CELLULAR COMPUTING FOR EVOLUTIONARY DYNAMICS

A DISSERTATION SUBMITTED TO MANCHESTER METROPOLITAN UNIVERSITY
FOR THE DEGREE OF MASTER OF SCIENCE
IN THE FACULTY OF SCIENCE AND ENGINEERING

2019

By
[redacted]
Department of Computing and Mathematics

Contents

Abstract vii

Declaration viii

Acknowledgements ix

Abbreviations x

1 Introduction 1

2 Literature Review 3

2.0.1 Genetic Algorithms 3

2.0.2 Synthetic Biology 6

2.0.3 Biological Simulation Software 9

2.0.4 Related Works 14

2.0.5 Conclusion 21

3 Design 22

3.0.1 Functional and Non-functional Requirements 22

3.0.2 Kappa Template 29

3.0.3 Genetic Algorithm 30

3.0.4 Data Dictionaries 33

4 Implementation 37

4.0.1 Global Settings 37

4.0.2 Individual Class 38

4.0.3 Genetic Algorithm Utilities 38

4.0.4 Kappa Integration 41

4.0.5 Primary Loop 42

4.0.6 Dictionary Optimisation 43
4.0.7 Distribution Plotting 44

5 Evaluation 46
5.0.1 Performance Evaluation 46
5.0.2 Genetic Algorithm Evaluation 50

6 Conclusion 63
6.0.1 Future Work 63
6.0.2 Closing Thoughts 65

References 67

A Terms Of Reference, Ethics Forms & Additional Material 68

List of Tables

3.1 Functional requirements for the developed solution 27
3.2 Non-functional requirements for the developed solution 29
3.3 Data dictionary containing global variables for genetic algorithm. . . . 35
3.4 Data dictionary containing variables for individuals in the genetic algorithm. 36

5.1 Details of all settings used for the control experiment. 51
5.2 Results of genetic algorithm experiments. 61

List of Figures

2.1 Sample of sbml code that is used to define a model for E.Coli cells - VK (2006) 10
2.2 Sample of Kappa template available on Kappas web based UI. - Kappa-Language (n.d.) 12
2.3 Resulting contact map of template described in figure 2.3, note how agents and contact sites are represented as well as which agents interact with certain sites. For example, agent A has two contact sites, c and x, where both of agent cs sites bond with site c and agent x bonds with site x. 13

3.1 Generated contact map for template file. R (activator) will bond to A (promoter) which will then commence the production of M (mRNA) based on how many sites are currently bonded to. Produced M agents will begin output of P (protein). 30
3.2 Proposed workflow for genetic algorithm 31

4.1 Implementation of the individual class. 38
4.2 Implementation of mutation function. 40
4.3 Implementation of crossover function. 41
4.4 Implementation of kappa within the primary loop. 42
4.5 Implementation of the main function used during the primary loop. . . 43
4.6 Implementation of distribution comparison functionality within the solution. 45

5.1 Visualisation of results from Kappy and multi-processing experiments. 49
5.2 Visualisation of results from the dictionary optimisation experiment with a retest threshold of 60. 49
5.3 Collection of average level of fitness of each generation during the control experiment. 51

5.4 Progression of average level of fitness during the first 10 generations of the genepool size experiments. 54
5.5 Progression of highest level of fitness during the first 10 generations of the genepool size experiments. 55
5.6 Progression of average level of fitness during the first 10 generations of mutation rate experiments. 57
5.7 Progression of highest level of fitness during the first 10 generations of mutation rate experiments. 58
5.8 Visualisation of plotting the simulation results from the individual with the hightest fitness during experiment IND-175 against the distribution set of the user required protein setting. This resulted in a fitness of 70.21 when using chi-squared. 59
5.9 Visualisation of plotting the simulation results from the individual with the hightest fitness during experiment FIT-TS against the distribution set of the user required protein setting. This resulted in a fitness of 99.99 when using two sample t-test. 60

Abstract

In recent years, advances in computer science has broadened the scope for who can use these advances to further their own research in different areas of scientific studies. One area that researchers are interested in utilising computer science is synthetic biology, a school of thought within biology that seeks to understand the genetic makeup of micro-organisms in order to re-purpose them for various applications that can potentially benefit modern society. Using computing technology within this area would enable researchers to evaluate their initial hypotheses and work around various obstacles. One method that has been proposed to aid researchers find viable genetic configurations is a genetic algorithm, a type of algorithm that gradually evolves a set of potential solutions in order to work towards a users requirements.

This project follows the development and creation of a genetic algorithm that is capable of working towards finding a cellular configuration that is able to produce a distribution of protein output that is similar to what is required by the user. The genetic algorithm was then evaluated to determine if it meets the primary goals of the project, which was carried out by evaluating its overall performance when optimisation techniques were applied and how modifying various settings within the genetic algorithm would affect the solutions ability to work towards a potential solution. Whilst the findings from evaluation found that the solution was able to deliver on the primary goals of the project, they also highlight potential areas that can be improved upon during future work.

Declaration

No part of this project has been submitted in support of an application for any other degree or qualification at this or any other institute of learning. Apart from those parts of the project containing citations to the work of others, this project is my own unaided work.

Signed: [redacted]

Date: 23/09/2019

Acknowledgements

I would like to thank the following people for assisting with this project:

Matteo Cavaliere, for acting as my supervisor and for his much appreciated expertise, guidance and feedback during the course of this project.

My friends and family, for their kindness and support throughout this endeavour into the field of computer science.

Abbreviations

GA	Genetic Algorithm
ODE	Ordinary Differential Equation
SDE	Stochastic Differential Equation
SVP	Standard Virtual Parts
ACO	Ant Colony Optimisation
DNA	Deoxyribonucleic Acid
SBML	Systems Biology Markup Language
GCDL	Genetic Circuit Description Language
GCC	Genetic Circuit Compiler
FR-x	Functional Requirement - x
NR-x	Non-functional Requirement - x

Chapter 1

Introduction

Over the past few years, development in computing technology has led to significant advances which has broadened the scope for how computing can be applied as well as broadening the scope for who can to utilise these advances to further their own scientific studies. One area in computing that is gaining increasing interest from a plethora of communities is machine learning. Machine learning has been used by many researchers to assist with their research in an attempt to address different kinds of challenges they may be facing, particularly in instances where a problem is either very complicated or does not follow an obvious kind of pattern. As well as advancing the technology itself, this technology has also been made more accessible to those who may not possess a background in computer science, thus enabling them to apply the technology to their area of study.

> 'Computational science and engineering also enable multidisciplinary design and optimization, reducing prototyping time and costs' - Reed & Dongarra (2015)

One area where researchers are interested in applying this kind of technology is the field of synthetic biology. In this field, researchers concern themselves with the synthesising of customised micro-organisms that are tailor made by biologists to address some form of required functionality. Some areas of society that findings from this field can be applied to include usage within medical fields, such as biosensors and medicine dispersal, and in chemical manufacturing, such as the processing of biofuels. One obstacle that researchers face in this area is needing to justify the purpose of their projects, which potentially can be overshadowed by large development costs and having to manufacture large quantities to recoup these costs.

> 'Second, the application area must have a sufficiently large volume to justify the cost of development' - Paul S. Freemont (2012)

The purpose of this project is to develop a software solution that will allow researchers in this field to narrow down how micro-organisms should be configured to provide the required protein output in response to a promoter in the system that it will be present in. The machine learning model that has been proposed to be used to address this problem is a genetic algorithm, a type of machine learning pattern where the algorithm is able to work towards a potential solution for a problem over a period of time and uses mechanics that resemble that of biological evolution to refine the calculation of new potential solutions. First, a literature review will be carried out, which will include more in-depth discussion of the two fields being explored in this project along with analysis and discussion of related work, which will help determine additional requirements for the algorithm and can be used during evaluation of the final algorithm. Following on from this, the development of the genetic algorithm will be documented, which will include how it was designed to address the described challenge. This will include documentation for how the algorithm was initially designed followed by how the algorithm was implemented, including any optimisation that was required. This report will then conclude by evaluating the overall effectiveness of the final implemented solution, determining if the solution met the requirements of the project and suggestions for how this kind of implementation could be improved in future works.

Chapter 2

Literature Review

Before commencing with the design and implementation of a prototype for addressing the project criteria, research will first be carried out. During this section of the report, the key components of the project proposal will be discussed in depth, this will include discussions for genetic algorithms, synthetic biology and software that can be used to simulate biological systems. Following on from this, related works will be discussed where researchers have attempted to utilise evolutionary programming and other computing solutions to address tasks within the field of biology. The analysis that will be undertaken during this stage of the project will be vital as any insights gathered from this stage will ultimately be used to shape decisions made during the design stage of the project, particularly when outlining functional and non-functional requirements.

2.0.1 Genetic Algorithms

One area of machine learning that has gained increasing interest in recent years are genetic algorithms, a concept which was conceptualised in a larger field known as evolutionary computing, a term which was originally coined by Laurence J. Fogel in 1966. Genetic algorithms take inspiration from Charles Darwin's theory of evolution, wherein generations of objects, usually referred to as individuals, resembling some potential solution to a problem, are generated and bred together until a suitable solution is generated by the algorithm.

> 'But the very point of evolutionary programming is its versatility. Here is a procedure that includes a random process in order to generate a never-ending sequence of alternative hypotheses concerning suitable logic and decision-making.' - David B. Fogel (1966)

These kinds of algorithms are ideal for problems where the desired outcome is known by researchers but not necessarily what the optimal solution is for achieving it. A commonly used scenario for illustrating the kind of problems these algorithms can be used to tackle is the knapsack problem, where an algorithm determines the most useful combination of items to take with you on a camping trip before the weight limit is exceeded. One example of real world application of genetic algorithms was a project carried out by NASA, where evolutionary computing was employed in order to determine the most optimal antenna design for certain space exploration missions.

> 'Evolutionary algorithms draw inspiration from nature, an evolutionary algorithm starts with a randomly initialized population. The population then evolves across several generations' - Wong (2015)

> 'Most recently, we have used evolutionary algorithms to evolve an antenna for the three spacecraft in NASAs Space Technology 5 (ST5) mission7 and are working on antennas for other upcoming NASA missions, such as one of the Tracking and Data Relay Satellites (TDRS).' - Hornby et al. (2006)

A genetic algorithm is typically made up of five components that are used throughout the process in a continuous loop until an optimal solution has been produced.

Representation

The first is representation, this encompasses how a potential solution in the population is encoded in a code format. For example, a genome sequence could be represented within an algorithm as an array of integer values. These encoded values are used throughout the algorithm to carry out its functionality and try to work towards a potential solution. At the beginning of the process, these individuals will be generated with random traits.

Fitness Evaluation

The next aspect of a genetic algorithm is fitness evaluation. This stage of the algorithm is used to determine how well each individual in the population fulfils the criteria of the defined problem. The fitness function is defined by the researchers developing the algorithm and the formatting of the function will vary depending on the exact nature of the problem that the algorithm is trying to address. In the context of the knapsack problem, the fitness function would calculate a value that is comprised of the total value of the items that an individual solution has taken, it could also set the value returned as 0 if the solution exceeds the maximum weight limit.

> 'Genetic algorithms model evolution as a succession of changing gene frequencies, with contending solutions being analogous to chromosomes. The space of possible solutions is searched by applying transformations to the trial solutions as observed in the chromosomes of living organisms: crossover, inversion, point mutation, and so forth.' - Fogel & Fogel (1996)

Reselection

The value calculated during fitness evaluation can then be used to carry out the next stage of the algorithm, reselection. During this stage, based on the fitness value generated during the previous step, individuals from the population are selected to be used during the crossover stage for generating new individuals. There are different ways of implementing this step, one example is bias roulette selection. With this method, all individuals in the population will have a chance of being randomly selected, with those having a higher fitness value and therefore being better suited for addressing the problem, being more likely to be selected for crossover.

Crossover

When individuals have been selected, the crossover stage is then carried out, this is also sometimes referred to as the mating step of the algorithm. Here, aspects of individual represented solutions are combined with other solutions in order to generate new offspring solutions that contain traits from both of its parent solutions.

Mutation

The last step that is carried out before the process begins again is the mutation stage. At this point in the algorithm, there will be a small chance that part of an individuals encoded solution may be altered slightly. This step is vital for ensuring that there is some element of controlled randomness in the population as it will help with preventing generated solutions within the population from stagnating with a limited variation of individuals.

> 'Can we think of the variables of an object as DNA? Can objects make other objects and pass down their DNA to a new generation? Can our simulation evolve?' - Shiffman (2012)

These stages of the algorithm are repeated until an optimal solution has been found, which can be done through either an exit condition defined in the algorithm or through manual human intervention. By utilising this kind of framework, a genetic algorithm should, in theory, be able to work towards producing an optimal solution that is capable of potentially addressing the problem that the algorithm was made to address. Fitness evaluation and reselection in particular should allow for the algorithm to produce more suitable individuals as it works through each new generation, with the fittest of the population being selected for producing new offspring that should lead to more suitable solutions. If an algorithm properly implements all these stages, this should allow for an algorithm that is able to work towards a potential solution through a refined and intelligent trial and error process.

Considering how genetic algorithms function and the fact that they can be tailor built to address many kinds of problems, this type of algorithm design would be suitable for addressing the primary goal of this project. This kind of algorithm could be used in conjunction with carrying out simulations of many different genetic configurations until a potential solution has been found that meets the criteria of the users required protein output. Reselection should also allow for different genetic configurations being crossed over with each other, which should lead to the algorithm gradually working towards a solution that works towards the user's requirements. To make this algorithm viable for the task at hand, optimisation will likely be a top priority during the development stage of the project. To make such an algorithm viable for addressing the project goal, it will have to be able to work towards a potential solution within an appropriate time scale and whilst making efficient use of computing resources.

2.0.2 Synthetic Biology

Over the course of the last two centuries, scientists have made great progress in the field of biology, one key development includes the discovery of deoxyribonucleic acid (DNA) which serves as the building blocks in all living organisms. Since its discovery, scientists have carried out numerous research projects in various attempts to map and understand the genetic makeup of different organisms. One of the most famous projects carried out concluded in 2003, the human genome project, where a team of researchers published their findings demonstrating that they were able to map out a significant portion of the human genome. These developments have led to the creation of a new subfield within biology, commonly known as synthetic biology.

'Having a complete sequence of an organisms genome is essential for understanding how a cell works and how it can be re-engineered to be a chassis system for synthetic biology' - Paul S. Freemont (2012)

The purpose of synthetic biology is to synthesise repurposed micro-organisms in order to carry out customised functionality based on the intentions of those manufacturing them. One of the most common ways to achieve this is to implant a custom written gene sequence into an existing micro-organism that has been previously mapped out, this organism is then capable of carrying out this additional functionality. Some of the most commonly used micro-organisms used to achieve this are E.coli and bakers yeast. By doing this, scientists are able to manufacture customised cells that can carry out custom functionality by themselves or form part of a larger genetic circuit where many different organisms can collectively process more complicated tasks. The latter potentially paves the way to a new kind of computation that can take place within living organisms, this is sometimes referred to as wetware.

By utilising discoveries found in the fields of gene mapping and synthetic biology, scientists today are able to synthesise and manufacture micro-organisms to carry out a wide variety of applications in order to address concerns and problems that face modern society. The most obvious and common use for synthesised organisms is within the field of medicine where micro-organisms can be manufactured to aid with medical treatments. For example, in the paper by Anderson et al. (2006), researchers studied the potential for E.coli cells to be engineered to release a cytotoxic agent that can potentially treat cancer cells within a patient. The applications for synthetic biology are not limited to medical fields, researchers are also determining how manufactured micro-organisms can be used within fields such as fuel manufacturing, drug manufacturing and genetically engineered crops. For example, in the paper by Lee et al. (2008), researchers put forth their proposal for how synthesised organisms can be used in order to process plant matter into different kinds of fuels, which could be used to address growing concerns about society's continued reliance on fossil fuels. All developments within this field, as well as the growing scope of potential applications have led to current projections that by 2022, the projected market worth of synthetic biology will be worth 13.9 billion dollars.

'One potential application of inv+ E. coli is therapeutic bacteria for the treatment of cancer. By restricting the expression of inv to tumor sites, invasion could be confined to malignant cells.' - Anderson et al. (2006)

'The global market for synthetic biology will grow from nearly \$4.4 billion in 2017 to \$13.9 billion by 2022' - Bergin (2018)

As with other fields of research, the field of synthetic biology is not without its obstacles. The synthesis of engineered micro-organisms can be a costly and time-consuming process, which in turn requires strong justification from researchers before the creation of new component organisms can be explored. Outside of monetary complications, synthetic biology can also come under fire from those with strong ethical objections, some viewing the editing of genes in organisms as playing God. Recently, a researcher operating in China faced staunch criticism after gene editing two children's genes in an attempt to protect them from the HIV virus, a decision that may actually negatively impact the two childrens health in the long term.

'His announcement was met with condemnation from hundreds of Chinese and international scientists, who said any application of gene editing on human embryos for reproductive purposes was unethical.' BBC-News (2019)

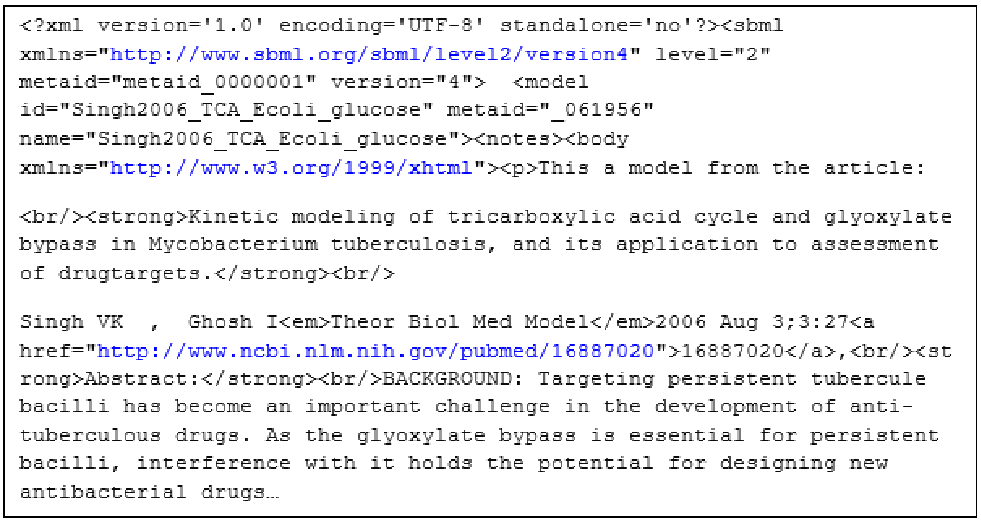
During this project, the proposed solution should aim to address these issues in order to provide researchers operating in this field a means for furthering their studies into their own proposals. The implementation of a genetic algorithm should allow for many different configurations of synthetic micro-organisms to be prototyped and simulated until one solution presents itself that fulfils the users required protein output. The user could then take the results from an optimal generated solution to a lab environment to begin the development of the required micro-organism. This should therefore address the current issues surrounding long development times and project costs by providing researchers with an optimal starting point for development and reducing time spent in the lab. In order to deliver on this, as described during discussion of genetic algorithms, the proposed solution should be capable of working towards a configuration that meets user requirements within a reasonable amount of time, otherwise the proposed solution would only increase development time and would not be suitable for addressing these obstacles faced by researchers. The proposed solution could also potentially aid researchers with addressing ethical concerns. The solution should allow for researchers to evaluate their initial hypothesises without the immediate need for synthesising micro-organisms. The findings from this initial stage of the project could then be used to assist researchers with the ethical checks their project will most likely have to undergo before being given permission from their organisation to proceed. The findings from this initial stage could also be used to help put public concerns at ease by providing evidence that certain proposals have been thoroughly tested and evaluated before the actual synthesising of micro-organisms takes place, meaning that researchers will only proceed with their work until they are satisfied that it is safe to do so.

