

Survivor Selection: 15 individuals

It is believed that these parameters are fair. These will be used unless stated otherwise.

The following three diagrams are 10 random attempts using the “fair” parameters:

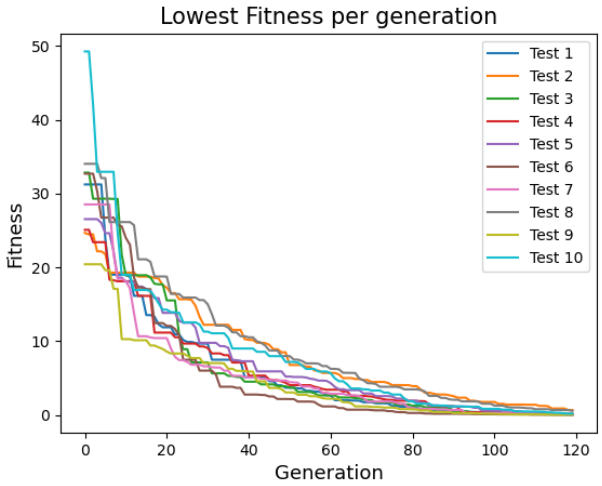


Figure 19 - The lowest fitness for each generation throughout the whole run- for all 10 attempts

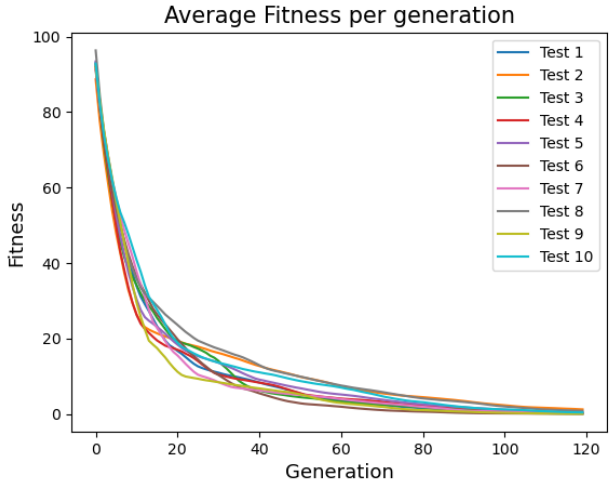


Figure 20 - The average fitness for each generation throughout the whole run- for all 10 attempts

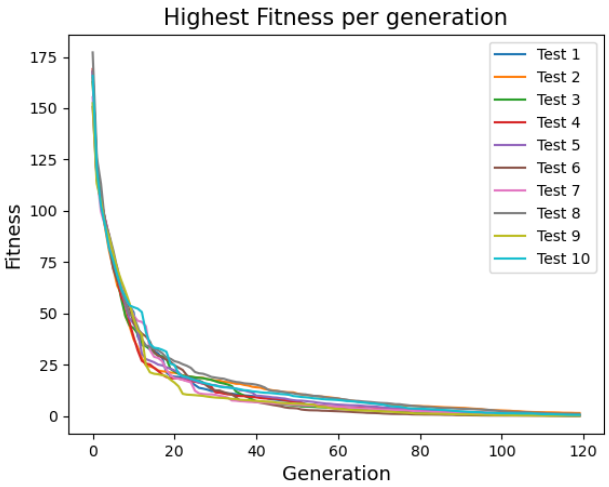


Figure 21 - The highest fitness for each generation throughout the whole run- for all 10 attempts

General observations:

- Figure 19, it is awfully hard to predict as to how the lowest value of each test performs. At around 10-15 generations all attempts seem to have a sudden similar drop in value. But after there is no logical pattern. It is believed that it is simply the GA just trying to find the minimum value, it is struggling as the function is more complex to solve, hence the odd values.
- Figure 20 and 21 have a similar trend in the sense that the data commences cluttered together and then spreads out temporarily before coming back together to the optimal solution. It is believed that whatever is occurring in figure 19 is the reasoning as to why figure 20 and 21 results disperse near the end. Regardless, the GA in the end does find the global minimum for the function in all attempts.

