Final Year Project Report

**Interim Submission – Final Report**

Genetic Algorithm

Hameed Roleola

A report submitted in part fulfilment of the degree of

**MSci (Hons) in Computer Science**

**Supervisor:** Dr Eduard Eiben



Department of Computer Science

Royal Holloway, University of London

November 27, 2022

**Declaration**

This report has been prepared on the basis of my own work. Where other published and unpublished source materials have been used, these have been acknowledged.

Word Count:

Student Name: Hameed Roleola

Date of Submission:

Signature: Hameed Roleola

Table of Contents

Contents

[Abstract 4](#_Toc120496741)

[Chapter 1: Introduction 5](#_Toc120496742)

[1.1 Aims and objectives 5](#_Toc120496743)

[1.2 Genetic Algorithms 6](#_Toc120496744)

[1.2.1 Constraint Satisfaction Problems: 7](#_Toc120496745)

[1.3 Encoding 8](#_Toc120496746)

[1.4 Theory of coalescence 10](#_Toc120496747)

[Chapter 2: Genetics Overview 12](#_Toc120496748)

[1.5 Genetic Drift 12](#_Toc120496749)

[Chapter 2: Design patterns 14](#_Toc120496750)

[2.1 MVC Design Pattern 14](#_Toc120496751)

[Chapter 3: Technical Aspect 15](#_Toc120496752)

[3.1 Technical Decisions 15](#_Toc120496753)

[3.2 Planning 15](#_Toc120496754)

[3.3 Software development 15](#_Toc120496755)

[3.4 Software Engineering 15](#_Toc120496756)

[3.5 Proof of concept 15](#_Toc120496757)

[3.6 Next term aims 15](#_Toc120496758)

[Chapter 4: Project Plan 17](#_Toc120496759)

[Chapter 5: Summary of project diary 18](#_Toc120496760)

[Glossary 19](#_Toc120496761)

[Bibliography 20](#_Toc120496762)

Abstract

***One of the most popular evolutionary computation techniques discovered is Genetic algorithms. A Genetic Algorithm is a type of metaheuristic slightly inspired by the process of natural selection coined by Charles Darwin. Genetic Algorithms are employed to develop high-quality solutions to optimization and search problems. This report sets out to find out how Genetic Algorithms can be used to solve constraint optimisation problems and thus help to solve real-world problems. The main focus of this interim report was on one optimization problem known as the Travelling Salesman Problem.***

Chapter 1: Introduction

## Aims and objectives

The Genetic algorithm based on Charles Darwin’s theory of natural selection, the mechanism that propels biological evolution, is a technique used for solving constrained and unconstrained optimization problems. In more simplistic terms it uses a series of evolutionary processes to solve either with or without given parameters set i.e., constraints. Genetic algorithms contribute to help find solutions to difficult problems and the application of real-world problems.

The aim of the project is to use develop an understanding of genetic algorithms and explore how they can be used to solve real world problems. In particular this project will be focusing on the use of genetic algorithms with constraint optimisation. For the first term I will be working specifically on one problem in this case being the Travelling Salesman Problem (TSP). The TSP is a problem is an example of a constraint satisfaction problem used for finding the shortest and most efficient route for a person to take within a given list of destinations. This problem is an example of how constraint optimisation has real world applications. Real world applications of this include the optimization of travel, vehicle routing and astronomy to help determine the movement of a telescope for the shortest distance between different stars.

So, the idea I will be working on in my project is ‘*The use and application of genetic algorithms in constraint optimization problems’*. The project will involve deep knowledge of genetic algorithms, their use in constrained problems, and their practical applications. I chose my project to be in GAs because of their application to real-world problems.

Real world problems can vary based on complexity and the exhaustiveness of recourses. An example of this is calculating the survival rate of different organisms, this seemingly tedious task can be handled more efficient with less costs using a genetic algorithm simulation.

For the interim report I decided to begin my focus on one constraint optimisation problem this being the ‘Travelling salesman problem’ formulated by William Rowan Hamilton. With this I will learning the basics of genetic algorithms and how they can be applied to constraint optimisation. I will be attempting to implement a genetic algorithm for the TSP with research to strengthen my understanding and possibly a visualisation for this problem. I intend to code this using python and other libraries included for example Tkinter. I also aim to write a report gaining insight into genetic algorithms and providing knowledge on this.

