Brunel University London

College of Engineering, Design and Physical Sciences

Department of Computer Science

Computational Biology Team

Application of Bio-Model Engineering To Model Abstract Biological Behaviours

A Doctorate Thesis

Supervisors:

Author: *Doctor Alessandro Pandini*

Leila Ghanbar *Professor David Gilbert*

leila.ghanbar@brunel.ac.uk *RDA:*

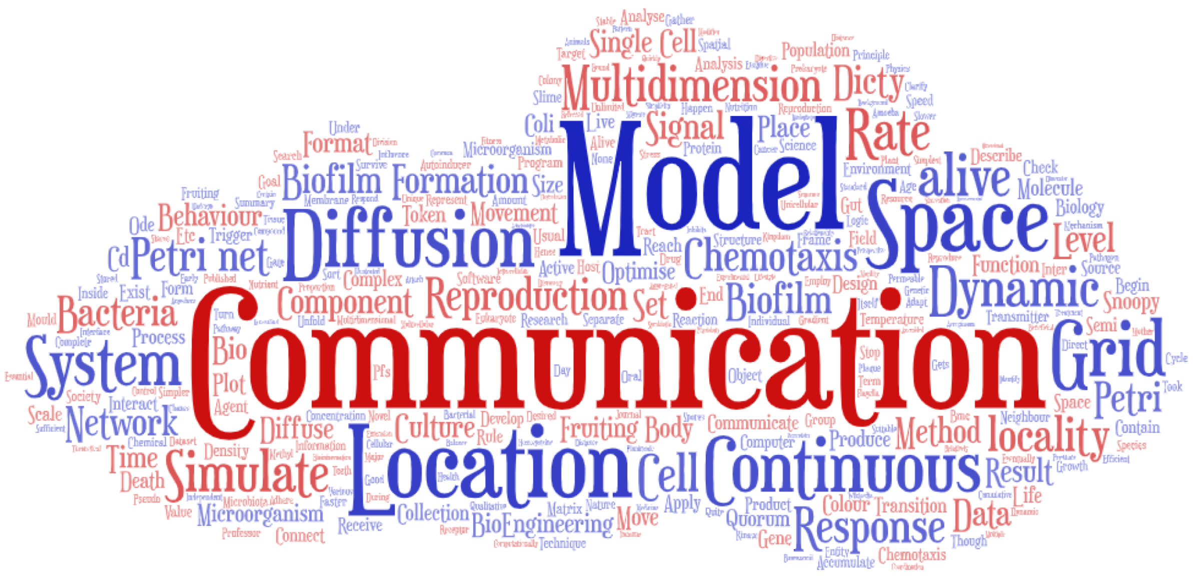
Doctor Rumyana Neykova

"I like crossing the imaginary boundaries people set up between different fields."

Maryam Mirzakhani

Professor of mathematics in Stanford University

(1977 - 2017)


Communication
Model
Location
Diffusion
Space
Continuous
Grid
System
Simulate
Response
Dynamic
Chemotaxis
Reproduction
Multidimension
Rate
alive
locality
Biofilm
Petri net
Network
Bacteria
Cell
Formation
Component
Signal
Dicty
Place
Level
Petri
Method
Result
Data
Culture
Fruiting Body
Single Cell
Behaviour
Format
Movement
Optimise
Microorganism
BioEngineering
Time
Death
Diffuse
Collection
Snoopy
Function
Molecule
Population
Transition
Quorum
Chemotaxis
Set
Bio
Plot
Cd

Abstract

Life in nature is defined by many characteristics. Whether something can move, communicate, response to the others, reproduce or die, indicate if it is alive or not. Among these features, communication can be considered the most basic and yet the most important as it happens both inside and outside an organism; between every molecule and every cell there are signals to be passed and to be responded to. Communication defines biology.

A network of molecules or a society of organisms are both complex systems. The smallest change in this snarled network affects the whole system and changes the output significantly. Comprehending and manipulating them in detail is time and resources consuming and involves human error. But there is a way to simplify the process of inspecting the living creatures.

Bio-model engineering lies at the crossroads of biology, mathematics, computer science, engineering and is a branch of systems biology. In this field of science, biological models are created and/or re-designed for simplification, abstraction and description of biological networks. Modelling these networks based on past experimental observations *in silico* with a set of pre-designed models and a collection of components would make this process faster and simpler.

This thesis contributes to science by providing a collection of model components built in Petri nets with Snoopy. These components each describe a specific behaviour and they can be used individually or as a combination. The set of behaviours in this collection include chemotaxis, reproduction, death, communication and response. These are a few of the most basic behaviours in nature that mark something as alive. These basic behaviours choose that a piece of stone is not alive but the small microscopic bacteria on it are.

Starting with small achievable steps, these components are modelled in abstract, meaning they demonstrate only the critical parts of the behaviours. Not only the models, but also the process of modelling and combining the components is provided from the adaptation and manipulation of a general protocol.

The components in this library are categorised based on their complexity. In this categorisation, the models have four levels, with each level more complex than the former. The more complex levels, are built from the simpler ones in a hierarchical manner. There are two application of the models to two different microorganisms, each from one of the main biological superkingdoms to demonstrate the practicality of this collection. The chosen microorganisms are from: the domain of Prokaryotes *E. coli* and Eukaryotes *Dictyostelium* a.k.a slime mould.

Each model contains a set of rate constants that define the speed of the reactions. A set of **expected** behaviours based on biological literature is defined for these models to be compared with the outcome result of the analysis of the models. The models are simulated by Spike, a command line programme for simulation of models built in Snoopy, and are analysed with R and Python. To achieve the expected results, optimisation methods are used to find the best rates possible in the models in order to achieve a defined behaviour. In this thesis the optimisation is applied to Dictyostelium model to achieve the best rates for the accumulation of Dictyostelium cells in one location to create fruiting bodies. Random Restart Hill Climbing and Simulated Annealing are the chosen methods for optimisation.

Key Words: Model component, hierarchical modelling, Petri net, optimisation, quorum sensing, biofilm formation, chemotaxis, communication, location, locality, space, grid, gradient, alive, basic behaviours, movement, reproduction, death, signal, dynamic, systems, networks.

Acknowledgements

Throughout this research I have faced many challenges and I could not possibly overcome them without the help of the great people around me. I would first like to offer my sincere gratitude to my supervisor, Professor David Gilbert, for providing guidance and feedback throughout this project as well as guiding me so positively and always making me feel confident in my abilities. I am extremely grateful for what he has offered me.

Also I would like to thank Dr Alessandro Pandini whose knowledge and experience have encouraged me in my academic research. I am also thankful to Professor Monika Heiner for her guidance through my research and constructive advice that directed me in every step of my research. I am grateful as well to my RDA, Doctor Rumayna Neykova who has been guiding me kindly and patiently.

I thank profusely Doctor Yasoda Jayaweera and Doctor Sarath Dantu for offering me their assistance in computational aspects of my research with patience and understanding. Thank you for all the time you spent to aid me with my research. Your insight and experience have inspired me and made learning more fun and easy.

Last but not least, I would like to thank my parents for their love and sacrifices for educating me and supporting me all the way through by always believing in me.

Contents

1 Introduction 1

1.1 Research Questions . . . 3

1.2 Aims and Objectives . . . 5

1.3 Novel Contributions to Science . . . 7

1.4 In the Next Chapters . . . 9

2 Literature Review 10

2.1 Systems and Synthetic Biology . . . 10

2.2 Bio-Model Engineering . . . 12

2.3 Choice of Microorganisms For Modelling . . . 13

2.3.1 Bacteria . . . 14

2.3.2 Dictyostelium: The Slime Mould . . . 15

2.4 Chosen Behaviours . . . 17

2.4.1 Quorum Sensing and Biofilm Formation . . . 17

2.4.2 Duplication, Chemotaxis and Death . . . 23

2.5 Bio-modelling Tools and Techniques . . . 24

2.5.1 Differential Equations . . . 25

2.5.2 Agent-Based Models (ABMs) . . . 26

2.5.3 π-Calculus 27

2.5.4 λ-Calculus 28

2.5.5 Statecharts 29

2.5.6 P systems models 30

2.5.7 CellML 30

2.5.8 Bio-PEPA 31

2.5.9 Petri net 32

2.5.10 Tool Box 32

2.6 A Review of Previous Models 32

2.7 Discussion 38

3 Methodology For Modelling, Simulation, Optimisation and Data Analysis 40

3.1 Chosen Software Tools for Modelling and Simulation 40

3.1.1 Snoopy 40

3.1.2 Different Classes of Petri nets: 41

3.1.3 Spike 43

3.1.4 Optimisation Method 44

3.1.5 Data Analysis Tools 44

3.2 Discussion 45

4 Model Library 46

4.1 Properties 48

4.2 Level-0: Basic Components 52

4.2.1 Level-0 Diffusion and Movement 53

4.2.2 Level-0 Duplication 53

4.2.3 Level-0 Death 53

4.2.4 Level-0 Transmitter 53

4.2.5 level-0 Receiver 53

4.3 Level-1: Simple Models 55

4.3.1 Level-1 Transmitter 55

4.3.2 Level-1 Receiver 56

4.3.3 Level-1 Chemotaxis 57

4.3.4 Level-1 Duplication and Death 58

4.3.5 Level-1 Limited Strip Glu 59

4.3.6 Level-1 Semi-permeable obstacle and Transmitter 60

4.4 Colouring the Models 61

4.5 Discussion 63

5 Application of Models from the library 66

5.1 Bacteria *E. coli* 67

5.1.1 Level-2 Transmitter and Receiver(TR) 67

5.1.2 Level-2 Duplication, Chemotaxis and Death (DCD) 68

5.1.3 Level-2 Combined Strip Glu, DCD and TR 70

5.1.4 Level-2 Biofilm Formation 73

5.1.5 Level-3 Quorum Sensing 74

5.2 Level-2 Dictyostelium: Slime Mould 75

5.3 Collection of Model Components 77

5.4 Discussion 78

6 Optimisation 81

6.1 Definitions . . . 82

6.2 Random Restart Hill Climbing . . . 85

6.2.1 Pseudocode . . . 86

6.3 Simulated Annealing . . . 87

6.3.1 Pseudocode . . . 87

6.4 Discussion . . . 88

7 Data Analysis and Results 90

7.1 Timing the Simulations . . . 91

7.2 Models . . . 92

7.2.1 Diffusion / Movement . . . 92

7.2.2 Chemotaxis . . . 94

7.2.3 Semi-permeable Obstacle and Transmitter . . . 95

7.2.4 Combined Strip Glu, DCD and TR . . . 97

7.3 Optimisation Results . . . 98

7.4 Principle Data Analysis . . . 98

7.4.1 PCA on Random Restart Hill Climbing . . . 99

7.4.2 PCA on Simulated Annealing . . . 101

7.5 Inspecting the Behaviour of Dicty . . . 105

7.5.1 Visualisation of Random Restart Hill Climbing . . . 105

7.5.2 Visualisation of Simulated Annealing . . . 110

7.5.3 Random Restart Hill Climbing Versus simulated Annealing . . . 114

7.6 Discussion 115

8 Summary, Conclusions and Further Works 117

8.1 Summary 117

8.2 Conclusion 119

8.3 Further Work 121

8.3.1 Biological and Non-Biological Applications 121

8.3.2 Stochastic and Complex Movement 121

8.3.3 Automated Modelling 122

8.3.4 Improving the collection 122

Bibliography **123**

A Acinetobacter baumannii 144

B Random Restart Hill Climbing Code 146

C Simulated Annealing Code 154

List of Figures

2.1 A general image of biofilm formation process 20

2.2 Spatial distribution of biofilm 23

2.3 Statechart of an air conditioner 29

4.1 Hierarchical Modelling 47

4.2 Four main Level-0 components 54

4.3 Level-1 Transmitter 55

4.4 Level-1 Receiver 56

4.5 Level-1 Chemotaxis 58

4.6 Level-1 Death and Duplication 59

4.7 Limited Strip Glu 60

4.8 Level-1 Semi-permeable obstacle and Transmitter 61

4.9 A summary of the process of colouring a model in coloured Petri nets. 63

5.1 Transmitter and Receiver Model 68

5.2 Duplication, Chemotaxis and Death 70

5.3 Transmitter and Receiver Model 72

5.4 Biofilm formation model 73

5.5 Li_Ghanbar model . . . 75

5.6 Model of Dictyostelium . . . 77

6.1 Flow Chart of Optimisation . . . 84

6.2 System Diagram of Optimisation . . . 85

7.1 Diffusion / Movement plot . . . 94

7.2 Heat Map Plots of Chemotaxis . . . 95

7.3 Heat Map Plots of Semi-Permeable Obstacle and Transmitter . . . 97

7.4 The behaviour of Combined Strip Glu, DCD and TR . . . 98

7.5 Plots of PCA of Hill Climbing . . . 100

7.6 Correlation between PC1, PC2 and PC3 in RRHC . . . 101

7.7 Correlation matrix of RRHC data . . . 102

7.8 Plots of PCA of Simulated Annealing . . . 103

7.9 Correlation between PC1, PC2 and PC3 in Simulated Annealing . . . 104

7.10 Correlation matrix of SA data . . . 105

7.11 Heatmap of simulation 4147 . . . 107

7.12 Dicty Behaviour in Simulation 4147 with Diffusion rate of 0.0001 . . . 108

7.13 Dicty Behaviour in Simulation 4147 with Diffusion rate of 0.001 . . . 109

7.14 Plot of maximum population density in SA . . . 113

7.15 Bar Plot of the Location of Maximum Population Density From SA Results . . . 113

7.16 Heatmap of Simulated Annealing With Maximum Population Density . . . 114

A.1 A.baumannii quorum sensing and biofilm formation . . . 145

List of Tables

2.1 The table of different software tools using different methods 33

2.2 Summary of the models discussed in 2. 39

4.1 The information of 1D, 2D and 3D grids for ColPN 49

4.2 The neighbouring functions for 2D and 3D grids. 50

4.3 The functions of the Sink Transition. 51

4.4 The function definitions for obstacles and the changed neighbour function 52

4.5 Creating a strip of food source on one side of the grid. 52

4.6 The table of Basic Components and their properties. 54

4.7 The table of Simple Models and their components and properties. 65

5.1 A summary of all the properties that are defined in this study. 78

5.2 The list of all model components. 80

7.1 Timing Spike Simulations 93

7.2 Timing of Optimisation 93

7.3 Table of PCA Analysis of Hill Climbing 99

7.4 Importance of components in PCA Analysis of Hill Climbing 100

7.5 Table of PCA Analysis of Simulated Annealing 102

7.6 Importance of components in PCA Analysis of Simulated Annealing 103

7.7 Random Restart Hill Climbing Outstanding Data 106

7.8 Movement of the Clump of Dicty Over Time on the Grid for model 3673 110

7.9 Accepted Simulated Annealing Rates 111

7.10 Top Ten High AP Solutions Sorted Based on the Maximum Population. 112

Chapter 1

Introduction

“Life” is a complex concept with many definitions from different aspects. However, biology has its own way to decide if something contains life and is “alive” by considering some characteristics such as whether or not it reproduces, moves, dies and/or communicates. Among these characteristics, communication could be considered one of the most basic and yet important features as it is required for the other behaviours to happen. Intercellular or intracellular communication is required for an alive organism to survive. Whether it is between the organelles, organs or between different organisms, the presence of this communication is crucial. Therefore, it can be said, that communication defines biology.

Biological networks within and without the living creatures are complex and difficult to comprehend. This complexity brings the necessity to create a simplified version of the networks, by designing models, and achieving a better understanding of them [52]. Modelling biological systems is a developing area of science in which using the data provided by experimental laboratories, the scientists create models, mathematical or computational, to get a better observation of the behaviour of a biological system as well as the detailed interactions inside it [72]. These models then could be used not only to describe and analyse the behaviours, but also to predict them [175, 169, 196].

By modelling the structure of biological systems less time and resource is spent compared to a biological laboratory, whilst the models provide the details behind the interactions inside them. These models can be useful in many areas from suggesting new therapeutic methods [175, 169] and preventive strategies for different diseases [163] to finding solutions for industrial problems caused by microorganisms such as formed biofilm on the hulls of the ships [28]. Thus, by modelling biological networks and analysing them, it is possible to provide testable approaches, reproducible models and results while saving time [177].

Thanks to the informatics and computational sciences, biology has been more informational than before. This allows to shift the focus on construction and re-construction of biological models using engineering principles. Bio-Model Engineering is a branch in systems biology, that designs, constructs and analyses computationally built models of biological networks [88]. In Bio-Model Engineering models can be constructed in different levels of complexity and abstraction from atomic to cellular structure and from inter- to intracellular scales [88, 78, 73]. In this thesis, using engineering principles, models are designed and constructed in different levels and scales in a multidimensional, multi-level and multi-scale environment.

To begin the modelling, choosing a simple case study is the first step. The simplest forms of life are the unicellular microorganisms. These microorganisms might live in isolation planktonically or in societies or colonies [200, 112]. In this thesis, the focus is on two different microorganisms each from one main domains of life: Eukaryotes and Prokaryotes. The chosen microorganisms from each domain are bacteria and Dictyostelium respectively. In chapter 2 a detailed explanation about these microorganisms is convened. The second step of modelling is to choose the basic behaviours to model. As the aim of this study is to simplify the understanding of biological networks, simple behaviours are chosen such as communication, response, movement, duplication and death which are modelled *in silico*.

The third step of modelling is to choose a tool and a platform. After considering different methods and tools which have been used before for modelling biological networks [a complete comparison is provided in chapter 2 section 2.5], the use of Petri nets as the tool and Snoopy as the software platform was decided. Snoopy supports different types of Petri nets such as Stochastic, Continuous and Coloured. Using these different types it is possible to model different aspects of the behaviours as required. Coloured Petri nets are especially useful as they provide a multidimensional grid which then could be used for spatial modelling [77]. Here, Coloured Continuous and Coloured Stochastic Petri nets are used mostly to present a multi-level, multi-scale and multidimensional model of basic biological behaviours.

These models are components in a collection which could be used individually and/or combined to test and analyse the biological system under different circumstances. Then, using different tools, such as Spike the models are simulated to produce output data in csv format. These data are analysed with different techniques, such as R and Python. Analysing the data from simulations, provides a primary understanding of the function of the models.

By understanding the functionality of the models, it is possible to go further into making the models function as expected. This means finding a target function and optimising the models. Optimisation is a mathematical programming technique to find the best solution for a problem. There are different methods for optimisation, which of those, Random Restart Hill Climbing and Simulated Annealing are chosen for this study. After writing the algorithm for each of these, the codes are written in Python. In this code, the rate constants of the models are changed. These rates define the action rates in the models, meaning they describe how fast an action should occur. The codes use Spike to simulate the models and then speculates the behaviour in order to decide whether to accept the new constants or not. The target function is defined to find the maximum number of the cells in one location and find a better combination of rates to achieve a higher number. The reason for this choice is to mimic the construction of fruiting body in unicellular microorganism Dictyostelium. The target function is not under the influence of the location. A detailed description is provided ins Chapter 6.

A part of this thesis has been published in BMC Bioinformatics Journal in 2019 [77]. In this Article, a model of biofilm formation was combined with a model of quorum sensing in *E.coli* using Coloured Petri nets to discuss the affect of distance on the communication of bacteria on a grid.

1.1 Research Questions

To call a system “alive” there are fundamental behaviours to consider such as movement towards a chemical trigger, reproduction, communication, response and death which are only some examples of these behaviours. This means these behaviours can be observed in any system that is alive and that makes studying these behaviours interesting. It is now possible to use computational and engineering methods to model these behaviours in order to obtain a better understanding of them. Modelling biological networks provide a better perceptive of the system while it saves time and resources. Constructing a collection of model components that contains the most simple behaviours could be helpful in providing an easy way to model networks for different purposes such as, in case of biological models, disease prevention or treatment. These models are abstract, they are reproducible and could be used for different studies. The motivation of this study is to create such a collection, to quicken the process of modelling and studying biological systems while finding the best combination of rates in the model in order to achieve the desired behaviour.

Considering the importance of microorganisms in human life, and their simplicity, these creatures are used in this study for the application of the model components. In a biological laboratory, the researchers should wait some time between 2 hours to weeks to achieve the desired results, excluding the human errors from the experiments. Using this collection of model components, this process is invigorated.

Not only modelling, but also analysing the models is a matter of importance. The models provide an abstract comprehension of the system and should behave as expected based on biological experiments observations. To get from modelling to analysis and achieving the desired outcome, optimisation is the bridge; which means using optimisation methods, it is feasible to achieve the desired behaviours.

Since biological networks are complex, it is easier to break the complicated networks down into small pieces and model them in separately. This means to construct a model that only contains the most important parts of that network. The motivation of this study is to:

- construct a library of models components to demonstrate the most basic biological behaviours such as communication, response, reproduction, movement and death;
- successfully combine the model components from the collection and achieve the expected behaviours based on biological literature;
- improve/re-design the components, and combine them with new models using engineering principles;
- Set an expected behaviour and achieve the best solution for it.

There are many methods and tools that could be used for modelling biological systems, which have been discussed in detail in Chapter 2 Section 2.5. In this study, Petri nets are chosen as the modelling tool and Snoopy as the platform as it supports different types of Petri nets such as Coloured Petri Net. Using this method a collection of abstract models and model components are constructed in different levels of complexity to be used individually or in a combination as required. Models that are constructed previously by other researchers have been also used in this study (see Chapter 4). To analyse the output data from the models, R is used and to check the accuracy of the model as well as to achieve an optimised solution for the defined target function, Python routines are employed (See Chapter 5 for the chosen prototypes. 6 for the methods of optimisation and Chapter 7 for the final results).

1.2 Aims and Objectives

Designing models of biological systems using engineering principles, provides a better comprehension of interactions inside and between the biological entities. These models also could be used to analyse or predict biological behaviours. The main purpose of these models is to study the biological components at a cellular level [99]. By using these models it is possible, for example, to predict a drug's influence on every molecule and each pathway that ends up forming the cell behaviour [7].

The aim of this study is to provide a general methodology to construct complex models from the combination of simpler components to describe some basic biological behaviours, using characteristics such as space and location to describe basic behaviours such as movement or communication. This study includes a step-by-step protocol that describes a method for modelling, a collection of properties that are the basis of modelling, stratified model components, categorised based on complexity and examples of application of the collection's components in different biological settings and optimisation for finding the best combination of rate constants for a desired outcome.

Looking at this methodology, from different angles, provides the objectives of this study which are to:

1. Construct a collection of model components in structured different levels of complexity. These reproducible models could be used to study different biological behaviours under different circumstances. From the aspect of system complexity, the models are four groups: one group of Properties and four Group of Models which are categorised based on their complexity:

   I Properties: The definitions and declarations of the models, such as Grid, Neighbour Function, *etc.*

   II Level-0: Simple Components which contain one place, one transition and the properties.

   III Level-1: Basic Models, which are the simple combinations of the Simple Components and a few properties.

    IV Level-2: Systems which are complex networks and a combination of a few Basic Components and/or Basic Models.

    V Level-3: Complex networks built from Level-2 models and a few properties.

2. Categorise the components based on biological complexity aspects into two groups:

    I Intercellular interactions, the interactions outside the cells such as:

      i. Movement and chemotaxis,

      ii. Duplication,

      iii. Death,

      iv. Communication with a focus on bacterial quorum sensing and Dictyostelium's accumulation around the signalling molecule cAMP.

      v. The response to the communication.

    II Intracellular interactions, the network inside the cell, such as:

      i. Production of signalling molecule. (This part of the model was constructed by Li *et al* and was assembled with one of the models from this study)

      ii. The interactions of the molecules inside the cell after the cell receives the signalling molecule.

3. Apply the models of chemotaxis, death, duplication and communication in two different domains of life, choosing one microorganism in each domain as case studies:

    I Prokaryote: bacteria in a general form. In this study the focus is not on any specific type or species of bacteria

    II Eukaryote: Dictyostelium known as "slime moulds".

4. Design a a step-by-step protocol adapted from a general protocol written by Professor Monika Heiner and Professor David Gilbert with introductions for modelling using Petri nets, Snoopy and Spike as the method as well as the guid to combine different components together.

5. Optimise the models using Random Restart Hill Climbing and Simulated Annealing methods as an employment of Python routine codes. Using optimisation, the aim is to find the best combination of rates in the model for a defined behaviour. This defined behaviour is explained in detail in 1.3.

An extended explanation of all of these models and how they are constructed and worked in in Chapter 4 and the biological applications these models can be found in Chapter 5. The models are stored as a structured data collection, categorised based on their level of complexity as CANDL files. CANDL files are exported from Coloured Petri nets, are small in size, can be edited simply and can produce ANDL files for unfolded models.

1.3 Novel Contributions to Science

The contributions of this thesis could be summarised as follows:

1. A reproducible structured library of model components to improve the understanding of biological systems. These models could be used to comprehend the biological behaviours and predict them under different circumstances. This library consists of:
   - I Constructed models in different levels, inter- and intracellular, and are abstract, which could be used individually or as a combination
   - II Abstract models of the most basic principles of life: movement, death, reproduction, communication and response.
   - III Detailed models of communication and response, taking quorum sensing and biofilm formation illustrated by examples of studying these behaviours.
2. A protocol of how to use and apply the models components from the collection to a variety biological scenarios.
3. A methodological approach for modelling biological network. Using Petri nets, Snoopy and Spike as a combination of tools and techniques to achieve the objectives.
4. Optimisation of a chosen model in order to find the best action rates to achieve a specific behaviour.

In detail, the novel contributions of this research is as follows:

1. **A reproducible structured library of model components to improve the understanding of biological systems as well as a protocol of how to use them.** These models can be used to comprehend the biological behaviours and predict them under different circumstances. After investigating about different potential methods and tools

for biological modelling [see Chapter 2], Petri nets are chosen as the tool to model this collection of model components. This library contains models which describe intracellular and intercellular levels of biological systems. For each level, a chosen behaviour is used as an application of the model components. The result is a library of abstract models of movement, reproduction, death, communication and response, and detailed models of communication and response. The models are designed in 2D and all of them could be applied in 3D as well. These sets of model properties which could be used for different scenarios by changing their rate and/or location. The model properties (explained fully in Chapter 4 Section 4.1) are the basics of the model. In other words, Properties of this collection are functions, definitions, coloursets, grid sizes and other variables such as Glu, Sink, Wall and pseudo-infinite grid. The model components could be used to study different scenarios of the desired behaviour. Also, by changing the rate of transitions or the location of these components it is possible to investigate the effect of these changes on the outcome of the model. [See Chapter 4 for the collection of components]

2. **protocol of how to use and apply the models components from the collection to a variety biological scenarios.** The process of design, construct and assemble of the models is explained in this study in detail. Using the components of the library, complex systems are created to describe multiple biological behaviours in one model. The models describe two different microorganisms from different biological superkingdoms: bacteria and Dictyostelium. [1] [See Chapter 5 for application of the components.]

3. **A methodological approach for modelling biological network. Using Petri nets, Snoopy and Spike as a combination of tools and techniques to achieve the objectives.** After investigating different potential methods and tools for biological modelling, Petri nets and Snoopy are chosen as our tools. Snoopy has a built-in simulator which could be used for abstract and simpler models. But in the case of 2D models on larger grids, Snoopy may take time for simulation due to its graphical user interface. That is why the developers of Snoopy, created a command line program which can promote the speed of simulation for the more computationally intensive models, called Spike. Spike provides csv data files which then could be analysed using different tools such as R. In this study, all the data files accorded from Spike are called "output data". [See Chapter 3 for the methodology used in this thesis.]

[1]In bacteria, the model of Biofilm Formation is not created from the components of the library as it was initially created in detail and combined with the model of signal production to exhibit quorum sensing as an intracellular communication and response.