2.0.3 Biological Simulation Software

As computing technology has advanced, scientists have developed various software packages that can be used to simulate biological processes in order to aid with biological research. These packages have been designed with researchers in mind, who may not necessarily have a developed skillset for advanced work with computers, such as being able to program complex applications. By developing these packages to be more accessible to researchers, this means that more researchers in this field are able to take advantage of the benefits that these packages provide, which should allow for greater efficiency, improved research time and allowing researchers to evaluate their initial theories before proceeding into the lab. All of these advantages that these software packages provide should ultimately benefit the field as a whole. Considering the limited timescale for this project, different software packages will be considered for integration with the proposed genetic algorithm. Some of the key points that will be taken into consideration for these packages is how they allow users to define experiments, how they efficiently they can run simulations and how well they are able to simulate a biological system, which would include simulating any additional factors that could influence the protein production being simulated in this solution.

Mathsbml

One option available to researchers is Mathsbml, a package for Mathematica that is built upon the systems biology markup language (sbml). The sbml language, which follows an XML format, allows researchers to model biological systems that can then be used in conjunction with Mathsbml to carry out a simulation. This same language can also be used with many different software packages, meaning that these models can potentially be simulated in different software packages based on the researchers needs. Mathsbml allows for loading a model into a simulation, running it over a period of time and plotting the findings out onto a graph, which will show the user defined values over a period of time units. This allows researchers to determine how synthetically



```
<?xml version='1.0' encoding='UTF-8' standalone='no'?><sbml
xmlns="http://www.sbml.org/sbml/level2/version4" level="2"
metaid="metaid_0000001" version="4">  <model
id="Singh2006_TCA_Ecoli_glucose" metaid="_061956"
name="Singh2006_TCA_Ecoli_glucose"><notes><body
xmlns="http://www.w3.org/1999/xhtml"><p>This a model from the article:

<br/><strong>Kinetic modeling of tricarboxylic acid cycle and glyoxylate
bypass in Mycobacterium tuberculosis, and its application to assessment
of drugtargets.</strong><br/>

Singh VK  ,  Ghosh I<em>Theor Biol Med Model</em>2006 Aug 3;3:27<a
href="http://www.ncbi.nlm.nih.gov/pubmed/16887020">16887020</a>,<br/><st
rong>Abstract:</strong><br/>BACKGROUND: Targeting persistent tubercule
bacilli has become an important challenge in the development of anti-
tuberculous drugs. As the glyoxylate bypass is essential for persistent
bacilli, interference with it holds the potential for designing new
antibacterial drugs…
```

Figure 2.1: Sample of sbml code that is used to define a model for E.Coli cells - VK (2006)

customised micro-organisms will perform under certain circumstances.

Writing new models with the sbml format is not usually written in the same manner as conventional software. The recommended approach is to use a software package that utilises the sbml as an aid for constructing models, which is done in a manner that resembles that of object orientated design i.e. being able to define functions, parameters and rules for models within a project. One of the advantages to using sbml, a language that is widely used, is that there are many repositories available that contain well documented biological parts and processes that can be utilised into projects as required by researchers. And, as previously mentioned, these parts can be used with any software package that utilises sbml as its base language, such as Mathsbml.

Something to note about Mathsbml is that it utilises Ordinary Differential Equations (ODEs) to carry out its simulations. This means that each time a simulation is ran, provided that the settings within the sbml file remain the same, the exact same results will always be returned. The benefit to this is that less simulations would be required, which could allow for a genetic algorithm with a faster turnover time. But there are shortcomings to using this mathematical model for simulating biological systems, which will be discussed in detail following the discussion of the next software suite.

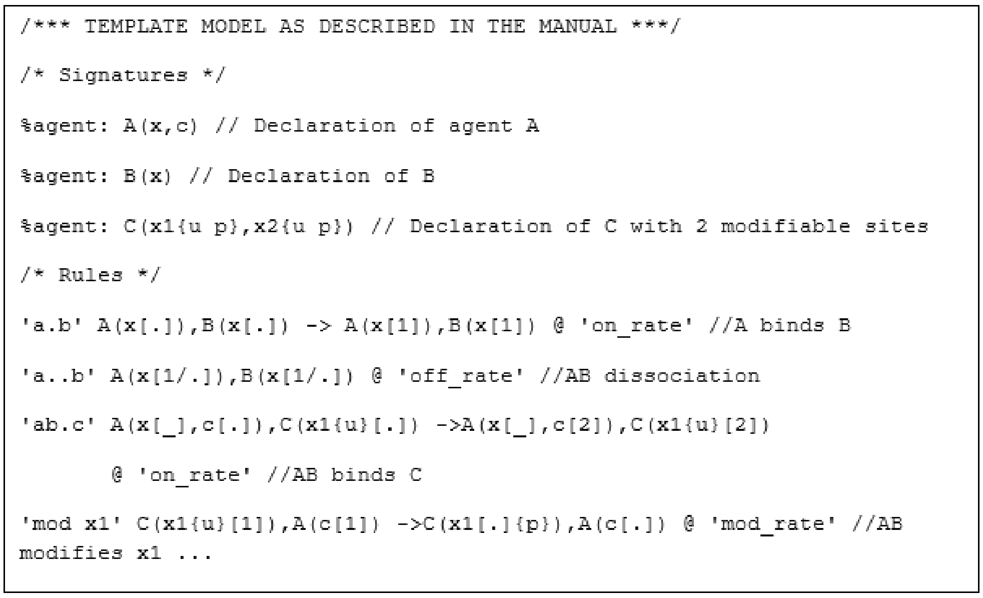
> 'MathSBML is an open-source package for working with SBML models in Mathematica. It provides facilities for reading SBML models, converting them to systems of Ordinary Differential Equations for simulation and plotting in Mathematica, and translating the models to other formats.' - Shapiro (2007)

Kappa

Another software package that is available for simulating biological systems is Kappa. Kappa uses its own language which, like sbml, allows researchers to define biological models that can then be simulated. Although software packages utilising the kappa language are not as widespread as the sbml language, there are multiple options available for researchers. The kappa simulation software is available to download off the website, can be configured to run as a web-based service on a server or users can access a web based application through their browser.

Kappa scripts are written in a format where each aspect of the simulation is initialised as a form of variable, meaning it does not follow an object orientated design in the same manner as Mathsbml. Within these scripts, it is possible for users to define agents within a biological system as well as other aspects of the agent that may be relevant to the simulation, such as the number of bonding sites that may be present. When a user has defined the agents that will be present within a simulation, the user is then able to define different rules for how agents within the simulation will interact and react under certain circumstances. For example, it is possible to define different rules for protein production for an agent in the simulation where its rate of production will be affected by the current states of its bonding sites.

Whilst the range of software is not as extensive as sbml offers, there still exists different options for carrying out projects with Kappa. The web UI, for example, can be used to both write out new simulation scripts and visualise how agents with interact with each other via the contact map utility. Simulations can then be ran through this web based UI or a local executable, visualising the state and number of different agents within a simulation, which are defined as observables within the simulation script. Once a simulation has been stopped by the user or reached a user defined point, such as a number of events occurring, it is then possible to download a file containing a time series of data. This data can then be loaded into the users choice of program for data analytics in order to plot the data out onto a graph, the web application is capable of doing this automatically.



```
/*** TEMPLATE MODEL AS DESCRIBED IN THE MANUAL ***/

/* Signatures */

%agent: A(x,c) // Declaration of agent A

%agent: B(x) // Declaration of B

%agent: C(x1{u p},x2{u p}) // Declaration of C with 2 modifiable sites

/* Rules */

'a.b' A(x[.]),B(x[.]) -> A(x[1]),B(x[1]) @ 'on_rate' //A binds B

'a..b' A(x[1/.]),B(x[1/.]) @ 'off_rate' //AB dissociation

'ab.c' A(x[_],c[.]),C(x1{u}[.]) ->A(x[_],c[2]),C(x1{u}[2])

       @ 'on_rate' //AB binds C

'mod x1' C(x1{u}[1]),A(c[1]) ->C(x1[.]{p}),A(c[.]) @ 'mod_rate' //AB
modifies x1 ...
```

Figure 2.2: Sample of Kappa template available on Kappas web based UI. - Kappa-Language (n.d.)

What sets Kappa apart from Mathsbml is that rather than utilizing ODEs to carry out its simulations, it instead uses Stochastic Differential Equations (SDE). This means that, unlike Mathsbml, when a simulation is carried out in Kappa on the same model multiple times, each simulation will return different results. By making use of SDEs, Kappa is able to simulate a biological system with background noise, which can influence the performance of the micro-organisms defined by researchers in their models. This would mean that during the genetic algorithms process, multiple simulations would have to be ran on the same configuration in order to get a general idea for how that configuration would perform in the environment it is intended for. This would require that additional optimisation is carried out in order ensure that these additional simulations do not hinder the overall performance of the algorithm.

> 'On the other hand, the stochastic framework takes into account the random interactions of biochemical species. More specifically, stochastic models are used to mathematically capture stochastic variations and noise, two properties inherent to biological systems.' - Paul S. Freemont (2012)

ODE vs SDE

The use of different differential equations by these two packages raises the question of which of these solutions would be the most beneficial to the project in question. Using

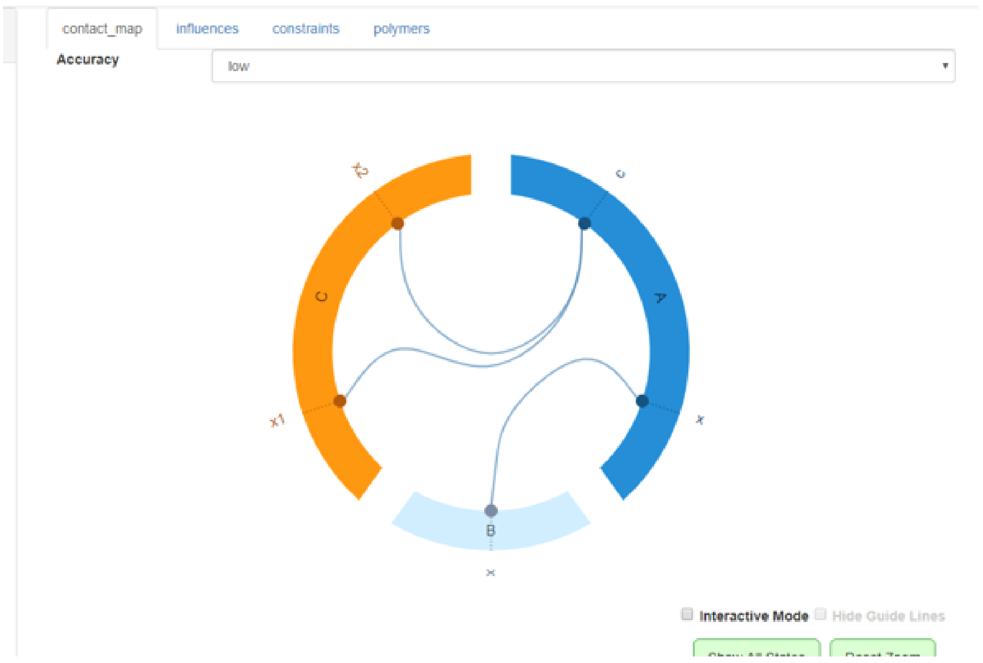


Figure 2.3: Resulting contact map of template described in figure 2.3, note how agents and contact sites are represented as well as which agents interact with certain sites. For example, agent A has two contact sites, c and x, where both of agent cs sites bond with site c and agent x bonds with site x.

a package such as Mathsbml, that uses ODEs, would likely allow for much faster performance as only one simulation would be required to retrieve an output value for each configuration within the algorithm, as running multiple simulations with the same settings will return the same result. However, the downside to using a package that utilises ODEs is that it will be unable to take external factors, such as noise, into account. Any form of biological system is highly complex, where many kinds of biological processes within a system can influence and affect other biological processes. Therefore, using a software package that uses ODEs to carry out simulations would return results that, while they could provide an initial idea of how a model performs, would fail to return results that would be reflective of how a synthesised micro-organism would perform within an actual biological system. On the other hand, a simulation package such as Kappa allows for models to be evaluated in simulations where noise can impact the performance of user defined models, which therefore provides a far more accurate idea of how a model would perform. The downside to this is that multiple simulations would be required to get a general idea of how different configurations would perform, which will require efficient optimisation where possible to ensure that the algorithm will perform optimally and work towards a potential solution within a reasonable time-frame.

With these considerations in mind, Kappa will be used to carry out the simulations for the genetic algorithm. Whilst this will require additional work to make it viable within the scope of a genetic algorithm, the fact that it utilises SDEs to carry out simulations means that it will return data that is far more accurate of how certain models will perform in their intended environment and will therefore be more beneficial to researchers.

2.0.4 Related Works

Before the next stage of the project can begin, related work will be discussed and analysed where these concepts and techniques have been applied to address similar objectives. The insights gathered during this section will be vital for understanding the kind of work that has been recently undertaken and where they could potentially be improved upon during this project and in future works. The analysis of these related works will also be used to refine decision making during the design of the proposed solution, such as the functional and non-functional requirements.

Tuning Receiver Characteristics in Bacterial Quorum Communication: An Evolutionary Approach Using Standard Virtual Biological Parts - Hallinan et al. (2014)

In the project carried out by Hallinan et al. (2014), researchers developed an evolutionary algorithm to deliver a solution that could help researchers identify which standard genetic parts should be used to output a specific response curve, similar to the goals set out for this project. As well as this, Hallinan et al. (2014) also set out to determine how viable it was to adjust found solutions to refine the output curve that resulted from certain solutions generated in the algorithm. Their solution was comprised of multiple components. The first was the genetic algorithm that was used to work through different configurations, avoiding a brute force approach whilst gradually refining its search by only retaining solutions that were deemed the most fit in their respective generations. Researchers also utilised a REST based server for storing a repository of standard virtual parts (SVPs) that could be used by the genetic algorithm to generate new potential solutions. All parts stored within this repository also came with comprehensive documentation, with a total of 2996 parts being available to use. Lastly, biological simulations would be carried out using sbml and an ODE based simulator, COPASI. Researchers note that their reasoning for opting for ODE over SDE was that they felt the models they were studying would not be viable for use in an algorithm that utilised SDE.

> ’Two versions of the algorithm were developed. The first had the aim of modifying only the final amplitude reached by the output curve, while the second aimed to modify the shape of the output curve to match that of a specific target.’ - Hallinan et al. (2014)

Hallinan et al. (2014) initially found success when they set their developed algorithm to work to toward a solution that produced a final value that met that of the target set, which would normally be obtained in less than 300 generations within the algorithm. However, they found that they were less successful when it came to their second goal of attempting to modify the response curve. When attempting to address this second objective in one of their experiments, Hallinan et al. (2014) found that out of 101 runs of their algorithm, only 31 of them would return with a successful result within less than 1000 generations of individual solutions. Of the runs that failed to produce successful solutions, Hallinan et al. (2014) found that these runs were being terminated due to either the simulation software or the algorithm itself eventually crashing under strained use. One other shortcoming noted by the researchers was that their solution was limited to what was available within the repository storing the standard parts, meaning that in order to expand the scope of options that the genetic algorithm can work with, new standard parts must be discovered, documented and stored into the standard parts repository.

> ’The possible behaviour of the models, and therefore the size and nature of the solution space, is constrained not only by the topology of the circuit, but also by the characteristics of the parts available in the SVP repository.’
> - Hallinan et al. (2014)

Hallinan et al. (2014) go on to note that more complex genetic circuits may require not only more sophisticated algorithms but also parts that are not currently available, this is something that could be addressed during this project by working with a rule-based language such as Kappa rather than relying on SVPs. Something to note is that researchers using this algorithm may find that the solutions it produces may not be as effective as they are described by the algorithm. This is because Hallinan et al. (2014)

opted to use an ODE based simulator over one that uses SDE. As previously discussed, whilst ODE based approaches allow for faster performances and less simulations being required for results, these approaches are unable to consider background noise that will most likely be present in an actual biological system.

> ’However, more sophisticated algorithms, producing more complex circuits may require parts not yet available.’ - Hallinan et al. (2014)

Overall, Hallinan et al. (2014) were able to make a promising start in producing a solution that can produce target outputs from genetic circuits and were able to highlight the challenges of using these kinds of algorithms to tackle more complex tasks. The findings in this project show that it is possible to utilise genetic algorithms to carry out this kind of work, but does emphasise the importance of optimisation. Therefore, optimisation will be a top priority both during the design and implementation of the solution produced during this project, especially considering the researchers in this project encountered issues when working with ODE based simulators, something that should be considerably quicker to use than SDE based simulators.

Metabolic Design and Engineering Through Ant Colony Optimization - Lincoln et al. (2015)

In the project carried out by Hallinan et al. (2014). the researchers developed an algorithm that was inspired by the theory of evolution in nature. Of course, there are other processes in nature that scientists look to when devising new potential ways of carrying out computation on complex tasks. In the paper by Lincoln et al. (2015). the researchers developed an algorithm that can work out the best configurations for E. coli cells in order to produce the required amount of succinct acid with a certain level of growth, which can then be used for applications such as biofuels and pharmaceuticals. The algorithm that the researchers developed utilised a type of evolutionary algorithm known as Ant Colony Optimization (ACO). These types of algorithms mimic the movements of ants moving between different nodes, with their movements being strongly influenced by simulated pheromones. By utilising this type of algorithm, Lincoln et al. (2015) were able to develop an algorithm that was able to simulate the behaviour of E. coli cells based on how they were configured.