With prior research it can truly be stated the project may prove to be very demanding but with a good use of time and research I believe this will be helpful. Key concepts on the theory of evolution will be explored like genetic drift, mutations, fitness functions and crossover functions. In the end, I plan to gain comprehensive background knowledge from the project with regards to how evolutionary driven technology can be used to help solve real-world problems.

## Genetic Algorithms

**Briefly describe Genetic algorithms and define any terms used in the subject (also put these in the Appendix as a Glossary)**

Charles Darwin’s theory of natural selection served as the fundamental foundation for evolutionary algorithms and search heuristics namely the genetic algorithm.

The genetic algorithm (GA), proposed by John Holland in 1975 [R] is a method that utilises natural selection, the mechanism that propels biological evolution, for resolving both constrained and unconstrained optimization problems. A population of unique solutions otherwise known as ‘chromosomes’ is repeatedly modified by the genetic algorithm. The genetic algorithm chooses members of the present population to serve as parents at each stage and utilises them to produce the offspring that will make up the following generation. The population "evolves" toward the best option over the course of subsequent generations.

The genetic algorithm can be used to tackle several optimization problems, including those where the objective function is discontinuous, nondifferentiable, stochastic, or highly nonlinear and are not well suited for typical optimization algorithms. When some components must only have integer values, mixed integer programming problems can be solved using the evolutionary algorithm.

Three fundamental rules—selection, crossover, and mutation—are used by a genetic algorithm to produce the next generation from the current population:

* **Selection:** chooses the parents, who will contribute to the population of the following generation.
* **Crossover**: combine two parents to create the next generation's offspring.
* **Mutation:** subjects each parent to random modifications.

Diagram

Description automatically generated

Figure 1: Flowchart of genetic algorithm

Here is an example of a simple Genetic algorithm referenced from ‘*An Introduction to Genetic Algorithms by Mitchell*’ –

1. Start with a randomly generated population
2. Calculate the fitness of each chromosome in the population
3. Repeat the previous steps until an offspring has been created
   1. Select a pair of chromosomes from the current population
   2. Cross over the pair at a randomly chosen point to form two offspring
   3. Mutate the two offspring at each locus
4. Replace the current population with the new population
5. Go to step 2

### Constraint Satisfaction Problems:

The main focus of this project regarding genetic algorithms will be constraint satisfaction problems.

Searching can often be executed more easily in cases where the solution rather than corresponding to an optimal path, the only requirement is to satisfy a set of defined conditions. These problems are known as Constraint Satisfaction Problems (CSP). For example, in the n-queen problem (chess-based problem) the set condition is that no two queens in the same column, row or diagonal can attack each other. If much more than this was required, it would then become a more general problem rather than a CSP.

**Problem definition**

A CSP consists of:

* A set if variables X = {x1, …, xn};
* Each variable xi, a finite set Di of possible values (Domain) and
* Set of constraints restricting values that the variables can take

The following form can be used to define a constraint satisfaction problem (CSP) in a (finite domain). Find values for the variables that satisfy each constraint given a collection of variables, a finite set of possible values for each variable, and a list of constraints. An example of this occurs in production scheduling. To ensure that each work is finished by the specified deadline, jobs must be processed on machines that can only handle one job at a time. Additional examples follow from the notion that an optimization problem can be stated as a series of CSPs. The solution to a CSP includes consistent and complete assignment. Where a consistent assignment dictates that an assignment does not violate any constraints and a complete assignment is where every variable is assigned.

## Encoding

Genetic algorithms are useful for solving a variety of problems that are classified as NP-complete and NP-hard. The overall forms if these types of algorithms are described in different places such as “Adaptation in Natural and Artificial Systems” by J Holland referenced below.

For the implementation of GA there are important design criteria needed to be considered, one of these is known as encoding:

How a given solution is represented depends on the encoding. A gene is any one of the many components that make up an individual. This representation—hence the gene—can be fixed, binary, real, or any other meaningful data structure. The representation used depends on the case.

Encoding is frequently the most difficult component of using genetic algorithms to solve a problem. The typical approach to express a solution in genetic algorithms is as a string of zeros and ones; this is the most frequent encoding method, as the initial genetic algorithm research employed this style of encoding.

Binary coding/ mapping

An example of a common encoding used for genetic algorithms is binary encoding when a gene is represented in terms of bits 0s and 1s.

Fitness Function

Fitness functions are used to determine how well solutions (such chromosomes) perform. The idea that fitness functions are unique to a particular issue is a key characteristic.

In this case when doing the travelling sales problem, I considered using two fitness functions one from ‘Eric Stoltz’ and the other from ‘YJ Park’.