4. **Optimisation of a chosen model in order to find the best action rates to achieve a specific behaviour.** The model chosen for optimisation is the Dictyostelium which is built from the components of the model library in this study. The models contain some constant rates which define the rate or speed of each action in the model. The expected behaviour is for the Dictyostelium cells to move towards a chemical trigger, cAMP, and accumulated around it. This behaviour has been observed in Dictyostelium in nature for the production of a fruiting body. For optimisation method, Random Restart Hill Climbing and Simulated Annealing are used as the methods in order to compare these two methods in finding the best possible solution which in this case is the highest possible accumulation of the most Dictyostelium cells in one location. The code is written in Python and could be reused by other researchers. [See Chapter 3 Section 3.1.4 and Chapter 6.]

1.4 In the Next Chapters

The following chapters will discuss this study in detail. Chapter 1 explains the aim and objectives of this study as well as the novel contributions it provides to science. Chapter 2 provides biological studies for a better understanding of biological systems that have been modelled. Also in this chapter there is a computational background study for different methods and tools which could have been used alternatively for this study and the reasoning behind the choices of tools and techniques for this study. Chapter 3 is the methodology that introduces the tools and methods used for modelling, data analysis and optimisation. Chapter 4 provides the collection of models and a protocol that includes all the required information to produce the same models in Petri nets and Snoopy and in Chapter 5 model components from the collection are used to be applied to real-life scenarios in two different microorganisms: bacteria and slime mould. Chapter 6 provides the algorithms for the python routines, using two methods of Random Restart Hill Climbing and Simulated Annealing. In Chapter 7 is the data analysis and visualisation of outcome data from the models with R and the result of optimisation of the models using python routines. Finally, in Chapter 8 a summary of the study is provided as well as some suggestions for further research in this area .

Chapter 2

Literature Review

In this chapter, the different fields of this study are thoroughly explained. In Section 2.1 systems and synthetic biology and how this work is related to these fields are described. Then in Section 2.2 the relatively-new concept of Bio-Model Engineering and how this concept have been used in this study is discussed. In Section 2.3 the reason for choosing each microorganism as the case study prototypes is explained. In the next section, 2.4, the biological behaviours that are chosen for this study are explained in detail. In Part 2.4.1, the type of communication and response and in Part 5.1.2 Duplication, Chemotaxis and Death behaviours are explained from a biological perspective. In the final two sections, there are two reviews: a review of the previous models that have been constructed in the same areas (Section2.6) and a review of other possible methods and tools that could have been used for this study. Then the reason of choosing Petri nets and Snoopy is clarified (Section 2.5).

2.1 Systems and Synthetic Biology

Biological and biochemical systems are complex, detailed, and in different levels of structure. These systems need to be deeply understood and examined and this understanding could be achieved faster and more efficient when there is a combination of biological experiment, computational sciences and engineering principles. This crossover of different areas of science creates systems and synthetic biology [24, 124]. Even though both systems biology and synthetic biology are the combination of these fields, there are some differences between them.

With the study of protein structure and DNA sequencing in 1950s, research in biology took a turn in how biologists analysed the biological systems and applied computational methods to understand the logical explanations behind the biological networks. This era was the beginning of systems biology in the following decades [209].

In systems biology - also called computational biology or computational systems biology, a better understanding of the living systems is the main objective. As a result of research in this field it has become possible to describe the dynamics of the complex biological systems in detail or abstract, analyse and/or predict them. In these systems, usually there are many elements interacting with one another, making complicated networks that result in forming a behaviour. Therefore, the result of this compound network, depends on each and every involved element [107, 186, 104].

Systems biology was initially used to provide an understanding of genes, reactions, and protein structures and localisation. But as time passed, it expanded to more complicated and detailed networks. Over time, systems biology developed synthetic biology with the main focus on reconstruction of biological systems for description and prediction of the results after the changes which mainly sourced inside the genetics of that biological network [104, 105]. The term "synthetic biology" was first used in 1912, however, it has recently become a theoretical and technical field of science that connects engineering, computing, maths and biology. In the synthetic biology, by using mechanical tools and biological knowledge, new organisms are created with new characteristics. In this field the scientists look at the biological networks as a system formed from different individual elements. These basic elements are used to create new systems [51].

With the development of systems biology, and its combination especially with computational studies, it has expanded biology from a descriptive research to an analytic and predictive approach [134] using models. A model could be a simplified version of a system, to ease the process of studying that system, or a detailed structure to provide a deep understanding of the details in the system. Some modellers such as Stan, believe that the model can only be the essential aspects of a network [181]. However, detailed models have also proven to be useful in the last years, especially when it comes to biology.

Materi and Wishart explain that a computational-biological model is only useful when it can save money and time, could be understood better than the pure biological system, the results could be matched with biological data and it can identify missing parts or components in a network [134]. The systems biology models have been successful in different fields such as evolutionary biology, epidemiology and ecology [104]. A model with these features could be used for different purposes such as drug discovery, personalised medicine and personalised nutrition projects [157, 134, 107].

In computational and systems biology, the biological systems could also be predicted with simulations and data analysis while the accuracy of the results could be examined in biological laboratories with the quantitative and qualitative data provided by the model. The goal is to design a model which can describe, analyse and predict the system's processes and its results under various circumstances [20, 107].

2.2 Bio-Model Engineering

At the intersect of modelling, biology, computer science, mathematics and engineering, Bio-Model engineering lays. As a part of both systems and synthetic biology, in Bio-Model Engineering the models are constructed, designed and developed in order to study, analyse and predict the enquired systems under different circumstances [99]. In this field, different versions of the same model are developed in order to study the hypothetical conditions in each version [88].

One of the most important aspects of Bio-Model Engineering is to construct a library of model components which is generalised enough to be used for different studies. These models can be in different scales, and different levels from molecular to whole-cell structures in space and time [22] and could be used individually or as a combination. The models created in bio-model engineering then go through structural analysis and model checking [23, 88].

This study is closely similar to the aspects of Bio-Model Engineering. Here, using graphical methods, a library of model components from the most basic concepts of life in an abstract setting is constructed. Based on the meaning of "alive" in Wikipedia, anything that could be called alive needs to have seven characteristics: Homeostasis, organisation, metabolism, growth, adaptation, response and reproduction. The whole article can be read at Wikipedia. In this study, These aspects are modelled as below:

- Movement as a *response* to a chemical trigger, i.e. chemotaxis;
- *Reproduction* which in this study is duplication of unicellular microorganisms;
- Death as a part of *growth*, as it is a change over time;
- Communication and response to the communication which are parts of *Metabolism* and

Response.

Using Petri nets, detailed illustrations of the biological systems are shown which are easy to understand and follow.

2.3 Choice of Microorganisms For Modelling

Microorganism is a term used for microscopic organisms which might be unicellular or live as a colony or a cluster that can include microorganisms from both biological superkingdoms: Prokaryote and Eukaryote (based on two-empire classification system). These microorganisms include bacteria, archaea, amoeba, protists and even viruses which are non-cellular organisms. Prokaryotic, literally meaning "before Kernel" are the organisms that do not have a nucleus surrounding their genetic information. Eukaryotic, the "true nucleus", have a membrane separating the DNA from the rest of the cell, making these cells more complex than Prokaryotes. [168, 25].

Prokaryotic cells are divided to two major evolutionary kingdoms: Bacteria (=Eubacteria) and Archea (=Archaebacteria). The latter has not received much attention since they usually live under specific circumstances and are difficult to culture in the laboratory. But the bacteria kingdom, has been studied well since they can grow almost everywhere, are used in industry, and most importantly have interactions with human body [25].

Interestingly, most of the domains of microorganisms from both superkingdoms could be found in human body[179, 192] which makes studying them an important area of science, especially from human health and industry perspectives. The microorganisms that are in direct interaction with human body are rather symbiotic or pathogenic.

In this study, two different microorganisms are chosen to build the models based on them as the prototypes or case studies. These two microorganisms are bacteria from the Prokaryote kingdom and Dictyostelium a.k.a. slime mould from the Eukaryote kingdom. In Section 2.3.1 the importance of bacteria in human life and in Section 2.3.2 Dictyostelium is and what makes it a good prototype, are explained

2.3.1 Bacteria

There is a huge population of symbiotic microorganisms living inside and on the surface of human body, containing around $10^{13}$to $10^{1}4$ cells. These microorganisms have been living with us ever since birth [98, 172, 46] and their construction have been developing since then [46]. These microorganisms, exceed the number of our own cells with the ratio of 1 to 10 [174, 11]; however, there are some debate about this ratio as some report it as 1 to 1 [27, 174]. In 1960, these bacteria started to be known as “mircobiota” and “microbiome” [158]. In this study they are referred to as “microbiota”. Microbiota could be found almost everywhere in the body, nonetheless, the concentration of this population is inside the digestive tract. Most of these microorganisms, around 1014 CFU/g, are bacteria [194, 35] [CFU = Colony-forming unit].

Microbiota have a direct interaction with human cells, wherever they live, which is why studying them is crucial. While they provide vitamins and effect the behaviour and body’s function [111], they can be used to prevent some diseases or, they can cause them [189, 192]. Recent studies have confirmed that microbiota is related to brain development [132], irritable bowel syndrome (IBS) [56], liver disease [130], oral cancer [208], colorectal cancer [130, 102, 35, 53], diabetes [70], Crohn’s disease [27], Parkinson’s disease [171] and encompassing ulcerative colitis [85]. Accordingly, microbiota plays an important role in human well being.

The digestive system contains the greatest population of microbiota, becoming one of the most important populations of microbiota in human body and is known as gut microbiota. The construction of these microorganisms is unique for each individual and is directly influenced by diet, although it also depends on lifestyle, use of antibiotics, infections, genetics, stress and other factors. The gut microbiota modifies during childhood and eventually is stable throughout adulthood [111, 188, 145].

In the past decades, most of the studies on gastrointestinal diseases was about pathogens and how they cause diseases; but while these microorganisms can cause diseases, they could be also beneficial [128, 190]. Recently the impact of gut microbiota on human health, their modifications and their usage for human health improvement and treat diseases, have become an interesting field of study [172, 56]. In fact, studies have shown that the gut microbiota play essential roles for humans to survive [188, 9, 189]. Gut microbiota and the host’s immune system interactions assist the immune system to develop, while the immune system in return aids the microorganisms with their composition which results in a complicated vast network, that connects brain, liver and gut and other organs all together for a better function of human body [145, 133, 97].

Another important population of bacteria in human body, is the oral microbiota; the bacteria in oral cavity. In 1970s a study on human dental plaques showed that oppose to the common belief of that time, bacteria do not live completely free and independent. They communicate with each other, build a society and adhere to their surface with biofilms [175].

There is no stop for biofilm formation, since it is a natural process in the bacteria. Biofilm formation can be "disturbed" with brushing. If left undisturbed, The biofilm becomes confluent [13]. Biofilm made by bacteria on the teeth and gums can cause plaques; and plaques can cause dental caries or periodontal disease [218]. In a healthy person there is a balance between the gum tissue and the biofilm formation and there is a symbiotic relationship between the host and the bacteria. But when this balance is elapsed there will be some problems for human health [175].

Biofilms are the result of the microorganisms communication, especially bacteria. The communication, known as quorum sensing, happens when the concentration of the microorganisms increases. Quorum sensing and Biofilm formation are one of the main focuses of this study, as an example of communication and the response to it. The detail about Quorum sensing and biofilm formation is explained in section 2.4.1.

Considering bacteria's importance from many aspects, as well as the simplicity of data collection about this domain, bacteria was chosen as one of the prototypes. When modelling detailed models are bacterial communication and response, the bacteria species should have been specified. As *E. coli* is one of the best recognised bacteria in the domain, the detailed models are built based on this species.

2.3.2 Dictyostelium: The Slime Mould

Among Eukaryotes plants and animals are the most known which are multicellular organisms. However, there are many unicellular species in this superkingdom. Dictyostelium (Dicty) is a social amoeba that is unicellular in parts of its life cycle, and multicellular in other parts, making this species very interesting. It is one of the well known microscopic Eukaryotes commonly known as the slime mould [112, 200]. Dicty is one of the most studied model microorganism for chemotaxis due to its similarities to mammalian cells [144].

Naming Dicty as a social amoeba is because this unicellular microorganism communicates with other Dicty cells in the area. Under normal circumstances, when there is no pressure, Dicty moves around randomly in a diffusion-like behaviour [63, 200]. But under pressure or starvation they gather together and create a multicellular aggregated society to survive. As a result of this communication, a fruiting body is created which bears the spores. In the final stage of the life cycle of this amoeba, the spores are detached from the fruiting body, off the tip of the fruiting body structure [59, 92]. The main focus in this study, is the stage when Dicty cells communicates and aggregates and eventually create the fruiting body.

The signalling molecule produced by Dicty is called cyclic adenosine monophosphate (cAMP). The mechanism of this communication is similar to bacteria and hence it is also called quorum sensing: the communication based on the population density. Under pressure, Dicty cells produce cAMP into the intercellular environment. Reaching a threshold, the cells start moving to where the concentration of the signalling molecule is the highest and create the multicellular structure [59, 200].

Even though the mechanism of the communication is the same in bacteria and Dicty, the physiology of the movement itself is completely different. In bacteria, the movement happens with the help of motility motors called flagella. However, in Dicty, as it is an amoeba with the ability to change its shape, it slides on the surface by creating extension of its plasma [144]. In this study the mechanism is the focus and not the physics of the movement and how it happens. But this difference could be helpful in further modelling, when the focus is more towards the physical characteristics of movement rather than the interactions between the cells.

There is another difference between the movement of the two microorganisms in the collection of models in this thesis. In bacterial movement, the trigger of the movement, a.k.a. the food source, is placed in one side of the modelling grid and the food molecules diffuse on the grid. the bacteria sense the food and move towards it. As a result, the concentration of bacteria is, at the end, around where the food is located. On the other hand, in the Dicty model of movement, the source of the chemotaxis trigger is the Dicty cells that produce cAMP. So, while they are producing these molecules, they move as well and slowly their random movement changes to chemotaxis as the concentration of the signalling molecule increases at some point on the grid. This means the location of the final lump of Dicty on the gird could be changed on every run of a stochastic model. This is explained in more detail in Chapter 5.

2.4 Chosen Behaviours

There are many behaviours in biological systems that could be modelled. In this study some of the most basic behaviours and examples of them that happen in microorganisms are chosen to be modelled. As mentioned before, communication and response to it are two of the chosen behaviours in this thesis. As an example of this behaviour quorum sensing and biofilm formation in bacteria were picked and modelled in detail. The other behaviours include duplication, movement/chemotaxis and death which are modelled in abstract. This section explains each of these behaviours in detail.

2.4.1 Quorum Sensing and Biofilm Formation

In this section quorum sensing and biofilm formation processes are explained. As the models of these two behaviours are in detail in the collection of components, these two are elaborated in detail, with mentioning all the biological parts that are used in the models.

Microorganisms tend to communicate with one another in different ways. One of the well studied communication methods is quorum sensing (QS). QS is a cell-to-cell communication based on cell density in the environment. To communicated, microorganisms produce signalling molecules and receive them simultaneously as soon as the density of the molecule is reached a threshold [80]. There are many different responses to this type of communication, varying based on the organisms. For example, in bacteria one of the many outcomes of QS is the formation of an intercellular matrix called biofilm and in Dicty it is aggregation of the cells together [112, 68]. This responses are as a result of regulations in gene transcription and cell behaviour [72, 58]

This chemical broadcasting communication is dependent on density of the communicators which is why it is called quorum sensing: communication when the density of cells around reaches to a quorum [150, 103] QS was first observed in early 1970s [84]. It is related to virulence, bioluminescence, swarming, motility, antibiotic resistance and biofilm formation [103, 191].

Signalling molecules in QS are the language between the microorganisms [217]. The microorganisms who have the suitable receptor will understand and respond to this language. In other words, if they have the receptors for that certain type of the molecule receive it, they are able to respond to it. These diffusible molecules are known as autoinducers(AI) in bacteria [191, 5].

To this day three different classes of autoinducers have been discovered: acyle homoserine lactones AI-1 produced by Gram negative bacteria, peptide based AI-2 the inter-species signalling molecule produced by both Gram negative and Gram positive bacteria and aromatic AI-3 which is the inter-empire signalling molecule for cell to cell communication between bacteria and their host [94, 43]. Vast studies have been done on AI-1 and AI-2 as they are the most common types of autoinducers among bacteria. Though, not all strains of bacteria produce all these types [184, 91, 103]. Some bacteria produce only one type or even none at all; but they may have the receptor for them. For example, *E.coli* does not produce AI-1 but it can respond to it [203, 90, 103].

Among different types of autoinducers, AI-2 is the inter-species signalling molecule among different bacteria which can be produced and detected by both Gram positive and Gram negative bacteria and acts as a common language for them[150].

At the beginning of the journey of modelling biological systems, the first model constructed was a detailed model of quorum sensing and biofilm formation. This model, which the steps of its construction is explained in detail in 5.1.5, is made of two parts: quorum sensing, constructed in 2006 by Li *et al.* [114] and biofilm formation which was constructed by our group [77]. As this model is detailed, in this section quorum sensing and biofilm formation mechanism iare explicitly explained.

The Gram negative bacteria *E.coli* is one of the well-known microorganisms that could be found almost everywhere, including human body as a pathogen or a commensal bacteria [193] which produces AI-2 through a particular pathway that involves an enzyme called *LuxS* which is the starter protein for the production of AI-2 from methionine [211, 50].

Wang *et al.* [204] and Sun *et al.* [183] explain the pathway of the production of AI-2 in *E.coli* as follows: Methionine adenosyl Transferase *(MetK)* changes Methionine to S-adenosine methionine *(SAM)* by using an *ATP* and changing it to *AMP*. *SAM* can be affected by two enzymes with two different destinations:

1. *SAM* can be a methyl donor for Methyl Transferase enzyme. This enzyme effects on a methyl acceptor, which causes a production of methyl. Here, *SAM* itself is changed to S-adenosyl-homocysteine *(SAH)*. *SAH* can be toxic if accumulated. Therefore, the cell rapidly changes it to adenine and S-ribosyl homocysteine *(SRH)* by Pfs, which is anucleosidase. Pfs here uses a water molecule to synthesise a polyamine. Then *LuxS* acts on *SRH* and causes the production of homocysteine. This homocysteine can be recycled to methionine and 4,5-dihydroxy-2,3-pentamidine *(DPD)*. The second product can undergo

some rearrangements to produce AI-2 [204] by *LuxS* protein [166]

2. The other pathway that *SAM* can undergo, wouldn't end up to produce AI-2. *SAM* decarboxylase *(SpeD)* would change *SAM* to Decarboxylated SAM *(De-SAM)* by releasing CO_2. After that Spermidine synthase *(SpeE)* would act on *De-SAM* and change it to 5'-Methylthioadenosine *(MTA)* while changing Putrescine to Spermidine. The same *Pfs* enzyme would act on *MTA* and would change it to 5'-methylthioribose *(MTR)* while taking a water molecule and synthesis of polyamines [204].

As soon as the entrance of AI-2, it is phosphorylated by a Kinase named *LsrK* (LuxS Regulator Kinase). This phosphorylated AI-2 appears to be the factor, which effects on the transcription of *LsrR* (lsr transcription Regulator), a DNA binding transcriptional regulator [113]. Phosphorylated AI-2 binds to *LsrR* and makes it separated from the operon of *lsr* genes (LuxS regulator genes). On the other hand, Phosphorylated AI-2 binds to a two-component regulatory system (a coupling mechanism to sense and respond to environmental changes), *QSeBC* (Quorum Sensing *E.coli*regulator B and C) to control biofilm formation. When non-pathogenic *E.coli* produce AI-2 in the intestine, it induces an inflammatory response in the body, which is quickly stopped. Since AI-2 can act as an inter-kingdom signal, when AI-2 system is activated the signal molecule is quickly removed from the environment to stop other strains of bacteria to use this molecule and prevent the change of their behaviour through this molecule. In *E.coli*, AI-2 effects on virulence factors, motility, pathogenicity and biofilm formation [204, 166, 148]. Figure 2.1 shows a simple model of what happens in the cell after AI-2 is imported.

Considering the bacterial life which includes four different phases, the production of AI-2 happens in the middle to the end of the second phase, exponential growth phase [91]. When AI-2 is produced it is exerted to the extracellular matrix to congregate there. The more number of AI-2 producers, the more density of AI-2 in the environment [82]. When it reaches a minimum threshold [90], the bacteria discover that there are enough of them around to start detection and import of the signalling molecule [139, 184, 141]. Not all types of bacteria in the environment can produce AI-2, but the ones who have the receptor have the ability to detect it [183].

The changes start when AI-2 enters the cell through a protein complex gate called *LsrABCD*. AI-2 activates the two- component system Quorum Sensing *E.coli* regulator B and C (QSeBC). *QSeBC* has two proteins: one is bound to the membrane and is a histidine kinase sensor for

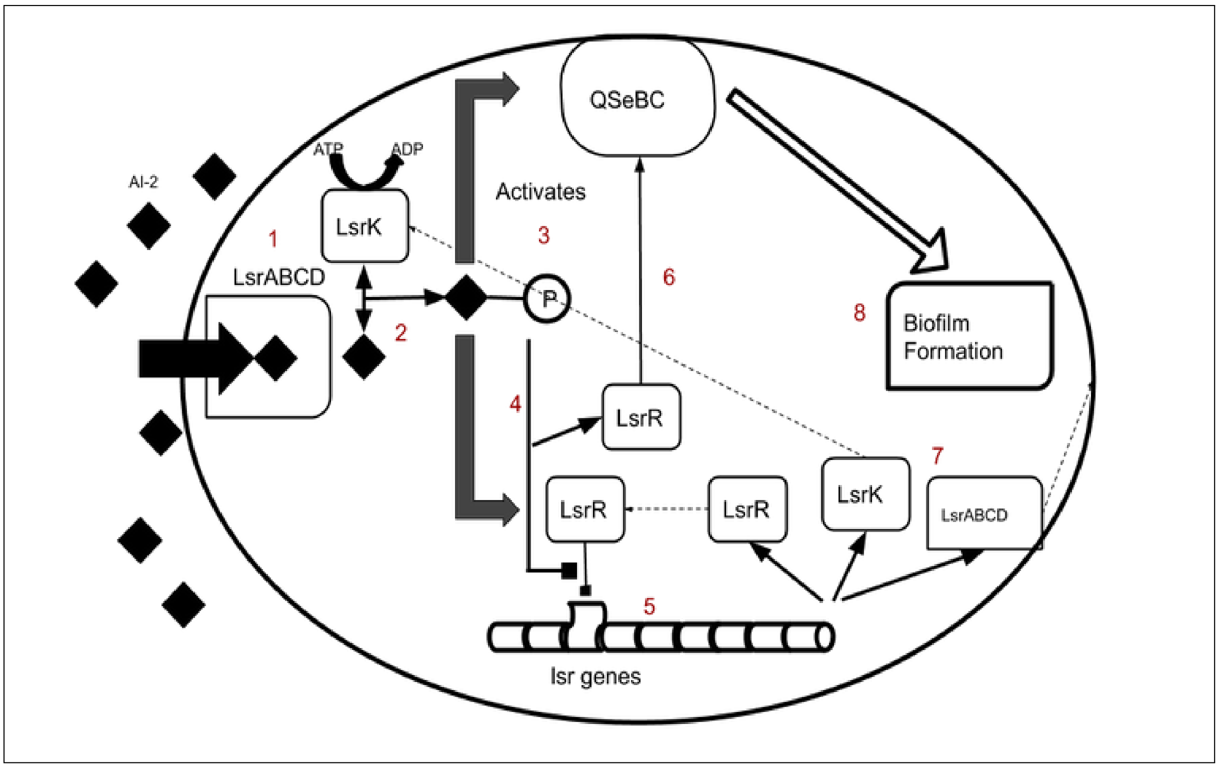


Figure 2.1: As soon as the AI-2 enters the cell through *LsrABCD* (1) it is phosphorilated by *LsrK* (2). The phosphorilated AI-2 activates *QseBC)* (3) or is attached to *LsrR* (4) and separates it from the *lsr* genes operon so now the genes can transcribe (5). The result of the depression of *lsr* genes is the production of proteins of *Lsr* family such as *LsrABCD*, *LsrK* and *LsrR* (7) to involve more AI-2 molecules. *LsrR* interacts with *QSeBC* for biofilm formation (6) and also *QSeBC* is activated by AI-2 which both of them result in biofilm formation (8).

environmental changes and one is a regulator for target genes. This system is capable of activating the process of biofilm formation [180, 148, 43].

On the other hand, the imported AI-2 is phosphorylated by *LsrK* upon entering the cell. The Phosphorylated AI-2 *(AI-2-P)* binds to *LsrR* on the prompter of *lsr* genes and separates the promoter from the gene. *lsr* genes (LuxS Regulator) are a group of genes with one promoter (the transcription start area) in the DNA of bacteria. One of the proteins that are made from the expression of this family gene, is *LsrR* (lsr Regulator Protein). In the quorum sensing-deactivated situation, when AI-2 is not inside the cell, *LsrR* is bound to the promoter to stop the transcription [95, 113].

Now that the promoter is free, the transcription of *lsr* gene family, which includes *lsrA, lsrB, lsrC, lsrD, lsrK, lsrR*, starts [148, 113]. The results of the transcription are:

- More *LsrABCD* will be produced, so more AI-2 can enter the cell,
- More *LsrK* will be produced, thus more AI-2 can be phosphorylated, and
- More *LsrR* will be produced so that more AI-2-p can bind to it.

Novak states that *LsrR* also can activate *QSeBC* and cause biofilm formation [148]. However, it is not clear how *LsrR* can activate *QSeBC* while bound to the promoter, or it is after it is separated from the promoter by AI-2-P. It is also unclear if AI-2 is it consumed when activating *QSeBC* or it acts as a catalyser to activate this two-component system. On the other hand Li explains that that "as soon as AI-2 enters the cell it is phosphorylated by *LsrK*" [113]. This causes an inconsistency here. If AI-2 is phosphorylated "as soon as" entrance when does it find the chance to activate *QSeBC*. These unclear points made some problems for us, to model biofilm formation in the cell by AI-2.

The first observations of biofilm goes back to 1683 when Antoni van Leeuwenhoek described the bacteria from his dental plaques as "an unbelievable great company of living animalcules, a-swimming more nimbly than any I had ever seen up to this time, the biggest sort bent their body into curves in going forwards" [175, 64]. However, the study of biofilm did not start until the late 1970s when the biofilm was accidentally observed on dental plaques. In late 1970s after quorum sensing rejected the theory of planktonic lifestyle of bacteria, biofilm as a matrix protective layer consisting of societies of bacteria was observed [175].

Biofilm formation is one of the results of quorum sensing among microorganisms [167]. Biofilm is a hydrated matrix polymer layer of extracellular polymeric substances (EPS) which its components are DNA, proteins and other small molecules [212, 68]. Bacteria secrete this protective layer to attach to the surface and/or each other [65] in order to communicate, improve life style and survive by making them resistant to antibiotics and other environmental changes [116, 94] while being a diffusion barrier for small molecules [123, 129]. Interestingly, as biofilm is a surviving plan in case of starvation [4], the metabolic activity of the cells in biofilm decreases notably. The reason is to decrease the needed energy for metabolic pathways while being protected inside the biofilm [35, 54].

Bacteria who are embedded in biofilms are resistant to antibiotics for different reasons such as the production of the matrix, modified metabolic pathways, gene transfer between bacteria or higher anti-oxidative capacity [175, 15]. Recent studies have shown that the amount of required antibiotic to destroy biofilm is 250 times more compared to when the bacteria is not surrounded by biofilm [137]. By using too much of antibiotic on bacteria, they improve antibiotic resistance as well. These studies prove why biofilm is not treatable with antibiotic [175]. As the relationship between biofilm and pathogenicity gets more serious, researchers focus more and more on biofilm formation in different pathogen bacteria [49, 182, 201].

It has been reported that biofilms are related to infections in teeth, urinary tract and skin [110, 123, 30]. It can cause a problem as simple as plaques formation on the teeth or as complex as cancer in different parts of the body [208]. Chen *et al.* [35] showed the relationship between colorectal cancer and biofilm formation due to chronic inflammation caused by biofilm. Because of biofilm, there might be chronic infections, failed treatment or slow healing processes [136]. However, it should be noted that biofilm *per se* is not the problem [115]; it causes the problem when it is rather produced inside the mucus layer of digestive tract, or slows and/or prevents the treatment strategies or causes infections in the body [160, 27]. That said, biofilms are not always trouble makers. for example in case of dental plaque, they are not necessarily causing any diseases, if the tissue and bacteria are kept in a balance [175].