> 'In this work, we propose a new method to identify the best sets of gene knockouts for metabolite production. Specifically, we combine Minimization of Metabolic Adjustment (MoMA) with ant colony optimization (ACO).' - Lincoln et al. (2015)

By implementing this kind of algorithm, the researchers were also able to introduce functionality similar to SDEs within their solution as ACO allows for an element of randomness within the algorithm due to the usage of simulated pheromones. Although this only occurs during the first few iterations of the algorithms, later iterations will begin to refine its search towards potential solutions. Incorporating functionality similar to SDEs in this solution would greatly benefit those who wish to use this solution as it is able to take noise within a biological system into account, meaning the generated solutions it offers will be more indicative of what will occur in the actual environment as opposed to if they had utilised ODE functionality. One of the decisions that Lincoln et al. (2015) made during their project was the implementation of a REDIS database. REDIS allows for data to be stored within an in-memory database by using key value pairs, which allows for much faster data access than other more traditional formats. Each time their algorithm produced a viable solution, this would be stored within a local instance of REDIS, which meant that the results for a previous experiment could be quickly accessed if the same configuration was encountered again. This allowed for greater optimisation within the solution by Lincoln et al. (2015) and greatly emphasises the importance of optimisation within these kinds of solutions when carrying out these kinds of complex calculations.

> 'The optional local search is performed to improve upon known solutions by locally searching the solution space and is usually included in most ACO algorithms in order to improve upon known good solutions.' - Lincoln et al. (2015)

After initial implementation, Lincoln et al. (2015) found that their solution was able to improve upon previous works with the solution being able to calculate E. coli configurations that resulted in better succinic acid production and growth rates than previous studies. One of the key areas that the researchers suggested could be improved during future work was to optimise the solution further to allow for more sophisticated parallelized functionality, which should allow for much larger scopes to work in for calculating more complex configurations of cells. Something to note about this project is that it utilises a dataset that contains various information regarding E. coli cells such as genes and potential reactions. Whilst this does allow for a solution can effectively calculate E. coli configurations, the solution requires other datasets to adapt it for calculating configurations for different kinds of cells. And even if a different dataset is utilised, this still means that the solution is only able to calculate viable configurations within the scope of pre-existing knowledge. As with Hallinan et al. (2014)'s usage of SVPs, this is something that can be potentially be expanded upon during this project via the usage of a rule-based language like Kappa rather than relying on existing datasets which would restrict the scope of potential solutions being produced by the implemented algorithm.

> 'The top five predicted best knockouts in five sets of twenty runs not only had a higher succinic acid production rate, but a higher growth rate as well when compared to reaction knockouts reported from Chong et al.' - Lincoln et al. (2015)

Overall, Lincoln et al. (2015) were able to produce a solution that can calculate potential configurations for E. coli cells that could be used for various applications. The solution produced by Lincoln et al. (2015) does also have the capability of being configured to use different datasets that document different kinds of cells, meaning that the solution is not strictly limited to carrying out calculations on just E.coli cell. One crutial decision that enabled Lincoln et al. (2015) to deliver their solution was having optimisation as one of their top priorities, as demonstrated through thoughtful usage of utilities such as the REDIS database package. This reinforces what was discussed for the project carried out by Hallinan et al. (2014) regarding optimisation being a top priority during this project. Lincoln et al. (2015) were able to demonstrate how making choices such as reading data from an in-memory source rather than from a database or file can greatly improve access times and therefore improve performance of the solution. This will be taken into consideration during the design stage of this project by designing a solution that avoids overly taxing actions during the algorithms main processes, such as opting to reduce the amount of file accessing.

A Genetic Circuit Compiler: Generating Combinatorial Genetic Circuits with Web Semantics and Inference - Waites et al. (2018)

During their project, Lincoln et al. (2015) opted to design and implement their own solution for running simulations for determining the most optimal configuration for E.coli cells. As previously discussed, their solution could potentially be expanded to use different datasets for different types of microorganisms, but what could be carried out within the simulation itself could be potentially limited. Developing new solutions that allow for far greater flexibility can be very time consuming and expensive to develop, hence why some researchers such as Hallinan et al. (2014) opt to use existing software to carry out their research. Researchers who work in this domain can greatly benefit from solutions that have been designed in mind for those with technical knowledge of biology but limited experience with programming languages. The project by Waites et al. (2018) aims to provide a solution for allowing researchers to carry out experimentation on genetic circuits that use high level language to allow for greater ease of use.

> 'Code generation from this high-level description to a low level language for simulation greatly reduces the scope for error in coding simulations' - Waites et al. (2018)

Waites et al. (2018) delivered on the aims of their project by developing a new language called Genetic Circuit Description Language (GCDL) that can be used by researchers to design and document genetic circuits in a structured and modular approach, which is compiled via their Genetic Circuit Compiler (GCC). The circuits defined in this language can then be parsed, transformed into the required format and ran through an SDE based simulator such as Kappa, which was used during the project. This functionality is achieved by utilising a dictionary which contains elements for determining how to reformat the syntax of the written GCDL code into a different modelling language such as Kappa, which means with additional work, this approach could be applied to many different simulation packages providing researchers in this area potential additional options. Waites et al. (2018) define their new modelling language with the intention of other researchers taking a more abstract approach when carrying out their research for creating and analysing new genetic circuits and to also help reduce the number of errors when simulations are carried out. Along with this, Waites et al. (2018) also advocates for including additional metadata with models constructed in this language and storing them in a repository of virtual parts, in a similar manner to how Hallinan et al. (2014) conducted their project.

> 'For efficiency, and economy of representation, we claim that the description of a computational model should include minimum information necessary for simulation. However, in order to use these models in an automated design processes, additional metadata, or annotations, about the

meaning of different modelling entities is needed.' - Waites et al. (2018)

Before commencing with a demonstration of the language, Waites et al. (2018) proceed to extensively describe the components and syntax that makes up GCDL including parts, promoters and methods for defining rules within a model. With the components of GCDL defined and documented, the team then demonstrates its usage by using the language to create a model that simulates mRNA molecules producing new proteins in response to a promoter being activated within a genetic circuit, the same genetic process that this project aims to refine via a genetic algorithm. Their solution then proceeded to translate this model from GCDL into the Kappa syntax where it was then simulated via the Kappa simulation software.

> 'Many compiler implementations are possible; ours innovatively combines the logical inference that is native to the semantic we with the use of templates to generate the target program.' - Waites et al. (2018)

Waites et al. (2018) were able to make a promising start to providing a system capable of delivering on their goals, however the researchers do go on to make several recommendations for future work. Following the initial implementation, Waites et al. (2018) noted that their solution requires additional validation to take place as the GCDL is translated into different formats to ensure the models being simulated are valid. For example, the researchers note that it was possible to create a model that utilised DNA with invalid gene sequence ordering. Waites et al. (2018) also note that their choice of modular abstract design, while it does allow for much greater ease of use, is unable to deliver on more complex biological processes and interactions, one example they cite was not being able to take the physical and chemical structure of DNA into account. Following discussion about these shortcomings, Waites et al. (2018) went on to discuss potential options for automated design and optimisation of genetic circuits in a manner that strongly resembles that of a genetic algorithm, something that this project seeks to address.

> 'A method for doing this, which we only sketch here, is to define a suitable fitness or distance measure on the output of simulations with respect to the desired specification.' - Waites et al. (2018)

Overall Waites et al. (2018), were able to deliver on a system that paves the way for a system that allows for abstract genetic circuit design, with potential compatibility with various software packages such as Kappa. The work that the researchers have carried out demonstrates that Kappa has the potential to form part of an automated process that could be used to refine searches for the most optimal circuit design as previously discussed. The work carried out by Waites et al. (2018) also makes a strong case for flexible design giving researchers more options when using their solution. Therefore, the solution developed during this project should aim to have a degree of flexibility that allows for different kinds of simulations and variables to be ran through it. For example, the user should be able to define their own set of variables with user defined tags and values that can be used to replace values in a template file. While Waites et al. (2018)'s solution uses a more general language that can be translated into other modelling languages, this project will only utilise one modelling language, the one used by Kappa, meaning the only validation that be required will be when a template file is ran through the simulation software, which should be handled by the software itself. In future work, as the scope of the genetic algorithms functionality potentially expands, more validation may be required to ensure that more complex templates generated by the algorithm are valid and therefore will not hinder the process. It may also become potentially possible to incorporate software packages that are able to take additional factors for genetic circuits into consideration such as their physical shape and chemical makeup. For this initial version of the algorithm, a simple template file will be used in order to evaluate the initial effectiveness of the algorithm to determine if such functionality has a place in the field of designing genetic circuits.

2.0.5 Conclusion

Carrying out this research has helped to establish what is currently available to researchers, what work has been carried out in related fields and the many challenges that face those who carry out these kind of projects. Initial findings during this stage suggest that is viable for genetic algorithms to be utilised for the optimisation of synthetic micro-organism configurations, provided that the noted aspects of such an algorithm are properly implemented and with effective optimisation. The insights gained will be a useful guide for driving the design decisions of the software solution that will be developed during the course of this project. As well as this, it may very well be possible to make improvements in areas that have been highlighted during this section, particularly in the way of utilising SDE for carrying out simulations and ensuring that effective optimisation is implemented into the solution.

Chapter 3

Design

With the literature review carried out, it was now possible to commence designing the solution that would aim to deliver on the goals of the project. As discussed previously, insight from the research will be utilised in order to design a prototype that delivers on certain requirements to ensure that it is able to sufficiently meet the goals of the project and to address additional points that were highlighted during the literature review. Carrying out this stage of the project should allow for smooth implementation by putting into place a suitable development plan as well as identifying potential pitfalls that may be encountered, such as identifying possible areas with a slow turnaround time in the code.

3.0.1 Functional and Non-functional Requirements

Before outlining what will take place during development, additional requirements will be defined in order to ensure that the solution is able to properly deliver on the goals of the project, to address any potential issues that previous work faced and to potentially expand on previous work. These requirements will be grouped into two separate sets of requirements, functional and non-functional requirements. These will be used to aid with design of the solution and will also be used during the conclusion of the project in order to evaluate the effectiveness of the developed prototype.

ID	Requirement	Notes	Priority
FR-1	Each individual in the Genetic Algorithm (GA) must have a representation that is reflective of a potential solution for cellular configuration and be compatible with Kappa.	This requirement concerns how configurable variables within the kappa template file will be represented in code format. This will likely take the form of an array in each individual that stores the tag within the template to be changed and the value it will be replaced with. For the implementation of this prototype, this will only concern two fields within the template file, the bonding and un-bonding rates of cells, but could be expanded in future work. Based on discussion of genetic algorithms and all related works during the literature review.	High
FR-2	Solution must be capable of generating a random set of individuals when initialising the GA, the number generated should be based on user specification.	This should occur at the beginning of an experiment within the solution, with values in the representation of a solution being randomly generated based on a user's defined template fields. For initial implementation, this will only concern the two discussed fields in FR-1, but should account for the possibility that users may wish to use different kinds of variables in their experiments. Based on discussion of genetic algorithms and on related work by Hallinan et al. (2014) and Lincoln et al. (2015) during the research section.	High

ID	Requirement	Notes	Priority
FR-3	GA must contain functionality for evaluating fitness of a potential solution that an individual presents by comparing the likeness of two sets of distributions.	This should be accomplished by applying statistical testing between two sets of data, which will be a set of Kappa simulation results from an individual and the user required protein output set. Different kinds of statistical testing will be considered when implementing this requirement. Based on discussion of genetic algorithms and on related work by Hallinan et al. (2014) and Lincoln et al. (2015) during literature review.	High
FR-4	GA should log which individual had the highest level of fitness for each generation.	This should allow users to evaluate the progress of each experiment with the algorithm and allows them to work with a set of data upon concluding.	Low
FR-5	GA must be capable of selecting individuals from each generation for crossover in a way that is reflective of their overall fitness.	The most likely candidate for selection within this function will be bias-roulette, where each individual has a chance of being selected with those having a higher fitness rating having a higher chance of being selected. This should, in theory, enable the algorithm to gradually work towards viable solutions that researchers may consider. The effectiveness of this functionality may be undermined if FR-3 is not properly implemented. Based on discussion of genetic algorithms and on related work by Hallinan et al. (2014) and Lincoln et al. (2015) during literature review.	High

ID	Requirement	Notes	Priority
FR-6	GA must be capable of carrying out crossover between two different individuals to generate a new individual that will be entered into the next generation of potential solutions.	Implementing this will allow the solution to generate new solutions based on which parents are selected during FR-5. This should mean that offspring solutions are generated from the solutions that were considered the most suitable in the current generation. Based on discussion of genetic algorithms and on related work by Hallinan et al. (2014) and Lincoln et al. (2015) during literature review.	High
FR-7	GA must allow for mutation in order to prevent total stagnation.	Implementing this should allow for a degree of controlled randomness in the experiment in order to reduce the likelihood that all individuals in a generation contain the exact same attributes. Based on discussion of genetic algorithms and on related work by Hallinan et al. (2014) and Lincoln et al. (2015) during literature review.	High
FR-8	GA should be capable of executing multiple simulations via Kappa and storing the results to an array for each individual.	As Kappa utilises SDEs, multiple simulations will be required in order to get a general idea of how an individuals configuration will perform. Based on discussion of biological simulation software and related work by Waites et al. (2018) during literature review.	High

ID	Requirement	Notes	Priority
FR-9	GA should be optimised to ensure that it can work towards potential solutions in a timely fashion.	The solution will likely have to work with many individuals in each generation, each requiring many simulations. If the solution is not properly optimised, it will slow to a crawl and will be unable to work within a suitable timeframe. This will likely be addressed by reducing the amount of file operations and implementing multithreading. Based on discussion of biological simulation software and works by Hallinan et al. (2014) and Lincoln et al. (2015) during literature review.	High
FR-10	Solution should allow for plotting distributions to charts.	Implementing this should allow researchers to visually evaluate the suitability of generated solutions from the algorithm and see how they compare to the specified required protein output.	Low
FR-11	Solution should be capable of reading from a kappa template file and updating values based on each individuals configuration.	Although the initial prototype will be developed using the same template file, it should account for the fact that researchers may wish to run different kinds of experiments during future use. This should be achieved by allowing the researchers to define which template to work with and which variables should be changed. Based on discussion of biological simulation software and related work by Waites et al. (2018) during the literature review.	High

ID	Requirement	Notes	Priority
FR-12	Solution should allow researchers to define their desired protein output from Kappa experiments.	It is likely that researchers will want to determine the optimal configurations of cells for different levels of protein outputs. During the implementation of the prototype, this will likely be achieved by allowing the user to enter their desired output as a variable. This is based on discussion of related work by Waites et al. (2018) during the literature review.	High
FR-13	GA should have configurable options to allow researchers to tailor fit the solution to their experiments.	Allowing users to configure settings within the GA will allow for configuring the solution to run different kinds of experiments and expands the scope for the hardware it can potentially run on. This will likely be achieved by allowing the user to adjust settings such as the genepool size and number of simulations ran for each individual. Based on discussion of related works by Lincoln et al. (2015) and Waites et al. (2018) during literature review.	Low
FR-14	Solution should allow users to define settings for Kappa to use each time an experiment runs.	Like FR-13, users should also be able to configure settings for Kappa simulations. This will likely be achieved by allowing the user to adjust settings such as the number of time points and exit condition for the simulation. Based on discussion of related work by Waites et al. (2018) during literature review.	Low

Table 3.1: Functional requirements for the developed solution

ID	Requirement	Notes	Priority
NR-1	Provide a solution that is able to determine potential cellular configurations based on researcher requirements.	This is the primary goal of the project. This should be achievable by addressing all of the high priority functional requirements in table 3.1.	High
NR-2	Provide a solution that allows researchers to evaluate their initial hypothesises before commencing work in the lab.	This requirement concerns issues raised during discussion of synthetic biology, such as costs and ethical obstacles. This should be achievable by providing a solution that is capable of working towards a potential solution that can be used as starting point when a researchers project progresses to the lab environment.	High
NR-3	Provide a solution that offers a level of flexibility that allows researchers to carry out different kinds of experiments with Kappa.	Whilst the overall objective of the project is to deliver a solution that can work towards potential configurations for achieving certain protein outputs, the possibility that this solution could be used for different kinds of Kappa based experiments should be taken into consideration. This could be achieved by offering flexible configuration options in the solution such as which fields in the template will be replaced and what they will be replaced by. Based on discussion of all related works during the literature review.	Low

ID	Requirement	Notes	Priority
NR-4	Provide a solution that does not require an extensive knowledge of programming to operate.	Although the solution will require some basic knowledge of programming, the more the solution is able to hide away and handle the more complex processes of the algorithms, the more accessible it will become to those working in the field of synthetic biology. Based on discussion of all related work during literature review.	Low

Table 3.2: Non-functional requirements for the developed solution

3.0.2 Kappa Template

In order to evaluate the viability of using the Kappa simulation software with a genetic algorithm, a basic template file will be utilised during the course of this project. The aim of this template file is to simulate the common molecular process of cells producing mRNA cells, which in turn produces protein, in response to a promoter within the molecular environment. Some of the key aspects that this template will use to simulate this process is the bonding and un-bonding between cells, the varying production rates of mRNA based on how the primary agent is currently being bonded to and cellular degradation. In regards to the genetic algorithm that will eventually work with this template, two variables will be modified for each individual, ka_on and ka_off, these refer to the previously described bonding and un-bonding rate. Should a viable solution be found within the genetic algorithm, these variables can be used to determine the required configuration of gene sequences within the cells that will be used to produce the required protein output. Lastly, the template file will be configured to return the total number of protein molecules within the simulation, which will be returned to the set of values for an individual in the genetic algorithm. By incorporating these aspects into the template, this should allow for simulations in Kappa that replicate the required biological process whilst allowing for modification that makes it compatible with the processes of the genetic algorithm. Implementing these should also contribute towards requirements FR-1, FR-8, FR-11 and FR-12 described in table 3.1 and NR-1 and NR-3 described in table 3.2.

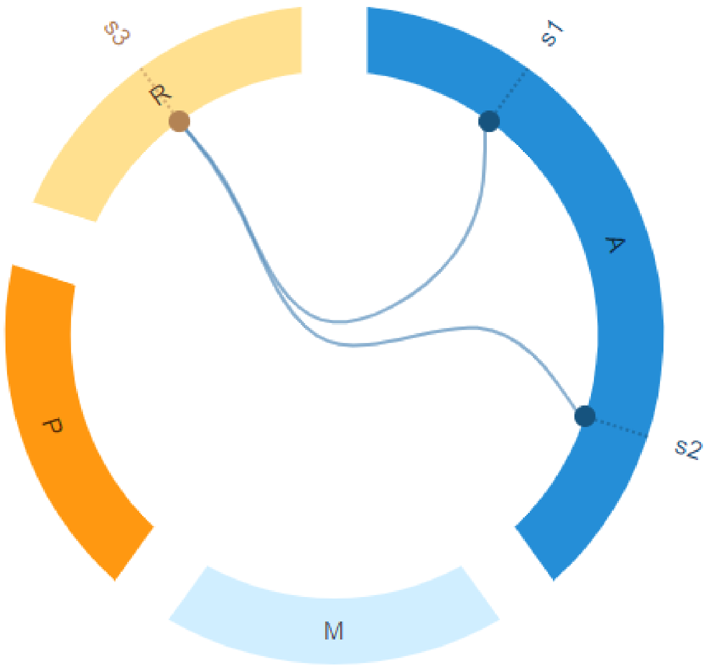


Figure 3.1: Generated contact map for template file. R (activator) will bond to A (promoter) which will then commence the production of M (mRNA) based on how many sites are currently bonded to. Produced M agents will begin output of P (protein).