Text

Description automatically generated

After researching I found that there were already suitable fitness functions created and it would be difficult and a waste of time to implement a completely different one as a result, I decided to use the one below from Eric Stoltz. This function makes use of inverse route distance.

Text

Description automatically generated

Here is an alternative fitness function this fitness function makes use of inverse total distance.

I choose to go with the option from ‘Eric Stoltz’ as I don't believe the other method makes good use of the actual distance between the routes rather it uses a generic formula. So, the encoding in this case would not be correct in the first method we are actually trying to minimize the route distance.

Crossover

Text

Description automatically generated

Discuss later

When there is not enough information available to solve a problem, GA can be implemented. To achieve the optimal performance using GA, the problem must be represented with the proper encoding, the appropriate fitness and crossover functions must be determined, and the parameter values must be appropriate.

## Theory of coalescence

It represents how coalescent events on a gene genealogy are distributed.

The coalescence theory states that all alleles of a gene must have descended from a single allele. In a population of constant size with random mating, the idea often applies to neutral or relatively neutral genes. The generation period and population size affect the coalescence time, which is the distance from the most recent common ancestor.

Diagram

Description automatically generated

This figure depicts the complete genealogy of a population of “ten haploids” taken from [co]. As shown on the figure the lines trace the ancestries of the three sampled linages to a single common ancestor. In other terms all of the alleles in the final generation go back and coalesce which means merged together in this case merged together in the past. So, we can see the coalescent even or the single allele that gave rise to all of the descendant alleles. We can see that there is no natural selection here so in a way coalescence is inevitable. It should also be noted that the average time to coalesce is dependent on the size of the population.

Figure 2 depicts basic principle behind coalescent

Chapter 2: Genetics Overview

## Genetic Drift

Genetic drift can be described as the random variation in the frequency of an existing gene variant in a population.

There are two major types that can occur which are: the founder effect and bottlenecks. The founder effect occurs when a small subset of a larger population establishes a new population. A population bottleneck arises when the size of a population is drastically reduced, causing a shift in the distribution of alleles.

First, we will discuss what causes genetic drift, smaller populations of organisms are significantly more likely to experience genetic drift. As seen on the graphs the lines represent the frequency of alleles in a population. From the initial graph we can see that when the population is small and a lot of alleles exist, each of the alleles can easily go extinct or become fixed in the population. By contrast when the population is large an allele is less probable to disappear entirely since more individuals contain it, reducing the likelihood that all of them would become extinct (seen in the last graph).

Graphical user interface, chart, line chart

Description automatically generated

Figure 3: Random genetic drift graphs

Discuss bottleneck and founder in more detail.

**Population Bottleneck**

The Bottleneck effect is when you have some sort of major disaster or effect that in turn kills of a lot of the population resulting in a severe decrease in a populations size. Examples of these are competition and spread of disease which leads to a massive decrease in population size. So only a little bit of the population is able to survive. The organisms that did not die now determine the allele pool. Some alleles become more frequent simply because they are the only ones left. An example of this can often not discussed be seen in [GD] with the case of antibiotics. Antibiotics kill pathogenic bacteria in your system regardless of their alleles and massively help to reduce harmful bacteria, preventing the spread of disease. However, if the patient is not consistent with usage and quits to early then a small population of antibiotics will survive. Potentially, the allele frequencies in this much smaller population of bacteria could be  significantly dissimilar to those in the original population. Selection or further genetic drift causes the allele frequencies to change, and the new alleles will be dominant in the population.

**Founder Effect**

The same idea of a population becoming small, but the founder effect isn’t a result of a natural disaster rather a new population is formed in a new location. If this new population does not interact with and reproduce with the main population, the allele frequencies in this population will indeed be substantially different from those in the parent population leaving variations being disproportionately different. A result of the founder effect, many islands have species that only exist on a single island. For example, perhaps if two birds of a species land on an island, the diversity there will be reflected for simply by their alleles. Although these alleles may initially dominate, population mutations will eventually result in new adaptations. This new adaptation remains in the founding population. The two groups potentially become so different over time that they are unable to interbreed. This is a common way that species diverge.