Biofilm is not always in biological systems and in microbial scale. They can exist anywhere, on the hulls of the ships, inside the pipes or on ants' bodies, or even on food [206, 8]. With the recent development of using bacterial products in industry, biofilm has been used for beneficial purposes for instance, in waste water treatment [212]. Interestingly, research shows that biofilms can be quite advantageous for some plants' growth [176]. Biofilm is not always corrupting for human either. Commensal bacteria in human gut produce biofilm naturally as well [115, 27]. In the field of design, the researchers have worked on modelling a position-based dynamic design for bacteria growth. They used synthetic biology and engineering to model biofilm formation and show the growth, communication and interactions with environment to find a way to use biofilm's features for human use [12]. All these said, biofilm can be beneficial or harmful depending on the place it is formed and the amount it has been produced.

Elias and Banin [64] show that there can be three types of structure for biofilm: They can be two different biofilm societies separate from one another, compact together or layered. There are not enough studies about the different types of the biofilm construction and different components in different bacteria but generally biofilm consists of proteins, small DNA and polysaccharides. To stop the infections and problems caused by biofilm different strategies such as using bacteriophages (phage therapy), dietary changes to adjust microbiota composition, prevention or suppression of biofilm has been suggested [35, 115]. In the figure 2.2 the three different types of biofilm formation is shown:

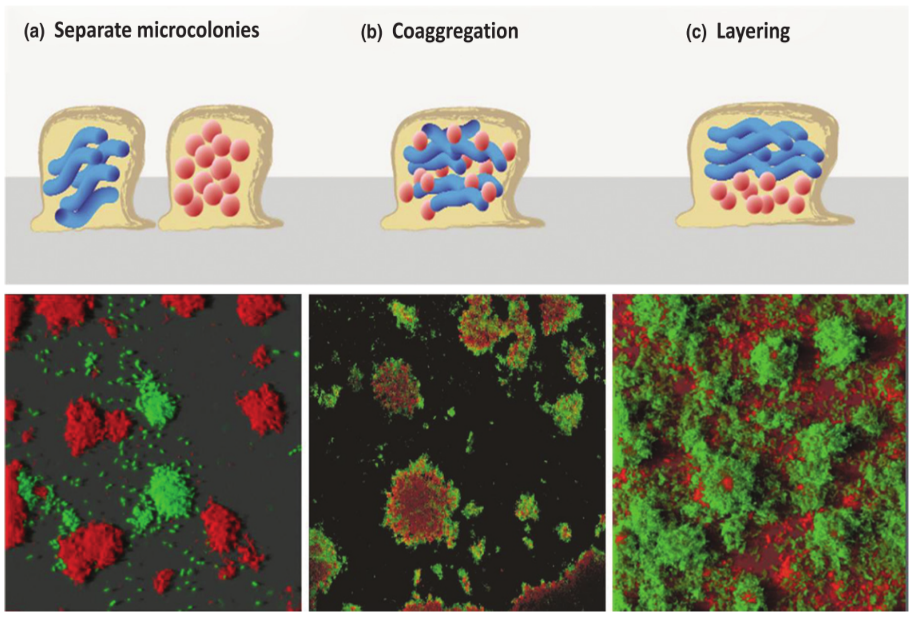


Figure 2.2: Spatial distribution within mixed-species biofilms. species is mixed in biofilms can organise in several ways: (a) separate mono-species microcolonies, (b) co-aggregation (c) arranged in layers. from [64].

2.4.2 Duplication, Chemotaxis and Death

Both eukaryotic and prokaryotic cells have the ability to sense the trigger molecules in their intercellular space which results in moving towards that molecule [135]. The movement in microorganisms has been studied over the last years and is not a new subject. The trigger molecule can differ from a nutrition source to a signalling molecule [2, 55]

It is not a new discovery that proves bacteria recognise the food source and migrate towards it [2]. In case of chemotactic movement starting with attraction towards a food source, some bacteria duplicate as a part of their life cycle, especially when the concentration of food is higher [147]. Also, unlike the common thought, bacteria are not immortal. Studies show that in nature, when the circumstances are favourable for the bacteria they get old and lose the ability to reproduce. Studies confirm that bacteria could go to the death phase of life cycle, while having enough nutrients, only because they have aged and lost the ability of reproduction [81]. These studies show that the bacteria move towards the food, reproduce which is age-dependant and die in the environment which this situation is another the aim of the study to be shown in a model. The model in this thesis is in a chemostat (stable nutrient) phase; which means food is provided to the bacteria constantly. Therefore, in this model the death of bacteria happens due to ageing rather than lack of nutrients.

In other cases such as Dicty, the movement might be towards the food or towards the high concentration of signalling molecules. As a survival strategy, Dicty cells first move towards their food, which can be bacteria or other sources, and when they are starved, they start to produce and export the signalling molecules and then migrate to the highest concentration of the molecule in their environment [59]. Similar to bacteria, Dicty undergoes binary division when the nutrition is available and the growth rate decreases when it is starved in order to conserve the energy for movement towards the food [122, 216].

2.5 Bio-modelling Tools and Techniques

Modelling biological systems in biology saves time and resources while providing a detailed understanding of complex biological systems. Models in systems biology can be in various levels and scales, from atomic structure [153] to genome scale [131]; from cell structure to multi-cellular organism [74]. They can describe biochemical networks as well as protein-protein interactions. Modelling does not end here. It can be used to describe receptors of a cell, movements and reproduction of cells or in a higher level, tissues and a whole organisms. while all these models propose a valid prediction of the system [202, 107]. There are many tools and techniques that could be used for these purposes.

After modelling the system, they can be engineered and adapted with synthetic biology. Unlike systems biology in which only the biological networks are described, in synthetic biology some changes are initiated in order to reach a specific result. The change can be inhibition of a pathway or introducing a new gene to the cell [93]. The goal of this section is to improve the knowledge of different computational tools and techniques for modelling biological systems and comparing them to reach a final conclusion: which technique is the best for approaching the aim of this study to model communication theory in biological systems. A summary of this review in the table in section 2.5.10, where the tools column shows the software tools which are a platform for a specific framework and Method column shows the method(s) these tools use, such as ODEs and PDEs, etc, Table 2.1.

In this section the other potential tools as well as the reason why these tools were not chosen for the objectives of this study are discussed. Also, in the end the reason for the choice of tool is advocated.

2.5.1 Differential Equations

Differential Equations (DEs) are mathematical methods for modelling. For many years this method have been one of the primary choices for biological modelling. DEs are usually used for predictive modelling when one of the factors is time, in order to analyse the behaviour of a system over time. This method consists of two parts: functions and derivatives; the physical quantities and the rate of change and the equations represent the relations between the two. *i.e* it shows how the derivatives are related to a function based on one or more variables The function dictates the rules of the model. Considering that the rules are set in the equations, the outcome of the models based on the same equations is always the same [47, 149].

If the DE has only one variable it is called Ordinary Differential Equations (ODEs) and if it has more than one it is Partial Differential Equations (PDEs) [149].

Ordinary differential equations (ODEs)

ODEs are mostly used for modelling a continuous behaviour of a biological system over time [202, 47]. They also can be used for spatial modelling, using mathematically defined compartments. These compartments need to be in a 'well-mixed' defined interaction [47]. The general form of these equations is as Equation 2.1 where shows the evolution of x over time t based on functions $u(x)$ [149]. To create a continuous quantity, the number of components need to be significantly high [47].

$$\begin{aligned} \frac{d^4u}{dx^4} + \frac{d^2u}{dx^2} + u^2 = \cos x \\ variables = u(t, x, y) \end{aligned} \tag{2.1}$$

Partial Differential Equations (PDEs)

While ODEs are the evolution of one variable over time, PDEs are multi-variable functions. In these functions, one derivative is constant while the other ones are changing. PDEs enable modelling space that could be used for modelling biological systems such as diffusion to show the changes over time and space [47]. A simple PDE is shown in Equation 2.2 in which u is the function and x, t and y are the variables [149].

$$\begin{aligned}\frac{\partial u}{\partial t} &= \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2}{\partial y^2} - u \\ variables &= u(t, x, y)\end{aligned} \tag{2.2}$$

Why not DEs? The use of DEs have been increasing in the past few years for engineering, biology and mathematics [47, 149] but they have their shortcomings. In case of ODEs, there is only one variable against time. This means geometry and space could not be added to the equations. For a continuous behaviour the number of components should be high, which is not achievable for all scenarios. The problems with space, however, can be tackled by using PDEs. But comparing the two in the speed of simulation, ODEs are usually faster. PDEs, being more complex with more variables, take a much longer time to simulate. If the models are too complex, the simulation is too computationally intensive [47].

2.5.2 Agent-Based Models (ABMs)

ABMs are a bottom-up modelling method that use encapsulated agents in an environment with defined rules. Agent-based models are used for many purposes such as marketing or biological systems modelling [214, 127, 16] and is one of the most-used methods in systems biology [83] especially when equation-based methods are not the ideal approach as the details of a system are unknown [6], since it is a behaviour-based adaptive modelling method [7, 213]. This means that the model, following the defined rules can change the behaviour based on the outcome or even create new rules [126]. In this method agents are introduced that are components with specific characteristics and rules that are traceable. These agents are the adaptive part of the method that are intelligent and have decision-making abilities and can change based on the environment or create new rules [126]. In the models, usually the agents are placed next to other agents that may interact with one another or with the environment. These interactions might adapt over time [126]. Unlike other methods that are usually models that create data, ABMs can be built based on data. This means the model can look for a pattern in the data and defined the rules of the model based on that [7].

ABMs could also be used to design spatial models, usually in 2D grid with spatial details [7, 178]. They are mostly used for stochastic modelling and new rules could be added to the model, by adding new agents that changes the existing agents [7].

Why not ABMs? As it was mentions, each agent in ABMs carries a set of rules, encapsulated in an environment. This means carrying a lot of information that makes the simulation quite intensive computationally [6]. The focus of ABMs is on the differences between the agents, but when there is too much of difference between the defined information for each agent, not only this slows down the simulation but also removes ABMs as a choice when it comes to designing a homogeneous model [213]. (See also: Pros and cons of Agent-Based Modelling).

2.5.3 π-Calculus

π-Calculus is a mathematical process calculus method from process calculi or process algebra family for distributed concurrent systems, introduced in 1980 to describe theoretically interactions and communications in biological systems when the resources in the model change over time [18, 197, 108, 199]. It is also ideal for designing large systems using smaller subsystem [156].

π-Calculus uses channels as a communication link between pairs of processes but these channels can only be in pairs. This means for modelling broadcast communication, π-Calculus probably is not the ideal tool [199]. Since π-Calculus is compatible with Gillespie simulation algorithms, it is one of the most reliable tools for modelling biological systems stochastically [32, 210]. Stochastic π-Calculus are modelled independently as components, instead of a model of individual reaction which simplifies the models and, therefore, the simulation. This means that the models are independent from their environment [32, 161]. π-Calculus also allows new names to be introduced to the modelled system and allows the model pass complex messages [125]. In a comparison between π-Calculus and Petri nets, it was discussed that Petri nets are more reliable for modelling biological models containing space [197, 29].

An example of a simple model created by π-Calculus is shown in Equation 2.3 from an introduction to π-Calculus written by Parrow [151]. This models expresses that the server that sends a along b is $\bar{a}b$. The client that receives some link along b is S. Then S sends data along $b(c)$. The interaction that is being modelled here, when the client sends a signal to a printer, is shown as $\bar{c}d.P$. There are some extension of π-Calculus as well such as stochastic π-Calculus [159] or applied π-Calculus [1].

$$\bar{b}a.S|b(c).\bar{c}d.P \xrightarrow{\tau} S|\bar{a}d.P \tag{2.3}$$

Why not π-Calculus? Even though π-Calculus has been used for biological modelling, it shows a weakness when it comes to modelling communication as π-Calculus cannot model communication unless the communicators are paired. π-Calculus does not have a graphical user interaction (GUI) and is based on mathematical equations to define the behaviour of the model. Being a mathematical technique, π-Calculus is only limited to a range of researchers and scientists who can understand and work with it [156]. Similar to DEs, in complex models, the simulations are computationally intensive, especially when the models include space [199].

2.5.4 λ-Calculus

Introduced in 1930s by Alonzo Church, λ-Calculus is one of the simplest and 'smallest' programming languages [66, 164]. This programming language consists of one function and one transformations rule. Being a universal language that can express any computational function, it is equivalent to Turin Machine. λ-Calculus defines the expression by a name or a variable application and function [164]. In λ-Calculus the main concept is a "name" or a "variable" which is an identifier of an expression [164, 66]. λ-Calculus is simpler than π-Calculus since the only keywords in this language are λ and dot. Also, Unlike π-Calculus functions are not given any names λ-Calculus. So every time the modeller wants to use a function, they should write the whole function again [164, 138]. A simple model in λ-Calculus can be seen in Equation 2.4 from a work done by Rojas [164]. In the function definision, E is the same expression and λ is the identifier. In this model, the first line is the expressions, the second the identity function, the third the application and the final line is the body of the function definition.

$$\begin{aligned}(\ldots((E_1E_2)E_3)\ldots E_n)\\ \lambda_{x.x}\\ (\lambda_{x.x})y\\ (\lambda_{x.x})y=[y/x]x=y\end{aligned} \tag{2.4}$$

Why not λ-Calculus? λ-Calculus is a very simple programming language. It has not been used often by the biologists in the past few years either. Considering that the functions are not given any names, it can only be used for simple models [138]. Also, similar to π-Calculus it is not suitable for users without mathematical or computational knowledge and does not provide GUI.

2.5.5 Statecharts

Used by Agent-oriented Software Engineering (AOSE), statecharts has been used for modelling interactions. Statecharts are a programming language for modelling complicated reactive biological systems in which the structure is hybrid with continuous and stochastic behaviours consisting many interactions. The execution of the statecharts begins at the *start* state and continues as a sequel of steps. Statecharts was originally introduced by David Harel for modelling software systems. Statecharts are hierarchical dataflow diagram or an activity-chart which each state presents an activity representing the main activities, *i.e.* AND, OR states or other basic states [79]. In the statecharts the Reactive systems interact with their environment and are not independent from it. A reactive system does not behave based on a pre-prepared chain of instructions. Instead the system reacts in parallel to the new inputs, which is the output of the reaction before [67]. The statecharts have been used for modelling T-cell activation and development in thymus before. But the chain linked interaction structure of the language is useful to model DNA and the other molecules interactions in a cell as well. Using statechart it is possible to model a biological system's behaviour in time and it can show how the objects in the model interact and change under different circumstances. Statechart has GUI but it can have other approaches for formalism based on the modeller's need [67]. Figure 2.3 represents a simple statechart of an air conditioner created by Spanoudakis that was published in 2020 in Lecture Notes in Artificial Intelligence book.

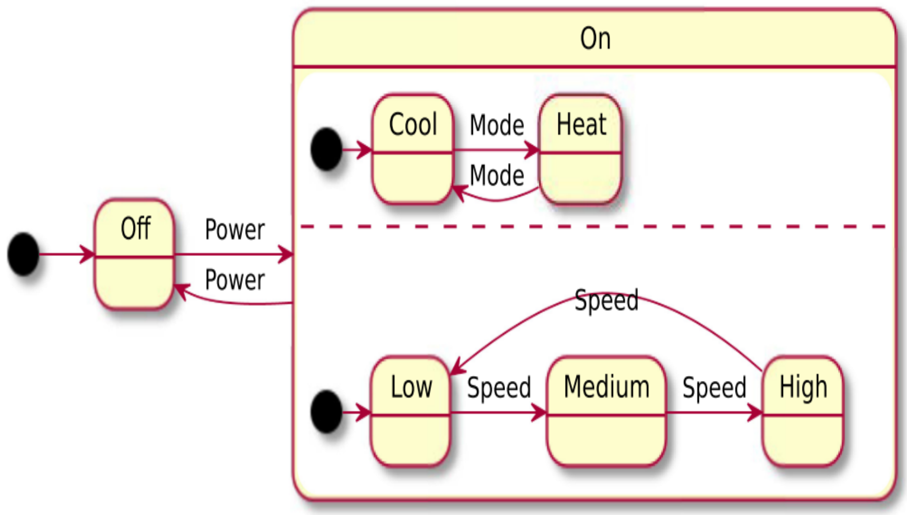


Figure 2.3: Statechart of an air conditioner in Lecture Notes in Artificial Intelligence published in 2020 [79].

Why not Statecharts? Statecharts are a sequence of steps that occur in a model, rather than the structure of the model. Currently, this method does not include space or location. It is an agent modelling and logic based that is mostly used for software engineering rather than biology. To present the process, or communication of the agents through messages or a blackboard, statecharts are ideal. But for broadcasting communication, statecharts show weakness [79].

2.5.6 P systems models

P-systems are distributed parallel computational models to describe the structure of a living cell. Basically P systems consist of a main membrane which embeds several other membranes called skin and regions which is a space between the membranes. Each membrane contains certain objects and these objects follow evolution rules in a random parallel manner [165]. It models the cells with compartments and each reaction takes place in different compartment of the cell. Because of its multi-scale feature, this method have been used to show biological interactions and networks. P-systems can be used to predict the concentration of an object, which may represent proteins and small molecules [19, 165, 162]. It is noteable that P systems are mostly for representing the structure of a system and not a detailed model of it [165]. There are different software tools which support P-systems, which based on their website [P-system software tools] the latest one is for 2013.

Why not P systems? The processes in P-systems are discrete yet not accurate since the main motivation of this framework is to explore the mathematical and computational aspect of biological systems [19, 44]. There are not many systems biology research using P systems in the recent years indicating that this tool is probably out of date or not suitable for studying biological systems. as mentioned before, this method is used for the construction of a call rather than the details [165]; which is probably the main reason of this method not being used anymore.

2.5.7 CellML

The System Biology Markup Language (SBML) is a representation format based on eXensible Markup Language (XML) for modelling biological systems in silico[86]. CellML is another XML-based mathematical modelling language for system biology which can simulate models based on the given underlying mathematical equations. Usual models are text files with mathematical formalism in them and they cannot be used by the other scientists to get the same published results in another platform. In the models made with CellML all the information about the model are stored in one place which gives CellML the feature of being reusable, not to be mistaken by reproducible. It consists of components, connections, groups and metadata where all of them together can make a model in CellML format, thus in XML. CellML itself does not have GUI, yet it can be exported to other formats and be used in the software tools which have GUI [86]. The main aim of CllML was to create a virtual physiological human. Ever since the launch of the project, they have been adding different types of models from different literatures in their database in order to reach this goal [119]. CellML is still growing this date and the second version of it was released in 2020 [44].

Why not CellML? As mentioned before, CellML does not provide a reproducible model that could provide the same result. Even though the language of the model supports the format of XML, it is a editable text file which needs expertise to work on. One of the aims of this study is to introduce a way that could be used by people with no knowledge with mathematics and computer and clearly CellML does not fulfil this mission.

2.5.8 Bio-PEPA

PEPA is a process algebra that was originally used for analysis of computer systems but since 2008, it was applied to systems biology to model biological systems [41] for stochastic computational modelling and analysis. Bio-PEPA is designed to describe the biological behaviour and simulate and analyse the models stochastically with probabilistic model-checking[39]. Bio-PEPA are used to model events, or changes in the biological system to analyse a biochemical network. Bio-PEPA can be changed to other formats of modelling, such as ODEs, for analysis in different platforms such as MATLAB. With Bio-PEPA it is possible to design an abstract model of the species and reactions which can process concurrently[42].

Why not Bio-PEPA? Even though Bio-PEPA became popular in 2008 and 2009 by a group of scientists who were interested in modelling biochemical reactions [41, 39, 42, 38], it does not seem to be of any new uses in the recent years. Bio-PEPA originally was very system and was focusing on the events and changes of a system. It is a good choice for models in which there are changes in the pathway, such as biochemical reactions, rather than biological systems that describe a dynamic process[39, 38, 40].

2.5.9 Petri net

Petri net is a multi-scale bipartite graph originally introduced by Carl Adam Petri in 1962 [33] consisting of two types of nodes: places and transitions [87, 76] used for multi-level modelling on a multidimensional environment [118]. These two nodes are connected to one another using a discrete arc [17, 72]. The places can contain tokens. The number of tokens can only be integer in stochastic and non-negative real numbers in continuous. Transitions can carry functions that defines their action rate and in case of coloured Petri nets, their locations. In a biological setting, transitions can represent the processes (*e.g. biochemical reactions*) while the places are the entities (*biochemicals in a reaction*). This method can be used for both qualitative and quantitative modelling and simulations [187, 177] and it is independent from mathematical techniques as it has a GUI [121, 17, 48].

The transitions in Petri nets are the reactions and the actions rates, or firing rate is a number that defines how long it takes for the transition to activate. In Stochastic models this number can only be an integer while in the continuous ones it can be a Real number[21].more explanation about how the action rates work is provided. Based on the chosen tool that supports Petri nets, there are different types that could be used as required. More about different types of Petri nets can be found in Chapter 3.

2.5.10 Tool Box

To use all the methods discussed in the previous section, specific platforms are needed. Below is a table that introduces different software tools as well as which method these software tools use.

2.6 A Review of Previous Models

There are a range of methods for bio-modelling. These methods are explained in section 2.5 in detail. Using these methods some models have been constructed and published over the past years in the same area as our study. In this section the models that are constructed after the the beginning of the 21st century are chosen to be discussed. Here it is explained explain how

Software Tool	Method	Citation
Lsodar	ODE	[142]
Omix	ODE & PDE	[96]
Morpheus	ODE & PDE	[96]
SmartCell	PDE	[96]
MesoRD	PDE	[96]
FLAME	ABM	[178]
SPARK	ABM	[7, 178]
NetLogo	ABM	[178]
RePast HPC	ABM	[178]
Mobility Workbench (MWB)	π-Calculus	[199]
Microsoft	π-Calculus	[18]
PICASSO	π-Calculus	[14]
STLC	λ-Calculus	[152]
SCXML	Statecharts	[106]
P-Lingua	P systems	[154]
PMCGPU	P systems	[195]
MeCoSim	P systems	[195]
OpenCOR	CellML	[75]
Virtual Cell	CellML	[120]
Bio-PEPA Workbench	Bio-pepa	[39]
Cell Illustrator	Petri nets	[96]
MonaLisa	Petri nets	[96]
Snoopy	ODE & Petri nets	[89]

Table 2.1: The table of different software tools using different methods based on Section 2.5 .

they were constructed. It is important to note that there are many models in the area of quorum sensing and biofilm formation and fewer in the area of movement and duplication.

Perez *et al* have published a review article of all the mathematical models of quorum sensing biofilm formation prior to 2015 [155]. Using this review, along with other related studies, a brief review of the past works on bio-modelling of related models is gathered in this section *i.e* quorum sensing, biofilm formation, duplication and movement/chemotaxis.

James *et al* in 2000 [101] designed a non-linear ODE model with the focus on the regulatory system in a single cell in *Vibrio fischeri*. The observation on this bacteria that was first studied in the light organ of some marine fish, showed that in high cell density these bacteria communicate with each other which results in maintaining light production in the light organ [90]. James *et al* developed a model of *lux* genes control *V. fischeri* in the simplest form possible. They compared the regulation of *lux* genes in two different situations: a) When a free-living cell does not have the autoinducer in the environment and b) when there is autoinducer added artificially to the environment and the bacteria is exposed to it [101].

In 2001 Eberl and his colleagues developed a Deterministic spatio-temporal Continuum Model for biofilm formation [155]. His mathematical model is a density-dependent equation to show a spatial growth of biofilm in a heterogeneous society. They do not mention any specific bacteria and all of the work is completely mathematical with no biological verification. However, after the analysis of the model it seems that the model is able to predict the irregular biofilm formation in 1D, 2D and 3D. They also present a density-dependant diffusion, environmental conditions and nutrition in their model [61]

Another model in 2001 was developed by Nilsson *et al* to describe the autoinducer concentration in bacterial cells and how biofilm changes in time as a result of bacterial growth rate and diffusion of autoinducers. Nilsson's mathematical model is based on one species of Gram-negative bacteria who form biofilm. The bacteria are completely identical in terms of size, shape, production and degradation. The goal of their study was to understand the effects of biofilm formation on signal production. The result of the model is theoretically in agreement with the expected result of the bacteria in a high rate nutrient culture [155, 146].

Deckery and Keener in 2001 designed a mathematical model of quorum sensing which is an eight dimensional ODE. in their model they focused on the kinetics of *Las* system in *P. aeruginosa* [155]. They analysed and simulate their results numerically and tried to include biochemistry of the bacteria as well. They concluded that the quorum sensing depends on the size of the colony and density, which is expected. The results are completely mathematical with no verification from biological lab [57].

In 2002 Chopp *et al* developed a 1D spatio-temporal model of biofilm formation and quorum sensing [155]. He worked on Gram-negative bacteria *P. aeruginosa* who has two separate quorum sensing systems. In the model they separate biofilm in two compartments. One is where the active cells exist and one is the inert biomass. They also show the diffusion of oxygen and the limitation of nutrition [37]. In an extended model Chopp models the limited substrate and the production of autoinducers. The used method in this study is Reaction-diffusion equations which are a type of parabolic partial differential equations [37].

Ward *et al* in 2003 developed a model that is a PDE non-linear model which was analysed numerically and the results were in agreement with experimental data [155]. The models describes the early stages of biofilm formation considering the cells growth on a solid surface and making a compact society. after that they developed a model of the overall formation of the biofilm [207].

Viretta and Martin in 2004 presented a qualitative model using linear differential equations method. In this model they explored the process of quorum sensing in detail in the Gram Negative bacteria *P. aeruginosa* trying to understand the complexity of this process deeper [201].

Li *et al* in 2006 designed a stochastic model of autoinducer production in *E. coli* using Petri nets[155]. The interesting fact about their model is all the markings, rate constants and tokens numbers come from a verified reference. The model is based on experimental data and the results were experimented in biological labs [114].

Muller *et al* in 2008 developed another differential equation stochastic model to show the threshold effect of autoinducers in quorum sensing. The results are numeric and mathematical and data is model-based. Interestingly they revealed that the molecules produced by each cell is sufficient so that the bacteria can distinguish their own signals [155, 140].

Netotea *et al* in 2009 developed a 2D agent-based model of biofilm formation to show that the bacteria *P. aeruginosa* communicate using diffusible autoinducers. They move toward a nutrition source and the autoinducer threshold controls the movement. There are both biological and computational observations in this study: The biological observation focuses on comparing the wild and the mutant bacteria while the computational observations construct a framework for early stages of quorum sensing [143].

Duddu *et al* in another work in 2009 worked on a 2D continuum model of biofilm using Extended Finite Element Method [155]. Duddu's model also included fluid flow around the biofilm on the surface, the shear stress of the biofilm-fluid interface and the reaction of substrate. The biofilm is assumed to be a continuous medium with two compartments: active and inactive, the same assumption Chopp used in his model in 2002. The fluid in their model has a constant viscosity and the shear stress is also considered [60].

Vaughan *et al* in 2010 developed a finite element 2D model of diffusion, degradation and production of signalling molecules in biofilm formation [155] similar to Duddu's model and method.

Their bacteria target was *P. aeruginosa* and their focus was mostly on the distribution of the autoinducer and the effect of a fluid environment on the quorum sensing and biofilm formation. In their model they included the fluid velocity of the culture media and studied biofilm formation in different concentrations of autoinducer and different roughness of the culture medium. The results are all computational simulations [198].

Fredrick *et al* in another model in 2010 is a 2D mathematical model of quorum sensing in a slow-flow environment [155]. They developed the first mathematical model for a two dimensional biofilm with flow and nutrient-dependent growth and their results were computational simulations only [69].