3.0.3 Genetic Algorithm

The research carried out during the literature review helped identify key components that should be present in a genetic algorithm and allowed for drawing out the projected workflow that an algorithm such as this should undertake. This workflow is visually represented in figure 3.2. Each element within this workflow will now be discussed in additional detail in order to establish what exactly is required from each stage. Properly integrating each of these stages into the algorithm should contribute towards most requirements in table 3.1 and NR-1 in table 3.2.

Create X random individuals

At the beginning of the process, the algorithm should automatically create a number of individuals with a randomly generated set of variables, which will be required for the algorithm to begin its primary functionality. The settings for this stage should take

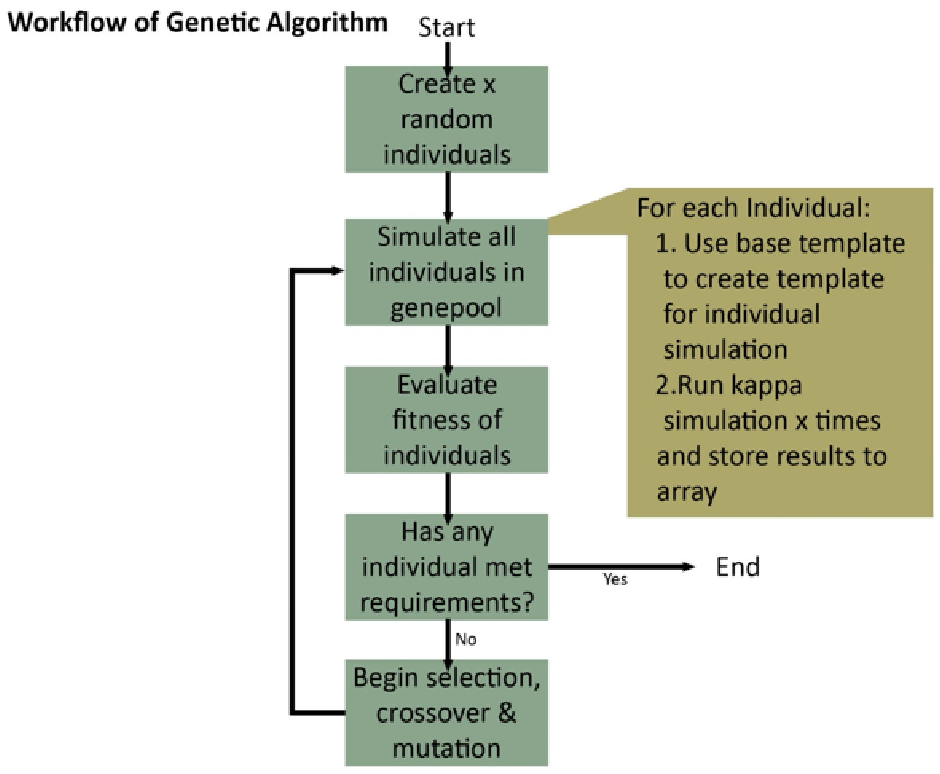

Workflow of Genetic Algorithm
Start
Create x random individuals
Simulate all individuals in genepool
For each Individual:
1. Use base template to create template for individual simulation
2.Run kappa simulation x times and store results to array
Evaluate fitness of individuals
Has any individual met requirements?
Yes
End
No
Begin selection, crossover & mutation

Figure 3.2: Proposed workflow for genetic algorithm

into account how many individuals the user requires (which should be configurable to accommodate for different hardware and time constraints), which values should be used to randomly assign to individuals and an array that these individuals can be stored into (hereby referred to as the genepool). Implementing this element will contribute to fulfilling FR-1 and FR-2 described in table 3.1 and NR-3 in table 3.2.

Simulate all individuals in genepool

At this point, the algorithm will enter into the first stage of the primary loop. At this stage of the process, the algorithm should carry out a number of simulations for each individual, which will involve modifying the base kappa template and updating it with the stored values within the individual. Following the completion of each simulation, the final count of protein cells present in the simulation should logged into an array of results within the individual, which will later be used to evaluate the overall fitness of the individual. Of all of the stages within the genetic algorithm, this stage will most likely require the most optimisation due to the large number of simulations that will take place. Failure to consider this will result in a sluggish algorithm and not fulfilling the requirement of FR-9 in table 3.1. Some additional settings that could be implemented for this stage includes the number of simulations that should be ran and the settings for kappa to use, including the exit condition. Implementing this stage into the algorithm should contribute towards requirements FR-8, FR-13 and FR-14 in table 3.1 and NR-2 and NR-3 in table 3.2.

Evaluate fitness of individuals

Once all simulations have been carried out, each individual should now contain a set of results of the final protein total from each simulation. It should now be possible to apply statistical testing in order to determine the similarity between these result sets and a set of data that contains the users required protein output, which should be a configurable setting. The result of this statistical testing should be converted into a value that sufficiently communicates the individuals overall fitness, which will be used for the next couple of stages in the algorithm. One setting that could be implemented for this stage of the algorithm is which type of statistical test to use. Its unlikely that one statistical test will be suitable for every kind of experiment using this solution, so multiple tests should be implemented to give the user a set of options when defining their own experiments. Implementing this stage should contribute towards fulfilling requirements FR-3, FR-4, FR-5, FR-12 in table 3.1 and NR-3 in table 3.2.

Has any individual met requirements?

At this stage, the algorithm should determine if a potentially viable solution been found based on user specification. This should be carried out before the genepool is repopulated with newly created individuals. Incorporating this stage will require a setting to allow the user to specify the minimum level of fitness that should be reached before returning an individual and exiting the primary loop. If this requirement is not fulfilled, then the algorithm will then commence with the next stage of the process. It could also be possible at this stage to take note of the individual within the current generation that had the highest level of fitness, even if it did not meet requirements, as this could be potentially useful information to the user. Implementing this stage of the algorithm should contribute requirements FR-4, FR-13 in table 3.1 and NR-1 and NR-2 in table 3.2.

Begin Selection, Crossover & Mutation

Should it be the case that a viable solution has not yet been found, the algorithm should then proceed to select individuals based on their level of fitness, cross them over to create new potential solutions and repopulate the genepool with these newly created individuals. Mutation should also take place here to determine if any individuals settings are randomly updated. Once this has been completed, the algorithm will then return to the start of the primary loop. One setting that should be implemented for this stage is the mutation rate, which should allow a user to control the likelihood that mutation will place. Incorporating this final stage of the process should work towards fulfilling requirements FR-5, FR-6 and FR-7 in table 3.1 and NR-1 and NR-3 in table 3.2.

3.0.4 Data Dictionaries

With each required stage of the genetic algorithm established, it was now possible to properly document which variables should be present in order for the algorithm to carry out its processes. Here, global variables for the entire process as well as variables for individuals will be documented including notes about their functionality and data type.

Variable	Type	Notes
Genepool	Array	Used to store all individuals, size based on setting of Genepool_Size. Contents of array will be refreshed each time crossover has completed in the algorithm.
Genepool_Size	Integer	Used to set how many individuals will be present in the Genepool array.
Min_Fitness_Threshold	Decimal	Used to set the exit condition for the genetic algorithm by determining if an individual solution has achieved a fitness equal or higher to this assigned value.
Genepool_Best_In_Generation	Array	Used to store the individual that obtained the highest level of fitness in a generation. Each entry will include the number of the generation the individual was present in and an instance of the individual object.

Variable	Type	Notes
Generation_Average_Fitness	Array	Used to keep track of the average level of fitness across each generation, this will be used during later stages of the project to evaluate performance of the algorithm.
Generation_Average_Time	Array	Used to keep track of the average turnaround time for simulating all simulations for individuals for each generation, this will be used during later stages of the project to evaluate performance of the algorithm.
Curr_Generation	Integer	Used to keep track of which generation the algorithm is working with. Will also be used to store individuals into the Genepool_Best_In_Generation array.
User_required_protein	Array	Used to store the set of results containing the users required protein output. Will be used during fitness evaluation and can potentially be used for plotting onto distribution charts for comparison with individuals.
Individual_test_count	Integer	The number of simulations that should take place for each individual in the genepool.
Mutation_count	Integer	Used to log the number of mutations that occur within each generation.
Kappa_exit_condition	String	Used to define when a simulation in Kappa will stop and return results. For example, stopping the simulation when more than 30,000 events have occurred.
Num_of_datapoints	Integer	Used with Kappa to determine when a new datapoint should be logged in the simulation. For example, for every 10 datapoints, log user defined variables to results.
Template_file_path	String	File location of Kappa template file to use for simulations within the algorithm.

Variable	Type	Notes
Num_values_to_change	Array	Used to store set of numerical variables that will be changed within the template and what to randomly assign to each individual. Formatting of objects in this array is [[VARIABLE_TAG],Min_Value,Max_Value].
Mutation_rate	Integer	Used to modify how likely a mutation will take place during the generation of a new Genepool, the smaller the value, the more likely a mutation will take place.
Fitness_Test_Setting	Integer	Used to determine which statistical test should be used during the genetic algorithm.

Table 3.3: Data dictionary containing global variables for genetic algorithm.

Variable	Type	Notes
ID	Integer	Used to assist with identifying each individual within the genepool. Will also be used for debugging.
Temp_values	Array	Used to store randomly assigned values to be used when updating the base template file for kappa and serves as the individuals representation. Formatting of objects within array is [VARIABLE_TAG,Assigned_Value].
Fitness	Decimal	Used to store the level of fitness for an individual following fitness evaluation within the algorithm. This will be used to carry out reselection.
Sim_results	Array	Used to store the final protein count for each simulation that takes place for the individual during the algorithm. This array will be used with User_requried_protein variable described in Table 3 in order to carry out statistical testing for fitness evaluation.

Table 3.4: Data dictionary containing variables for individuals in the genetic algorithm.

Chapter 4

Implementation

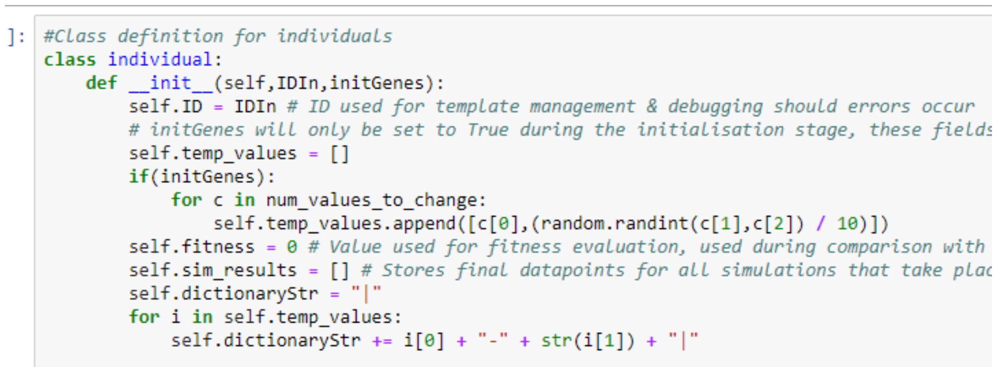
With further requirements established and the design in place for the proposed solution, it became possible to begin implementing the genetic algorithm to address the objectives of the project. To achieve this, the solution was developed using the Python programming language. Python provides a flexible set of utilities and is well suited to carrying out high performance tasks, something that will certainly be a factor as development moves towards performing simulations using Kappa. This section of the report will focus solely on documenting the development of required features discussed during previous chapters, where issues were encountered and what was carried out to address these unexpected challenges.

4.0.1 Global Settings

Work began by implementing all of the required variables that were discussed in table 3.3, as these needed to be present before work could begin proper with the genetic algorithm as a considerable amount of processes within the algorithm rely on these variables being present. These have been included towards the top of the file as this will allow users to immediately adjust the settings of the solution to meet their needs. These settings are also kept in one single place to aid with ease of access as it will not require significant navigation around the solution to configure the simulation. Implementing these variables should address requirements FR-12, FR-13 and FR-14 in table 3.1 and contribute towards addressing NR-3 found in table 3.2.

4.0.2 Individual Class

Following on from the implementation of the global classes, the focus then shifted towards implementing the class that will be used to represent individual solutions within the algorithm. This involved implementing the variables described in table 3.4. With the implementation of these variables, potential solutions could now be represented in code format within the algorithm, which should allow for carrying out the functionality expected from a genetic algorithm. As well as implementing these variables, functionality has been included to allow for automatic generation of an individual with randomised values, which are defined in the num_values_to_change global setting and will be used before proceeding into the primary loop of the genetic algorithm. This functionality can also be overridden so random values wont be created, which will be utilised during the implementation of crossover within the algorithm, avoiding unnecessary initialisation of template values during crossover which should aid with optimisation. To aid with debugging and providing feedback to the user, additional functionality was defined in order to easily display the values of individuals within the Genepool, either individually or for the entire contents of the array. The implementation of these variables should address requirements FR-1 and FR-2 in table 3.1.



```
]: #Class definition for individuals
   class individual:
       def __init__(self,IDIn,initGenes):
           self.ID = IDIn # ID used for template management & debugging should errors occur
           # initGenes will only be set to True during the initialisation stage, these fields
           self.temp_values = []
           if(initGenes):
               for c in num_values_to_change:
                   self.temp_values.append([c[0],(random.randint(c[1],c[2]) / 10)])
           self.fitness = 0 # Value used for fitness evaluation, used during comparison with
           self.sim_results = [] # Stores final datapoints for all simulations that take plac
           self.dictionaryStr = "|"
           for i in self.temp_values:
               self.dictionaryStr += i[0] + "-" + str(i[1]) + "|"
```

Figure 4.1: Implementation of the individual class.

4.0.3 Genetic Algorithm Utilities

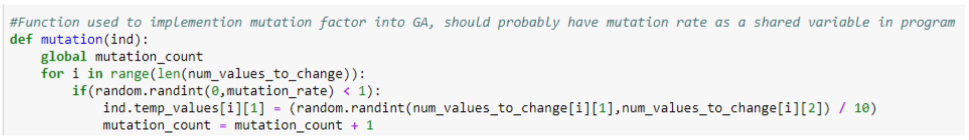
With the individual class being defined within the solution, work could now begin on developing the genetic algorithm. Before implementing the algorithm itself, functionality that will be used during the algorithm will be defined, which would involve implementing the aspects that genetic algorithms contain as discussed during chapter 2.

Fitness Evaluation

The first function that was implemented was the calculateFitness function, which will be used by the algorithm to evaluate the overall fitness of an individual potential solution when compared to the values defined by the user in the user_required_protein global variable. The aim of this function is to carry out some form of statistical test using both the returned results for an individual from Kappa and the user defined protein values in order to determine how similar both sets of distributions are. The more similar these two distributions are, the higher the fitness value should be. The proper implementation of this function is crucial as it will be used to both provide the user some indication of how suitable an individual configuration is for addressing their needs and the generated fitness value will also be used during other processes within the algorithm such as reselection. To experiment with different statistical tests and to provide users additional options, three different kinds of tests were implemented into the function. These include two sample t-test, chi-squared and Kullback-Leibler Divergence. A user can select which statistical test to use by using the fitness_test_setting global variable. Each of these implemented functions will attempt to format and return a fitness value that can be easily understood by the user and can be used for reselection. In order to aid with avoiding stagnation of solutions during usage of the algorithm, the minimum value that can be returned from these functions is 1, this will allow for some small possibility that an individual with a low level of fitness can still be chosen during reselection. The implementation of this function should address FR-3 in table 3.1 and should contribute towards addressing NR-1 in table 3.2.

Mutation Function

The next function that was implemented was the mutation function, which will be used following the creation of a new individual via crossover. For each setting an individual has, a check will be ran to determine if that setting is randomly reinitialised. This check is carried out by generating a random number between 0 and a user defined value i.e. 100. If that number is less than 1, then the value will be reinitialised. Implementing this functionality introduces a much needed aspect of genetic algorithms as this should aid with preventing stagnation of potential solutions within the algorithm. Implementation of this function should address FR-7 in table 3.1.



```
#Function used to implemention mutation factor into GA, should probably have mutation rate as a shared variable in program
def mutation(ind):
    global mutation_count
    for i in range(len(num_values_to_change)):
        if(random.randint(0,mutation_rate) < 1):
            ind.temp_values[i][1] = (random.randint(num_values_to_change[i][1],num_values_to_change[i][2]) / 10)
            mutation_count = mutation_count + 1
```

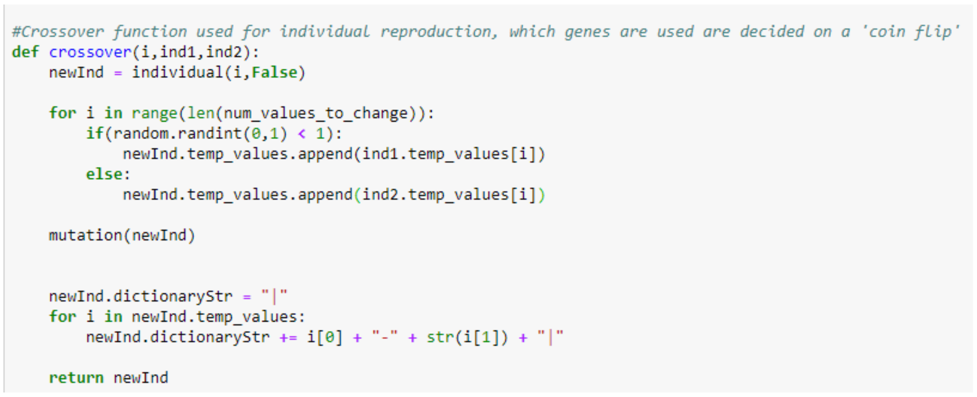
Figure 4.2: Implementation of mutation function.

Individual Crossover

Following on from mutation, the next set of functionality that was implemented was the crossover function. Here, the values of two individuals will be crossed over with each other in order to generate a new individual, which should contain the settings of either one or both of its parents. To implement this, two individuals are passed into the function, which is then followed by the creation of a new child individual. To determine which value should be taken from a parent for each setting, a random integer is generated of either 0 or 1, which is then used to determine which parent to take a setting from. Following on from this, the mutation function is ran on the newly generated individual and is then used as the returned value of the function. The implementation of this function should address FR-6 in table 3.1.

