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 13, 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 Where other published and unpublished source materials have been used, these have been acknowledged.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](#_Toc118646647)

[Project Specification 5](#_Toc118646648)

[Chapter 1: Introduction 6](#_Toc118646649)

[1.1 How to use this template 6](#_Toc118646650)

[1.2 Genetic Algorithms 6](#_Toc118646651)

[1.2.1 Constraint Satisfaction Problems: 7](#_Toc118646652)

[1.3 Encoding 7](#_Toc118646653)

[1.4 Theory of coalescence and Genetic Drift 7](#_Toc118646654)

[Chapter 2: Genetics Overview 8](#_Toc118646655)

[2.1 Theory of coalescence and Genetic Drift 8](#_Toc118646656)

[Chapter 3: Design patterns 9](#_Toc118646657)

[3.1 MVC Design Pattern 9](#_Toc118646658)

[Chapter 4: Technical Aspect 10](#_Toc118646659)

[4.1 Technical Decisions 10](#_Toc118646660)

[4.2 Planning 10](#_Toc118646661)

[4.3 Software development 10](#_Toc118646662)

[4.4 Software Engineering 10](#_Toc118646663)

[4.5 Proof of concept 10](#_Toc118646664)

[4.6 Next term aims 10](#_Toc118646665)

[Chapter 5: Project Plan 11](#_Toc118646666)

[Chapter 6: Summary of project diary 12](#_Toc118646667)

[Bibliography 13](#_Toc118646668)

Abstract

This document serves as a layout and formatting template for your project report. It does not tell you how to write it, or what it should contain. It explains how it should be formatted and typeset. Please refer to your project booklet for information about report sizes, contents and rules.

***NOTE: in your report, you should replace this with an appropriate Abstract for your project report.***

Project Specification

Aims: To implement a GA (or, preferably, a range of variations of the GA) and to apply it to various optimisation problems (e.g. constraint satisfaction). The program should follow a model-view-controller design pattern. It should be possible to design, implement, and evaluate a (range of) genetic algorithms.

Background: A genetic algorithm (GA) is a simulated model of evolution: techniques of this type are widely used to solve optimization problems. "Solve" is perhaps the wrong word here: 'try to improve a solution' might be better, for although these algorithms are models of evolution, they often don't work very well. GAs are so widely used because they are straightforward to program, and in optimisation problems, the only harm an ineffective algorithm can do is to waste computer time.

Early Deliverables

1. Proof of concept programs: First implementation of GA for constraint satisfaction/function optimisation.
2. Report on Design Patterns.
3. Report on Genetic Algorithms.
4. Report on encoding various problems for GAs.
5. Report on the Theory of coalescence and genetic drift.

Summaries

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 the

## 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

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

<https://www.hindawi.com/journals/cin/2017/7430125/>

<file:///C:/Users/LBX/Downloads/A_New_Approach_using_Binary_Encoding_to_Solve_Travelling_Salesman_Problem_with_Genetic_Algorithmsfootnote_This_report_is_English_translation_of_paper_.pdf>

## Theory of coalescence and Genetic Drift

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.

Chapter 2: Genetics Overview

## Theory of coalescence and Genetic Drift

Genetic drift is 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.

# Design patterns

## MVC Design Pattern

# Technical Aspect

## Technical Decisions

## Planning

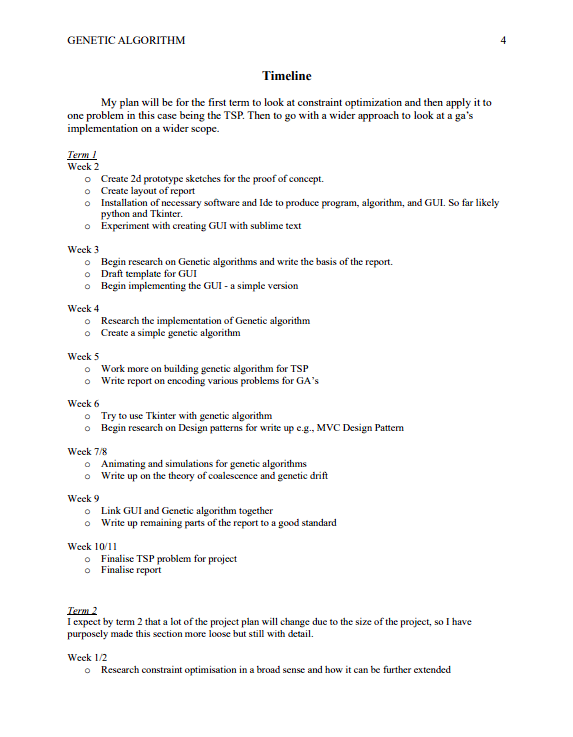
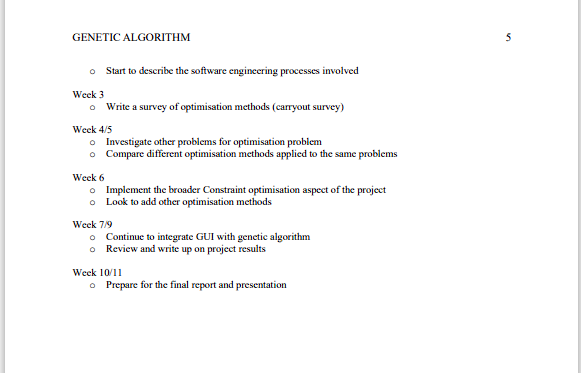
## Software development

## Software Engineering

## Proof of concept

## Next term aims

# Project Plan

Comments

# Summary of project diary

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.