Fredrick extended his work on a reaction- diffusion equation for biofilm in 2011 with numerical solutions [155]. They describe the modelled biofilm structure as a deterministic density-dependent equation. the model describes the quorum sensing results in biofilm formation in a narrow conduit to mimic solid pore or plant vessels. They used experimental data for the parameters of the model but analysed the model with computational simulations. The used method in this study is Reaction-diffusion equations [69].

In 2012 Ward and King developed a system of PDEs model. They considered that the biofilm consists of bacteria and water. The volume fraction of bacteria is assumed to be uniform and the internal stress between bacteria and bacteria is less compared to the bacteria and water. Their purpose was to describe biofilm formation and quorum sensing process in bacteria using computational and numerical results [155] [205].

In 2013 Schaadt *et al* designed a parameter independent multi-level ODE formalism to model quorum sensing in P. aeruginosa which produces AI-1 to show the complicated network of the communication process in this bacteria. In their model they did not include the changes in environment and considered that all cells in their model receive the same amount of autoinducer, which shows that their model does not describe space, location and distance. However, they have included aspects of the quorum sensing such as the threshold, degradation and inhibitors [170].

In 2015 Emerenini and colleagues developed a dynamic spatial mathematical PDE model to show quorum sensing and biofilm formation to show the relation between the cell dispersal and produced biofilm size. In their model, the society of bacteria is homogeneous, meaning that all bacteria are the same to simplify their model and thus, the simulations [65].

Matthew Edgington in his thesis in 2015 [62], used a four-dimensional nonlinear ordinary differential equations approach to model the chemotaxis in bacteria as a single cell. In his study, he mostly focused on the mechanism of chemotaxis, inspecting the flagellar motors of the bacteria. He then used Agent Based models for studying the bacteria in a population. Using the mathematical models Edgington proposes new pathways for chemotaxis which are in agreement with experimental data.

In 2017 David Gilbert, Monika Heiner and Leila Ghanbar modelled an *E. coli* K12 in Petri net as a quantitative modelling tool. Petri nets enable us to model stochastically or continuously, while using coloured Petri net gives space to the model so that we can have models in 1D, 2D or 3D. This method can be developed for personalised models for humans to be used in personalised medicine and personalised nutrition [76]

The next model of biofilm was published in 2018, by Zhang *et al* which is a dynamic flux-based model of biofilm communities and a "kinetic free formulation" based on genomics data. The model consists of different models, such as general biofilm model, metabolic model and exchange flux model. This is a novel method to replace the classic methods consisting of kinetics functions [215].

Another model of biofilm was designed in 2018 by Bardini *et al.* [15] which showed the biofilm resistance to antibiotics in hybrid Petri net formalism on Nets-Within-Nets to show the different antibiotic protocols influence and different levels of resistance of biofilm. However they do not use locations and space, they have used Coloured Petri nets to show separate bacteria cells in one place by colouring the tokens in that place. the coloured tokens carry information on the identity of species and the possibility of the presence of resistance, unlike Snoopy that the coloured tokens carry information of the space and location of each token [15, 118].

Gilbert *et al* in 2019 created a combined model of quorum sensing in *E.coli* from Li's model in 2006 [114] and a new part of biofilm formation [77]. This model that was designed in Petri net using Snoopy Software as a platform, described the biological components of the system as well as the space and location. The previous model by Li was a non-spatial stochastic model of quorum sensing via AI-2 with limited nutrition. This model, added spatial properties using coloured Petri nets to the model and analysed the behaviour.

The latest model of biofilm is proposed by Brown *et al.* in 2019 which is an agent- based model presenting the production of biofilm by *Haemophilus influenzae* bacteria *in vivo.* Simulating their models they showed that the reason of different size of cluster of the bacteria *in vivo* and *in vitro* is because of the elimination of single planktonic cells by the host. They also predicted how the processes in the model effects the infection [25].

Brown *et al.* in 2019 constructed the latest available model of biofilm. It is an agent-based model of biofilm formation of *Haemophilus influenzae* bacteria *in vivo* reasoning that the difference in the size of cluster of the bacteria *in vivo* and *in vitro* is for the elimination of single planktonic cells by the host [26].

In 2020, for the most recent available model, is a chemotaxis model of bacteria *E.coli* using a unique approach called Vivarium done by Agmon and Spangler [3]. In this approach they created multi-scale models in a spatial environment. This software uses a variety of methods. The focus of their paper is on the biophysical features of chemotaxis, describing the flagella which is the movement engine in bacteria.

In this study, first a collection of model components were built. The components of this library describe different biological behaviours and are used individually or as a combination to reproduce some scenarios and contains various models. The components of the library include movement, chemotaxis, reproduction, communication, response to communication and death. These models are on a multidimensional grid, describing a multi-level and multi-scale model to study both intercellular and intracellular interactions in microorganisms. They are designed so simple that could be used by individuals with no computational or mathematical background.

Table 2.2 presents all the models created in 21st century in a summary.

2.7 Discussion

In this chapter the differences between different science fields and where this study exists were discussed and a biological and computational background for this study was provided. The biological literature provided in this chapter is required for a better understanding of the models in this study. In the computational background, two types of reviews were presented: one for the other alternative tools and why they were not chosen while reasoning the choise Petri net for this study and the other is a review of the previous models done in the same area of science of this study. In the next Chapter, software tools used for this study are vastly explained as well as how the modelling process have been done. Then it follows in the next chapters with the library of models, data visualisation and optimisation pseudocodes.

Year	Modellers	Model	Method	Citation
2000	James *et al*	QS	Non-linear ODE	[101]
2001	Eberl *et al*	BF	DE	[61]
2001	Nilsson *et al*	QS & BF	DE	[146]
2001	Deckery and Keener	QS	ODE	[57]
2002	Chopp *et al*	QS & BF	Reaction-diffusion equation	[37].
2003	Ward *et al*	BF	PDE	[207]
2004	Viretta and Martin	QS	Linear DE	[201]
2006	Li *et al*	QS	SPN	[114]
2008	Müller *et al*	QS	DE	[140]
2009	Netotea *et al*	BF	ABM	[143]
2009	Duddu *et al*	BF	DE	[60]
2010	Vaughan *et al*	BF	DE	[198]
2010	Frederick *et al*	BF	Reaction-diffusion equation	[69]
2011	Frederick *et al*	QS & BF	Density dependent equation	[69]
2012	Ward and King	BF	PDE	[205]
2013	Schaadt *et al*	QS	Multi-level ODE	[170]
2015	Emerenini *et al*	QS & BF	PDE	[65]
2015	Edgington	Chemotaxis	Non-linear ODE	[62]
2017	Gilbert *et al*	QS	PN	[62]
2018	Zhang *et al*	BF	Flux-based model	[215]
2018	Bardini *et al*	BF	PN	[15]
2019	Gilbert *et al*	QS & BF	PN	[77]
2019	Brown *et al*	BF	ABM	[26]
2020	Agmon and Spangler	Chemotaxis	Vivarium	[3]

Table 2.2: The table summarises all the models mentioned above. Acronym: QS = Quorum Sensing, BF = Biofilm formation, Dicty = Dictyostelium, DE = Differential equation, ODE = Ordinary Differential equation, PDE = Partial Differential equation, PN = Petri net, SPN = Stochastic PN, ABM = Agent-Based Model .

Chapter 3

Methodology For Modelling, Simulation, Optimisation and Data Analysis

In this chapter the chosen software tools are introduced for modelling, simulation and data analysis in Section 3.1. Also the optimisation methods are explained briefly in this chapter in Section 3.1.4. In Chapter 6 a detailed explanation of the optimisation methods as well as the pseudocodes are provided. The aim of this chapter is to clarify the different tools used in this study. In the next chapter the different parts of the library of model components are introduced as well as a step-by-step protocol for how the colouring and combination works.

3.1 Chosen Software Tools for Modelling and Simulation

In this section the chosen tools of this study are introduced. After introducing the tools used in this thesis, in Section 3.1.5 and Section 3.1.4 the methods for data analysis and optimisation are discussed.

3.1.1 Snoopy

Snoopy is a non-commercial free software which runs runs on Windows, Linux and macOS and is a software tool for hierarchical modelling in Petri nets. This software supports different classes of Petri net such as classic/standard, stochastic, continuous and most importantly Coloured Petri nets. Snoopy has different solvers for each type of the Petri net built-in and can simulate the models. It also produces CSV data files as well as plots and graphs. In standard or classic Petri net, Snoopy is equipped with the animation mode that can trace the movement of the tokens in the model. Snoopy models can be exoroted to ANDL (Abstract Net Description Language) and CANDL (Coloured Abstract Net Description Language) which are human-readable text files for easier altering and simulation of the models [89, 36, 76] and can import models from other different formats of Systems Biology Markup Language (SBML) [89]. Exporting and importing large and detailed models are slower in Snoopy as it has a graphical user interface.

There are several different types of Petri nets that Snoopy supports. The classes of Petri nets that are used in this study are described below:

3.1.2 Different Classes of Petri nets:

There are different types of Petri nets for biological modelling which are particularly supported by Snoopy Software. The types that have been used in this study include: Standard (qualitative) Petri nets (PN) which do not have the concept of time or space and are the classic type of Petri nets, Stochastic Petri nets (SPN), Continuous Petri nets (CPN), and Hybrid Petri nets (HPN) combining both stochastic and continuous; and finally, Coloured Petri nets (ColPN) [45].

a. Standard Petri Nets are the classic type of Petri nets, used for constructing models in a qualitative manner, meaning that the semantics of this class does not include time and the network is described by its topology [78]. The difference between the qualitative Standard Petri net and the quantitative Petri nets (such as continuous) is in the formalism. While the qualitative is defined based on linear differential equations, quantitative Petri nets are based on ODEs [34]. Standard Petri nets are quite simple to use and easy to understand. In Snoopy, as one of the platforms that supports Petri nets, it is possible to activate the animation mode which demonstrates the tokens moving between the places and transitions. Providing a graphical user interface (GUI) and simplicity, make this type of Petri net rather easy for the individuals with no experience or knowledge of computational modelling as well as for beginners in the field of modelling with Petri nets. This type of Petri net does not require any mathematical knowledge either [89]

b. Stochastic Petri nets (SPN) are used for quantitative modelling. This class of Petri nets is similar to the PN, with the addition of Time as a quantitative property and the rates to the transitions which exists also in Continuous, Coloured and Hybrid Petri nets [17]. In Snoopy software, besides the animation mode, this type also provides simulations using different simulators, such as Gillespie, as well as plots and output data which can be exported into CSV data files [134, 89]. The stochastic models can run for one or multiple simulations. By increasing the number of runs the outcome plots are smoother and the result is closer to a continuous output [21].

c. Continuous Petri nets (CPN) are another quantitative type to observe how the model behaves continuously over time. In this type, not only the action rates, but all the numbers can be non-negative real numbers. Due to this continuity, this class of Petri net is usually used for studying the overall behaviour of an organism rather than than the individual behaviours [88]. For example, in this study, this class is used for studying the population density rather than the actual number of the cells in the society. This class is time-dependant as well [23] Continuous Petri nets can be mapped to ODEs and can be simulated using ODE simulators [21, 78].

d. Hybrid Petri nets (HPN) are the combination of SPN and CPN. Usually, in nature a biological system is both stochastic and continuous. To construct a more realistic model, HPNs are an ideal option. This type of PNs allow the co-existance of discrete and continuous places and transitions in the same model [93]. This type is especially used in the combined model of quorum sensing and biofilm formation which is explained in 5.1.5 and was published in 2019 in BMC Bioinformatics Journal [77]

e. Coloured Petri nets (ColPN) are specifically useful when modelling a repeated network in a multidimensional model. In these situations, it is possible to edit a folded version of the model and unfold it, in the end, to study the model as a whole [73, 10]. After the work on the folded version is done, it is possible to export it to a non-colour PN and observe the behaviour of the model as a whole. ColPN is a class of PN that enable modelling in 2D and 3D environments by colouring the tokens; which means giving each token a specific set of data or colours that facilitates the representation of space and location. ColPN represents a compact and folded version of a complex model where each object in the system becomes a coloured place to have a location defined by that colour [117].

ColPNs support can be stochastic or continuous. However, it is important to note that in the Coloured Continuous Petri nets(ColCPN), only the number of the tokens can be real numbers more than zero and not the grid. Creating a 2D or 2D grid in ColPN using analysis tools such as R shows that the grid is discrete. The models built in ColPN incorporate the concept of space on a grid and the location is defined based on the coordinations of the grid. On each location, places are set and the transitions are the bridges between two locations of the grid. That is why the location on places is shown as "Place_1_1" while on transitions it is shown as "Transition_1_1_1_2". This means the transition is connecting coordination 1_1 to 1_2.

In Snoopy, when the models are built, they exist in all of the locations by default. By changing the marking and defining the location, only the marked locations are active. It is important to note that currently, in Snoopy only the tokens move on the grid while places and transitions are immobile. When a grid is defined, all the locations contain the models, but when the location of a place is defined, the other places on other locations are inactive. In other words, on the coordinations of the grid that no places are set, the model exists, except it is switched off.

3.1.3 Spike

Models built in Petri nets might be so complex that their simulation is time and memory consuming and computationally intensive [36]. Especially when it comes to coloured Petri nets, where the models are multi-level, multi-scale and multidimensional, simulating with Snoopy is not the best idea.

Sipke is a command-line simulation tool that uses Abstract Net Description Language (ANDL) and Coloured Abstract Net Description Language (CANDL) files to simulate models exported from Snoopy. ANDL and CANDL are human readable text format files exported from Snoopy Petri nets that contain the model's properties [21, 36]. Based on the website of the developers of Spike, it is "a tool for efficient and reproducible simulation of stochastic/continuous/hybrid Petri nets, coloured or uncoloured ones" [General Description of Spike Website]. Spike has a specific configuration script, a text file called SPC. In SPC files the modeller can alter some of the parameters of the model, such as the action rates, the gird size, etc, instead of initiating changes on the original model in Petri net. Spike also has the feature of scanning through a range of parameters to check the effect of the changes on the model outcome. However, considering the unlimited numbers and possibilities, optimisation with this method is currently impossible. Another feature of Spike is that it can simulate models in parallel or sequentially.

All the changes in the SPC files are saved before the start of the simulation which results in reproducible simulations. The simulations are happening on a server and there is no GUI to slow down the simulation. Spike supports Stochastic, Continuous, Hybrid and Coloured Petri nets for simulation. In the SPC files the different types of solvers can be chosen as required [36].

3.1.4 Optimisation Method

Spike was used for scanning among different parameters. But in the case of complex systems when multiple parameters need scanning, Spike would not be efficient as it would provide many datasets that need to be analysed one by one. Therefore, to make the process of scanning faster, optimisation methods were applied. Python routine codes for Random Restart Hill Climbing and Simulated Annealing were employed to scan over different constant rates of the model in order to find the best possible solution for the expected behaviour..

A heuristic technique is an approach to solve a problem by getting approximately the best solution. It is used when solving a problem would take an unlimited amount of time or the number of solutions are unlimited. In these situations, approximate best solution is the best option to use [109].

In this thesis, two methods of optimisation are employed to find the best solution for the objective of this study, which is to achieve the maximum number possible of Dictyostelium cells in one location on the grid to mimic the formation of fruiting body in this interesting microorganism. At this stage of life, Dictyostelium forms a semi-multicellular society to protect itself under difficult situations such as starvation. For more details on the biological concept of this Eukaryote see Chapter 2 Section 2.3.2. For a detailed discussion about the optimisation methods and psuedocodes, see Chapter 6

3.1.5 Data Analysis Tools

To analyse the output data from the models, or rather checking if the behaviour of the model is as expected as well as for data analysis and visualisation, R is used. To analyse the time series data taken from Snoopy and Spike R coding program was applied to create 2D and 3D plots as required. With R also, heatmaps and plots are plotted to study the behaviour of the models based on the location of the tokens in the model. These heatmaps are set together to create a movie that shows the development of the model over time, with the focus on the changes in their locations over time.

3.2 Discussion

In this chapter different tools that have been used in this study were discussed as well as the chosen methods for optimisation. There are many methods for optimisation, and in this study two of the most popular ones have been chosen: Random Restart Hill Climbing, which is simple but may not be accurate since it may provide the local maximum instead of the global one, and Simulated Annealing, which compared to Hill Climbing is more accurate and searching as many answers as possible, it might provide a better answer. The expected result of these optimisations is to find the best constant rates for the four transmitters of the Dicty model, in order for the Dicty to clump on the grid.

In the following chapters the collection of model components will be introduced and different levels of models will be discussed following by examples of application of these model components to a close to real-life scenario. In this study the applications are biological but that does not limit the adaptation of these models only to biological experiments. These models could be applied to any non-biological scenarios.

Chapter 4

Model Library

Specific behaviours define if a system can be considered alive. Some of these behaviours include movement towards a chemical trigger, reproduction, death, communication and response which all are modelled in this thesis. The reason for the choice of these specific behaviours is simply due to their importance for survival. This set of behaviours is only an example of basic requirements for life. If there is no movement towards a chemical trigger, the organism might starve to death. In critical situations the organisms need to communicate and respond to one another to survive. They need to reproduce to pass on their genetic information to the next generation and finally they will die.

In this chapter different parts of the collection of model components including properties and different levels of model components, the process of modelling and combining the models are explained in detail. Here, using the properties, the models are constructed from scratch and each model component is shown in a figure for a better understanding. In the next chapter, combining these components and the construction of systems in two case studies is explained.

Putting together components and properties, can lead to the construction of models in different levels. In this study the collection of models components is categorised into four groups:

1. Properties: The base of modelling which includes definitions of functions, constants and coloursets.
2. Level-0: Basic components which are usually made from one place and one transition. If they are modelled in space (ie. in ColPN) then they also include a property.
3. Level-1: Simple models which are the combination of basic components together with more

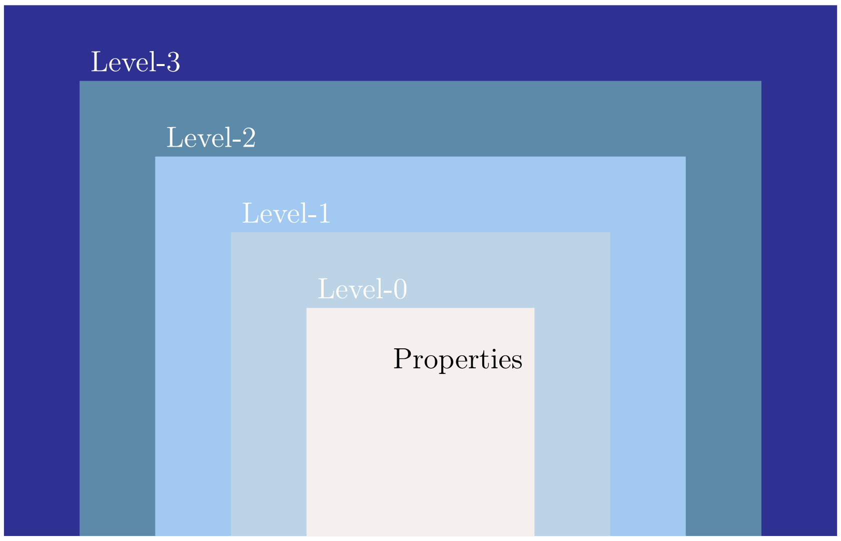
than one property.

4. Level-2: Systems and complex models which are combination of Level-1 and/or Level-0 with a few properties.
5. Level-3: Complex networks which are created from Level-2 models as well as properties. These could contain lower level components as well. They are explained in Chapter 5

All the components of this collection of models are stored as a data collection in CANDL format. CANDL file are small in size and they are human-readable text format files, and are easy to edit. From CANDL file it is possible to export to other formats such as ANDL or Coloured Petri nets.

The modelling method in this thesis is hierarchical. As it can be seen in Figure 4.1, Properties are the most basic part of the models and exist in all the levels. Level-2 models are built as a combination of Level-1 and Level-0 components and Level-1 components are built of Level-0 components. It is important to note that a Level2 model can also be built from Level-0 components only.

Figure 4.1: Modelling in this study is hierarchical, meaning the more complex models are built from simpler components.



It is important to note that the categorisation of each level, is based on how complex they are structurally and it is independent from how complex or detailed the model is. The question is how to decide on the Level of a model? To answer this question, first it is important to understand that the modelling in this thesis is hierarchical, meaning each model is built from simpler components from lower Levels, unless it is a Level-0 model which is the simplest component. To identify the Level of a model, first the most complex component should be spotted. Once the Level of the most complex component in the model is decided, the model in question is categorised on one Level higher. For example, in the Chemotaxis model, Diffusion which is a Level-0 model can be found. This is the most complex component that can be found, so Chemotaxis is a Level-1 model. Another example is the Quorum Sensing model which includes Biofilm Formation model and Signal Production. Biofilm Formation is categorised as a Level-2 model, so the Quorum Sensing model is placed at Level-3. A detailed elaboration of the structure of each model is provided in this chapter and the next. Table 5.2 at the end of Chapter 5 provides a summary of all the models in this thesis and their categorisation.

4.1 Properties

To build the models in this library, the basic definitions of the model need to be clarified first. These terms will be used in the following chapters. These basic definitions are called "properties" in this study, and are the first steps of modelling in Petri nets. Here, the concept of action rate, grid, neighbourhood function and coloursets and based on them, Wall, Obstacle and Sink are introduced.

Action Rate Action rate, firing rate or rate constant is a number that is set for the transitions that define how fast the action would happen. Using optimisation methods it is possible to find the best number of the action rates in order to achieve the expected behaviours from the models. The expected behaviour is based on the biological literature and is different for each model, depending on what the model demonstrates. For example,The expected result of the movement model of bacteria is different from Dicty model of movement.

The Grid The grid is a 2D or 3D panel that its size is defined in the coloursets in Snoopy made of x, y and z (in case of 3D) axes. These axes define the location of the tokens and places mathematically. For example, a place could be placed on (2,3) which means its location on the 2D grid is: x = 2 and y = 3. As mentioned before, the grid's structure is not continuous, even in a continuous model. On the grid it is considered that a token is located in the centre of the square location. The transitions are placed between the grid locations, connecting the one place on one location to the next, based on the definition of Neighbour functions. In that case the location of the transition that connects (1,2) to (2,2) will be "1_2_1_2".

In this study, for faster simulations and saving time, all the models are built on a 2D grid. The grid is only used in ColPN, as they are the only type of Petri nets that can demonstrate space. To define the size of the grids, after defining "Constants", the coloursets are created as a "product" of the constants to make a 2D and 3D grid. The definition of the grid by size and dimension is as shown in Table 4.1:

Dimension	Size	Colourset
Grid 1D	D = 5	[no coloursets required for 1D grids]
Grid 2D	D1 = 5	CD1 = 1-D1 [x axis]
	D2 = 5	CD2 = 1-D2 [y axis]
Grid 3D	D1 = 5	CD1 = 1-D1 [x axis]
	D2 = 5	CD2 = 1-D2 [y axis]
	D3 = 5	CD2 = 1-D3 [z axis]

Table 4.1: The information of 1D, 2D and 3D grids for ColPN

Neighbour Function These functions are used to describe the movement of tokens in Petri nets through the transitions. Neighbour functions can be used in both 2D and 3D grids. On 2D grids, the tokens can move to either 4 or 8 directions moving to the next location from their initial location and on 3D grid the token has the option to move to 26 locations around it. The tokens do not jump one location. This means if the token is moving from (3,4) to (3,6), it has to pass (3,5) as well. It is important to note that the grids introduced here and used in this study are always squares or cubes. This is not necessarily always the case. The grid can be circular and cylinder as well. The definition of the functions used in this thesis can be found in Table 4.2:

Function name	Function definition
neighbour2D4	(a=x & b=y-1)|(a=x & b=y+1)| (b=y & a = x-1)|(b=y & a=x+1)
neighbour2DB	(a=x-1|a=x|a=x+1) & (b=y-1|b=y|b=y+1) & (!(a=x & b=y)) & (1<=a & a<=D1) & (1<=b & b<=D2)
neighbour3D26	(a=x-1 | a=x | a=x+1) & (b=y-1 | b=y | b=y+1) & (c=z-1 | c=z | c=z+1) & (!(a=x & b=y & c=z)) & (1<=x & x<=D1) & (1<=y & y<=D2) & (1<=z & z<=D3)

Table 4.2: The neighbouring functions for 2D and 3D grids.

The syntax provided in Table 4.2 explains how the tokens are allowed to move on the grid. x and y define the initial location coordination and a and b are the new locations. So, for example, when the function is written is $(a = x \ \& \ b = y - 1)$, this simply means that the token is permitted to stay in the same x while goes down one location in the y axis.

The Sink The square/cubic grid is a limited closed area as big as the defined constants and coloursets. To produce a pseudo-infinite grid, a set of transitions can be added to the edges of the grid which removes the tokens from the edges of the 2D and the faces of the 3D grid. These transitions are called "Sink" transitions. In this study, the Sink is used when there is a need for the removal of some tokens from the grid. This transition's job is to prevent the accumulation of the tokens in the grid if required. The function defined for this transition for 2D and 3D grid is shown in Table 4.3.

Another way of defining the location of the Sink transitions is using a colourset called "Edges". This colourset is the "Union" of 4 Simple coloursets: top, bottom, right and left. These simple coloursets carry the locations of the edges of the grid. However, using this method the unfolding of the models can be only done by the "Gecode intern" engine in Snoopy. The output of this engine is not compatible with Spike. Therefore, in case of the more complex models that simulations with Spike is required, using this method is not recommended.

Grid size	Function definition
2D Grid	((x=1) & (y>=1) & (y<=D2)) | ((x>=1) & (x<=D1) & (y=D2)) | ((x>=1) & (x<=D1) & (y=1))
3D Grid	(x<=1 & x>=D1 & y<=1 & y>=D1 & z=1)& (x=1 & y<=1 & y>=D1 & z<=1 & z>=D1)& (x<=1 & x>=D1 & y<=1 & y>=D1 & z=D3)& (x=D1 & y<=1 & y>=D1 & z<=1 & z>=D1)& (x<=1 & x>=D1 & y=1 & z<=1 & z>=D1)& (x<=1 & x>=D1 & y=D2 & z<=1 & z>=D1)

Table 4.3: The functions of the Sink Transition. Based on these function definitions, the Sinks are located at the edges of the 2D and faces of the 3D grid.

The Wall Adding the "Sink" transition, makes a pseudo-infinite grid. By removing this transition on one side, the "Wall" is created which can simulate a surface where the bacteria can adhere to. From side that the wall is located, nothing can leave the grid. Wall can be considered a surface that the cells are attracted and adhered to. In a biological laboratory it can be considered the inner wall of a test tube which is coated with food source to attract the microorganisms to it.

The Obstacle Obstacle is a part of the canvas (grid) which nothing can cross it from the outside. It is defined as a block which can be located anywhere as required, based on its function. Obstacle also can mimic a semi-permanent membrane which only passes permitted molecules to the outside while not allowing anything to enter it. By setting the Massaction of the transitions as zero, the obstacle turns into a solid block that nothing can cross it. The neighbour function when using obstacle changes. The definition of these functions is as shown in 4.4.

Function name	Function definition
Obstacle	(a=x-1|a=x|a=x+1) & (b=y-1|b=y|b=y+1) & (!(a=x & b=y)) & (((a>=ObsX1 & a<=ObsX2 & b>=ObsY2 & b<=ObsY1)) | ((x>=ObsX1 & x<=ObsX2 & y>=ObsY2 & y<=ObsY1)))
Neighbour2D8_obstacle	(a=x-1|a=x|a=x+1) & (b=y-1|b=y|b=y+1) & (!(a=x & b=y)) & (! (a>=ObsX1 & a<=ObsX2 & b>=ObsY1 & b<=ObsY2)) & (! (x>=ObsX1 & x<=ObsX2 & y>=ObsY1 & y<=ObsY2)) & (1<=a & a<=D1) & (1<=b & b<=D2)

Table 4.4: The function definitions for obstacles. In these Functions, $ObsX1$, $ObsY1$, $ObsX2$ and $ObsY2$ are the locations of four angles of the obstacle rectangle.