Reselection Functionality

The last of these functions to be implemented was the selectionAndReproduction function. Here, a new genepool should be generated via a process of reselection and crossover. To begin with, the function will generate a new array that will be used to implement bias roulette functionality into the genetic algorithm. This was done by adding a number of instances of each individual from the genepool based on the fitness value that is generated by the calculateFitness function. For example, if individual A has a fitness of 65, 65 instances of that individual should be appended to the roulette array, whereas individual B with a fitness of 1 only gets added once. This means that individuals with a higher fitness value should have a greater chance of being selected for crossover. The roulette array is then used to randomly select individuals to carry out crossover with in order to generate new potential solutions. These newly created individuals are then passed into an array that will be used to create a new genepool for the next generation of individuals. The implementation of this function should address FR-5 in table 3.1.



```python
#Crossover function used for individual reproduction, which genes are used are decided on a 'coin flip'
def crossover(i,ind1,ind2):
    newInd = individual(i,False)

    for i in range(len(num_values_to_change)):
        if(random.randint(0,1) < 1):
            newInd.temp_values.append(ind1.temp_values[i])
        else:
            newInd.temp_values.append(ind2.temp_values[i])

    mutation(newInd)


    newInd.dictionaryStr = "|"
    for i in newInd.temp_values:
        newInd.dictionaryStr += i[0] + "-" + str(i[1]) + "|"

    return newInd
```

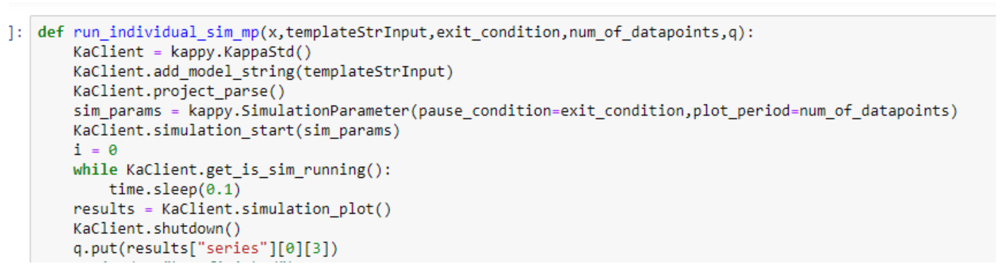
Figure 4.3: Implementation of crossover function.

4.0.4 Kappa Integration

The next set of functionality that was implemented was integrating Kappa into the solution. First, the Kappa wrapper for python, Kappy, was imported into the project. This allows for initialising Kappy objects within a Python program, that enables the solution to pass in a Kappa script into the object as a string and define additional simulation parameters such as the exit condition. This object can then be used to execute a simulation which will return a dataset containing a time-series of variables defined in the script. In the scope of this project, for each individual, the solution will read the base template from a kappa file, replace any tags within the script with the values defined for the individual and then pass this updated script into the Kappy object.

Integrating Kappy into the solution allowed for reducing the amount of file accessing when building a new template, as the only time reading a file is necessary during this process is when the base template needs to be loaded into python, which will only occur once before the primary loop of the algorithm begins. Using Kappy also means its not required to read output files from Kappa as the data from these simulations are returned as collections within Python. In order to evaluate the overall effectiveness of using Kappy, a less efficient function was also integrated into the solution. Here, a Kappa executable would be invoked via system commands and requires that additional template files are written in order to carry out simulations for each individual. The expectation was that the Kappy integration would be far more efficient than this second method due to less file accessing being required and also due to the streamlined integration into Python. The overall effectiveness of this integration will be evaluated in the next chapter. Otherwise, the integration of Kappa in the solution should address

FR-8 and FR-11 in table 3.1 and should contribute towards addressing NR-2 and NR-3 in table 3.2.



```
]: def run_individual_sim_mp(x,templateStrInput,exit_condition,num_of_datapoints,q):
       KaClient = kappy.KappaStd()
       KaClient.add_model_string(templateStrInput)
       KaClient.project_parse()
       sim_params = kappy.SimulationParameter(pause_condition=exit_condition,plot_period=num_of_datapoints)
       KaClient.simulation_start(sim_params)
       i = 0
       while KaClient.get_is_sim_running():
           time.sleep(0.1)
       results = KaClient.simulation_plot()
       KaClient.shutdown()
       q.put(results["series"][0][3])
```

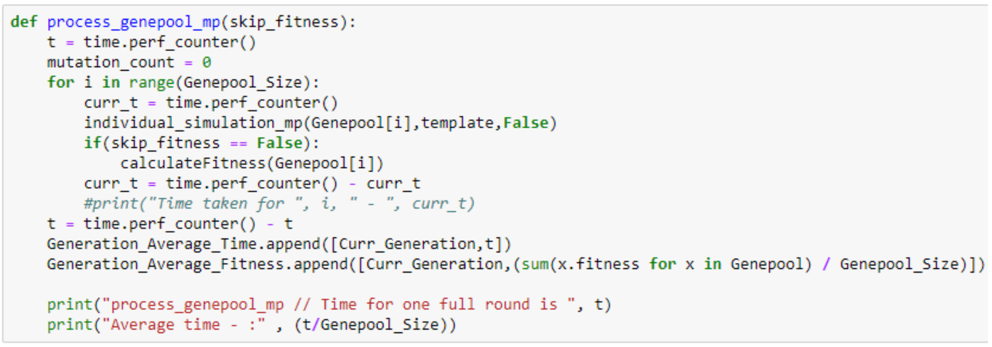
Figure 4.4: Implementation of kappa within the primary loop.

4.0.5 Primary Loop

With all of the required elements for the genetic algorithm being present and with Kappa functionality integrated, attention then turned towards implementing the primary loop of the algorithm. It is here that during each generation, all individuals will have a set number of simulations carried out in Kappa, be evaluated via the fitness function and then, provided an optimal solution has not been found, the generation of a new genepool will take place using the reselection, crossover and mutation functionality. Before generating a new generation, the individual with the highest level of fitness will be logged into the Genepool_Best_In_Generation array, which will allow the user to keep track of how an experiment has progressed and will address FR-4 in table 3.1.

This section of the solution will most likely be the most resource intensive section due to the large number of simulations being carried out for each individual in the genepool. To address this, multiprocessing was implemented into the section of code that runs through the user defined number of simulations. Here, sub-processes will be created for each individual simulation required. Upon completion of a simulation, each sub-process will read the final count of protein agents within the simulation and pass it back into a queue object, which will then be converted into a collection and stored within the individuals sim_results variable.

Implementing multiprocessing into this section of the solution should provide a much quicker turnaround time for compiling a set of simulation results that can then be evaluated against the user_require_protein global setting during the fitness function. Although it is possible that the effectiveness of this functionality could be limited by the available hardware that the solution is ran on, therefore, this solution will be evaluated on a system that utilises a multi-core processor in order to get the best performance with multiprocessing. To further evaluate the effectiveness of implementing multiprocessing into this function, a second function was integrated into the solution that will carry out one simulation at a time rather than assigning each to a separate sub-process. The expectation here is that the function integrating multiprocessing will have a much quicker turn-around time as it will be able to take full advantage of a multi-core processor in order to carry out multiple simulations simultaneously. The integration of this functionality should address FR-8 and FR-9 in table 3.1.



```
def process_genepool_mp(skip_fitness):
    t = time.perf_counter()
    mutation_count = 0
    for i in range(Genepool_Size):
        curr_t = time.perf_counter()
        individual_simulation_mp(Genepool[i],template,False)
        if(skip_fitness == False):
            calculateFitness(Genepool[i])
        curr_t = time.perf_counter() - curr_t
        #print("Time taken for ", i, " - ", curr_t)
    t = time.perf_counter() - t
    Generation_Average_Time.append([Curr_Generation,t])
    Generation_Average_Fitness.append([Curr_Generation,(sum(x.fitness for x in Genepool) / Genepool_Size)])

    print("process_genepool_mp // Time for one full round is ", t)
    print("Average time - :" , (t/Genepool_Size))
```

Figure 4.5: Implementation of the main function used during the primary loop.

4.0.6 Dictionary Optimisation

After initial testing of the primary loop of the genetic algorithm, it became clear that further optimisation would be required in order to make the solution viable with a larger number of individuals in the genepool. Before further optimisation, it would take an average time of 4.5 seconds to carry out 100 Kappa simulations per individual. Whilst this is fairly quick given the complex nature of the Kappa simulations and considering the number of simulations required, the total amount of time to process a generation of individuals quickly added up. Initial testing showed that to process a generation of 100 individuals, it would take an average time of 7.5 minutes. To address this, the solution was examined to determine where further optimisation could be implemented in order to improve performance.

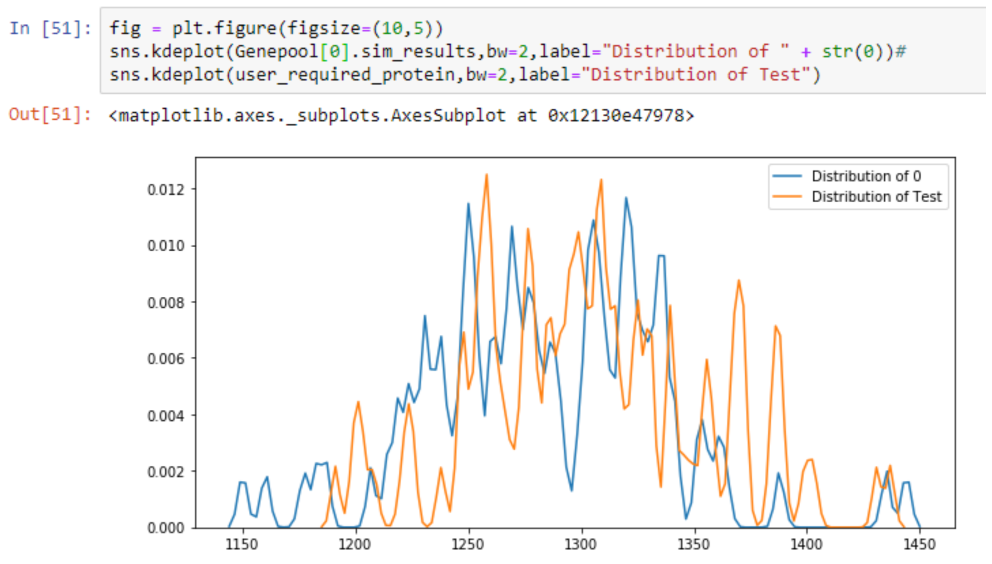
After examination of the code, a question was raised that could potentially pave the way to better performance, once an individual configuration has been simulated in Kappa, does it necessarily need to be simulated again? Once an individual has completed its simulations, the results could be stored into a dictionary, where the data can be quickly accessed again should the same configuration be encountered in other generations, without the need to rerun Kappa to evaluate it. The thought process for implementing this dictionary was inspired by Lincoln et al. (2015)'s usage of REDIS in their project.

However, something that needs to be taken into consideration before implementing this functionality is the fact that Kappa utilises SDEs to process its simulations, which as previously discussed, will mean that each simulation will return different results due to simulated background noise. One potential solution for addressing this concern would be to run a large number of simulations to establish a far more general idea of how a certain configuration would perform. However, considering the hardware limitations of the environment the algorithm will operate in, this solution is unfeasible. The next solution that was considered was to limit which configurations are re-simulated through Kappa. A new global variable could be implemented into the project that would allow users to configure the minimum fitness threshold for determining if a configuration should be simulated again. For example, if the user were to set this value to 60.0, any configuration that met or exceeded this would be simulated again, otherwise the previous results would be obtained from the dictionary and simulation would be skipped.

This solution provides a reasonable compromise between improving turn around time in the solution and the nature of SDEs in Kappa. It should also allow users to have further control over which configurations they can discount from their research, as they can essentially skip over configurations that have a low level of fitness when evaluated against their required distribution of protein. These changes were integrated across the solution, and initial testing showed that performance would improve as the solution progressed through each generation. These changes will be discussed further in the next chapter. The implementation of this optimisation should further address FR-9 and FR-13 in table 3.1.

4.0.7 Distribution Plotting

With all of the key functional requirements for the genetic algorithm in place, additional functionality could be implemented into the solution to aid researchers with their evaluations. This was done by implementing functionality that would allow researchers to compare the distribution of results from an individual against the distribution of required protein as specified by the user. It is recommended that users use this functionality in conjunction with other functions, such as displaying the details for all individuals within a generation when determining which configurations they should compare to their required protein output. Functionality for showing every distribution from a generation was explored, however it became clear that this functionality would be useless for users due to the sheer number of distributions being present on a single plot. The implementation of this functionality should address FR-10 in table 3.1 and should contribute towards addressing NR-2 in table 3.2.



```
In [51]: fig = plt.figure(figsize=(10,5))
         sns.kdeplot(Genepool[0].sim_results,bw=2,label="Distribution of " + str(0))#
         sns.kdeplot(user_required_protein,bw=2,label="Distribution of Test")

Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0x12130e47978>
```

Figure 4.6: Implementation of distribution comparison functionality within the solution.

Chapter 5

Evaluation

With all of the required functionality implemented into the developed solution and with all functional requirements from table 3.1 addressed, focus could then shift towards evaluating the solution that was developed during this project. During this section of the report, the developed solution will be evaluated against the original goals of the project and against the additional requirements that were identified following analysis of related literature. Any shortcomings that were encountered during evaluation will be noted and any future work that could address them will be discussed in the next chapter.

5.0.1 Performance Evaluation

Before evaluating the suitability of the developed genetic algorithm, the performance of the solution will first be evaluated. If the solution is not able to operate within an acceptable time frame, it will become unviable for anyone wishing to use this solution to evaluate their initial hypothesis before progressing their research into the lab, which would fail to address one of the primary goals for this project.

Kappa Wrapper Vs. No Wrapper

Testing began by evaluating the effectiveness of implementing the Kappy wrapper into the solution against functionality that carried out simulations by using system commands to trigger a Kappa simulator. To evaluate the difference in performance between these two different ways of implementing Kappa into the solution, two performance tests were carried out, each using their respective method for executing simulations. For each of these tests, the solution would process a genepool with 10 individuals with

100 simulations being carried out one at a time for each one, totalling 1000 simulations. Once an experiment had been completed, the solution would take note of the time taken to complete each of these tests and the average amount of time to process each individual. The expectation here was that the function using the Kappy wrapper would yield a much faster turnaround time due to its streamlined integration into Python and requiring no file operations for reading and writing Kappa files. These initial tests were carried out on a windows based system that utilised an Intel i7-6700 CPU with 4 cores and a clocking speed of 3.40 GHz.

The results of this experiment showed that the Kappy function was able to process its workload at a significantly faster rate than the function that used system commands to interact with a Kappa executable file. The experiment showed that the Kappy function was able to process a total of 1000 simulations in 161.18 seconds with an average of 16.11 seconds for each individual. Whereas the non-wrapper function took a total of 295.76 seconds with an average of 29.57 seconds for each individual. These results have been visualised in figure 5.1 alongside the results of the next experiment. The results of this experiment demonstrate how the implementation of Kappy within the solution can yield a greater turn around time for processing individuals and was able to achieve a speedup factor of 1.83 over the non-wrapper function. The results of this experiment are likely due to Kappy allowing for a streamlined implementation into Python which removes the requirement for writing additional Kappa template files, reading from the output files and allows data from simulations to be immediately returned as data collections within Python.

Serial Vs. Multi-Processing

Whilst the results of the previous experiment demonstrated that implementing Kappy into the project resulted with an initial improvement in performance, the average turn around time of 16.11 seconds showed there was still much room for improvement. This is likely due to the fact that each of these functions were implemented as serial functions and were therefore not taking full advantage of the systems available resources, particularly that of the multi-core processor. To evaluate the potential benefit from implementing multi-processing into the solution, another test was devised. Here, like the previous test, two functions would be evaluated for how quickly they could process 10 individuals with 100 simulations each. Both functions would utilise Kappy, discussed in the previous experiment, with one function using serial-based functionality and the other utilising multi-processing, with a process being generated for each required simulation. To carry out this experiment and make use of this multi-processing functionality within Python, the solution was transferred to a Linux based system that utilised an Intel i7-4770 CPU with 8 cores and a clocking speed of 3.40 GHz. The expectation from this experiment was that the multi-processing function would greatly improve the turn-around time for processing individuals when compared to the serial-based function. It was also expected that, even with improved hardware, the performance of the multi-processing function would likely max out CPU resources due to the sheer amount of sub-processes that would be created when the test began, possibly limiting its potential turnaround time.

The results of the experiment showed that the multi-processing function was able to process its workload at a far greater rate than the function that only utilised serial functionality. The experiment showed that the multiprocessing function was able to process its entire work load in 49.77 seconds with an average time of 4.97 seconds for each individual. Whereas the serial function was only able to process its workload in 163.3 seconds with an average time of 16.33 seconds for each individual. These results have been visualised in figure 5.1 alongside the results of the previous experiment. The results of the experiment demonstrate how the implementation of multi-processing within the solution has continued to improve performance with the multi-processing function achieving a speedup factor of 3.28 over the serial function. The results of this experiment are likely due to the multi-process function being able to take full advantage of the CPU resources available to it. The usage of better hardware will have also helped with achieving a faster turn-around time. Something to note from this experiment was that the multi-processing function did indeed max out the CPU when this experiment was underway, which suggests that with more optimal resources, such as a distributed computing system, the solution could potentially achieve an even greater turn-around time.

Dictionary Optimisation Evaluation

The last experiment that was carried out to evaluate performance of the solution was tracking how long it would take the solution to process a series of generations of individuals following the implementation of dictionary optimisation. Here, the solution would be set to process 10 generations of 150 individuals with 100 simulations each, a total of 15000 simulations during each generation. The solution was configured to only retest configurations if their fitness value exceeded or met 60, which would be

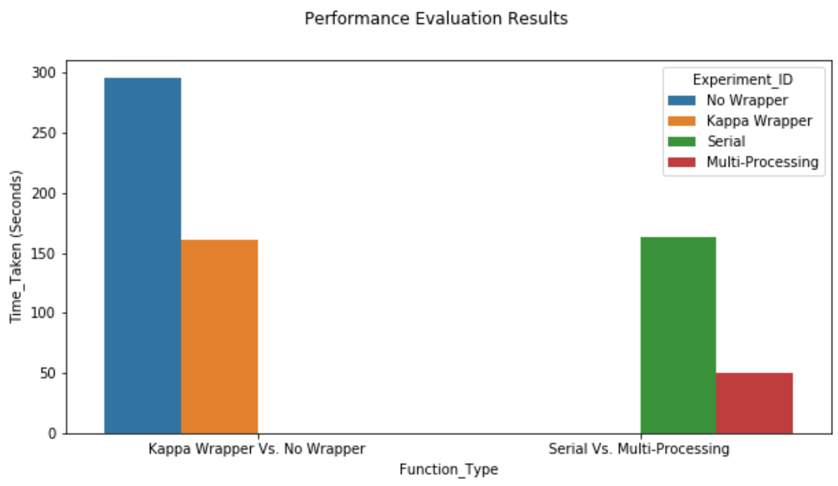


Figure 5.1: Visualisation of results from Kappy and multi-processing experiments.

calculated by utilising chi-squared during fitness evaluation. The expectation from this experiment was that initially, it would take a few minutes to process the first few generations, and as the solution would progress through each new generation, the performance would gradually improve as the solution should skip over configurations that are considered unsuitable based on user settings.