Population Results

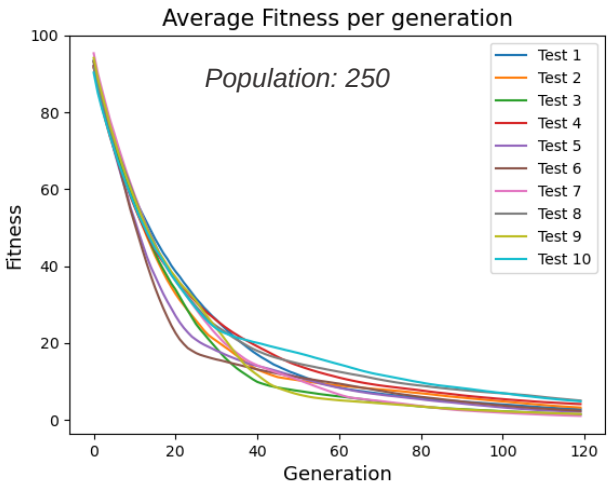


Figure 22 - High Population: Average Fitness

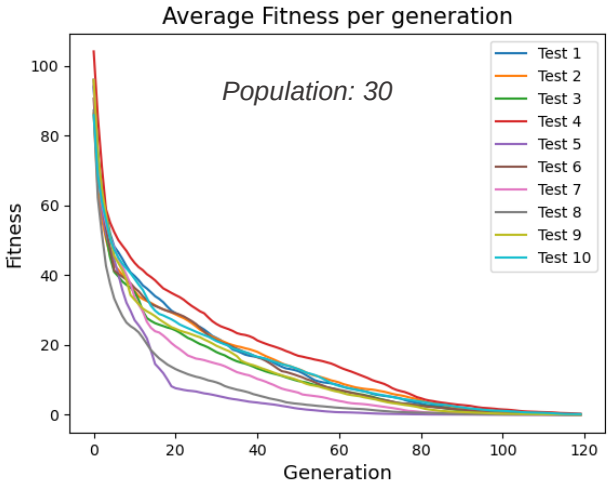


Figure 23 - High Population: Average Fitness

Observations from Figures 22 and 23:

- Population seems to have a great impact for this function. The reason as to why population can be of this major factor is because the GA has such a large area to search compared to the other 2 functions. As well as having a rougher search area to explore. Therefore, by having a smaller population it takes larger time to collect good results.

Crossover Results

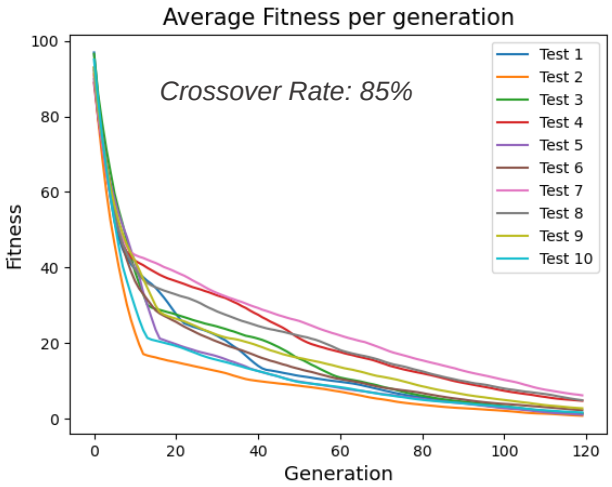


Figure 24 - High Crossover: Average Fitness

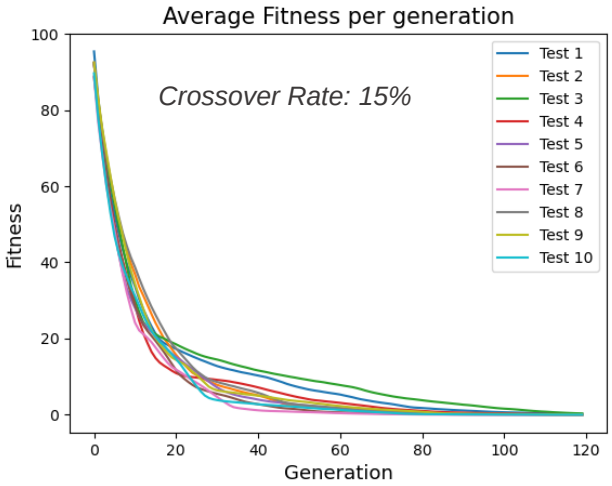


Figure 25 - Low Crossover: Average Fitness

Observations from Figures 24 and 25:

- For this function crossover doesn't seem to help the development of the chromosomes, as seen in figure 24. It has a harming effect rather than any beneficial.
- Therefore, for this function it would rather be better to use a lower crossover rate as that seems to deliver more consistent results.