The Glu Glucose is one of the carbon sources used by the microorganisms. In the models of this study, "Glu" simulates source of the food that produces food and is the trigger of chemotaxis for the microorganisms. To define the location of the Glu, the variable "Gluxy" was defined. This Variable exists in the colourset "GluS" which consists of two constants of Glux and Gluy. To make a strip on one side of the grid Glux is defined as an integer as big as D1 (the size of the grid) and Gluy is set as an interval [1-D1]. The information of Glu is summaried in thable 4.5

Constants	Colorsets	Variables

Glux = D1
Gluy = [1=D1]
	GluS (Glux, Gluy)	Gluxy (in GluS colourset)

Table 4.5: The information needed for creating a strip of food source on one side of the grid.

4.2 Level-0: Basic Components

Level-0 (Lvl0) models are the simplest stage of modelling. These models are abstract and without detail and could not be broken down to any simpler models. They are usually made of one or two place(s) and transmitter(s) and sometimes they include one property. The Basic Components of the collection are summarised in Table 4.6.

4.2.1 Level-0 Diffusion and Movement

Diffusion is a simple model of changing the location of tokens in Petri nets, or rather the movement of the tokens on the grid. The random movement without a controlling trigger is the same as diffusion. This component only consists of a place and a transition and Grid as the property. The transition has a Neighbour function which depending on the dimension and requirement of the modeller it can be neighbour2D4, neighbour2D8 or neighbour3D26. Figure 4.2-a shows the model of diffusion/movement.

4.2.2 Level-0 Duplication

Lvl0 Duplication model contains one place and one transition connected to each other with two arcs. The out going arc from the place to the transition carries the weight 1 but the incoming arc from transition to the place carries the weight of two. This means for any one tokens that goes to the transition, two tokens will come out. The model can be seen in Figure 4.2-b.

4.2.3 Level-0 Death

Lvl0 Death model is a very simple construction which consists of one place and one transition. The transition does not have any functions defined but the rate of it could be changed as required. Figure 4.2-c shows the Lvl0Death model. In a coloured model, As this model does not have movement attached to it, the death only happens in one location defined by the colourset of the places. Since this model is in the ColPN, this model also includes the Grid property.

4.2.4 Level-0 Transmitter

Lvl0 Transmitter is a part of the model that produces signalling molecules. In the case of quorum sensing, it is the part of the cell that produces the autoinducers. The simplest way to show Transmitter, is two places connected by one transition. The model is shown in Figure 4.2-d.

4.2.5 level-0 Receiver

Receiver model is more complex than transmitter. This basic component responds to the signalling molecule by initiating some changes, such as biofilm formation. In the Receiver model,

Basic Component	Properties	Section	Figure
Lvl0 Diffusion and Movement	Grid	section 4.2.1	Figure 4.2-a
Lvl0 Duplication	Grid	section 4.2.2	Figure 4.2-b
Lvl0 Death	Grid	section 4.2.3	Figure 4.2-c
Lvl0 Transmitter	Grid	section 4.2.4	Figure 4.2-d
Lvl0 Receiver	Grid	section 4.2.5	Figure 4.2-e

Table 4.6: Table of Basic Components and their properties.

there is a place for receiver and a place defined as "RecACC" which is where the tokens from receiver place are accumulated, the same way that the biofilm could be accumulated on the grid. The signals emerge from a transition, so they are unlimited but could be changed to a limited entity if required. The model can be seen in Figure 4.2-e.

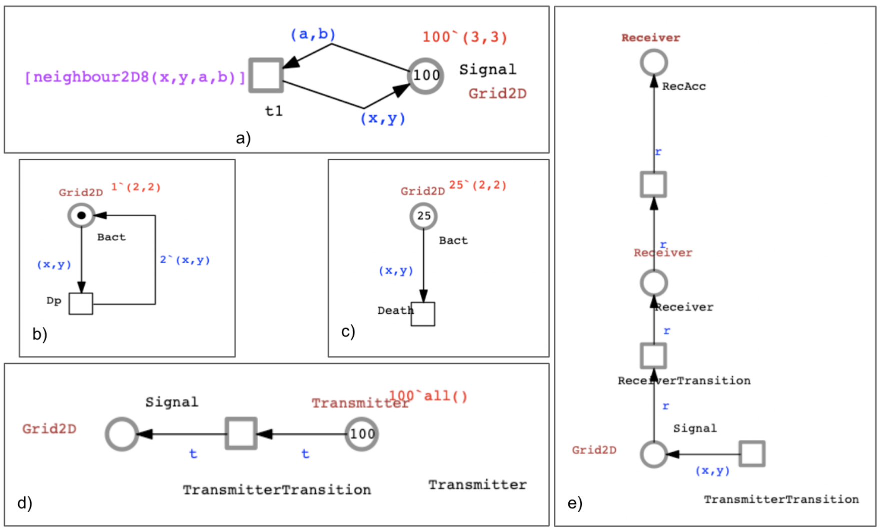


Figure 4.2: Four main Level-0 components. a) Diffusion, b) Duplication, c) Death, d) Transmitter, and e)Receiver.

4.3 Level-1: Simple Models

Level-1 (Lvl1) components are models made of one or two Level-0 basic components and one or more properties. They are simpler than complex models (Level-2 and Level-3 models), and could be broken into basic components. Table 4.7 at the end of this chapter provides a list of all the simple models in this thesis.

4.3.1 Level-1 Transmitter

The Lvl1 Transmitter, is the combination of basic Transmitter, Diffusion/Movement and the Sink property. For studying this model under different conditions, one option is to alter the rate action of diffusion, Sink or TranmsitterTransition which produces the signal. This model mimics the broadcasting communication of the bacteria without any receivers for the signalling molecules. Even though the Sink is usually present for the removal of Glu from the grid, here it is used for removing the signal from the environment to avoid its accumulation. Figure 4.3 demonstrates the SM Transmitter model.

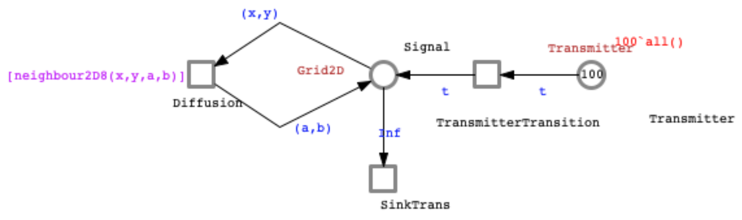


Figure 4.3: Level-1Transmitter: a combination of BC Transmitter, Diffusion and Sink

Required components:

- Level-0 Transmitter (4.2.4),
- Diffusion / Movement (4.2.1),
- Properties: Sink, Grid.

4.3.2 Level-1 Receiver

The Lvl2 Receiver is the Receiver and Diffusion/movement combined. This model, Figure 4.4, shows the microorganisms which carry the receptor for the signalling molecule, move randomly on the grid and are not fixed in one place. Since this model is in colour and contains movement, it is coherent that the Grid would be included in this Simple Model as well.

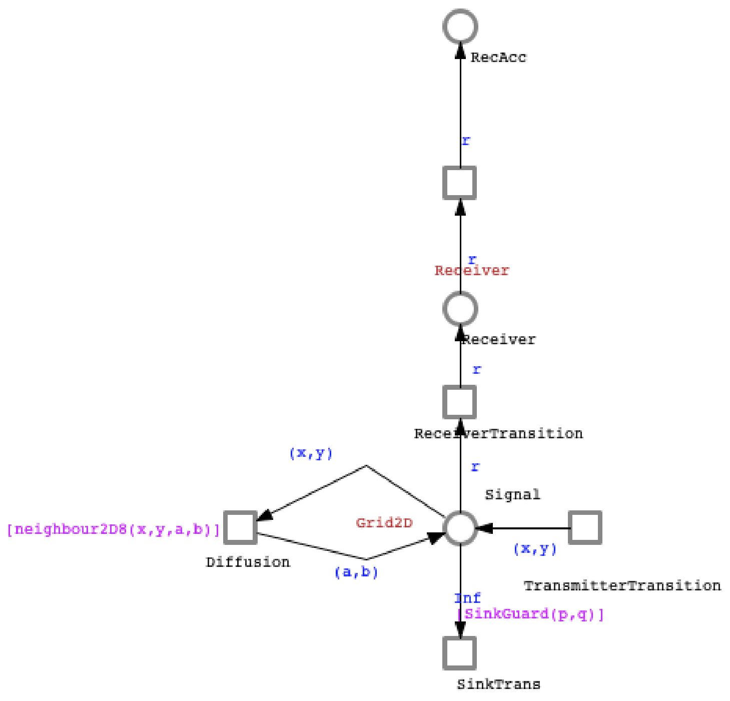


Figure 4.4: Level-1 Receiver which consists of BS Receiver and BC Diffusion/Movement, as well as Grid as a property.

Required components:

- Level-0 Receiver (4.2.5),
- Diffusion / Movement (4.2.1),
- Properties: Sink, Grid.

4.3.3 Level-1 Chemotaxis

Chemotaxis, moving towards a chemical trigger, has been observed in many organisms. While communicating, some microorganisms such as bacteria move toward the food source (chemotaxis), accumulate and produce signalling molecules [135]. Some types of communication such as quorum sensing, when the density of the signalling molecule in the environment reaches a certain threshold, the microorganisms which have the receptor for it, start importing and then respond to it in different ways such as biofilm production [110].

The aim is to model chemotaxis, the movement towards food or any other chemical trigger, via the detection of the gradient of it in the environment on the gird. In order to do so, unlimited food is produced for the microorganism by connecting a transition to “Glu”. For the microorganisms to sense the food, it is essential that the food also diffuses in the grid, however, to prevent its accumulation on the grid we connect it to the Sink transition, placed on the edges, to leave the system. The chemotaxis model is shown in Figure 4.5.

The chemotaxis model is the combination of diffusion and movement plus the properties of Glu and Sink. Properties are explained in section 4.1 above.

Required components:

- Diffusion / Movement (4.2.1),
- Properties: Sink, Grid, Glu.

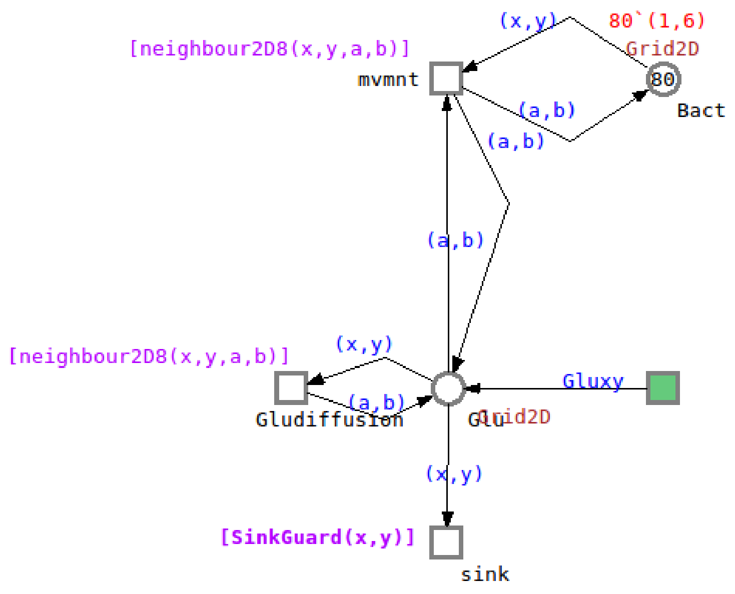


Figure 4.5: Level-1 Chemotaxis which is a combination of diffusion, movement, Glu and Sink.

4.3.4 Level-1 Duplication and Death

Simple models of duplication and death are combinations of components death and duplication plus diffusion/movement. Figure below, 4.6, shows how this combination looks. These models indicate that the bacteria move randomly on the grid while going through reproduction. To make these two behaviours dependent on a chemical trigger such as food, adding SM Chemotaxis is possible which will create a system since it would be a combination of two Simple Models.

Required components for Death:

- Level-0 Death (4.2.3),
- Diffusion / Movement (4.2.1),
- Properties: Grid.

Required components for Duplication:

- Level-0 Duplication (4.2.2),

- Diffusion / Movement (4.2.1),
- Properties: Sink, Grid.

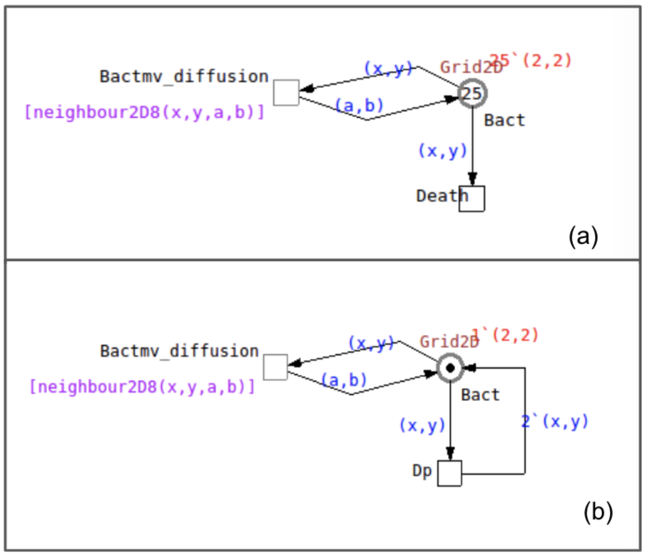


Figure 4.6: Level-1 Models of a) death, and b) duplication.

4.3.5 Level-1 Limited Strip Glu

This simple model is a combination of Glu, Diffusion/Movement and Sink. Using the Gluxy variable, as explained in the previous chapter, the location of Glu is defined on the right side of the grid, and creates a strip of food source on one side of the grid. From there, The food, Glucose or any other chemical compounds diffuses to the rest of the grid and eventually leaves the grid through the Sink on the other three edges. To make the Glu unlimited, we can add a transition before GluSource Place, using Gluxy as a variable on the arc to make sure the location is as required. The model can be seen in Figure 4.7.

Required components:

- Diffusion / Movement (4.2.1),
- Properties: Sink, Grid, Glu, Wall.

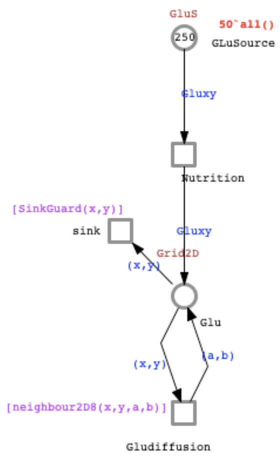


Figure 4.7: Level-1 model of Glu which is located as a strip in one side of the grid with the variable Gluxy and Sink to remove the excessive Glu out of the grid.

4.3.6 Level-1 Semi-permeable obstacle and Transmitter

The Semi-Permeable Obstacle model is a combination of Diffusion/Movement and Obstacle properties and Transmitter Basic Component. The obstacle is a simulation of the semi-permeable membrane of a cell. It has a lower action rate (massaction) than diffusion to be semi-permeable and a zero mass action to be a block on the way of diffusion of the molecules on the grid. If it is not zero, then the molecules will enter the area in a lower speed. The model can be seen in Figure 4.8. Even though Sink is not in this model, it can be added for studying it under another condition.

Required components:

- Level-0 Transmitter (4.2.4),
- Diffusion / Movement (4.2.1),
- Properties: Obstacle, Grid.

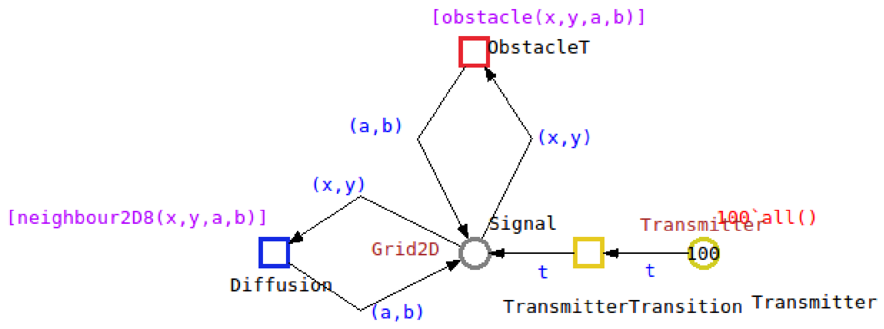


Figure 4.8: Level-1 Semi-permeable membrane model a combination of Diffusion, Obstacle and Transmitter.

4.4 Colouring the Models

In ColPN in Snoopy, giving specific data to each token to define their location is called colouring them [117]. This method gives every token a specific location and is used for multi-level and multidimensional model construction. ColPN represent a compact and folded version of a complex model where each object in the system becomes a coloured place and transition to have a location defined by colour [117]. The models made in ColPN are more realistic, since they contain space and location and the correct model can be used in many areas such as drug discovery and personalised medicine and nutrition or other research projects. It is possible to combine hybrid and colour Petri nets together to study larger biological systems and when the model is ready it can be exported automatically to hybrid Petri nets for further simulations [118].

Even though the focus of this study is mostly on the biological use of these components, it is possible to use these models in non-biological content as well. The process of colouring the models is a generic process which could be used for colouring any models.

The process of colouring in Continuous ColPN for a 2D diffusion model is as follows:

1. Open Snoopy Software. This is the “First Window”.

2. From the tab on the top of the window, click on File and then New.

3. Choose Coloured Continuous Petri Net. A new window with a white screen will be opened. This is the Screen

4. Define the “Constant” from the Declaration that is a panel on the left side of the First Window. For example: D1 = 5 and D2 = 5.

5. Define the “Simple Coloursets” from the Declaration on the First Window based on the constants from the Declaration on the first window. For example: CD1 = 1-D1 and CD2 = 1-D2.

6. Define the grid from the Declaration on the First Window by clicking on“Compound Colourset” which is a type “Product” of the Simple Coloursets. For example: Grid2D = CD1,CD2

7. Define “Variables” on the First Window based on the coloursets. If CD is the size of the x axis and CD2 the size of the y axis then x = CD1 and y = CD2. This means the variable x belongs to CD1 and the variable y belongs to CD2. These variables are used for the arcs.

8. Define the “Function" on the First Window based on the neighbouring functions provided in Table 4.2.

9. Choose “Place” from elements on the tab at top of the First Window and put a place on the Screen. Double click on the place and after writing a name for the place, on the tap above the window click on “Markings” and choose the colourset from the list. The Compound Colourset should be chosen.

10. From elements choose “Transition” and place one on the Screen. Double click on the transition and after typing a name for it, select “Gurds” from the top tab and underneath the Guard, write the name of the function with the variables. In the case of the examples above, it would be “Neighbour2D8(x,y)”

11. From the elements choose “Arc” and drag an arc from the place to the transition. Double click on the arc and clock on the “Expression” on the top tab. underneath the expression, define the variables based on the location of the arc as required. For example, (x,y) which is located in CD1 and CD2 coloursets.

12. Drag an Arc from the transition to the place and do the same steps as the other arc.

13. Now, the model is complete. This is a model of diffusion on a 5x5 grid. After saving the model, it is possible to export it to a CPN and see the unfolded version of it. Also this model can be exported to folded CANDL or unfolded ANDL file for further simulations.

Figure 4.9 demonstrates a summary of the required information for modelling in Snoopy.

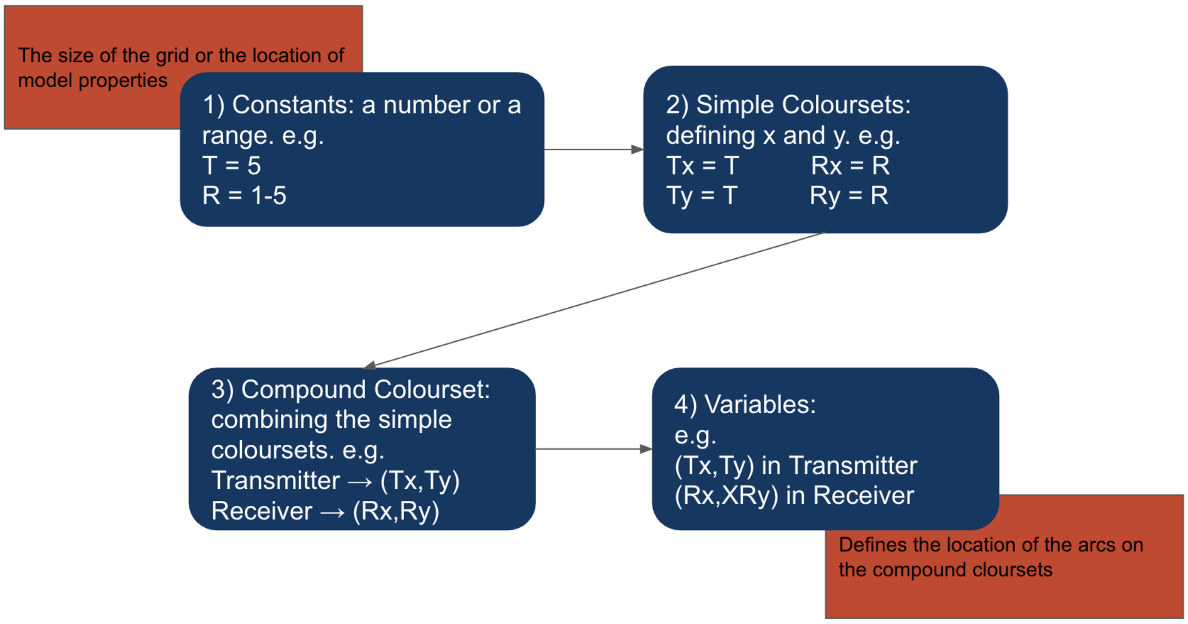

The size of the grid or the location of model properties
1) Constants: a number or a range. e.g.
T = 5
R = 1-5
2) Simple Coloursets: defining x and y. e.g.
Tx = T Rx = R
Ty = T Ry = R
3) Compound Colourset: combining the simple coloursets. e.g.
Transmitter → (Tx,Ty)
Receiver → (Rx,Ry)
4) Variables:
e.g.
(Tx,Ty) in Transmitter
(Rx,XRy) in Receiver
Defines the location of the arcs on the compound cloursets

Figure 4.9: A summary of the process of colouring a model in coloured Petri nets.

Table 5.1 is a summary of all the types of Properties that have been defined in this study. The CANDL file of these properties are stored in the data collection. It is important to remember that once the CANDL file of the properties are unfolded into ANDL files, the Grid2D property disappears as it is for labelling the places in the folded set.

4.5 Discussion

In this chapter the four levels of modelling were introduced. Properties that are the base of all the models, Level-0 which are simple and cannot be broken into simpler models and Level-1 model component which are made from simpler models and a few properties. Even though these models can be used individually, they could be applied in different scenarios to create biological systems. In the next chapter, these models will be applied to bacteria and Dicty as has been mentioned before by combining them in different circumstances. Among the applications, there are two models that are not modelled in the same manner; meaning their sub-models or components do not exist in the collection of model components. These models are the formation of biofilm and the combination of production of AI-2 and biofilm formation. However, due to their complexity, they are categorised as systems.

Model	Basic Components	Properties	Section	Figure
Lvl1 Transmitter	Transmitter, Diffusion / Movement	Sink, Grid	Section 4.3.1	4.3
Lvl1 Receiver	Receiver, Diffusion / Movement	Sink, Grid	Section 4.3.2	Figure 4.4
Lvl1 Chemotaxis	Diffusion / Movemen	Glu, Sink, Grid	Section 4.3.3	Figure 4.5
Lvl1 Death	BC Death, Diffusion / Movement	Grid	Section 4.3.4	Figure 4.6-a
Lvl1 Duplication	BC Duplication, Diffusion / Movement	Sink, Grid	Section 4.3.4	Figure 4.6-b
Lvl1 Limited Glu Strip	Diffusion / Movement	Wall, Grid, Sink	Section 4.3.5	Figure 4.7
Lvl1 Semi-permeable Obstacle Transmitter	Transmitter, Diffusion / Movement	Obstacle	Section 4.3.6	Figure 4.8

Table 4.7: Table of Simple Models and their components and properties.

Chapter 5

Application of Models from the library

In this chapter the components of the model collection are used to create complex models or Systems. Systems are complex networks which are created as combination of some Basic Components and/or Simple Models with some properties and they can be broken down into smaller working models. These systems are applied to some real case scenarios in two microorganisms: bacteria, more precisely *E. coli*, and Dictyostelium or slime mould. These two are the representatives of each main biological superkingdom: Eukaryote and Prokaryote.

There is an exception of construction in the application in this study. Bacterial communication and response model (Quorum Sensing), Signal Production and Biofilm Formation, are not created from the components of the collection of models in this study. The Biofilm Formation model, Section 5.1.4 was originally created as a detailed model from scratch and was combined with the Signal Production model which was constructed by Li *et al.* [114], in Section 5.1.5. Since these models are detailed and complex and could be broken down to smaller components, they are treated as complex systems. Looking at Biofilm Formation Model, it can be broken down into possible Level-1 and Level-0 components. However, these components are not a part of the current library but they can be added in a further study. The Quorum Sensing model is an assembly of Biofilm Formation and Signal Production models, which both are in Level-2, and is categorised as the only Level-3 model in this study.

The other models such as transmitter and receiver as an abstract form of communication in a heterogeneous society in section 5.1.1 and Duplication, Chemotaxis and Death in section 5.1.2, Combined Strip Glu, DCD and TR as a combination of most of the components in this library in section 5.1.3 and the communication model of Dictyostelium in section 5.2 are all built from the

Properties, basic components (Level-0) and simple models (Level-1) in this study. In each part the process of combination is also provided if relevant. It is important to remember that the process is to introduce the required parts and recognise the repeated parts of the model that should be removed in order for the models to be combined. The provided guidance is rather a suggestion than a set of instructions. What matters is to acknowledge the constructing components of the model. At the end of this chapter a table is provided which summarises all the model components and systems in this study

5.1 Bacteria *E. coli*

Systems in this study consist of more than two Properties, two Basic Components and/or Simple Models together. One of the two microorganisms that were chosen for case study was bacteria *E.coli* and the application of the detailed model as well as the models built from the collection of models are presented int this section.

These models include:

- Transmitter and Receiver, 5.1.1
- Duplication, Chemotaxis and Death (DCD), 5.1.2
- Combined Strip Glu, DCD and TR, 5.1.3
- Biofilm Formation, 5.1.4
- Quorum Sensing and Response, 5.1.5

5.1.1 Level-2 Transmitter and Receiver(TR)

A combination of Lvl1 Transmitter and Lvl0 Receiver models as well as Lvl0 Diffusion, Grid and Sink properties results in “Transmitter and Receiver” model. In this model the Transmitter produces a signal. The signal is then diffused on the grid and then reaches the Receiver. In this model, two different Compound Coloursets are also defined for each of the models in order to define the location of each part of the model separately. The other method is to simply define the location in marking of the places while setting the colourset in Grid2D. Using different coloursets makes amending the values and constants easier, as they are generalised and defined through constants in the model. For this system, see Figure 5.1.

Required components:

- Level-1 Transmitter (4.3.1),
- Level-0 Receiver (4.2.5),
- Diffusion / Movement (4.2.1),
- Properties: Grid, Glu.

Combination Process: Open the required Transmitter model and Receiver model. Select all the elements in Receiver model and copy them in Transmitter model (both models should have the same format, e.g. both should be ColCPN). Remove diffusion, sink and transmittertransition from Receiver model and connect the signal place of Transmitter to ReceiverTransition of Receiver model with an arc. set the correct values on the arcs.