# Design patterns

Design patterns are characterised by their use:

* **Creational**: Used to create objects that can be separated from the systems that implement them.
* **Structural**: Deal with coupling, used to connect numerous dissimilar objects into large object structures.
* **Behaviour:** These patterns are concerned with algorithms and the assignment of responsibilities between objects.
* **Architectural:** Describes a system’s assemblies, layers or environment

**Object Scope:** Handles object relationships that are changed in real time.

**Class Scope:** Addresses relationships between classes that can be modified during compilation.

## MVC Design Pattern

[The Catalog of Design Patterns (refactoring.guru)](https://refactoring.guru/design-patterns/catalog)

# Technical Aspect

## Technical Decisions

Justify your choices for the software used to create prototypes

One of the main software used for my project was sublime text which was used for coding both the graphical user interface and the genetic algorithm program. I chose to use sublime text due to its simplicity and convenience. Prior to this I had not been aware of sublime text and had not used it, but upon research realised how convenient it is to use. You can work with folders already created and it works with multiple programming languages incorporating different build systems. As you probably know already my program uses different python modules, with pip install and sublime text this was easy to do so using modules like Tkinter was not a problem to implement. There are also many useful keyboard shortcuts again emphasising the convenience such as ‘cntrl /’ for commenting out code and ‘cntrl [’ for indenting large chunks of code.

Prior to this as I mentioned before I had not heard of sublime text, so I had other text editors in mind to use such as the python idle, eclipse and visual studio. Although I may use the python Idle in the future, I decided to use sublime text over the other text editors as it was more straightforward to use and as I previously stated its convenience. It has a user-friendly environment and is very intuitive to use, its straightforward to use and it is not difficult to learn how to use. It also has customizable syntax-specific colour schemes are used to display code. The only caveat to using Sublime text is if you are on the free version an annoying pop-up frequently appears.

Go into more detail

## Planning

Describing the beginning phases.

At the beginning of term, we were tasked with creating a project plan which is essentially the guideline and framework for the project outlining what you hope to achieve on a weekly basis. My project plan consisted of doing a lot of research and learning how to use the software I selected for my project namely sublime text. Thankfully learning to use sublime text was not too difficult so I didn’t have to dedicate a huge amount of time for it. However, due to the difficulty of the project I ad to dedicate a large portion of time to research and understand it. Which justifies why there was some delays in the earlier weeks for the development of the project. With this research time I managed to discover a bevy of research articles and some books on genetic algorithms. Thus, explaining how I was able to find many points for reference.

Naturally, there were days when I was unable to follow my project plan, which caused me to slip behind a little bit, but this was always anticipated because the project plan represented my best-case for what I could accomplish rather than more of a standard plan. As a result, anything less was primarily a great plus for me.

The standard for the online project diary differs slightly from the project plan in that it effectively served as a log of my activities, detailing what I accomplished each week. For the sake of understanding, the project plan is the **framework** for the project outlining what you hope to achieve on a weekly basis, whilst the project diary is a **log** of my weekly activities. As it was clearly emphasised to be important from the beginning, I made sure to dedicate time to it. So, despite not fully following the plan I made sure to update the diary weekly with my activities and what may have been a limiting factor such as me catching covid or using certain technologies which require too much time. I didn't report on a weekly basis because I frequently missed submitting a review of my week and my accomplishments during some weeks due to the heavy workload I was carrying out. At times the last thing I remembered was, of course, to make a report about my week's work because finishing big amounts of work every week was exhausting.

During my meetings with the supervisor, I always had my trusted tablet with me to take notes and write down feedback on certain areas. This came in handy when I was working on the fitness function and didn’t know where to begin but as I had my notes from meetings, I was able to go back and review my notes. From this I was able to properly research into fitness functions and discover different fitness functions that could be useful and outline them. Thus, demonstrating the importance of taking notes during meetings.