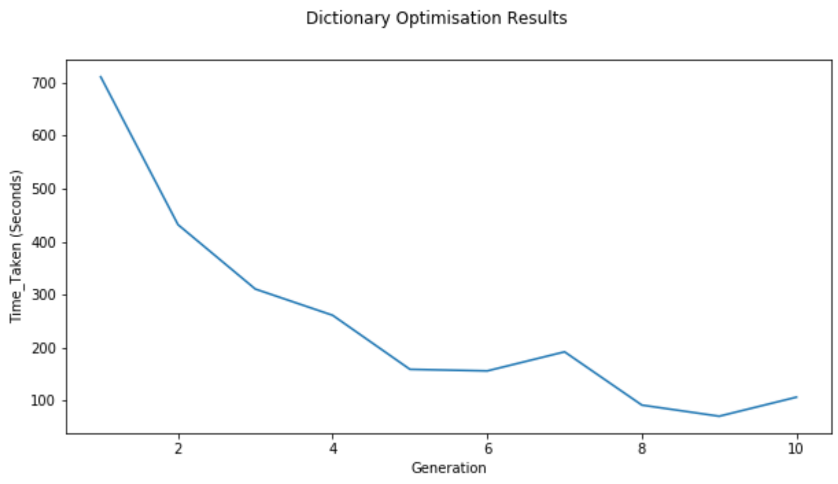


Figure 5.2: Visualisation of results from the dictionary optimisation experiment with a retest threshold of 60.

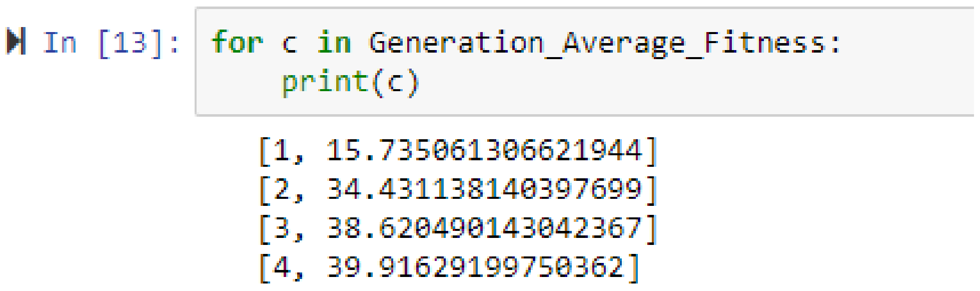
The results of this experiment showed that performance significantly improved over time as the solution worked through each generation. The first generation during the experiment took a total of 711 seconds to process all 150 individuals, and by the time the experiment had progressed to the tenth generation, the time taken to process all individuals took a mere 106.5 seconds to process, a speedup factor of 6.67. There was some fluctuation of time taken during some generations, this is likely occurring as the solution may be evaluating new configurations created via crossover and mutation and may also be re-evaluating configurations that exceeded a fitness of 60. The gradual improvement in performance becomes more apparent when these results were plotted out onto a graph, which can be seen in figure 5.2 and further highlights how the number of new and re-evaluated configurations likely varied during the course of the experiment. The results of this experiment illustrate how effective the implementation of a results dictionary within the solution is, which allows for working towards delivering a software solution that can enable researchers to evaluate their initial hypothesis within a reasonable time-span. With this final set of results, the criteria of FR-9 in 3.1 can be considered as firmly addressed during this project.

5.0.2 Genetic Algorithm Evaluation

With the performance of the solution evaluated and deemed as working at an acceptable manner, it was now possible to begin evaluating the effectiveness of the genetic algorithm that has been developed for this solution. For this series of experiments, the genetic algorithm was tested to determine how effective it is at working towards a required protein output, by using a combination of crossover and mutation, to produce a solution that meets user requirements. This will be achieved by the genetic algorithm by utilising the distribution sets that are generated for each individual after a number of simulations have been carried out on each configuration. These distributions represent the final total of protein molecules present when a simulation was terminated, these will be compared against the user_required_protein distribution set during fitness evaluation. Additional experiments will also be carried out to determine how adjusting certain settings within the solution can affect the genetic algorithms ability to work towards a solution. For each of these experiments, the settings of the algorithm has been adjusted to determine if the genetic algorithm is capable of returning a solution that meets or exceeds a fitness value of 70, a value which should be considered to be good starting point for achieving a certain protein output whilst allowing for some variation given the nature of SDEs being utilised by Kappa. The genetic algorithm has also been adjusted to work towards a maximum of 30 generations, as it is possible that some experiments may be unable to return a suitable configuration due to the settings for that experiment, which would mean that the genetic algorithm would run indefinitely. The results for all of these experiments can be found in table 5.2.

Control Experiment (CE)

Before commencing with experiments that revolved around adjusting one setting of the solution, a control experiment was first carried out in order to determine how the genetic algorithm would perform using settings that could be considered the default settings of the algorithm. For example, having a reasonably large genepool size of 150 and a mutation rate of 100. The full details for the settings used for the control experiment can be found in table 5.1 with additional details for these settings found in table 3.3. The results of this control experiment will be compared against later experiments which should allow for a better idea for how certain adjustments affect the performance of the genetic algorithm. The expectation here is that the algorithm should be capable of returning a configuration that meets fitness requirements within an acceptable timeframe, ideally under 15 generations.



```
In [13]: for c in Generation_Average_Fitness:
             print(c)

    [1, 15.735061306621944]
    [2, 34.431138140397699]
    [3, 38.620490143042367]
    [4, 39.91629199750362]
```

Figure 5.3: Collection of average level of fitness of each generation during the control experiment.

Setting	Value
Genepool_Size	150
Retest_Threshold	50.0
Individual_test_count	100
Kappa_exit_condition	’[E] >35000’
Num_of_datapoints	10
Num_values_to_change	[[”[ON_VAR]”,0,150],[”[OFF_VAR]”,0,150]]
Mutation_rate	100
Fitness_test_setting	2 (Chi-Squared)

Table 5.1: Details of all settings used for the control experiment.

The results from this experiment showed that the algorithm was capable of finding a configuration with a fitness value of 71.25 within four generations, greatly exceeding the expectation for how many generations would be needed to find a solution. Additional information from the experiment showed that the average fitness for each generation gradually improved over the course of the experiment, with the largest gain between the first and second generations. The results from this experiment demonstrate that the genetic algorithm is capable of working towards a potential viable genetic configuration within a reasonable time frame. Something to note is that it is possible that the number of generations used to find a configuration with these settings could potentially vary due to the nature of SDEs used by Kappa and random factors within the algorithm affecting individuals such as mutation and reselection.

Genepool Size Experiments (GS)

The first series of experiments that took place following the control experiment were carried out by utilising the Genepool_size setting within the algorithm. This setting will determine how many individuals will be present within the genepool in each generation, which will affect the number of available configurations that the algorithm can work with. Each of these experiments will work with a different number of individuals in the genepool, working in increments of 25. The expectation for these experiments is that the experiments using larger genepool sizes will be more likely to find a configuration that meets user requirements as they will have more potential configurations to work with than the experiments that used a smaller set of individuals. It is also likely that experiments utilising larger genepools will take much longer to process due to the dictionary optimisation taking a longer period of time before it becomes effective due to the greater variation in available configurations.

The results from these experiments showed that experiments with a genepool size of 125 or higher were generally capable of finding a viable configuration in less than 10 generations, whilst experiments with a genepool size of 100 or less were far more likely to reach 30 generations of configurations before they were able to find a suitable configuration. Plotting both the average and highest fitness for each generation during these experiments, as seen in figures 5.4 and 5.5, demonstrate that experiments with a genepool size higher than 100 were able to gradually work towards better individuals, whilst experiments with a size of 100 or less were more likely to result in an erratic genepool as the experiment progressed. Comparing these results to the control experiment show that experiments utilising larger genepool sizes were able to gradually improve the average fitness of their configurations in a similar manner to the control experiment, with some variation. The likely reason for experiments with a larger genepool size being successful is possibly because the genetic algorithm had far more configurations to work with, which will had led to a larger variety of potential configurations being simulated through Kappa. It is also likely that a larger genepool size also increased the likelihood of mutations taking place, thus further increasing the variety of possible configurations being simulated. The findings from this set of experiments could be acted upon during future works by developing the algorithm further so that it is more capable of working with much larger genepool sizes, i.e. enabling the algorithm to function on a distributed computer system to more effectively process larger sets. It's likely that the requirement for providing a solution that is capable of processing much larger genepools will become more prevalent should this solution be proposed for experiments that utilise individuals with far more variables used in simulations, i.e. multiple bonding rates or working with standard parts.

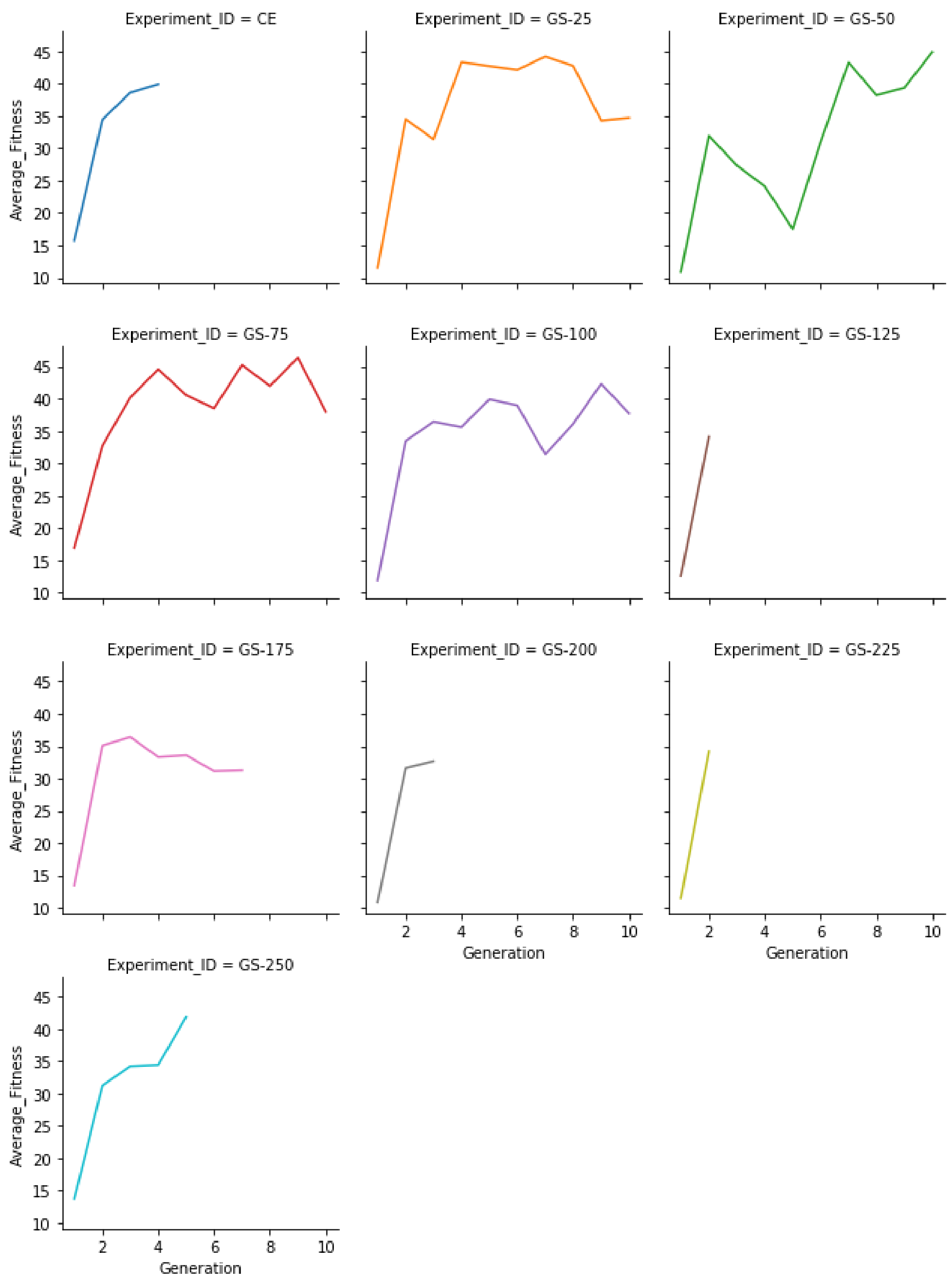


Figure 5.4: Progression of average level of fitness during the first 10 generations of the genepool size experiments.

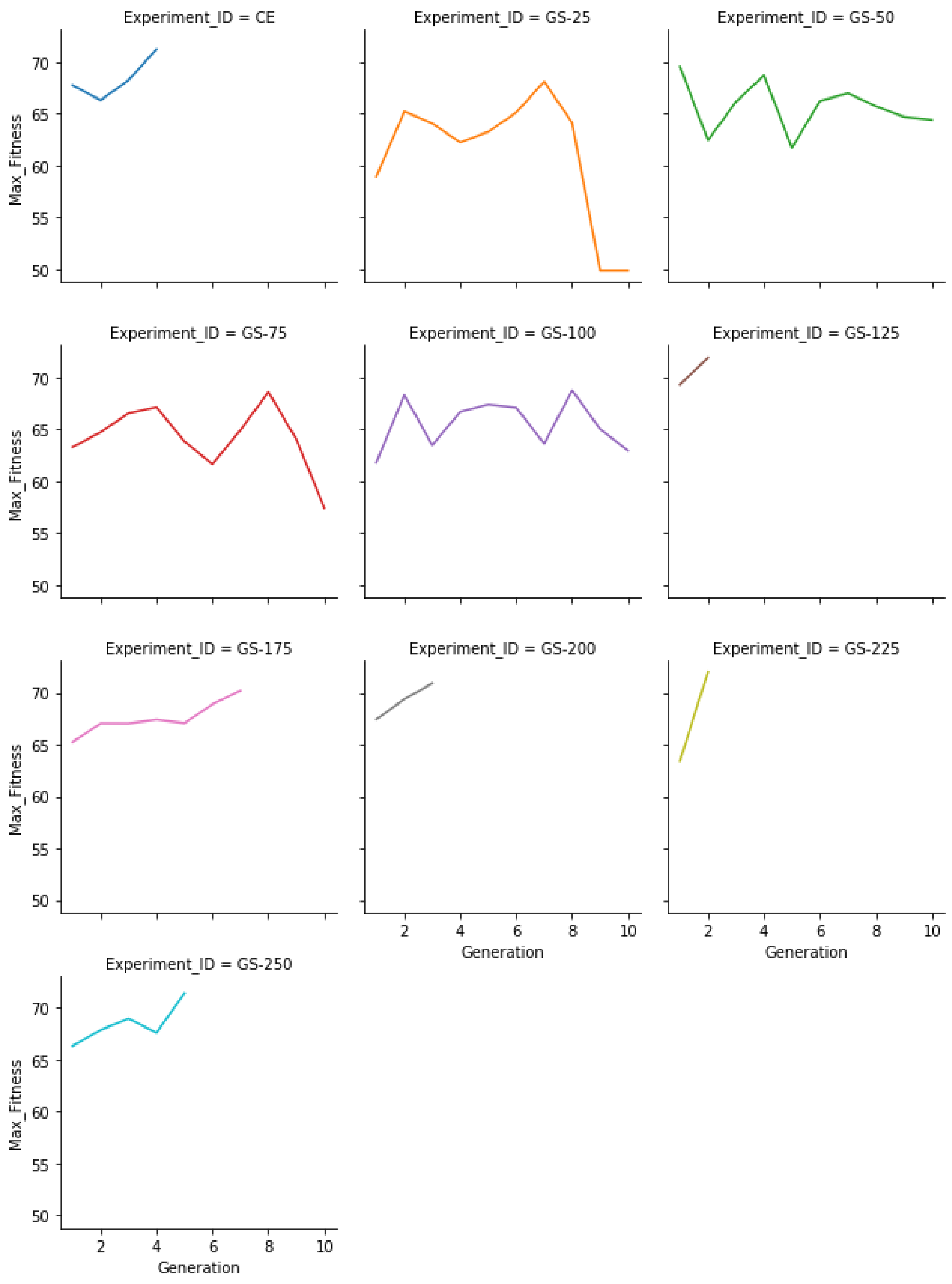


Figure 5.5: Progression of highest level of fitness during the first 10 generations of the genepool size experiments.

Mutation Rate Experiments (MR)

The next series of experiments that were carried out made use of the Mutation_rate setting within the algorithm. This setting determines how likely mutation will take place when a new generation is being populated. A lower value means that mutation is far more likely to take place whilst a larger value will mean mutations are less likely to occur. Each of these experiments will use a different mutation rate, working in increments of 25. The expectation for these experiments is that experiments that use a more reasonable mutation rate, around 100, will be more likely to return a configuration that meets user requirements. Experiments that use a mutation rate further away from the default value of 100 are expected to be unsuccessful, either due to too many mutations taking place resulting with an erratic set of configurations or not enough mutations taking place resulting in the algorithm stagnating.

The results from this set of experiments showed that all but one experiment was capable of finding a configuration that had a fitness value of at least 70. This exceeded expectations as it was expected that experiments using a smaller mutation value, resulting with more mutations, would encounter difficultly working towards viable configurations and resulting in erratic genepools. Instead, the experiments with a mutation rate of 75 or less were quite capable of finding a suitable configuration in a similar manner to the control experiment. Plotting both the average and highest fitness for each generation during these experiments, as seen in figures 5.6 and 5.7, demonstrate that all experiments, barring MR-175, were capable of gradually working towards a potential solution and were generally capable of maintaining a stable genepool. It is likely that the success of these experiments was due to the genetic algorithm having access to a more diverse range of configurations, similar to previous experiments with a larger genepool size, enforcing what was previously discussed. The only experiment that was not successful during this set of experiments was MR-175. Although some mutation still took place, it was far less frequent during this experiment, which likely resulted in the algorithm stagnating and failing to find a suitable configuration. This suggests that the algorithm does benefit from having some frequent mutation taking place throughout an experiment. This setting should be evaluated during future works to determine how effective this setting is when the genepool is much larger and has individuals with a wider range of variables as part of their configurations.