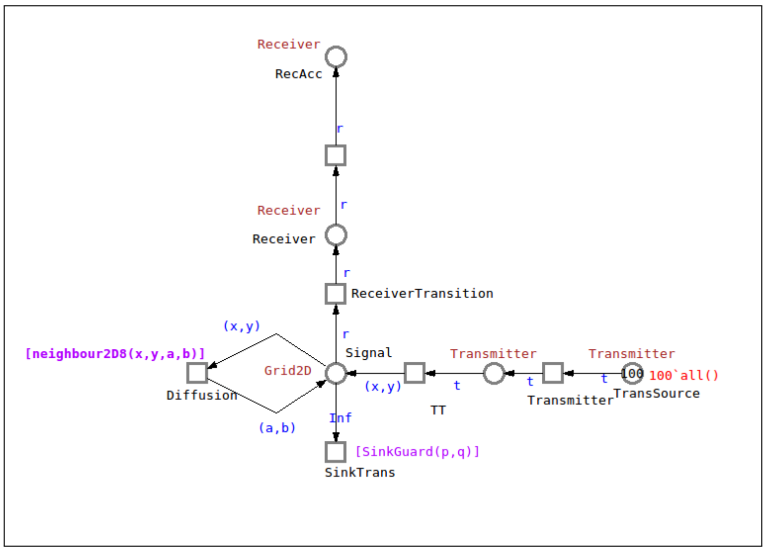


Figure 5.1: Simple system model of transmitter and receiver, defined in separate coloursets.

5.1.2 Level-2 Duplication, Chemotaxis and Death (DCD)

Movement, death and duplication are essential parts of microorganisms' lives. The mechanism of movement is different in each organism and cell. For example, bacteria produce flagella while Amoebae moves by extending its cytoplasm. The mechanism of the movement is not an argument for modelling in this collection, however the manner of it is. Meaning these microorganisms move towards a trigger in order to survive and that is what the chemotaxis model represents. This system, combines the chemotaxis with Lvl0 Duplication and Lvl0 Death. The duplication rate is dependant on the trigger as well, so that there is more duplication when the bacteria is closer to the source of the trigger. This model also includes, Glu, Sink, Grid and Wall Properties. Figure 5.2 shows the model.

Required components:

- Diffusion / Movement (4.2.1),
- Level-0 Death (4.2.3),
- Level-0 Transmitter (4.2.4),
- Level-1 Chemotaxis (4.3.3),
- Level-0 Duplication (4.2.2),
- Properties: Glu, Grid, Sink.

Combination Process: After making sure that all the models are in the same formate, Add Duplication, Death, Diffusion / Movement and Glu in one Petri net file. Connect death to the Bact place with (x,y) for its expression. connect division to Bact place with the out going arc (x,y) and ingoing 2‘(x,y). Connect Diffusion / Movement transition to Glu, with two arc and both (a,b) expression. When the expression of the arc to and from Diffusion is set as (a,b) it attracts the tokens but does not affect them in any other way. connect Glu to Sink transition with Inf expression, or use the Sink_Guard function for it.

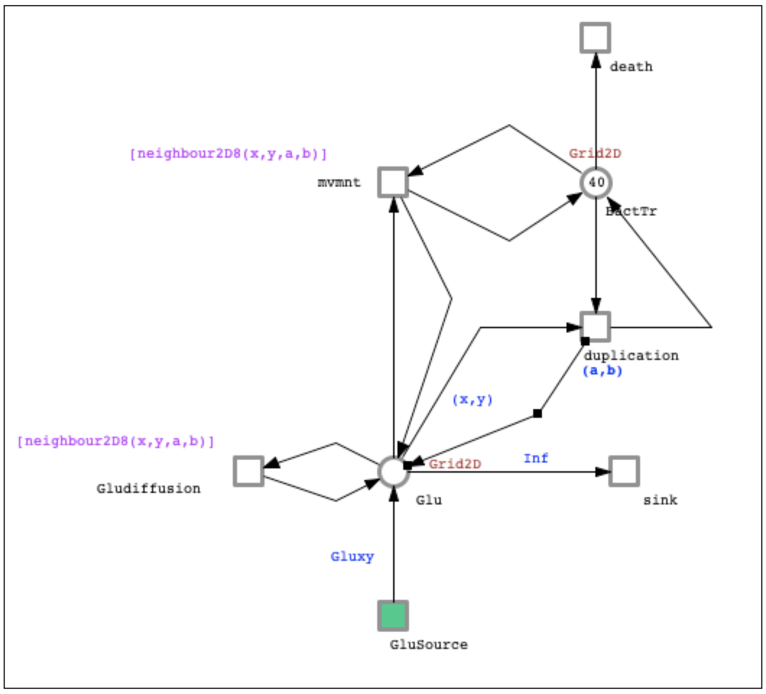


Figure 5.2: Duplication, Chemotaxis and Death model. division and movement are dependant on Glu and Glu is removed form the Sink placed on the edges of the grid.

5.1.3 Level-2 Combined Strip Glu, DCD and TR

The final model that has been applied to bacteria, is the combination of all Lvl0 the components in the library, except for detailed systems of AI-2 production and Biofilm formation. Figure 5.3 presents this system. This model includes Chemotaxis, Duplication, Death, Transmitter (BactT) and Receiver (BactTR) models, which produce AI-2 and biofilm respectively, as well as Glu, Sink, Wall and Grid properties. The model is created on ColCPN as the population of bacteria is presented as a density rather than the individual population.

Required components:

- Diffusion / Movement (4.2.1),
- Level-0 Death (4.2.3),
- Level-0 Transmitter (4.2.4),
- Level-0 Receiver (4.2.5),

- Level-0 Duplication (4.2.2),
- Level-1 Chemotaxis (4.3.3),
- Properties: Glu, Sink, Wall, Grid.

Combination Process: Make sure all the models' formats are the same. Create a base model in the desired Petri net format and define the necessary constants, coloursets, values, etc. Copy chemotaxis and paste it into the base model. Remove the Glu part of the chemotaxis model (Glu, sink and diffusion are repeated in Glu model) and connect Bact to Glu with 2 arcs both with (a,b) variables. Rename Bact to BactR representing Receiver. Once again paste chemotaxis model and do the same. This time rename Bact to BactTR representing Transmitter. Paste Division two times and connect it to both Bacts. Paste death two time and connect it to both Bacts. Remove the movement if the SM models were used. Add a transition for AI2 production (AI2production) and connect BactTR to it. Define its massaction rate based on the provided parameters in the base model. Add a place for AI2 (AI2) and connect it from AI2production transmition to AI2 place. Add a diffusion to this place and define a massaction rate for it. Add a transition next to BactR for biofilm formation (biofilmpro) and define a massaction for it. Then add the Biofilm place and connect it from the biofilmpro transition to biofilm place. Add diffusion to biofilm place and define a mass action. All new places are on Grid2D colourset. It is possible to use Receiver and Transmitter models as well. The model can be seen in Figure 5.3.

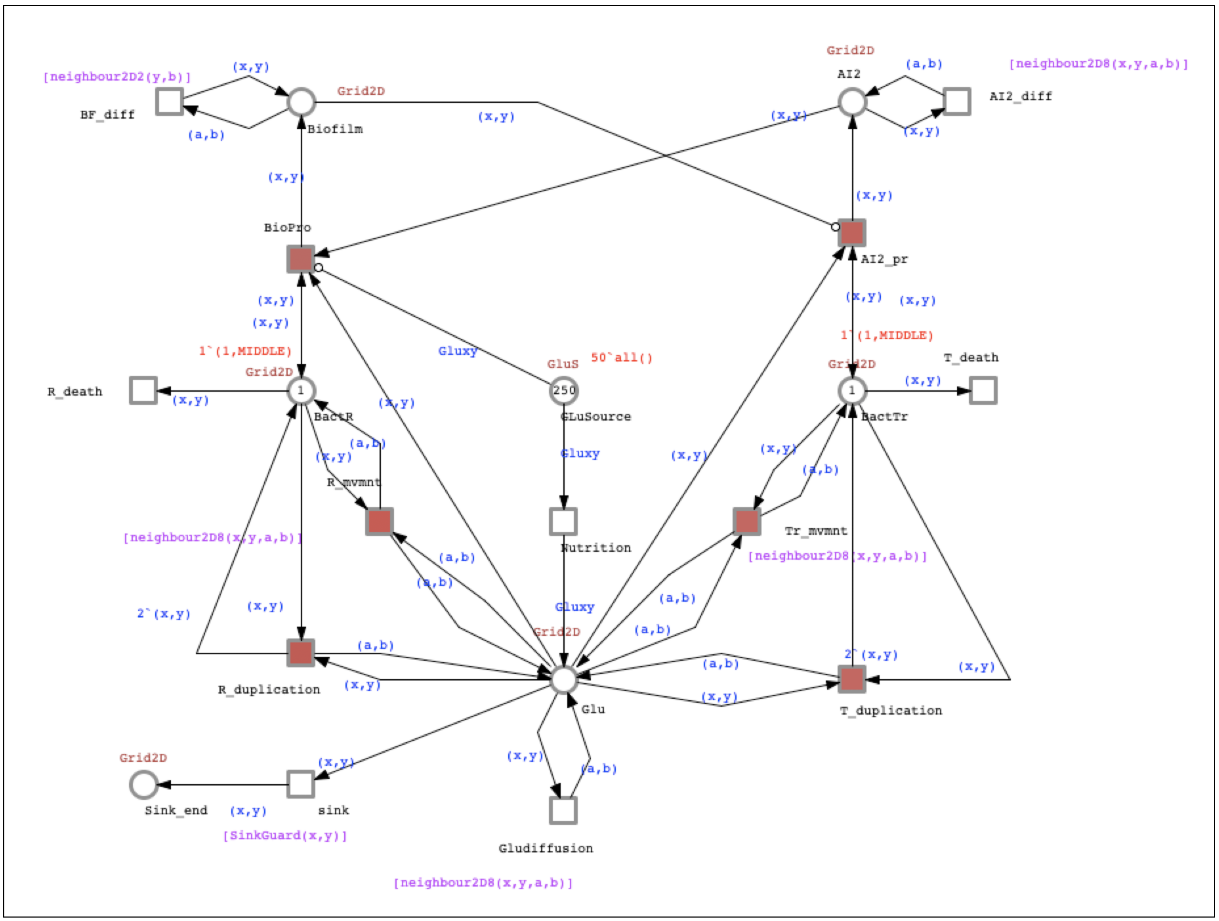


Figure 5.3: Simple system model of transmitter and receiver, defined in separate coloursets.

The structure of the model is as follows: GluSource is placed on the right side of the grid and is initialising the simulation. GluSource inhibits the production of biofilm (Biopro) Glu receives the token from Glusource and can diffuse throughout the whole grid. It exits the grid through “Sink” in order to prevent its accumulation on the grid. Glu initiates T_ and R_duplication and causes chemotaxis of both types of bacteria. Both BactTr and BactR use Glu to duplicate, However, Glu only attracts the bacteria towards the right side of the grid where food (carbon source) was initiated. BactTr produces AI2. AI2 diffuses on the grid. BactR produces biofilm only when the number of tokens in GluSource is around one (The inhibitor arc should start the biofilm formation when GluSource is zero but it does not happen. It seems it does not inhibits 100%) The produced biofilm diffuses on the grid as well as inhibiting AI2 production.

5.1.4 Level-2 Biofilm Formation

Biofilm is one of the responses to the bacterial communication. The Biofilm Formation model (BF) was originally made as a whole as a detailed model and its components currently do not exist in the collection of models of this thesis, except for the Diffusion. However, this model could be broken to smaller parts as Basic Components and that can be a part of another study. This model was designed in 2018 and was published in BMC Bioinformatics journal in 2019 [77]. BF describes the process of biofilm formation in *E.coli* or similar bacteria in detail. The biological study of this process was explained in Chapter 2 Section 2.4.1. Later on this model is combined with the model of production of AI-2 done by Li *et al* [114] to predict the behaviour of bacteria under different circumstances during quorum sensing and biofilm formation with a focus on the effect of the distance between the cells on communication. The model is shown in Figure 5.4.

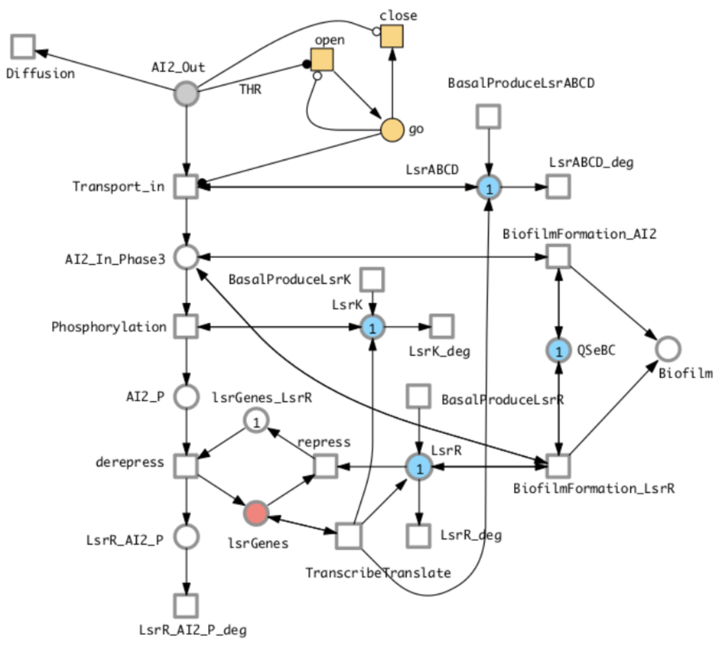


Figure 5.4: The model constructed by our group to show biofilm formation in bacteria.

The "go system" in this model is created by Professor Monika Heiner [77] which creates a gate in the cell. This gate only opens when the AI-2 reaches threshold which is defined for this system and is amendable as required. The other parts of the model describe the process of the production of biofilm which was explained vastly and in detail in Chapter 2.

5.1.5 Level-3 Quorum Sensing

This combined model, published in 2019 in BMC Bioinformatics Journal [77], was the first system designed in this study. This model is a combination of two detailed models. The quorum sensing model was designed by by Li *et al.* [114] in 2006 and was combined by the biofilm formation model in 2019. The only part of this model that exists in the collection of model components, is the Diffusion. The model was designed on both 2D and 3D girds. Hybrid Petri nets were used for modelling which means both continuous and stochastic features were present in the models. This model shows a broadcast communication occurring in two phases of bacteria life when they produce AI-2 and then as a response form biofilm. A system was created by Professor Heiner which made it possible to define threshold for the response to the AI-2 signalling molecule. Unless that threshold is reached, the bacteria do not allow AI-2 to enter. The model consists of gene transcriptions, protein productions and inter- and intra-cellular communication [77]. Due to being a detailed model in can describe may processed in the system. The model can be found in Figure 5.5. The two AI-2 places in this model are logic places (coloured in Grey), i.e. they represent the same molecules but are shown with two places.

On an abstract scale, this model is basically a detailed Transmitter, a detailed Receiver and a Diffusion / Movement model. Currently it is not possible to move complex systems in Snoopy. Which led us to designing an abstract version which could represent the movement. The abstract components are explained in 4.3.6.

Required Components:

- Biofilm Formation (5.1.4),
- Signal Production (from Li *et al* [114]),
- Diffusion / Movement (4.2.1),
- Properties: Grid.

Combination Process: Open Both models and make sure their formats are the same, meaning both are coloured Continuous Petri nets or Stochastic Petri nets. Select all the elements in Biofilm_LG and copy them. Paste the model in AIproduction_Li. Remove the Inflow transition from Biofilm_LG model. Double click on one of the AI-2 places and check the box next to logic. Do the same for the other AI-2 place as well.

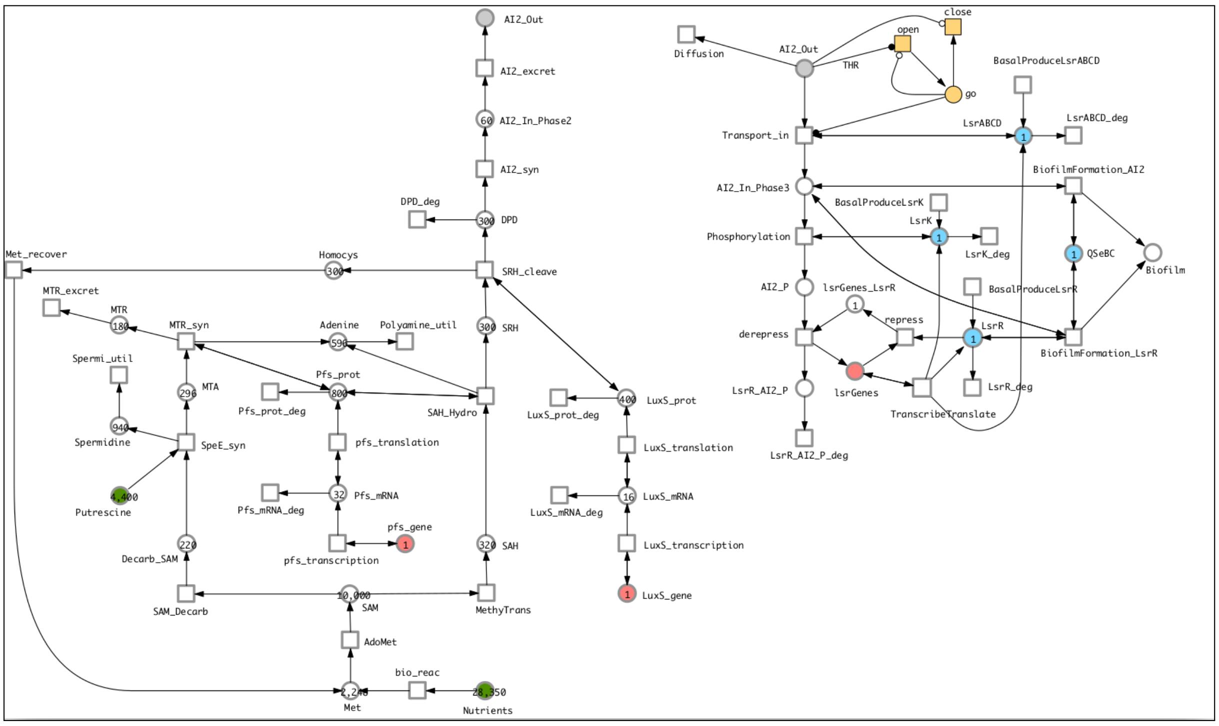


Figure 5.5: System model of AI-2 production and biofilm formation combination of two level-2 components. the AI-2 production part of the model was designed in 2006 by Li *et al.*.

5.2 Level-2 Dictyostelium: Slime Mould

Dictyostelium (Dicty) is a Eukaryote unicellular microorganism that has been a popular model organism for studying chemotaxis as a response to cAMP [31]. Dictyostelium cells produce cAMP in order to communicate with one another. The cells then are attracted to the location with the highest concentration of cAMP [31, 71]. Similar to bacteria, this movement is based on the density of cells meaning that the cells gather in the place with the most concentration of the signalling molecule. However, there is a difference between how chemotaxis is modelled in bacteria and Dictyostelium.

In the model of chemotaxis of bacteria, the bacteria move towards a stable source of food that has a gradient on the grid. While doing so, they communicate with each other and once they reach the surface and the amount of the signalling molecule has reached a threshold, then the bacteria start its changes, such as biofilm formation. In Dictyostelium model, the cells are attracted towards the concentrations of the signalling molecules which are being diffused on the grid. This means that the attraction is not in one place and does not show a stable gradient. As a result, the bacteria gather around the “Wall” while the Dictyostelium cells might gather anywhere on the grid as long as there is a high concentration of cAMP. The model can be observed in Figure 5.6.

This abstract model demonstrates the movement of Dictyostelium towards cAMP, while cAMP is diffused on the grid. It is important to note that Dictyostelium do not absorb the molecules and just sense them in order to move towards it. Using a Shell program, the Dictyostelium cells are located randomly on the grid. This shell program, written by Professor David Gilbert, defines the grid size and the population and produces CANDL and sps files. For axample, for a grid size of 15x15, around one third of the grid is occupied. However this one third is not an absolute value and varies between 25% to 30% . The shell program is a random algorithm that sets numbers 0 to 3 ion each location and then removes the numbers 2 and 3. In Random Restart Hill Climbing, the initial population of Dicty is 61, 27.11% of the grid and in simulated annealing this number changed to 69 which is 30.66% of the grid.

Once located on the gird, the cells start producing cAMP and naturally will move towards where the concentration of this molecule is the highest. The Sink transition is only connected to the cAMP in order to prevent too much concentration of the molecule on the grid. Otherwise, there will be too much cAMP everywhere and the movement would not happen.

Required components:

- Diffusion / Movement (4.2.1),
- Level-1 Chemotaxis (4.3.3),
- Properties: Glu, Sink.

Combination Process: Make sure all the models' formats are the same. Create a base model in the desired Petri net format and define the necessary constants, coloursets, values, etc. Copy chemotaxis and paste it into the base model. Rename yarayara to Dicty and Glu to cAMP. Copy and Paste Diffusion / Movement model. Remove the place connected to Diffusion transition and connect the transition to cAMP. Do the same with Sink. Add the transition between Dicty and cAMP for cAMP production. Connect Dicty with two arcs to this transition, so that the Dicty is not used for production of cAMP. connection this transition to cAMP with one arc. All these three arcs should carry the expression (x,y).

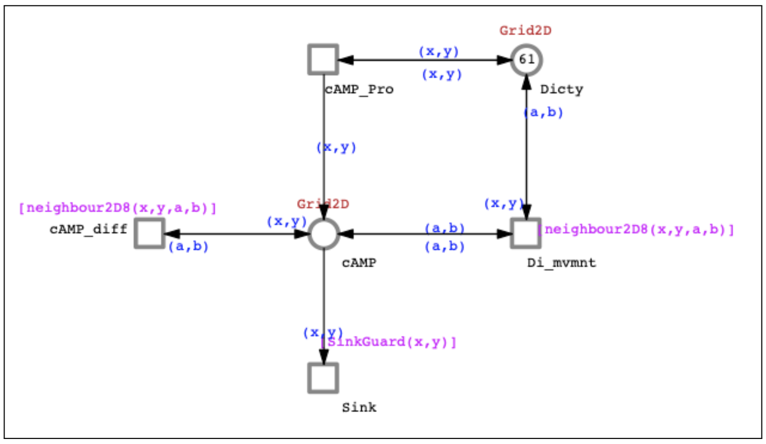


Figure 5.6: Model of Dictyostelium chemotaxis.

5.3 Collection of Model Components

The aim of this study is to create a collection of model components in different levels of complexity. These models have been introduced and discussed in this and previous Chapter. There are two different groups in the collection: the properties and the model components that are categorised based on complexity. Table 5.1 is a summary of the properties and their definition and use while Table 5.2 provides a list of all the models in this library and their components.

Property	Definition
Grid	The 2D or 3D space that is defined for the model.
Neighbour Functions	Define the movement of the tokens and how the transitions are set on each location.
The Sink	The transitions on the edges of a 2D grid or faces of a 3D grid that remove the tokens from the grid.
The Wall	By removing the sink from one side of the grid, a wall is created that from that side no tokens leave the grid.
The Obstacle	A membrane that can be located anywhere on the grid. It can be rigid or semi-permeable.
The Glu	The trigger for movement which resembles a food source. It is places on one side of the grid usually on the same side as the Wall.

Table 5.1: A summary of all the properties that are defined in this study.

5.4 Discussion

Two biological prototypes were chosen to employ the models from the collection of model components on them. These prototypes are from Prokaryote and Eukaryote biological superkingdoms and are used as simple samples to represent their families. Bacteria, and more precisely *E. coli*, is one of the chosen prototypes. Using the library of models, an abstract model demonstrating chemotaxis, duplication, death, communication and response was constructed. Also, a model of biofilm formation was designed, and assembled with another model built by Li *et al* [114] for a detailed model of quorum sensing and biofilm formation [77].

The other chosen biological sample is Dictyostelium, the slime mould. This microorganism have been studied for its unique and interesting communication behaviour. While it is a unicellular organism, it can join in to build a society as a survival strategy. At this time the cells gather together and eventually produce a fruiting body which carries their spores for reproduction purposes. The focus of this study is to simulate the migration stage of Dictyostelim life, when they are accumulated in one location to create the fruiting body. Using different parts of the model library, an abstract model was designed to show the communication towards the high concentration of cAMP and the accumulation of the cells.

Even though these models form the library have been used to demonstrate biological behaviour, it is possible to use them for other purposes, as the models are generic and abstract. The models in this thesis are all continuous. Therefore, the numbers that describe population are non-integers for population density and not individuals. Due to technical issues, it is not possible to study complex systems in stochastic. To simulate complex system Spike is required but the current version of Spike at the time of writing this thesis, cannot simulate complex stochastic models. In further works, this issue can be resolved with the aid of the developers.

In the next chapter, Chapter 7, the output data from the models and optimisation are provided and analysed using R and Python.

Table 5.2: Table of model components in the collection, from both categories of Basic Components and Simple Models

Complexity Level	Model	building Components	Properties	Section	Figure
Lvl0	Diffusion / Movement	N/A	Grid	4.2.1	4.2-a
Lvl0	Duplication	N/A	Grid	4.2.2	4.2-b
Lvl0	Death	N/A	Grid	4.2.3	4.2-c
Lvl0	Transmitter	N/A	Grid	4.2.4	4.2-d
Lvl0	Receiver	N/A	Grid	4.2.5	4.2-e
Lvl1	Transmitter	Lvl0 Transmitter, Diffusion / Movement	Sink, Grid	4.3.1	4.3
Lvl1	Receiver	Lvl0 Receiver, Diffusion / Movement	Sink, Grid	4.3.2	4.4
Lvl1	Chemotaxis	Diffusion / Movement	Glu, Sink, Grid	4.3.3	4.5
Lvl1	Death	Lvl0 Death, Diffusion / Movement	Grid	4.3.4	4.6-a
Lvl1	Duplication	Lvl0 Duplication, Diffusion / Movement	Sink, Grid	4.3.4	4.6-b
Lvl1	Limited Glu Strip	Diffusion / Movement	Wall, Grid, Glu	4.3.5 Sink	4.7
Lvl1	Semi-permeable Obstacle Transmitter	Lvl0 Transmitter, Diffusion / Movement	Obstacle, Grid	4.3.6	4.8
Lvl2	Transmitter and Receiver	Lvl1 Transmitter, Lv0 Receiver, Diffusion / Movement	Grid, Sink	5.1.1	5.1
Lvl2	Duplication, Chemotaxis and Death (DCD)	Diffusion / Movement , Lvl0 Death, Lvl0 Duplication, Lvl1 Chemotaxis	Grid, Glu, Sink	5.1.2	5.2
Lvl2	Combined Strip Glu, DCD and TR	Diffusion / Movement , Lvl0 Death, Lvl0 Transmitter, Lvl0 Receiver, Chemotaxis, Lvl0 Duplication	Glu, Sink, Wall, Grid	5.1.3	5.3
Lvl2	Biofilm Formation [a]	Diffusion / Movement	Grid	5.1.4	5.4
Lvl2	Dictyostelium: Slime Mould	Diffusion / Movement, Chemotaxis	Glu, Sink	5.2	5.6
Lvl3	Quorum Sensing and Response [b]	Signal Production, Biofilm Formation Diffusion / Movement	Grid	5.1.5	5.5

[a]This model is not built from the components of the library, as it originally was a detailed model. However it includes Diffusion / Movement Component

[b]This is a combination model of Quorum sending model designed by Li [114] and the Biofilm formation

Chapter 6

Optimisation

Optimisation is a target driven heuristic method to find a set of the best solutions for a problem [100]. Heuristic algorithms provide the closest answer to an optimised solution [109]. Different types of optimisation methods are chosen based on the problem. In this study two different methods were chosen for the same problem: Finding the optimised rate actions for the model “Dicty”. These two methods include Random Restart Hill Climbing and Simulated Annealing. These two methods are well-known, relatively simple to understand and employ, and computationally fast. While Random Restart Hill Climbing can look for the local (or possibly global) maxima, Simulated Annealing can explore the search space for the best possible solutions. The objective of the optimisation is to find the maximum number of Dictyostelium cells that can accumulate in one location on the grid to model the formation of fruiting body. The target function is based on the number of Dicty cells on the grid regardless of their locations. Based on this objective, these two methods seem effective and appropriate.

The code for these methods have been employed in Python. Considering the fact that Random Restart Hill Climbing is a simple method yet not very accurate, as it might provide a local maximum instead of the global maximum, Simulated Annealing method was also used. Using two different methods it is possible to compare the results and conclude which rate(s) have the most effect on the maximum number of Dicty in one location. In this chapter, after a brief section of definitions that have been used for optimisation, each method is explained and the pseudo codes for them for them is provided. The full codes are presented in Appendices B and C.