## Software development

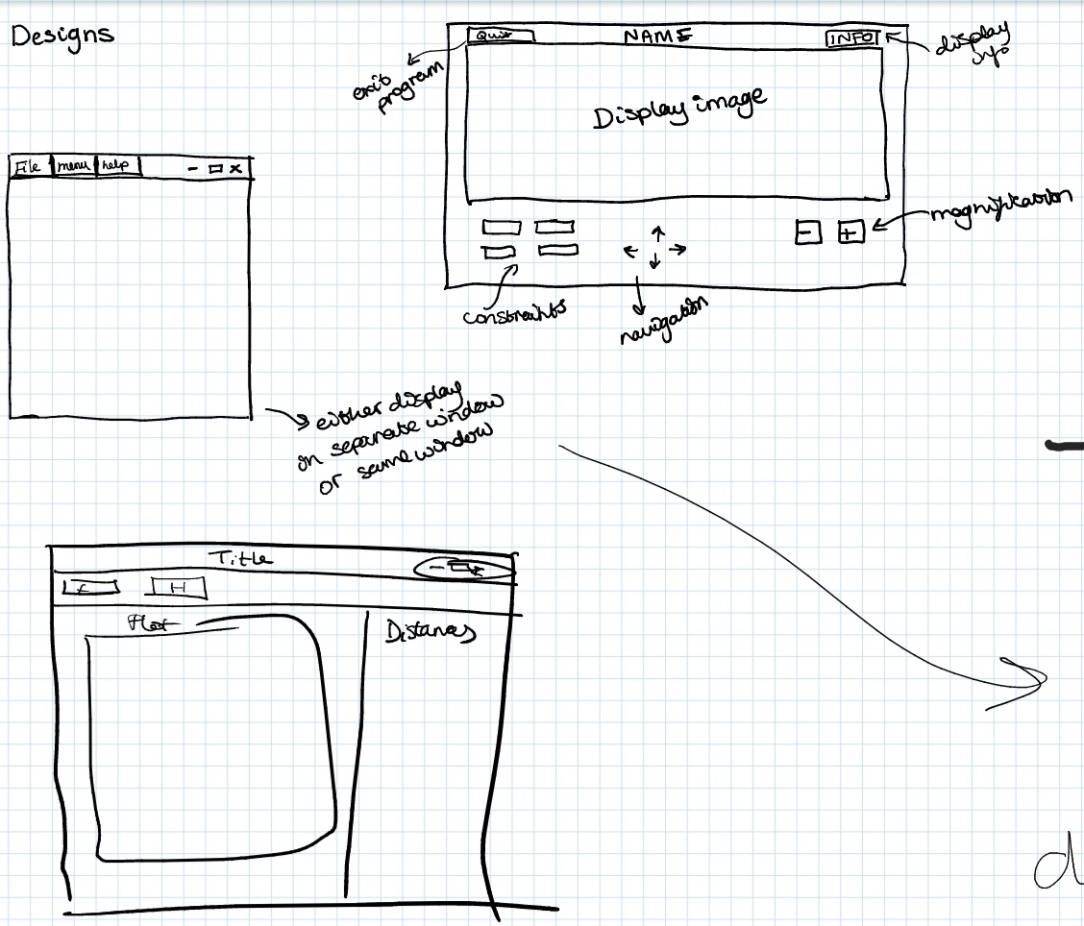
Discuss tools used i.e., Gitlab

From the beginning of term, I have been using a tool called Gitlab with access provided by the university, a platform for version control. The University provided me with a private repository where I had to commit all my code and I decided to commit my weekly logs there too. This is for ease of use and access. This includes the program, report, and project diary.

Using Gitlab allowed me to be able to have a place to consistently upload different components of my project whilst also ensure that if anything is deleted, I still have a backup. With the aid of version control I was also able to go back and look at previous drafts and be informed on what I need to change. This was a good way to present my work as evidence of what I had been doing. It is also safe for me since, in the unlikely event that my IDE crashes and all of my work is lost, I still have a backup option to restore all of my code. I was able to track any modifications I made to earlier versions of my code thanks to Gitlab.

## Software Engineering

I used an agile methodology while working on this project. This method performed best for my project since it enabled me to deliver what I had projected based on my project plan. The agile methodology operates traditionally by organising a project and breaking it down into manageable segments. At every stage, you process and plan in addition to continually testing and improving. For me, I had a thorough layout plan (project plan) from the beginning, so I knew exactly how I wanted my prototype to appear.



As you can see my 2D prototypes differ from my prototype developed. It is important to show how the project developed.