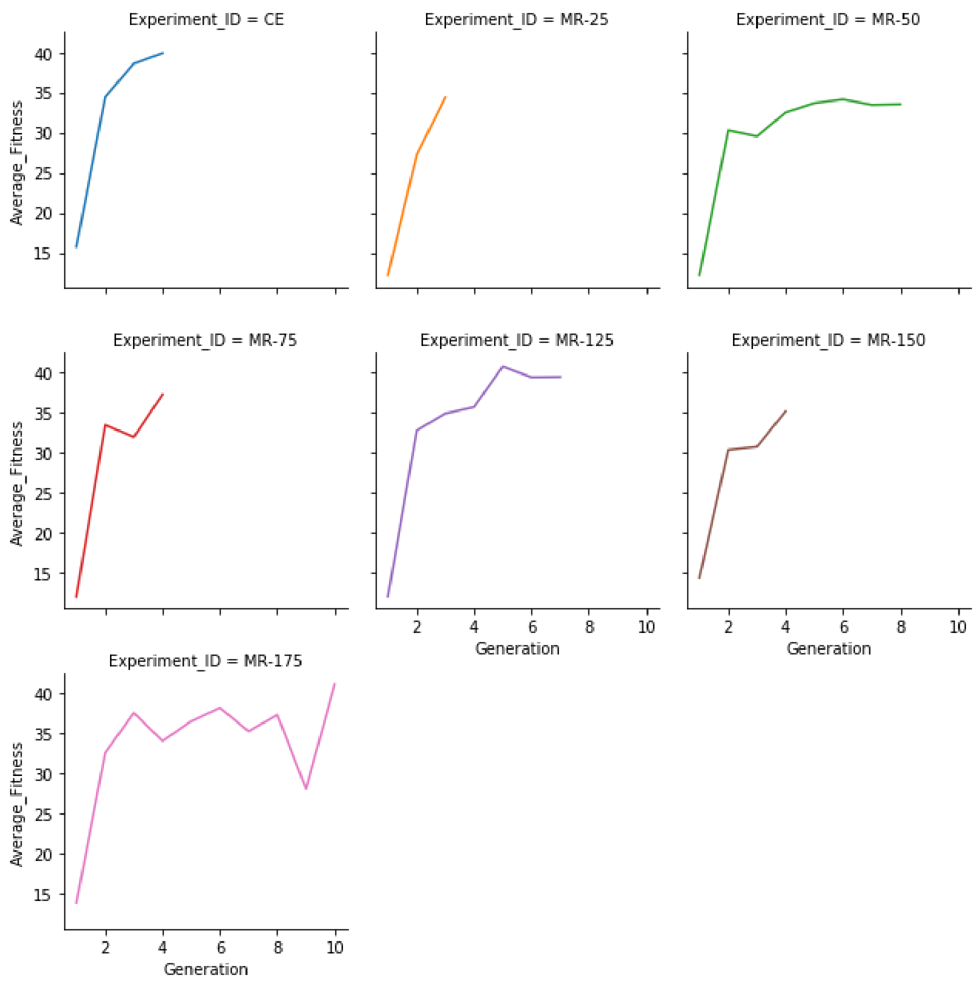


Figure 5.6: Progression of average level of fitness during the first 10 generations of mutation rate experiments.

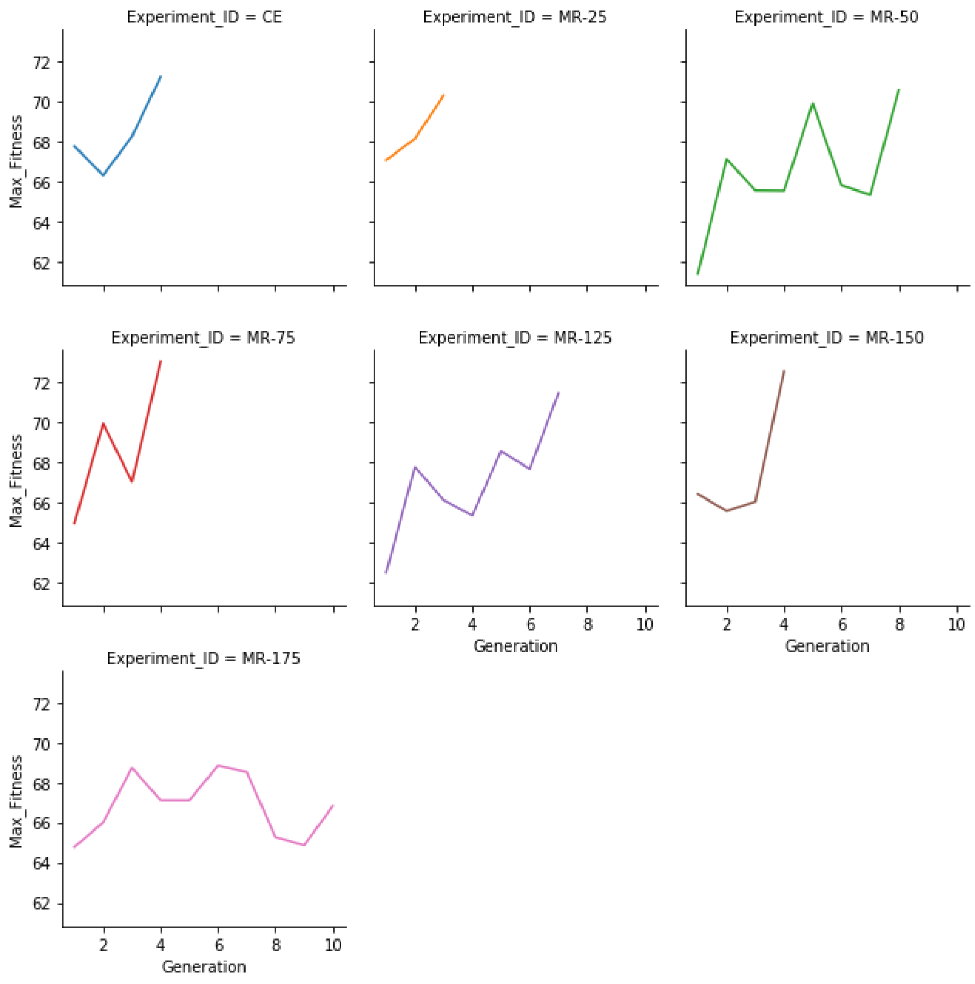


Figure 5.7: Progression of highest level of fitness during the first 10 generations of mutation rate experiments.

Fitness Test Setting Experiments (FIT)

The next series of experiments that were conducted were carried out using different kinds of fitness evaluation functions during each experiment. As outlined during implementation, three different options are currently available for fitness evaluation, two sample t-test, chi-squared and Kullback divergence. During all of the previous experiments, chi-squared was used to evaluate the fitness of each individual. To determine how modifying the kind of fitness evaluation method used will affect the algorithm, a couple of additional experiments were conducted. The expectation here is that each fitness evaluation method should be capable of returning a value that is suitably reflective of an individual's ability to achieve the required protein output. Also, the number of generations taken to find an optimal configuration may vary between each function as how they calculate their own fitness value will differ. For example, the result of a two sample t-test is based on the difference of mean averages between two samples whilst chi-squared is used to examine and compare frequencies of values between two sets.

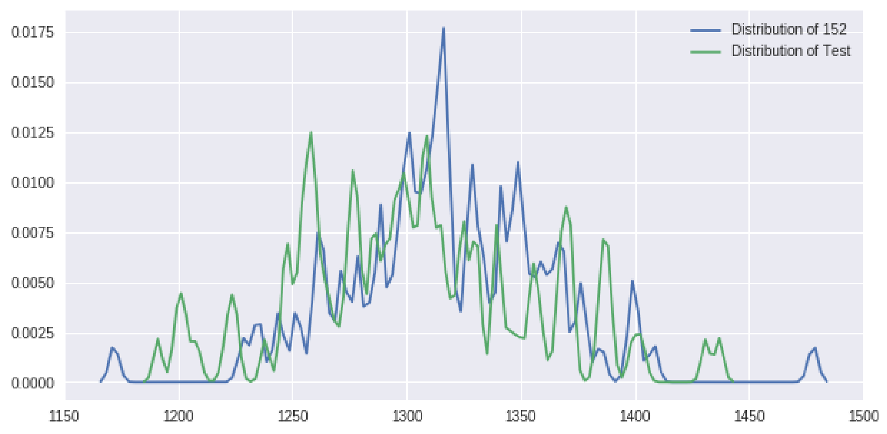


Figure 5.8: Visualisation of plotting the simulation results from the individual with the hightest fitness during experiment IND-175 against the distribution set of the user required protein setting. This resulted in a fitness of 70.21 when using chi-squared.

The results from each of these experiments highlight how integral the implementation of a suitable fitness function is within a genetic algorithm. The first experiment, using two sample t-test, was able to quickly find a configuration with a fitness value of 99.99 during the first generation. Plotting out this individual's results against the required level of protein shows that there is some validity to this value as both distributions were somewhat similar, although there was some significant variation between both distributions. From a user point of view, the expectation of a value this high would be that the distributions should be near identical. This can be seen in figure 5.9. Comparing this to the control experiment, which utilised chi-squared, returned a configuration with a fitness value of 71.25, which when that individual's set of simulation results are plotted against the required protein output distribution showed that the two distributions were fairly similar with some notable variation, meaning a value of around 70 seemed reasonable in this case. This can be seen in figure 5.8. As previously discussed, the reason for the significant difference between these two values despite similar distributions is likely due to the different methods for obtaining these values, which in this case has resulted in values that suggests chi-squared is better suited as a fitness function over two sample t-testing.

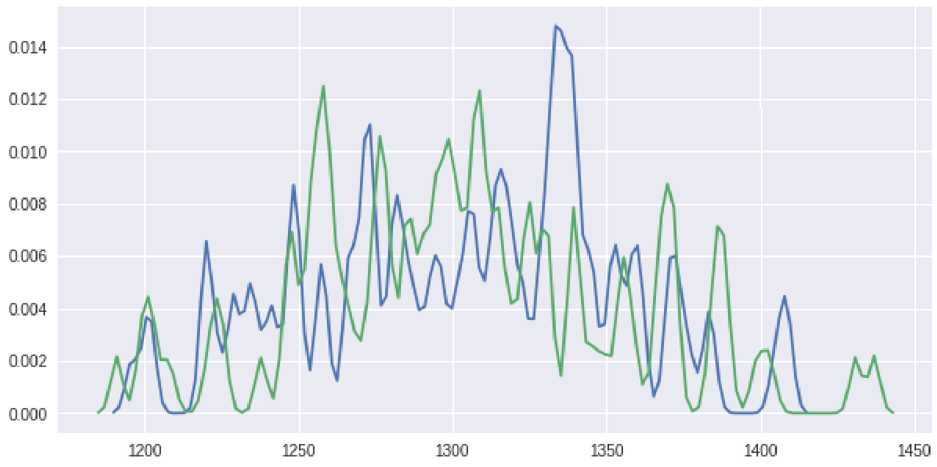


Figure 5.9: Visualisation of plotting the simulation results from the individual with the hightest fitness during experiment FIT-TS against the distribution set of the user required protein setting. This resulted in a fitness of 99.99 when using two sample t-test.

When carrying out the Kullback divergence experiment, the fitness function would return inaccurate values when evaluating individuals. For example one individual could have a fitness value of around 70 where the two distributions were reasonably similar whilst another individual could also have a fitness value of 70 and the two distributions would be nowhere near each other when plotted. The results of this experiment would suggest that further work is required to properly implement Kullback divergence as an option for fitness evaluation within the solution and would be otherwise unfair to completely dismiss this option until then. Otherwise, the results of these experiments suggest that, as of now, the most suitable fitness evaluation method was chi-squared due to its ability to evaluate individuals and return fitness values that seemed to properly reflect a configurations suitability for achieving a certain protein output.

Experiment ID	**Description**	**Generation Reached**	**Highest Fitness**
CE	Control Experiment	4	71.25
GS-25	Genepool Size - 25	30	68.136
GS-50	Genepool Size - 50	30	69.56
GS-75	Genepool Size - 75	17	70.43
GS-100	Genepool Size - 100	30	68.75
GS-125	Genepool Size - 125	2	71.90
GS-175	Genepool Size - 175	7	70.21
GS-200	Genepool Size - 200	3	70.90
GS-225	Genepool Size - 225	2	72.02
GS-250	Genepool Size - 250	5	71.43
MR-25	Mutation Rate - 25	3	70.31
MR-50	Mutation Rate - 50	8	70.59
MR-75	Mutation Rate - 75	4	73.04
MR-125	Mutation Rate - 125	7	71.46
MR-150	Mutation Rate - 150	4	72.54
MR-175	Mutation Rate - 175	30	68.8
FIT-TS	Fitness Test Setting - Two Sample T-Test	1	99.99
FIT-KULL	Fitness Test Setting - Kullback Divergence	1	86.17

Table 5.2: Results of genetic algorithm experiments.

Genetic Configuration Experiment

Carrying out these experiments has given a general idea of how the performance of the genetic algorithm varies when certain settings are altered. Ultimately, the primary purpose of this solution is to highlight genetic configurations that can potentially achieve a certain output of protein within a biological system. From these experiments, the experiment that returned a configuration with the best fitness was experiment MR-75, with a fitness of 73.04. Examining the output of this experiment showed that the configuration that achieved this had an ON_VAR value of 11.3 and an OFF_VAR value of 11.6. In order to determine if the genetic algorithm is viable for providing researchers with viable genetic configurations, the configuration from experiment MR-75 was examined further to determine if it provides a sufficient solution. In order to do this, a single individual was set up within the solution to use the values extracted from experiment MR-75. This individual was then processed 100 times with 100 simulations per epoch. The level of fitness was then extracted from each of these epochs and was used to calculate the mean average level of fitness that this configuration can achieve. The expectation for this experiment was that the level of fitness would likely be less than what was found in experiment MR-75 and would have a fitness value somewhere between 62 and 66.

The outcome of this experiment showed that the average level of fitness that this configuration was able to achieve was 57.99, which was significantly lower than what was expected. Examining the individual output of each of these epochs showed that the fitness values for this configuration varied greatly, ranging between 50 and 70. The outcome of this experiment suggests that even with 100 simulations being performed on individuals, processed individuals do not currently offer a sufficiently general idea of how certain configurations will perform within the intended environment. The solution would likely benefit from allowing more simulations to take place on individuals before carrying out fitness evaluation due to Kappas use of SDEs. This however, would require significantly more powerful and sophisticated hardware in order to ensure that the solution is still capable of working towards potential genetic configurations within a reasonable timeframe, otherwise it will become unviable for researchers to use.

Chapter 6

Conclusion

In conclusion, the solution that was developed was able to deliver on the primary goals of this project by providing a software solution for researchers that was capable of working towards potential biological configurations, which could be utilised in the field of synthetic biology whilst also capable of achieving this within a reasonable span of time. The solution was also developed to acknowledge the majority of additional criteria that was defined following the completion of the literature review found in tables 3.1 and 3.2.

6.0.1 Future Work

Whilst the development of this solution was overall successful, there is much room for improvement, particularly in some areas that were identified during the evaluation of the developed software solution.

Distributed Computing

Two key findings from the evaluation of the genetic algorithm were the algorithm being generally more effective at working towards a viable configuration when more individuals were present within the genepool and that the algorithm would likely benefit from carrying out more than 100 simulations in order to properly provide a general idea for how certain configurations perform. Whilst the genetic algorithm has been developed with a flexible genepool size and number of simulations in mind, the overall performance of the genetic algorithm was restricted by the hardware it was operating on. For example, experiments utilising more individuals in the genepool took much longer to process, even on a system that boasted a CPU with 8 cores and a clocking speed of

3.40 ghz. Something else to note, this was occurring with a series of experiments that only used 100 simulations with just two variables being modified for each individual in the genepool. It becomes clear that in order to make this solution viable for larger scale, more complex experiments to work towards viable configurations, it needs to be transferred to a system where it is able to far more efficiently carry out its processes. One possible option is to adjust the solution so that it is capable of carrying out experiments within a distributed computer system. This would mean the solution would be able to distribute the processing of individuals to other machines on the system, so multiple individuals could be processed simultaneously, as opposed to the current solution of only processing one individual at a time. This should allow for a solution that is capable of working with much larger genepools, more variables and running far more simulations for each individual. The biggest benefit to running more simulations is that it would allow for getting a far more general idea of how certain configurations would perform, and would also mean that no individual would have to be re-evaluated as it is in the current solution and all results could be stored to a local database of results, further optimising performance.

Additional Fitness Evaluation Options

As noted during evaluation, the only viable fitness evaluation method that is currently present within the solution is chi-squared. Two sample t-testing offered a value that was possibly overly generous whilst Kullback divergence did not operate as expected and did not offer a viable option for evaluating fitness. In order to determine if fitness evaluation can be made more accurate and to provide researchers additional options, additional fitness evaluation functions should be developed. Kullback divergence should be re-examined first before proceeding onto different options in order to ensure it is properly implemented into the solution and to determine if it would serve as a viable option for evaluating individuals. Should the scope of the solution expand in order to accommodate larger scale experiments, it will also be important to ensure that developed fitness functions are able to properly and accurately evaluate larger distribution sets returned from Kappa simulations.

User Interface

Whilst the solution was capable of delivering on the most crucial requirements, there was one requirement that was not sufficiently delivered upon. NR-4 in table 3.2 was the non-functional requirement for delivering a software solution that did not require extensive programming skills to operate. Whilst the solution has been developed to be as user friendly as possible, the fact remains that the solution still requires some knowledge of programming in order to operate. In order to address this during future work, a user-friendly interface should be developed in order to make the solution more accessible to more researchers. This interface should hide away the inner workings of the solution and should still allow the user to properly configure the solution to meet their requirements. During the development of an interface, additional tools could also be developed in order to aid researchers with their work, such as providing a way to easily generate a distribution set for user required protein.

6.0.2 Closing Thoughts

Overall, the solution could be considered as success as it delivers on the initial primary goals outlined at the beginning of the project and could eventually provide researchers a way of validating their initial hypothesis before proceeding into the lab environment, saving both time and money. The solution that was developed during this project has plenty of opportunity to be developed further in order to make it more viable for more demanding experiments and to keep up with addressing certain problems within the field of synthetic biology, a factor that will become more prevalent as the field continues to garner interest.

References

Anderson, J., J Clarke, E., Arkin, A. & A Voigt, C. (2006), 'Environmentally controlled invasion of cancer cells by engineered bacteria', *Journal of molecular biology* **355**, 619–27.

BBC-News (2019), 'He jiankui: China condemns 'baby gene editing' scientist (accessed 10/06/2019)'.
URL: *https://www.bbc.co.uk/news/world-asia-46943593*

Bergin, J. (2018), 'Synthetic biology: Global markets (accessed 17/06/2019)'.
URL: *https://www.bccresearch.com/market-research/biotechnology/synthetic-biology-global-markets.html*

David B. Fogel, Lawrence J. Fogel, M. J. W. (1966), *Artificial Intelligence Through Simulated Evolution*, John Wiley Sons Inc, New York, United States.