6.1 Definitions

Iteration: The number of loops of simulations. In Random Restart Hill Climbing, there are two Iterations given by the user. One for the Random Restart of the programme and one for Hill Climbing steps.

Rate constant: The rate of action for each transition in the model of question. There are four transitions in the Dicty model to be optimised and all of them are being amended during the optimisation. This means unlimited number of possibilities for 4 numbers and these numbers are in a range of 0 to 20.

Delta: After the starting point, specially in Hill Climbing method, the next answers are based on the previous one. Delta is a number defined in the algorithm that changes the current solution and creates the new solution.

Temperature: In Simulated Annealing, the Temperature decreases in each iteration, either way. This temperature is calculated based on the number of Iterations that is set as 100 at the beginning of the simulation.

Delta condition: Rates less that zero do not make sense in our model. Rate zero means deactivation of the transition and more than zero defines the activity rate. To avoid achieving less than zero rate action, a condition is set in the algorithm that changes the solution, if it is less than zero.

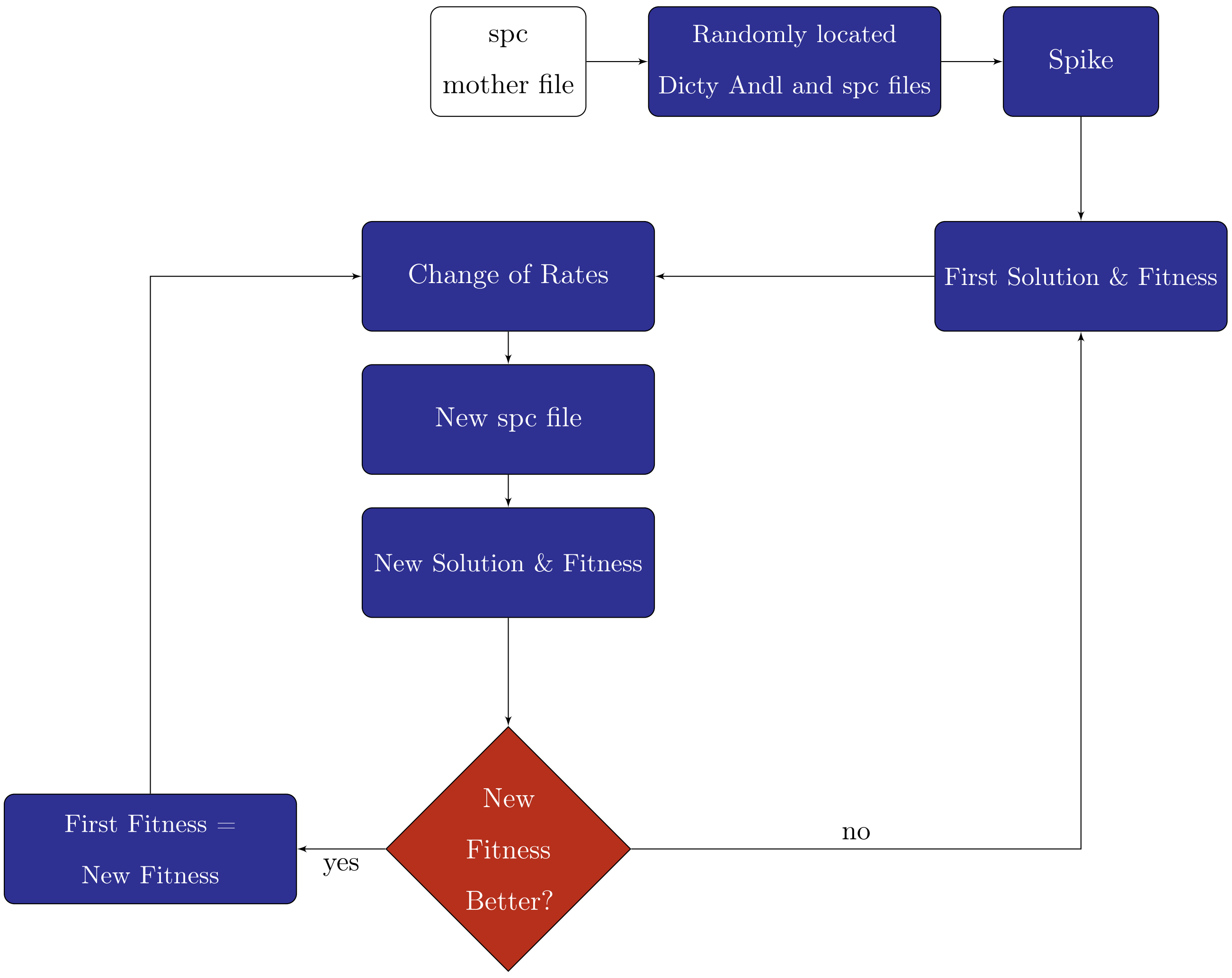
Acceptance Condition: The new solution is accepted only when it is more than the previous solution. To check that, the maximum amount of Dicty is checked on the grid, since the adjective is to achieve the highest number of Dicty possible. Checking that, if the new maximum number of Dicty is higher than the previous, it is accepted, otherwise it is rejected. In Simulated Annealing, since there is the Acceptance Possibility (AP), the acceptance condition is slightly different. In Simulated Annealing, if the new maximum number of Dicty is not more than the previous, the AP is calculated which is based on the temperature. the lower the temperature, the lower the possibility of accepting the new solution. Other than that, the new solution is rejected.

Values_index: Values_index is the csv file output from the optimisation which provides the required information such as the location of the maximum amount of Dicty, the old and new solution of the number of Dicty, the rates, the calculated AP, the number of Dicty present on the grid and the status of the solutions, whether they are accepted or rejected; and in case of Simulated Annealing whether they were accepted due to high AP.

Mother file: The mother files is the original spc file for Spike created before the simulation. In this algorithm, a copy is created and all the changes is intended on the copy file. Thus the mother file remains the same during the process of simulation.

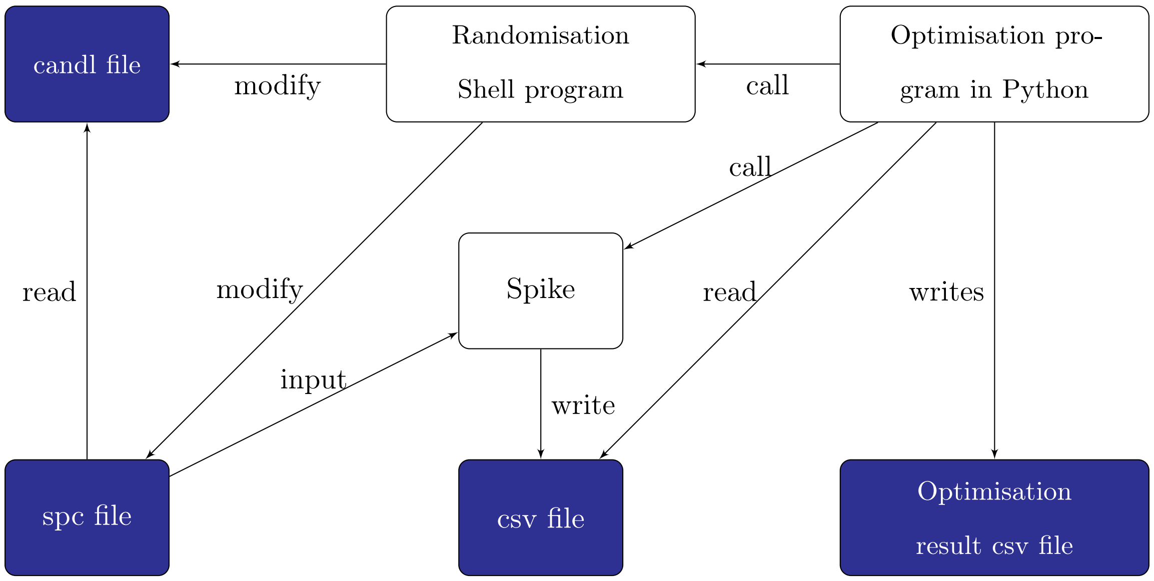
Workflow in Figure 6.1 exhibits the process of optimisation regardless of the method. The "Randomly located Dicty Andl and spc files" are created by the shell program that places Dicty cells randomly on the grid and creates the relative spc file. In Random Restart Hill Climbing this happens at the beginning of every random restart and in Simulated Annealing it. only happens once at the start of the optimisation. The Fitness is accepted when the new maximum population on the grid is more than the last.

Figure 6.1: Flow chart of optimisation. The process of optimisation regardless of method in this thesis.



Python code in both Random Restart Hill Climbing and Simulated Annealing call Spike and use the output from it. The output is inspected by the target function in the code and while it extracts import information from the simulation, it decides whether to accept the new solution and change the spc file or not and then Spike is called again for the new iteration. Figure 6.2 displays the system diagram of optimisation, exhibiting the software tools and the files used during the simulations. It is important to note that in Random Restart Hill Climbing this diagram starts from the beginning at every random restart whille in Simulated Annealing the shell programme is used only once at the beginning.

Figure 6.2: The files and Software tools used in Optimisation. The white squares represent software tools and the blue represent the files.


candl file
modify
Randomisation Shell program
call
Optimisation program in Python
call
read
modify
Spike
read
writes
input
write
spc file
csv file
Optimisation result csv file

6.2 Random Restart Hill Climbing

Hill Climbing (HC) is a heuristic optimisation method that begins the optimisation at a random point and starts climbing the hill by exploring the search space looking for better solutions. When reached a peak, the simulation will stop, concluding that the best solution is found. It is a local search method which only needs memory for the current run. This makes the process more efficient. But since the found peak might be a local maximum and not the global maximum, traditional HC is not the best method for optimisation. However, by adding the Random Restart(RRHC) to it, HC turns to be a more reliable method. In RRHC, the algorithm restarts randomly and starts over to a new search area. With this method, the possibility of finding global maximum increases.

Both HC and RRHC can only go up and reject the lower values; This might result in not finding the global maximum at all. The other issue with RRHC is that the algorithm cannot predict if there are no solutions so it will continue the optimisation with no end by changing the local maximum [173].

In Dicty model there is a set of four constant rates. The target function of RRHC is to change these four rates in order to find the best combination of them when the number of Dicty cells is as high as possible. Therefore, the condition for the solution to be accepted is to compare the maximum number of Dicty on the grid. if the new maximum number is more than the old one, the the new solution is accepted; otherwise it is rejected.

The final version of the code, calls a Shell program written by Professor Gilbert to define random locations for Dicty on the grid and this happens at each random restart. I removed one part of the acceptance condition of the maximum existing strictly at the centre of the grid and decreased the delta. Delta is the difference between the current rate constants and the new ones. The reason was, when delta was too high, the code could not find the best solutions in between the constants and most of the rates were rejected. Basically, the code would jump over the possible accepted solutions due to high delta. The code produces plots and an output file that shows the improvement of the code in each run.

6.2.1 Pseudocode

To increase the chance of getting the best number for each action rate, we decided to set the delta, not as a constant number but a random number in a range.

This rate is a random number between zero and 10, up to two decimal numbers. So now, there are two steps of randomised numbers. First, in the HC loop, the initial random numbers which replace in the mother file and the second is when a delta is randomly chosen and changes the current action rates. Algorithm 1 shows the pseudocode oof Random Restart Hill Climbing.

Algorithm 1 Random Restart Hill Climbing Algorithm

```
1: procedure RandomRestartHillClimbing
2:     for $x \leftarrow 1$ to STARTPOINT do
3:         $S \leftarrow Random\_Solution$                ▷ current solution
4:         $F \leftarrow Fitness(S)$                     ▷ current state's fitness
5:         for $i \leftarrow ITER$ do
6:             $S' \leftarrow SmallChange$               ▷ new solution
7:             $F' \leftarrow Fitness(S')$              ▷ new state's solution
8:             if F ' better F then then
9:                 $S \leftarrow S'$
10:            end if
11:        end for
12:    end for
13: end procedure
```

6.3 Simulated Annealing

Simulated Annealing (SA) is another optimisation method that is inspired by the physical annealing process, in which the temperature starts at a high point and decreases over the run. In SA, the initial temperature is set at a high amount. The algorithm looks for the maxima and accepts the values that are in the acceptance possibility distribution. When the temperature is higher, the acceptance possibility has a higher range and almost all of the values are accepted. As the temperature goes down, the values are narrowed down and only the best values are accepted. Eventually, among the best found valued, the simulation finds the global maximum.

Compared to HC, SA is a more accurate method as it tends to find the global maximum by accepting the worse solution with a probability and making the worst solutions' acceptance to be unlikely. The change in the accepted solutions is defined by "temperature". At high temperature, worse solutions than the last accepted soution are accepted but as the temperature decreases, the probability of this acceptance decreases. This method has more mathematical calculations which slows down the simulation. Also, the initial temperature which is set by the user, and the cooling rate must be calculated precisely. Otherwise, in the end, the final value might not be the best answer, since the initial temperature was not high enough or the cooling down was too fast. My colleague Yasoda Jayaweera has been assisting me as she had used this same method in her thesis. The current pseudocode is as below:

6.3.1 Pseudocode

The SA algorithms is based on an article by Stephen Swift *et al* [185] in which simulated annealing is used as an optimisation method. In this article they calculate that the best cooling is $c = 0.99994$. The initial temperature is set at 100. The Pseudocode for Simulated Annealing is shown in Algorithm 2.

Algorithm 2 Simulated Annealing Algorithm

```
1:  $Given : Number of ITERATIONS(ITER)$
2:  procedure SimulatedAnnealing
3:      $S \leftarrow Random\_Solution$                ▷ initial random rate constants
4:      $F \leftarrow Fitness(S)$                      ▷ maximum number of Dicty in the final time frame
5:      $\theta_0 = 100$                               ▷ $\theta_0 \leftarrow InitialTemperature$
6:      $c = 0.99994$                                  ▷ Cooling rate
7:      $t = \theta_0 * (c)^{ITER}$                    ▷ Final Temperature
8:      for $x \leftarrow 1$ to ITER do
9:          $S' \leftarrow New\_Solution$              ▷ new constants based on the previous solution
10:         $F' \leftarrow Fitness(S')$                ▷ maximum number of Dicty in the final time frame
11:         if F' better F then then
12:             $S \leftarrow S'$
13:         else if F better F' then then
14:             $e = 2.71828$
15:             $\Delta f = \frac{F-F'}{\theta}$
16:             $p = e^{-\Delta f}$
17:             $r = random(0, 1)$
18:             if p > r then then
19:                 $S \leftarrow S'$
20:             else
21:                 $pass$
22:             end if
23:         end if
24:         $\theta_i = c * \theta_{i-1}$
25:     end for
26: end procedure
```

6.4 Discussion

There are many methods for optimisation, and the method is usually chosen based on simplicity or functionality of the algorithm. In this study two of the most popular Methods are chosen to optimise the Model “Dicty”: Random Restart Hill Climbing and Simulated Annealing. The reason to use two different methods is to compare the results and the practicality of these two methods. Hill Climbing is simple but may not be accurate since it may provide the local maximum instead of the global one while Simulated Annealing is more accurate and searching as many answers as possible, it might provide a better answer but is more complex so it takes memory and time. The target function for optimisation is to find the best possible combination of rates to achieve the maximum number of Dcity cells on the grid in one location mimicking the creation of fruiting body in nature. In the next chapter, Chapter 7, the output data both from the models and optimisation simulations are provided.

Chapter 7

Data Analysis and Results

There are different ways of analysing the output data from the models. For simulation of complex models in this thesis, Spike has been used which was introduced in Chapter 3 Section 3.1.3. Also, to analyse the output data from Spike and Snoopy as well as the result of the optimisations, machine learning and big data analysis methods have been applied using R with a combination of Python when required. In the Python codes of optimisation, the code selects the most important parameters and saves them in a separate dataset which creates a sufficient data frame to be analysed, with all the important information included in it. The optimisation result is compared to the "expected" behaviour based on the biological literature. No biological data has been provided for this study for mathematical and computational verification, due to the unavailability of the experimental data, but based on the previous biological studies, it is possible to achieve the expected results from the models created in this thesis.

In this chapter, after a brief review of the timing of the simulations in Section 7.1, the output data of interesting models are provided, to prove that they work as it is expected of them. In the first part, Section 7.2.1 the output data from the Diffusion Movement is analysed. In Section 7.2.2 the output data of chemotaxis is visualised and compared with the diffusion. In Section 7.2.3 the Semi-permeable membrane obstacle model is visualised on a heatmap to demonstrate the functionality of the model. Finally, in Section 7.2.4, one of the most interesting models is analysed which is the system in which Glu is on one side of the grid and it is shown how the bacteria move on the grid to reach it. To avoid repetition of the result of the two models "Quorum Sensing and Response" and "Biofilm Formation" , the analysis of these models are not included in this thesis. The analysis of these models can be found in the study done by our group, published in 2019 in BMC Bioinformatics Journal, "Spatial quorum sensing modelling using coloured hybrid Petri nets and simulative model checking" [77].

On the second part of this chapter, 7.3, the output of optimisation simulations, Random Restart Hill Climbing and Simulated Annealing are analysed in order to find the best combination of constant rates that is found in 50000 runs.

7.1 Timing the Simulations

Different machines are used for the simulations in this study. Unfortunately there was no access to supercomputers for this thesis at the time of simulations or writing. To check how long each simulation takes in each machine, the time of each simulation is measured. The simulations were repeated on the most updated machines to provide the most updated information possible. Table 7.1 provides the time it takes for each model to be unfolded and simulated in Spike. To achieve these information the command *time* was used. The grid size in all of these models is 7x7. The machine runs a macOS Monterey (Version 12.1), its CPU 1.7 GHz Quad-Core Intel ® Core™ i7 and Memory of 16 GB. As it can be seen in Table 7.1, as the models get more complex, the time it takes to simulate increases. Looking at different levels of Duplication, it in obvious that these models and the systems that include them are taking more time as the models develop. This is probably because of the weighed arcs that exist in these models. There are two components from Systems that are not included in this table: "Biofilm Formation" and "Quorum Sensing and Response". These two models were simulated and analysed in BMC Bioinformatics Journal published in 2019 [77].

The other simulation is the optimisations. Two methods of optimisation were applied in this thesis: Simulated Annealing and Random Restart Hill Climbing. One of the reasons of choosing two methods, is to compare them. Computationally, Random Restart Hill Climbing takes considerably less time and memory compared to Simulated Annealing. Random Restart, starts the simulation once it reaches the provided number of iterations, in this case 1000. Simulated Annealing goes through the same simulation by decreasing the temperature. The reasons for this difference can be as follows:

- Hill Climbing does not require a memory to save the history of simulation while Simulated Annealing goes through all the iterations, by decreasing the temperature. This means this method consumes memory for the previous iterations and increases the time of the simulation.

- Simulated Annealing includes some simple mathematical calculations, such as the current temperature and the minimum temperature. These calculations happen in every iteration that can slow down the process. Hill Climbing does not consist these calculations.
- The output of the optimisation is saved into a CSV file. Each 1000 Hill Climbing has its separate file. This means the created and opened file is closed at the end of each 1000 iterations. On the other hand, the whole 50000 iterations of Simulated Annealing are written in one huge data frame in csv. This file is created and opened at the start of the simulation and is open for the whole process. This causes an increase in memory consumption and thus longer time for this method of optimisation.

Table 7.2 provides the time for each optimisation. These optimisations were done on a macOS machine running macOS version 10.3.1, its CPU 2.5 GHz Quad-Core Intel ® Core™ i7 and Memory of 16 GB. It is important to remember that the provided times are only estimated. In case of Random Restart, there are the pauses between the random restarts that are not calculated. It is important to note that the relationship between the time and the number of iterations is not linear. It means if it takes 15 hours for 5000 simulations of Simulated Annealing, it does not mean that it will definitely take 30 hours for 10000. Even though the difference between the estimated and actual time may not be significant, it is still there.

7.2 Models

Movement and Chemotaxis are two of the most basic and yet most interesting simple components in this study. In this part these two simple components as well as one Lvl1 model, Semi-permeable Obstacle and Transmitter and one Lvl2 model, Combined Strip Glu, DCD and TR are visualised and analysed.

7.2.1 Diffusion / Movement

When the models are small and simple, as it is in the model of Diffusion / Movement models, the built-in simulation of Snoopy is sufficient enough for understanding the behaviour of the model and produce numeric data. The numerical data could be exported from Snoopy as an image or a time-series data frame in CSV format. In this example, the plot of the simulation as an image is exported to provide an idea of how the plots in Snoopy work. In this model the grid

Table 7.1: The time it takes to simulate each model component from the library using Spike.

Model	Time
Diffusion	0.26 s
Lvl0-Division	0.58 s
Lvl0-Death	0.18 s
Lvl0-Transmitter	0.25 s
Lvl0-Receiver	0.31 s
Lvl1-Transmitter	0.31 s
Lvl1-Receiver	0.42 s
Chemotaxis	0.45 s
Lvl1-Death	0.30 s
Lvl1-Division	3.20 s
Limited Glu Strip	0.34 s
Semi-permeable Obstacle Transmitter	0.39 s
Transmitter and Receiver	0.54 s
Duplication, Chemotaxis and Death	1.84 s
Combined Strip Glu, DCD and TR	39.99 s
Dicty	0.76 s

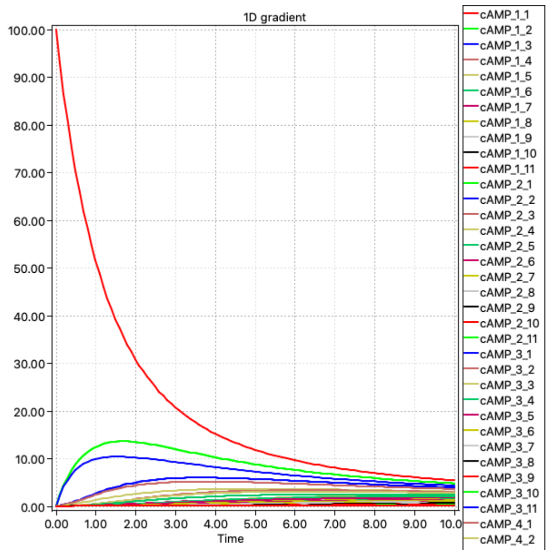
Table 7.2: Etimated time it takes for Optimisation to be done.

Method	Number of Runs	Time
Random Restart Hill Climbing	5000 Random Restarts	12 hours
	(5 RR and 1000 Iterations each)	
Simulated Annealing	5000 Iterations	15 hours

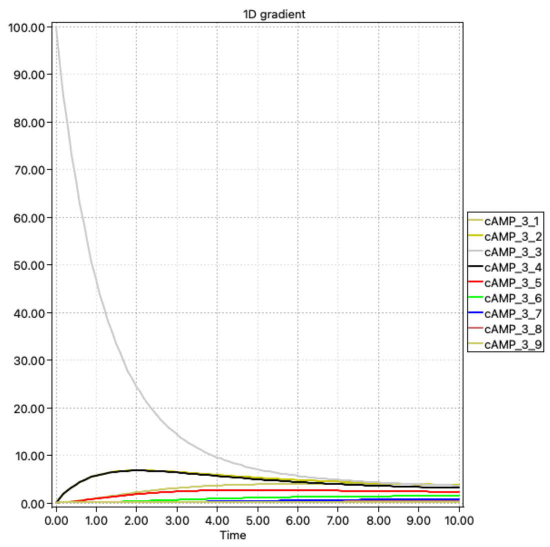
size (D1) is 11 and the initial location of the place is at (3,3) carrying 100 tokens in one location. In this model the diffusion rate is defined with the parameter k, which has different rates and could be chosen upon simulation in “Model Configuration” part of the simulation window.

In this continuous simulation the constant rate for the diffusion is set at 0.1. As it can be seen in Figure 7.1a, since the number of places is too many, the plot cannot exhibit all of the places in one image and it is not very informative either. However, to get an understanding of the model better, a closer look at the plot of all the location where x is 3 can be helpful, which can be seen in Figure 7.1b. What is obvious in this plot though is that the number of tokens in (3,3) decreases dramatically and increases in other places demonstrating the distribution of the tokens on the grid. Also, the locations closer to (3,3), such as (3,4) and (3,2) increase to a higher number faster compared to the further locations. This is due to the Neighbour2D8 function that has been defined in the model and allows the tokens to move in 8 different direction, but it does not allow them to jump one location. This means the closest locations to the initial placement, receive the most tokens, compared to the ones far away. Since the model is continuous non-integers are accepted as the number of tokens.

Figure 7.1: The plot of Diffusion / Movement model exported from Snoopy.



(a) Dicty Locations in simulation 4147 at the beginning of the simulation



(b) Location and number of Dicty in the last time frame of simulation 4147

7.2.2 Chemotaxis

Chemotaxis is a Level-1 model component which indicates the tokens should move towards a chemical trigger, which in this case it is the food source, Glu.The best way to analyse the data for a model that its focus is on movement, is heatmaps. These heatmaps in this section are created by Python, but a similar work can be done with R. For this model, Python codes were written by my colleague Doctor Sarath Dantu which created heatmaps for each time frame and from the heatmaps it created a mp4 file, placing the images in order of time and creating a movie that demonstrates the improvement of the model over time. This method is used for other models as well when the location or movement is an important aspect of the model. Figure 7.2 is the shots from 4 different time frames that show tokens (mimicking bacteria) at (1,6) move towards the food placed at (11,6).

In this model there are three different rates, Glu Diffusion, Chemotaxis and Sink,that could be altered as required. As it can be seen in Fugure 7.2, there is only one token placed at (1,6). This one token, can be found distributed at the Wall of Glu, with the most concentration at the centre of the Wall, where naturally the concentration of Glu is also the maximum. The Glu creates a gradient, which is the same behaviour in all the models that include Glu. This gradient is shown in Figure 7.4a.

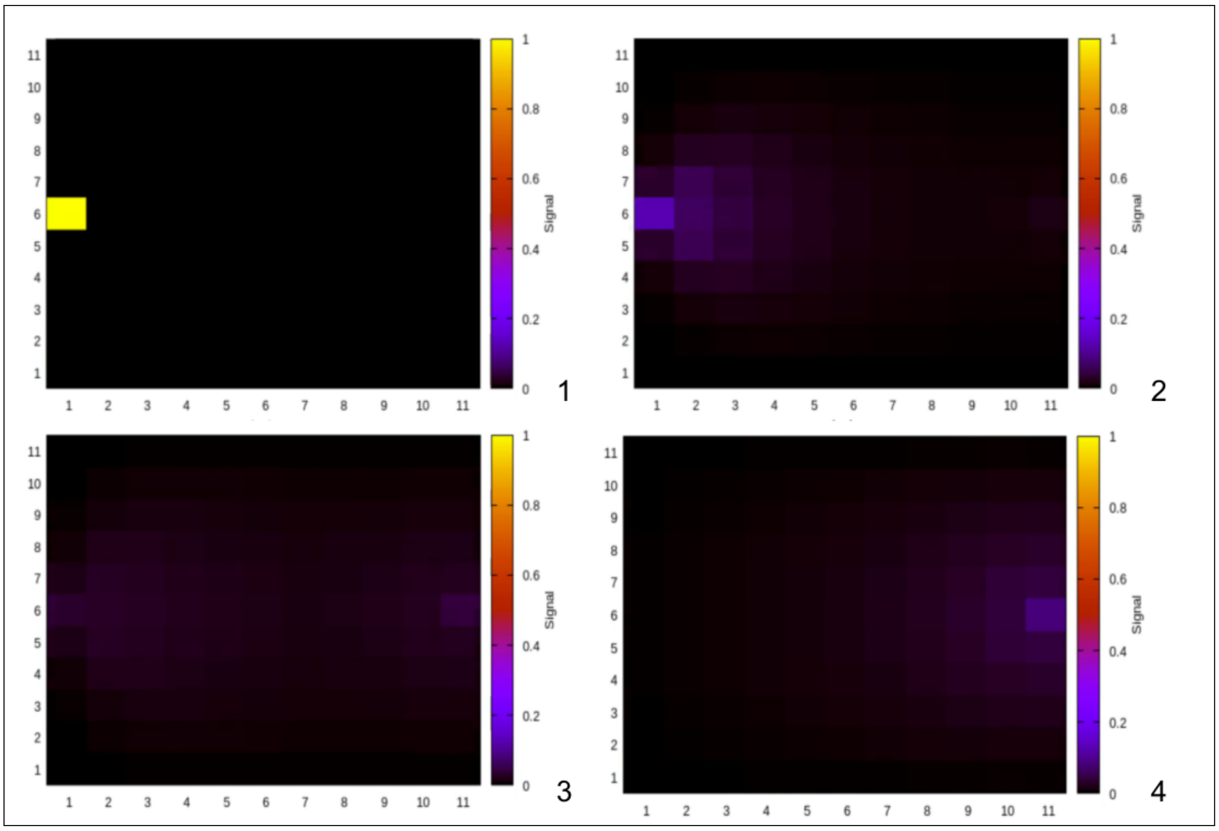


Figure 7.2: Heat maps made by python codes put together to show the progress of the model over time. Figure 1 is at the beginning and Figure 4 is the end. The grid size of this model is 11 x 11

7.2.3 Semi-permeable Obstacle and Transmitter

Designing obstacles on the grid can push the boundaries of modelling a little further. As challenging as it is to model the obstacles, these models are one of the most interesting components in the library. The Obstacle can be modelled in one of the three ways:

1. A semi-Permeable membrane consisting of transitions with a lower diffusion rate compared

to the other locations. This results in a slower diffusion in the defined area.