Mutation Results

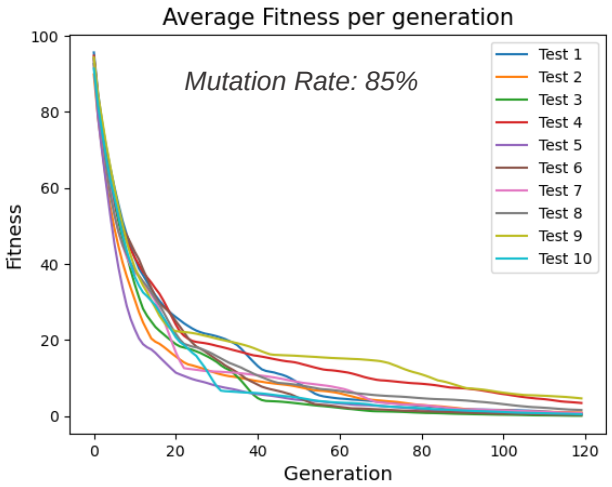


Figure 26 - High Mutation: Average Fitness

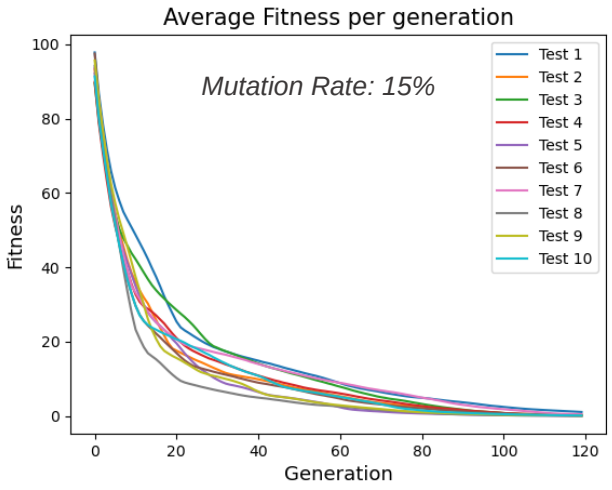


Figure 27 - Low Mutation: Average Fitness

Observations from Figures 26 and 27:

- Similarly, to the crossover operator. Higher mutation for the function seems to be worse to use than an actual benefit. Although the effect also shows to be more random, which is something to be expect from mutation
- Again, recommend using a smaller rate for such function. Regardless of its impact it still needs crossover and mutation as that will still help the function search for better values.

4. CONCLUSIONS

This report explored how Genetic Algorithms work through implementing a GA and by testing it with different functions/problems. It is reflected that there is a lot of power behind this technique. The fact that in all cases, no matter the tuning that was being tested the algorithm generally speaking performed very well by reaching to near optimal solutions.

The results from function 1 and 2 were not particularly too different as to what was being expected for the GA to perform. However, for function 3, it was interesting in particular to witness how the GA responded to crossover and mutation. This is because the knowledge gathered beforehand made it believe that these two operators are meant to be a beneficial factor towards the problem solving. But this was not the case and creates the thought as to how real-world problems would be affected by a GA, especially since those problems have more radical inconsistencies. Meaning that parameterisation would probably be more of a factor and would require a deeper analysis to get reliable results.

Further studies could include further analysis to the results that were obtained. There is still a large number of images that were saved, both low and high fitness for each operator and parameterisation examined. This could potentially reveal further explanation as to how some of these functions work the way they do. Additionally, could also try changing the parameters of other features, such as quantity for parent selection and survivor selection. Finally, another idea to further analyse the current application is to implement other types of operator techniques for parent selection, crossover, and mutation.

REFERENCES

Abuiziah, I. and Shakareneh, N. (2013) A Review of Genetic Algorithm Optimization: Operations and Applications to Water Pipeline Systems. International Journal of Physical, Natural Science and Engineering [online]. 7 (1), pp. 341-347. [Accessed 20 July 2020].

Bajpai, P. and Kumar, M. (2010) Genetic Algorithm – an Approach to Solve Global Optimization Problems. Indian Journal of Computer Science and Engineering [online]. 1 (3), pp. 199-206. [Accessed 20 July 2020].

Eiben, A.E. and Smith, J.E. (2016) Introduction to Evolutionary Computing. 2nd ed. Berlin, Germany: Springer.

Hartmann, A.K. and Rieger, H. (2002) Optimization Algorithms in Physics. 1st ed. Berlin, Germany: Wiley-vch Verlag Berlin Gmbh.

Marghany, M., Mansor, S. and Shariff, A.R.B.M. (2016) Genetic Algorithm For Investigating Flight Mh370 in Indian Ocean Using Remotely Sensed Data. Iop Conference Series: Earth and Environmental Science [online]. 37 (1), pp. 1-7. [Accessed 24 July 2020].

University of Wisconsin (2019) Types of Optimization Problems. Available from: https://neos-guide.org/optimization-tree [Accessed 24 July 2020]

Stanford University (2017) Introduction to Mathematical Optimization. Math [online] Available from: https://web.stanford.edu/group/sisl/k12/optimization/MO-unit1-pdfs/1.1optimization.pdf [Accessed: 24 July 2020]

Verma, R. and Lakshminarayanan, P.A. (2005) A Case Study on the Application of a Genetic Algorithm For Optimization of Engine Parameters. Proceedings of the Institution of Mechanical Engineers, Part D: Journal of Automobile Engineering [online]. 220 (4), pp. 471-479. [Accessed 22 July 2020].

Weise, T., Zapf, M., Chiong, R. and Nebro, (2009) Why Is Optimisation Difficult?. Nature-inspired Algorithms For Optimisation [online]. 193, pp. 1-50. [Accessed 19 July 2020].

Weise, T. (2011) Global Optimisation Algorithms - Theory and Application. 3rd ed. : .

APPENDIX A - SOURCE CODE & RESULTS

The source code that was implemented and that is referenced throughout the report can be found by using the following link:

https://github.com/JCR21598/BioComputation---Solving-Problems-using-GA

In the zip file, can find **all** the results for the different functions and the different parameterisation. Many of them were not covered and considered due to the already length of the report. Regardless they are left in the directories if the reader would want to go through them.