Fogel, D. B. & Fogel, L. J. (1996), An introduction to evolutionary programming, *in* J.-M. Alliot, E. Lutton, E. Ronald, M. Schoenauer & D. Snyers, eds, 'Artificial Evolution', Springer Berlin Heidelberg, Berlin, Heidelberg, pp. 21–33.

Hallinan, J. S., Gilfellon, O., Misirli, G. & Wipat, A. (2014), Tuning receiver characteristics in bacterial quorum communication: An evolutionary approach using standard virtual biological parts, *in* '2014 IEEE Conference on Computational Intelligence in Bioinformatics and Computational Biology', pp. 1–8.

Hornby, G., Globus, A., Linden, D. & Lohn, J. (2006), 'Automated antenna design with evolutionary algorithms', *Collection of Technical Papers - Space 2006 Conference* **1**.

Kappa-Language (n.d.), 'abc-pert.ka (accessed 03/07/2019)'.
URL: *https://tools.kappalanguage.org/try/?model=https%3A//raw.githubusercontent.com/Kappa-Dev/KaSim/master/models/abc-pert.ka*

Lee, S. K., Chou, H., Ham, T. S., Lee, T. S. & Keasling, J. D. (2008), 'Metabolic engineering of microorganisms for biofuels production: from bugs to synthetic biology to fuels', *Current Opinion in Biotechnology* **19**(6), 556 – 563. Chemical biotechnology / Pharmaceutical biotechnology.
URL: *http://www.sciencedirect.com/science/article/pii/S0958166908001420*

Lincoln, S., Rogers, I. & Srivastava, R. (2015), Metabolic design and engineering through ant colony optimization, *in* 'Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation', GECCO '15, ACM, New York, NY, USA, pp. 225–232.
URL: *http://doi.acm.org/10.1145/2739480.2754817*

Paul S. Freemont, Richard I Kitney, G. B. T. B. R. D. T. E. K. P. G.-B. S. (2012), *Synthetic Biology - A Primer*, Imperial College Press, London, United Kingdom.

Reed, D. A. & Dongarra, J. (2015), 'Exascale computing and big data', *Commun. ACM* **58**(7), 56–68.
URL: *http://doi.acm.org/10.1145/2699414*

Shapiro, B. E. (2007), 'What is mathsbml (accessed 17/06/2019)'.
URL: *http://mathsbml.com/mathsbml/*

Shiffman, D. (2012), *The Nature of Code*, The Free Software Foundation, Boston, United States.

VK, S. (2006), 'Singh2006_tca_ecoli_glucose (accessed 03/07/2019)'.
URL: *https://www.ebi.ac.uk/biomodels/BIOMD0000000222*

Waites, W., Msrl, G., Cavaliere, M., Danos, V. & Wipat, A. (2018), 'A genetic circuit compiler: Generating combinatorial genetic circuits with web semantics and inference', *ACS Synthetic Biology* **7**(12), 2812–2823. PMID: 30408409.
URL: *https://doi.org/10.1021/acssynbio.8b00201*

Wong, K.-C. (2015), 'Evolutionary algorithms: Concepts, designs, and applications in bioinformatics', *Nature-Inspired Computing: Concepts, Methodologies, Tools, and Applications* .

Appendix A

Terms Of Reference, Ethics Forms & Additional Material

This appendix contains the following additional elements produced for this project:

- Terms Of Reference Coversheet
- Terms Of Reference Body
- Ethics Form
- Risk Assessment

Additional material accompanying this project can be found **by clicking here**.
This additional material includes:

- Copy of Genetic Algorithm Code
- Experiment Datasets
- Kappa File Templates
- Dataplot Images

Department of Computing and Mathematics Computing and Digital Technology Postgraduate Programmes Terms of Reference Coversheet	
Student name:	[redacted]
University I.D.:	11024531
Academic supervisor:	Matteo Cavaliere
External collaborator(optional):	
Project title:	Cellular Computing for Evolutionary Dynamics
Degree title:	MSc Advanced Computer Science
Project unit code:	MCA_MSc.03
Credit rating:	120
Start date:	10/05/2019
ToR date:	26/05/2019
Intended submission date:	[redacted]
[redacted]	[redacted] 11/06/2019
Signature and date external collaborator (if involved):	

This sheet should be attached to the front of the completed ToR and uploaded with it to Moodle.

Terms of Reference

Project Aims

- Produce a program (preferably one that utilises a Genetic Algorithm) that can discover architectures of synthetic biological circuits that are able to provide the specified protein output from cells depending on data that is passed into the program, this will be in the form of a distribution plot which will be used by the user to describe the kind of protein output they require from synthetic organisms.
- Optimise solution in to allow for efficient discovery of suitable synthetic circuits in a reasonable frame of time and to ensure that resources used during runtime are effectively managed.

Learning Outcomes

- Apply skills of critical analysis to real world situations within a defined range of contexts
- Demonstrate a high degree of professionalism, e.g. initiative, creativity, motivation, professional practice and self-management
- Express ideas effectively and communicate information appropriately and accurately using a range of media including ICT
- Develop working relationships using teamwork and leadership skills, recognising and respecting different perspectives
- Manage their professional development reflecting on progress and taking appropriate action
- Find, evaluate, synthesise and use information from a variety of sources
- Articulate an awareness of the social and community contexts within the scope of the project.

Project Description

Synthetic biology is concerned with the design and construction of new biological entities (as well as with the redesign of existing biological systems). These entities can be engineered to address a wide range of problems being faced today by modern society. Examples can include synthetic organisms that are engineered to carry out administering of medication in a patient and organisms that have been engineered to aid with the manufacturing of biofuel. The assembly of new biological entities is often done by composing modular parts until the assembly producing the desired output has been obtained.

"*It is no surprise that the use of synthetic biology to design living systems will find numerous application areas with significant impact on society*" - Paul S. Freemont, Richard I Kitney, Geoff Baldwin, Travis Bayer, Robert Dickinson, Tom Ellis, Karen Polizzi, Guy-Bart Stan. 2012. *Synthetic Biology - A Primer.* London: Imperial College Press. Pg. 109

The manufacturing of new entities often encounters numerous challenges, such as requiring a significant amount of time for research, large development costs and being able to justify research with a large projected manufacturing output. One potential area of technology that may be able to assist with this challenge is machine learning. In recent years, development of machine learning models has allowed for providing researchers from a wide range of academic disciplines with solutions that can be used to solve a wide range of complex tasks, just as classification, deep learning and artificial intelligence. One kind of machine learning model that has gained interest in recent years is genetic algorithms, a type of machine learning model where the algorithm is able to determine the most suitable solution for a problem over a period and uses mechanics that resemble that of biological evolution to refine the calculation of new potential solutions.

"*Computational science and engineering also enable multidisciplinary design and optimization, reducing prototyping time and costs*" - Daniel A. Reed, Jack Dongarra. 2015. "Exascale Computing and Big Data." *ACM* 56-68. Pg.58

The goal of this project is to produce a program that discovers the architectures of synthetic circuits able to facilitate the evolutionary resilience of a cellular population. A program that is able to achieve the intended goals of the project would allow researchers in this field to determine ideal configurations for synthetic organisms for producing a required protein output, which will in turn save researchers development time, research costs and lower the number of requirements for research proposals in this area, such as being able to justify research with a significant projected manufacturing output.

Evaluation Plan

Towards the end of the project, the developed solution will be evaluated to determine how effective it was regarding meeting the goals defined in the project aims. The following criteria will be taking into consideration during the evaluation:

- Did the solution meet the goal of discovering synthetic circuits that can produce a distribution of protein output that resembles that of the distribution data passed into the solution by the user?
- Provided the solution can discover synthetic circuits, was the solution able to produce a solution that met the user's minimum similarity threshold (i.e. a minimum similarity rating of 95%)?
- How efficient was the solution at producing a solution required by the user? This will be determined by evaluating how effectively resources were managed and how quickly the solution was able to provide a solution based on user requirements.

- How effective were optimisation decisions that were made during the project? This will be evaluated by calculating the speedup and efficiency of the solution as optimisation is carried out (i.e. determining how effective multithreading implementation was compared to the initial serial solution).

Bibliography

Daniel A. Reed, Jack Dongarra. 2015. "Exascale Computing and Big Data." *ACM* 56-68.

Paul S. Freemont, Richard I Kitney, Geoff Baldwin, Travis Bayer, Robert Dickinson, Tom Ellis, Karen Polizzi, Guy-Bart Stan. 2012. *Synthetic Biology - A Primer.* London: Imperial College Press.

SECTION 1: Information Classification	**Y/N**
Has the data to be processed been classified according to MMU's Information Classification Policy? If so, please confirm the classification:	**N**
Have an information asset owner and manager been identified and engaged?	**N**
Does the project involve Personal data? This means data which relate to a living individual who can be identified a) from those data b) from those data and other information which is in the possession of, or is likely to come into the possession of, the university and includes any expression of opinion about the individual and any indication of the intentions of the data controller or any other person in respect of the individual. (An ICO-published flowchart for helping to identify personal data can be found here)	**N**
Does the project involve Sensitive personal data? This means personal data consisting of information as to a) the racial or ethnic origin of the data subject b) the data subject's political opinions c) the data subject's religious beliefs or other beliefs of a similar nature d) whether the data subject is a member of a trade union (within the meaning of the Trade Union and Labour Relations (Consolidation) Act 1992) e) the data subject's physical or mental health or condition f) the data subject's sexual life g) the commission or alleged commission by the data subject of any offence h) any proceedings for any offence committed or alleged to have been committed by the data subject, the disposal of such proceedings or the sentence of any court in such proceedings	**N**
Does the project involve other SENSITIVE data? This means data meeting the criteria for classification as SENSITIVE according to the MMU Information Classification Policy (found here). If any sensitive data types are confirmed, explain why the involvement of this data is essential to the system/service to be delivered by the project	**N**
Has a Privacy Impact Assessment been completed? This will confirm the sensitivity of the data being processed, and help in the selection of controls. Please refer to the PIA Screening Questionnaire to determine whether a PIA is required. If you answer yes to any of the questions then a PIA must be undertaken. Please contact Legal Services for advice on the completion of a PIA: legal@mmu.ac.uk. (A copy of the PIA template can be accessed on the University's Data Protection website).	**N**

SECTION 2: Information Storage, Processing, and Disposal	**Y/N**
Where will production and test data likely be stored? ***(check all that apply)*** a) On server(s) managed by MMU IT Services (ITS) b) On server(s) managed by another MMU department c) On server(s) managed by a third party Please summarise: **Has consideration been given to the security requirements for hosting?** Please summarise: Significant assurance can be taken from the presence of current and suitably-scoped certifications: ISO 27001 for information security management; ISO 22301 for business continuity management; ISO 27017 for cloud security; Cyber Essentials (mainly for HMG departments or providers of services to HMG). The Information Security Manager will be able to offer advice on the suitability of or necessity for certifications.	N
Has retention and disposal of the data being processed been considered? Please summarise: **Does the project involve the migration of data from one system to another?** Please summarise: Migration offers a unique opportunity to apply retention and disposal principles, and mitigate information risk by reducing the amount of data migrated.	N
Will data need to be encrypted at rest? If yes, please summarise: This can be required depending on the sensitivity of the data. Consideration needs to be given to the best method – whether full disk, database, file etc.	N
What data entry method(s) will be used? ***(check all that apply)*** a) End user input via Web interface b) End user input via client application c) File(s) transferred from other system(s) d) Other – please specify:	N
Will copies of data be transmitted out of the technology or service? **If so, what transmission method(s) will likely be used?** ***(check all that apply)*** a) Electronic file transfers to other information systems within MMU b) Electronic file transfers to entities external to MMU c) Output to spreadsheets/databases created by users of the IT product or service d) Other – please specify: For each method checked, briefly describe the purpose of the data transmission:	N

Will data be encrypted in transit? If yes, please summarise: Encryption for data in transit can range from password protecting an email attachment, utilising existing secure connection channels such as remote access, to a dedicated site-to-site VPN. It is important that the technology used to secure the data makes use of strong, modern cyphers, and that key lengths are appropriate. The Information Security Manager will be able to offer further advice.	N

SECTION 3: User Access Methods

What organisational units and user groups will have access to the technology or service?

Please detail below, considering all likely scenarios; e.g. normal use, support, compliance activity.

N/A

What groups or individuals external to the University will have access to the technology or service?

Please detail below, considering all likely scenarios; e.g. normal use, support, compliance activity.

N/A

What remote access is required?

Please detail the business requirement below.

N/A

How will users be authenticated to the system?

	Users authorised to access only their own data	Users authorised to access data of other individuals	MMU IT staff authorised to administer the system	Third parties authorised to administer the system
Link to AD accounts	☐	☐	☐	☐
User ID and Password	☐	☐	☐	☐
Software Token	☐	☐	☐	☐
Hardware Token	☐	☐	☐	☐
Other*	☐	☐	☐	☐

* If Other, describe here: N/A

How will users be authorised to access the system?

Please summarise: N/A

How will user activity be monitored?

Please summarise: N/A

Has access control – authentication, authorization and auditing – been included in testing?

Please summarise: N/A

Have default account details been changed?

N/A

How will user changes (joiners, movers, leavers) be handled?

Please summarise: N/A

Has secure operation of the system been included in user training and guidance as necessary?

Please summarise: N/A

SECTION 4: Third Party Services	Y/N
Will the project use third party services (for example, development, business analysis, project management, HR, other consultancy)? Please summarise: Third party services must meet MMU security standards. Where they reference certifications or accreditations, these should be checked. Consideration should be given to ongoing audit of security.	N
What is the contractual route being taken? a) Adoption of third party contractual terms b) Mandating of MM contractual terms Please summarise:	N/A
Will the deployed system use third party services for hosting? Please summarise, referencing any standards claimed or required, availability SLAs required, geographical location of the data centre:	N
Will the deployed system use third party services for data processing? Please summarise, referencing any standards adhered to:	N
Will the deployed system use third party services for support? Please summarise, referencing any standards adhered to, SLAs required: **Does this require a dedicated remote access solution, or access to MMU monitoring systems?** Please summarise:	N
Has the third party agreed to adhere to MMU security policies? Please summarise: **If not, have the third party's own policies and certification been checked for compliance with MMU's policies?** Please summarise:	N/A
Is vetting of third party staff required? Please summarise: Vetting may be required if third party data is involved and the third party has a requirement for vetting. Common levels of vetting are BPSS and SC, both generally mandated for access to HMG material at given classifications. MMU do not by default require more than basic ID and right to work checks for staff. Where data is being processed in a different legal jurisdiction, this may change.	N

SECTION 5: Information System Development and Deployment	Y/N
Where in-house or third-party development of a technical solution is required, have information security considerations been included in the specification, development and testing plans? Please summarise, including any standards adhered to:	N
Does the system require specific firewall changes? Please summarise: **Have these been documented?**	N/A
Are test and development environments segregated and adequately secured? Please summarise:	N/A
Does the proposed system require independent penetration testing? Please summarise:	N/A
Has the proposed system been included in vulnerability scanning routines? Please summarise:	N/A
Has the proposed system been included in backup routines? Please summarise:	N/A
Has the proposed system been included in patching routines? Please summarise:	N/A

The MANCHESTER METROPOLITAN UNIVERSITY
Faculty of Science and Engineering

RISK ASSESSMENT COVER SHEET

REFERENCE NUMBER: NPC / 270918 / JDE1.49
SCHOOL: Computing, Mathematics & Digital Technology
TITLE OF WORK: CMT Projects involving software development only
LOCATION OF WORK: John Dalton Building computing facilities, computers at student's own home etc.
INTENDED ACTIVITIES (attach methods sheets (e.g. standard operating practices) and work schedules to this form): General use of computers to develop and test software. Method sheets and work schedules not applicable.
PERSONS AT RISK (list names of all individuals (including status e.g. staff/student), and/or unit(s) / course(s) undertaking the activity. For students please indicate course and level, for staff give contact email / phone number): Postgraduate Student, MSc Advanced Computer Science, Level 7
HAZARDS (provide a summary of the hazards anticipated and attach detailed assessments with appropriate risk control methods to this form): Repetitive Strain Injury – work related upper limb disorder Back injury resulting from improper posture Eye strain Fatigue Stress Possible risk from 240v electrical mains supply *Are these hazards necessary in order to achieve the objectives of the activity?* **Yes** **Hazard Rating** (delete as appropriate): **Low**
HAZARDOUS SUBSTANCES/MATERIALS USED AND HAZARD CLASSIFICATION **(**appropriate COSHH data sheets / risk assessments must be attached to this form**):** **ALL CONTAINERS OF HAZARDOUS SUBSTANCES SHOULD BEAR CORRECT HAZARD WARNING LABELS**.

NAME OF MATERIAL *Please provide also approximate quantity and concentration if applicable.*	**HAZARD CLASS**	**HAZARD LABEL**	**DISPOSAL** *Hazardous materials must not be removed from laboratories. List disposal arrangements for* all materials listed below *in the location where* the work will be carried out:

RISK CONTROL METHODS (provide a summary of the hazards anticipated and attach detailed assessments with appropriate risk control methods to this form):

The hazards identified above are controlled by:

Facilities review when laboratories are commissioned
Induction session on H&S given to students by Technical Services Manager
School H&S information given in Student handbook
Posters in laboratories
PAT testing of equipment after three years
Annual H&S inspections

The laboratory workstations, whilst not legally required to be DSE compliant, (the continuous usage is too low to present risk) are fully compliant with current legislation. Monitors and keyboards are adjustable, chairs are adjustable and the lighting designed for both computer usage and associated reading activity. In each laboratory, there is an adjustable desk, suitable for wheelchair users, usually located in the next to the door.

Hazard Rating with Control Methods (delete as appropriate): **Low** *Will*

any specific **training** *be required (if* YES *give details)?* N/A

Are there any specific **first aid** *issues (if* YES *give details)?* N/A

PROCEDURE FOR EMERGENCY SHUT-DOWN (if applicable):

In the event of fire, flood or other emergency, evacuation of the laboratory would take place and the technical staff would subsequently make an assessment of the necessity of switch-off. As overall system control is vested in a separate server room, there would be little physical harm to any device in directly cutting the power to the mains for each individual lab.

Re-start of the lab may present problems of a technical nature but would not affect the personal safety or health of any individual.

IF OFF-SITE INDICATE ANY OTHER ISSUES (e.g. associated with: individual's health and dietary requirements (obtain off-site health forms for all participating individuals and indicate where this information will be located); social activities, transportation, ID requirements; permissions for access and sampling).

Not applicable – this form applies only to the laboratories listed

	NAME	**STAFF/STUDENT No.**	**DATE**
Originator	Nicholas Costen	01900261	270918

Supervisor	Matteo Cavaliere		29042019
Technical Manager			
Divisional / School Health and Safety Coordinator (p.p. HoS)			
DATE TO BE REVIEWED BY: September 2019			