## Proof of concept

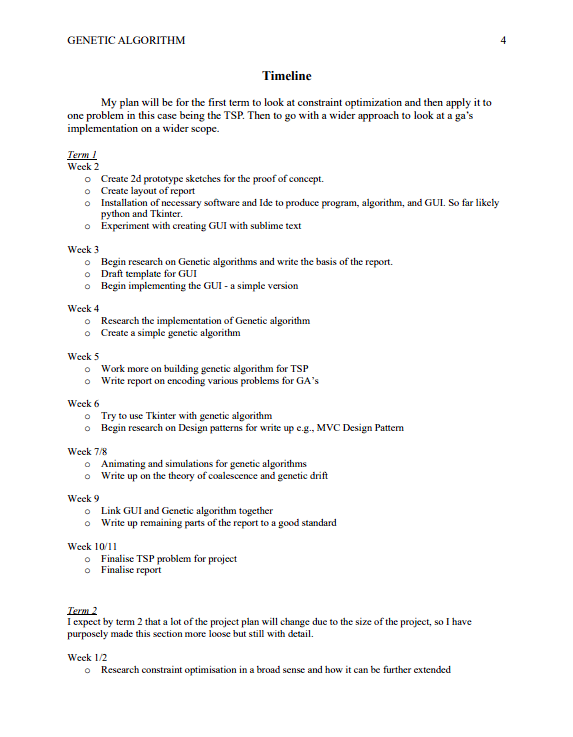
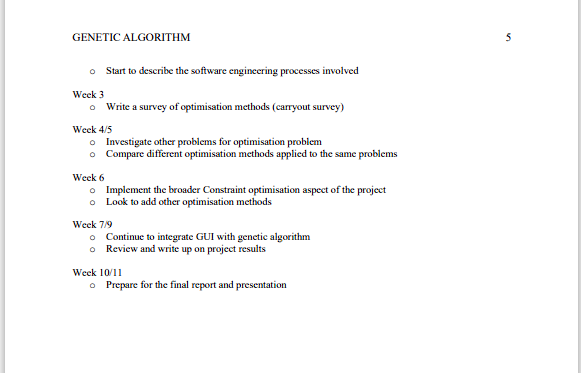
## Next term aims

Final Deliverables

* Research constraint optimisation in a broad sense and how it can be further extended
* Write a survey of optimisation methods (carryout survey)
* Investigate other problems for optimisation problem
* Compare different optimisation methods applied to the same problems
* Implement the broader Constraint optimisation aspect of the project
* Look to add other optimisation methods
* Continue to integrate GUI with genetic algorithm

# Project Plan

Timeline of the project plan

This is the original project plan for the project. The plan for the first term is more detailed then the second as I believe due to the cast nature of the project it can be more complex to deal with certain parts as such, I made sure that for second term I didn’t overdo it with tasks.

# Summary of project diary

I've been keeping track of project diary entries since the beginning of the term. Initially I did this by noting down what I was planning to do on my tablet. Then I found it would be a much better idea to create a digital diary on my GitHub repo and update it every week, I felt it would be best to update the diary in weekly blocks. It is stated that we should spend up to 20 hours each week on the project. Initially, I envisioned working 4 hours per day for 5 days, but I quickly realised that this was not realistic due in part to other modules, such as data analysis, that required time as well. As a result, I decided to change this to roughly 3.5 hours per 6 days. Working on the project would entail doing programming, writing a report, and researching concepts. Between the task I aim to spend an hour on each at time alternating at times to guarantee that certain aspects are up to standard. However, I will admit that often maintain this was difficult due other modules but also the workload. This explains why my project diary entries are well organised at the start of the term, but by the middle of November it was difficult to maintain track of this diary, not because my plan was not accurate, but rather the workload was becoming increasingly difficult.

Glossary

Words and stuff

**Selection:** chooses the parents, who will contribute to the population of the following

**Mutation:** subjects each parent to random modifications.

**Crossover**: combine two parents to create the next generation's offspring.

Bibliography

[1] - En.wikipedia.org. 2022. *Genetic algorithm - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Genetic\_algorithm> [Accessed 22 September 2022].

[2] - En.wikipedia.org. 2022. *Genetic algorithm - Wikipedia*. [online] Available at: <https://en.wikipedia.org/wiki/Genetic\_algorithm> [Accessed 22 September 2022].

[3] - Cs.mcgill.ca. 2022. [online] Available at: <https://www.cs.mcgill.ca/~dprecup/courses/AI/Lectures/ai-lecture05.pdf> [Accessed 27 September 2022].

[R] Holland, J.H. (1992) *Adaptation in natural and artificial systems: An introductory analysis with applications to biology, control, and Artificial Intelligence*. Cambridge: The MIT Press [Accessed 22 October 2022].

[GD] Kryvokhyzha, D. (2018) *Genome evolution and adaptation of a successful allopolyploid, capsella Bursa-Pastoris*. dissertation. Acta Universitatis Upsaliensis [Accessed 21 November 2022].

[CO] Rosenberg, N.A. and Nordborg, M. (2002) “Genealogical trees, coalescent theory and the analysis of genetic polymorphisms,” *Nature Reviews Genetics*, 3(5), pp. 380–390. Available at: https://doi.org/10.1038/nrg795.

[GD] Buckley, G. (2021) Genetic drift - definition, examples and causes, Biology Dictionary. Available at: <https://biologydictionary.net/genetic-drift/> [Accessed: November 28, 2022].