2. A membrane that only allows the tokens to leave its area but does not allow anything to enter.

3. A block that does not allow anything to cross.

Here, the first option is chosen for analysis. This obstacle is a semi-permeable membrane that has a slow diffusion. So, when the tokens enter in this area, their movement is slowed down. The obstacle is placed at the top of the grid in two layers. The movement of the tokens is random but gradually they will cover the whole grid. The expected result is to see a significantly slower movement of the tokens in the area of the obstacle at the top of the grid. This was achieved perfectly and is shown in Figure 7.3.

The grid size of this model is 7x7 and the initial number of tokens is set at 10 on (2,4). As it can be seen in Figure 7.3, the difference in speed of movement or diffusion at the top of the grid is noticeably slower than the rest of the grid. Again, this is a continuous model, allowing non-integer numbers. To see the same behaviour in a stochastic model, the initial number of the tokens should be as great as the grid size or more.

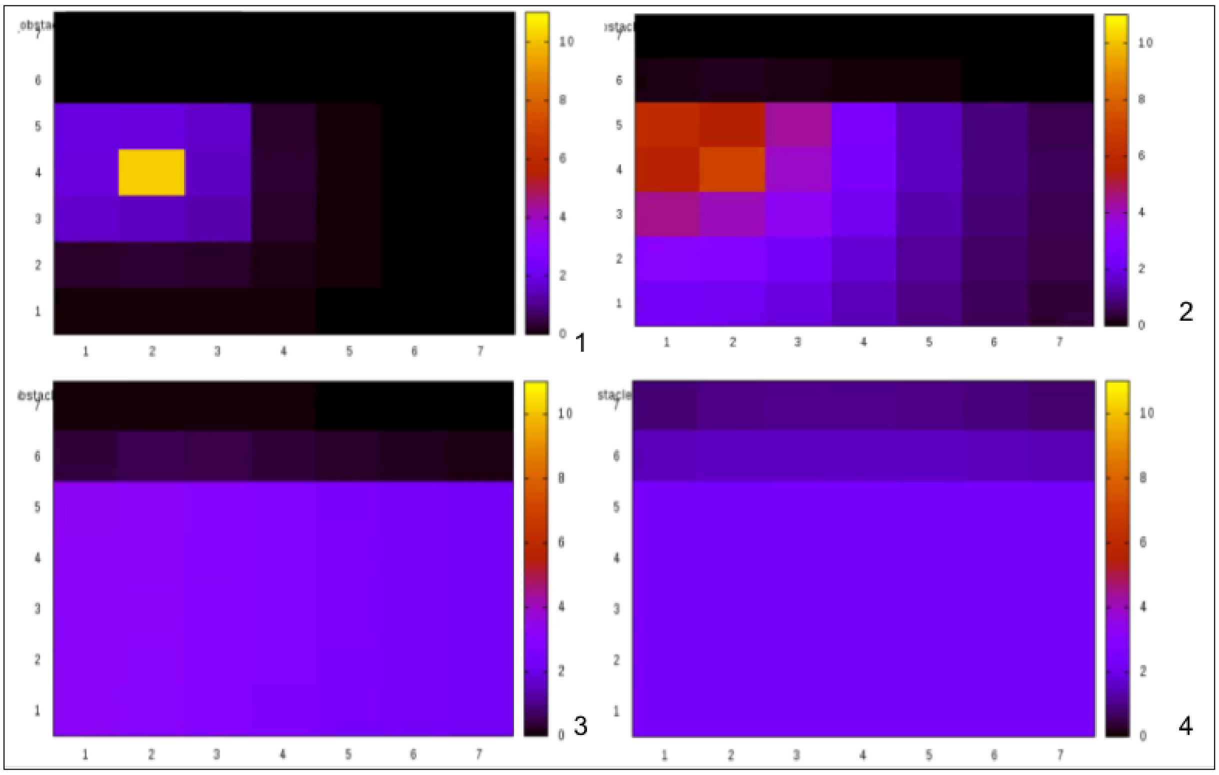


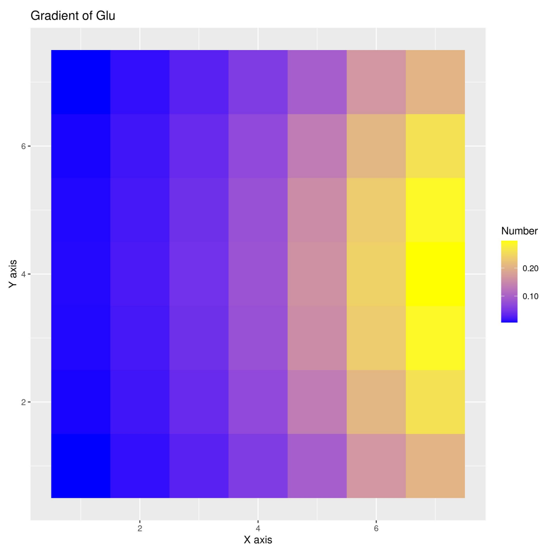
Figure 7.3: Heat maps made by python codes for the Semi-Permeable obstacle. The obstacles are placed on the top in 2 layers. There are no sinks in this model. It is visible that the speed of the diffusion is lower in the area of Obstacle.

7.2.4 Combined Strip Glu, DCD and TR

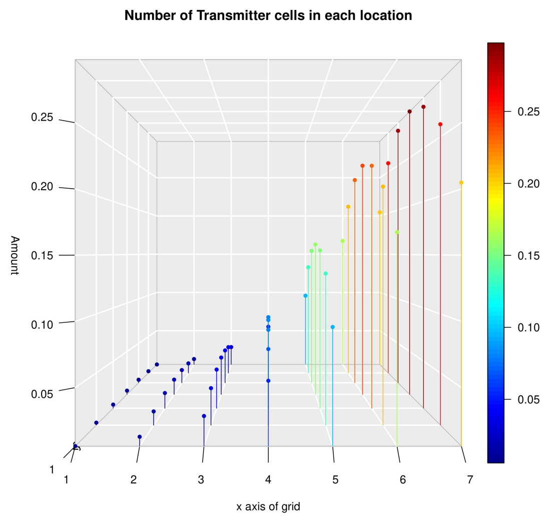
This model is one of the most complex models in the collection as it contains movement, death, duplication, Transmitter and Receiver; basically all of the Lvl0 components. To make sure that the model is behaving as expected, different methods of visualisation and analysis is done. For example, to check if the Glu creates a gradient, a heatmap is used that can be seen in Figure7.4 and to check if the tokens are attracted to Glu, a 3D plot is created, Figure 7.4a, that shows the number of bacteria in the final time frame in each location.

The Gradient of Glu can be seen in Figure, 7.4a. This heatmap shows how the number of Glu molecules is higher when closer to the source and less further away with a gradient that decreases gradually with the distance from the source. The 3D plot in Figure 7.4b, indicates the location and the number of transmitters (BactTR) on the grid using axis z to show the number of bacteria. The behaviour of BactTR and Receivers (BactR) is the same. They both move towards the food source and accumulate around it.

Figure 7.4: The behaviour of Combined Strip Glu, DCD and TR demonstrating the gradient of Glu, the movement of Transmitter and gradient of biofilm as a result of the communication



(a) The Gradient of Glu is visible in this heatpam. Close to the source, there are more Glu compared to the further locations.



(b) Location and number of Transmitter at the end of the simulation. As expected, the cells gather around the trigger of the chemotaxis and remain there.

7.3 Optimisation Results

Hill Climbing is an optimisation method that looks for solutions, checks for their fitnesses and accepts the better solution. The accepted solution is set as the default solution and is compared with the next one. If it is not better, the solution is rejected and the previous fitness remains unchanged. The algorithm for this optimisation is provided in Chapter 6.

7.4 Principle Data Analysis

Principle Data Analysis (PCA) is a data analysis technique that is used for data visualisation and noise reduction for large datasets. PCA is a suitable method for analysing big data with multiple dimension variables as it creates a smaller dataset based on principle components which can be used for understanding the correlation between different variables in the data. This method is unsupervised, meaning it is used for unlabelled data in hope of finding patterns, clusters and correlations in the data set.

7.4.1 PCA on Random Restart Hill Climbing

Before the analysis, the accepted populations of Dicty are filtered, since these are the data that we are interested in. Out of 50000 steps 383 observations are accepted. The reason of this low acceptance rate is that the new clustered population in one locations mostly lower than the last. For the new solution to be accepted the new clustered population should be higher than the last. In the most extreme scenarios the when the population is around 0.3, it still reaches to 0.7 which is the maximum number of Dicty achieved in these simulations.

In these simulations, the initial number of Dicty is 61, around 27% of the size of the grid. If there is Dicty in all 225 locations of the grid, then there will be 0.2 Dicty in each location. Hence, a population of 0.7 is more than 3 times the number that is on a equally distributed population of Dicty on the grid.

After filtering the data, PCA is applied to the accepted rates. Since this method can only be used on numeric data, PCA is only used on the “k_diff”, “k_mvmnt”, “k_pro” and “k_sink” columns, which define respectively the rates for diffusion, movement, sink and production of cAMP, in the output data from both Random Restart Hill Climbing and Simulated Annealing. With this method we hope to find the correlation between the rates. Table 7.3 shows the result of PCA on these four variables.

Table 7.3: The result of PCA analysis of RRHC.

	PC1	PC2	PC3	PC4
k_diff	-0.1433535	0.9282277	-0.34306807	-0.01214249
k_mvmnt	-0.5371974	0.1997441	0.75351264	0.32208692
k_sink	-0.6170094	-0.1329701	-0.07462533	-0.77204233
k_pro	-0.5569274	-0.2842791	-0.55583636	0.54778027

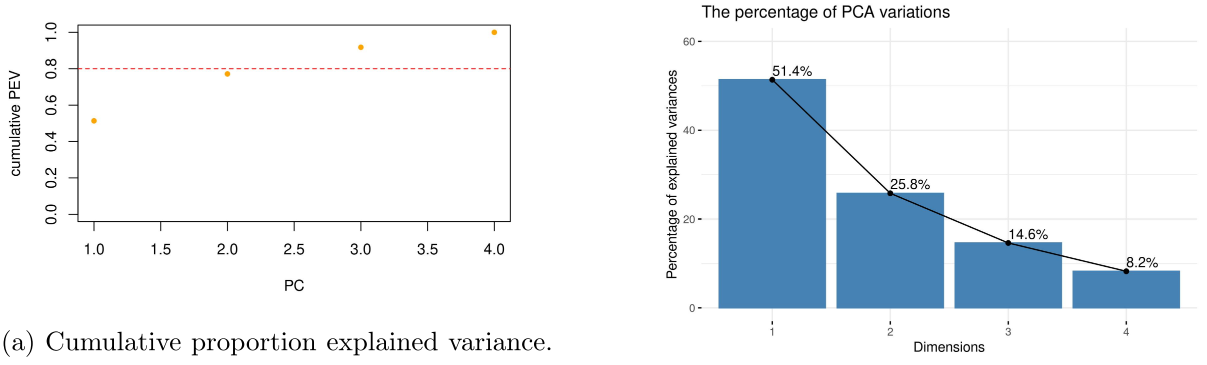
Table 7.4 presents the importance of components. Based on this table, the cumulative proportion shows that by PC3, 91% of the total variability is explained. Figure 7.5a which is the cumulative proportion of explained variance (PEV) and Figure 7.5b is a histogram of each PC that shows 91.8% of the variation is explained by the first two PCs. Based on the observations in the tables and figures, it can be concluded that PC1, PC2 and PC3 are efficient for the rest of

the analysis

Table 7.4: The importance of components table of PCA for Random Restart Hill Climbing.

	PC1	PC2	PC3	PC4
Standard deviation	1.4334	1.0158	0.7644	0.57384
Proportion of Variance	0.5137	0.2579	0.1461	0.08232
Cumulative Proportion	0.5137	0.7716	0.9177	1.00000

Figure 7.5: The result of PCA of RRHC shows that PC1, PC2 and PC3 are necessary for the analysis.

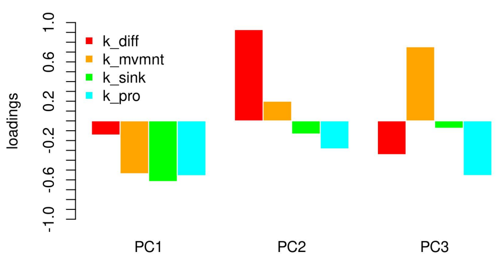


(a) Cumulative proportion explained variance. By PC3 more than 80% of the variability is explained.

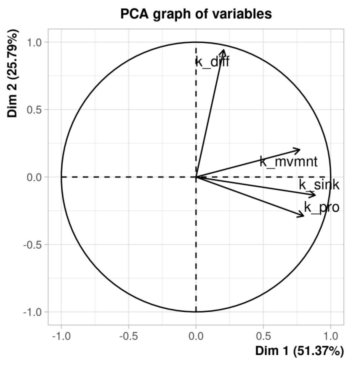
(b) Bar plot of PCs. Based on this bar plot PC1, PC2 and PC3 are sufficient for the data analysis

Since it seems that the first three PCs provide a satisfactory amount of the required variance, these three PCs are used to visualise the correlation between the different variables. The plots 7.6a and 7.6b show that the three variables of k_sink and k_pro are closely correlated while k_diff is completely different.

Figure 7.6: The correlation between PC1, PC2 and PC3 in RRHC in both plots shows that k_diff is the most different variable in the data set.



(a) Bar plot of correlation between the four variables in three PCs.



(b) Correlation circle showing the correlation between the four variables.

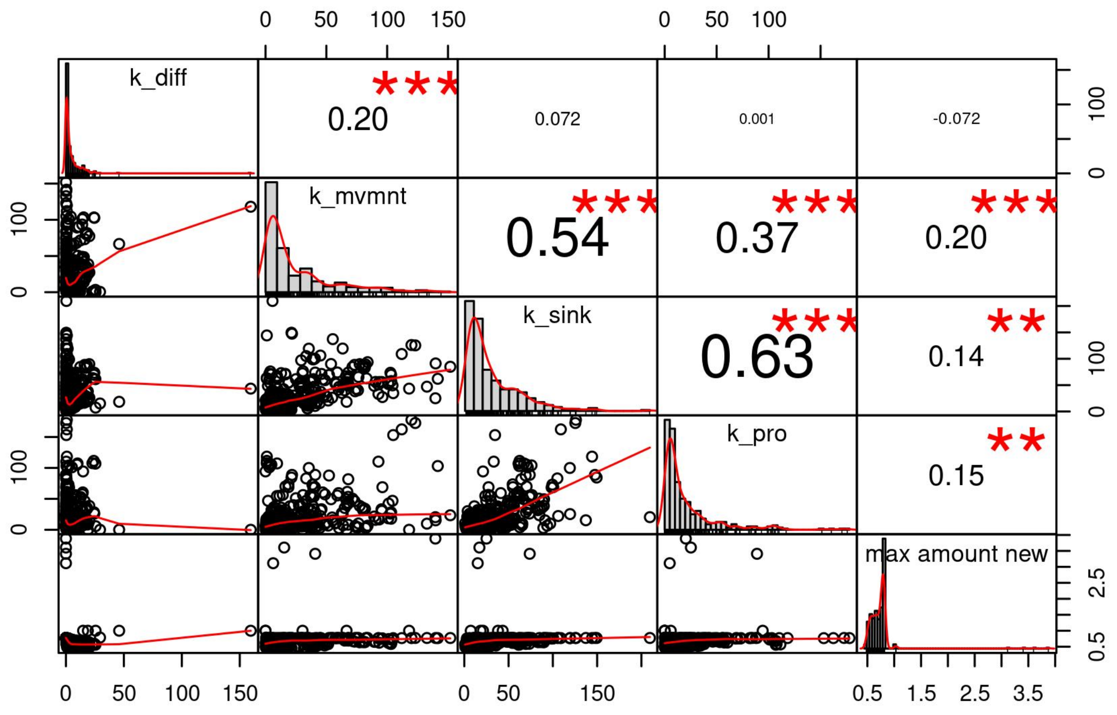
To check the effect of k_diff, the rate of diffusion, on the population density a correlation analysis was run on the original data. This matrix, shown in Figure 7.7, shows that the effect of diffusion on the population density is completely different to the other rates. Looking at this plot, however, shows there is no linear correlation between k_diff and 'max amount new" which is the population of Dicty. But there is a correlation between the diffusion and movement which makes sense. If there is no diffusion of cAMP on the grid, the Dicty cells will not move. If the diffusion is high, then the movement will be faster as well so that the maximum population density is achieved faster. To see the relationship between the diffusion and population density, other visualisation and analysis is done which is can be found in Section, 7.5

7.4.2 PCA on Simulated Annealing

Two different method of optimisation were used in this study. In the previous section we explored the PCA result of Random Restart Hill Climbing. In this section the same method is used to analyse the data from Simulated Annealing.

Simulated Annealing is an optimisation method that has the ability of accepting worse solutions, if these solutions have a high Acceptance Possibility (AP). The possibility of a number being accepted with high AP is higher when the temperature is higher. Therefore, mostly it happens in the first iterations. As the temperature decreases, the number of accepted solutions based on high AP also lessens. The calculations for AP is provided in Algorith 2. There are three groups in the dataset: rejected, AP high and accepted. The PCA analysis is done on the

Figure 7.7: Correlation matrix between rates and the population density of RRHC. This shows the difference in the effect of diffusion on dicty population density compared to the other rates.



accepted solutions.

Table 7.5 presents the result of PCA on the output data from SA and Table 7.6 provides the information about the importance of each PCA analysis. This table explains the proportion of explained variance in the dataset. As it can be seen PC1 explains 45% of the variability of the output data, PC2 explains 21% and PC3 20%. With a look at the cumulative proportion, it can be seen that by PC3, the total variability reaches 87% which is another proof of the fact that PC4 is the only principle component that will not be used in the data analysis.

Table 7.5: The result of PCA analysis of SA.

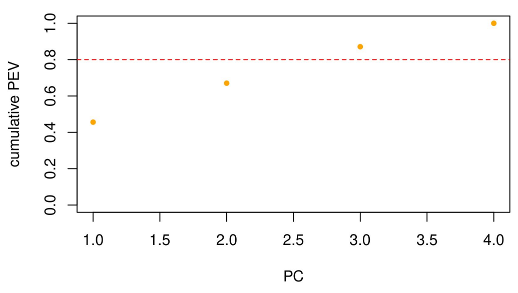
	PC1	PC2	PC3	PC4
k_diff	0.5975343	-0.1739437	0.1534465	-0.7675614
k_mvmnt	-0.4374830	-0.3024174	0.8404351	-0.1040251
k_sink	0.5101182	0.6414719	-0.1044463	-0.5633686
k_pro	-0.4374252	-0.6832285	-0.5091307	-0.2874788

Table 7.6: The importance of components table of PCA for Simulated Annealing.

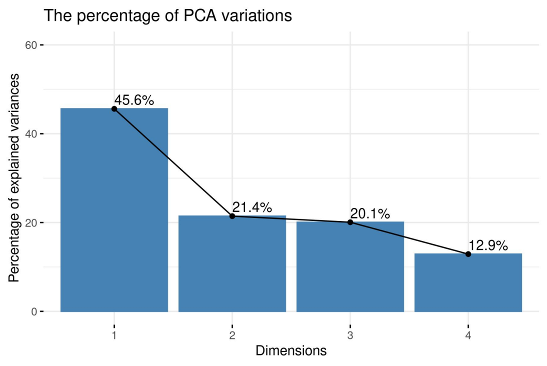
	PC1	PC2	PC3	PC4
Standard deviation	1.3506	0.9260	0.8958	0.7181
Proportion of Variance	0.4561	0.2144	0.2006	0.1289
Cumulative Proportion	0.4561	0.6704	0.8711	1.0000

For a better understanding, it is possible to visualise the result of the PCA which can be seen in Figure 7.8a and Figure 7.8b. These two figures show the plot of cumulative proportion to confirm the fact that the first three PCs are enough for the data analysis. As it can be seen in Figure 7.8b the cumulative proportion reaches more than 80% by the third PC.

Figure 7.8: The result of PCA of SA shows that only PC1, PC2 and PC3 are necessary for the analysis.



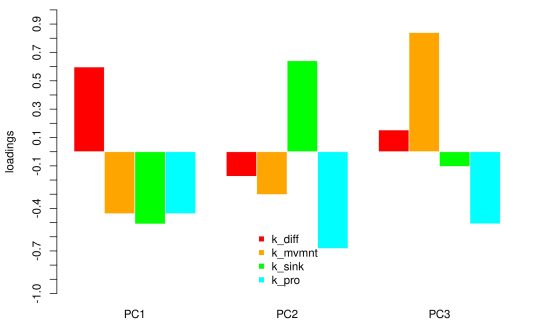
(a) Cumulative proportion explained variance. The total variability of the first three PCs is 87%.



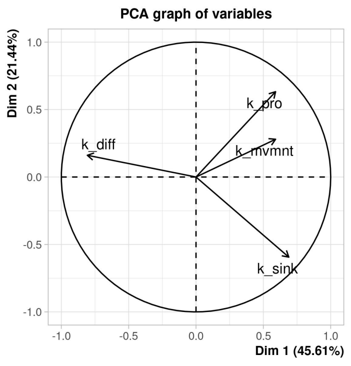
(b) Bar plot of PCs. Based on this bar plot PC1, PC2 and PC3 are sufficient for the data analysis.

To understand the correlation between the different variables, a correlation plot is provided in Figures 7.9a and 7.9b. As it can be seen in these plots, the behaviour of k_diff is completely different from the others. This is the same outcome from the analysis of RRHC. It can also be seen in both these figures that there is a similarity in behaviour between k_pro and k_mvmnt. This demonstrates the relationship between these two variables. When there is no production of cAMP, the movement does not happen. This behaviour is set as one of the expected behaviours of the model.

Figure 7.9: The correlation between PC1, PC2 and PC3 in SA in both plots shows that k_diff is the most different variable in the data set.



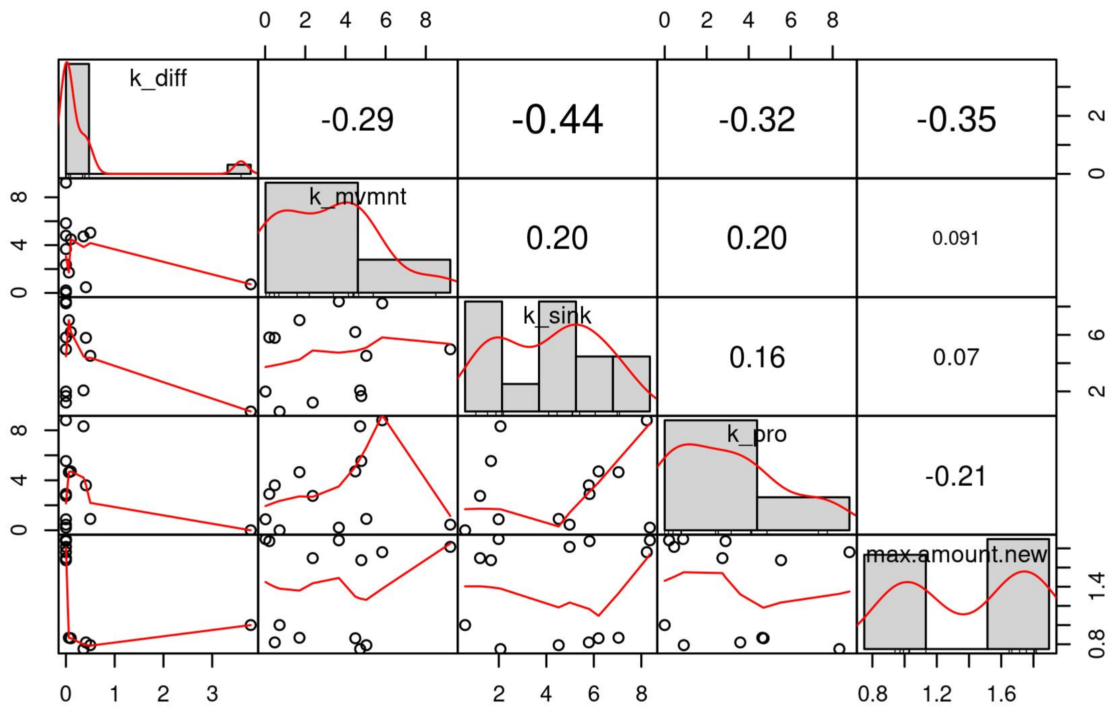
(a) Bar plot of correlation between the four variables in three PCs in SA



(b) Correlation circle showing the correlation between the four variables in SA which indicates how k_diff is completely different from the other rates.

Another way to study the correlation between the rates, is to create a correlation matrix between the rates and the population density which is shown in Figure 7.10. Considering that this analysis is only done on the 13 solely accepted iterations of simulated annealing iterations, this matrix provides a better view for the correlation that the matrix of RRHC. Based on this matrix, there is no linear correlation between the four variables and the population density of Dicty on the grid. This matrix does not show any linear relationship between the four variables and cluster of Dicty cells on the gird. This might be due to the few data points that have been analysed, compared to a much bigger accepted dataset for RRHC. Table 7.9 provides the information for the accepted Dicty in this optimisation method.

Figure 7.10: Correlation matrix between rates and the population density of SA. like RRHC, this matrix shows that diffusion has a different effect on dicty population density compared to the other rates.



7.5 Inspecting the Behaviour of Dicty

The models that are used for optimisation are continuous and the number of tokens ban be non-integer. Therefore, here “population” is used interchangeably with “population density”. In this section different visualisation methods are used to present tdifferent aspects of the model for a better understanding of its behaviour.

7.5.1 Visualisation of Random Restart Hill Climbing

RRHC contains 50 random restarts and each random restarts contains 1000 iterations. The aim of the optimisation is to find the most effective constant rate on the population of Dicty on the grid as well as finding the optimal combination of these rates to achieve the expected result. The reason for choosing 50000 runs of Hill Climbing was the time limit. In Further studies, this number can increase in order to find more and solutions. However, the observation of the output result of RRHC shows that the current best solutions might be the global maxima regardless of the number of random restarts. To avoid overwriting the data, each random restart has a tag made of random numbers.

Among the 50000 results, there are some interesting combinations that stand out. The maximum Dicty on the grid is 0.7865301, when none of the rates are 0. However, the population can go as high as 3.8861813 when there is no diffusion for cAMP. Table 7.7 summarises the interesting findings from RRHC:

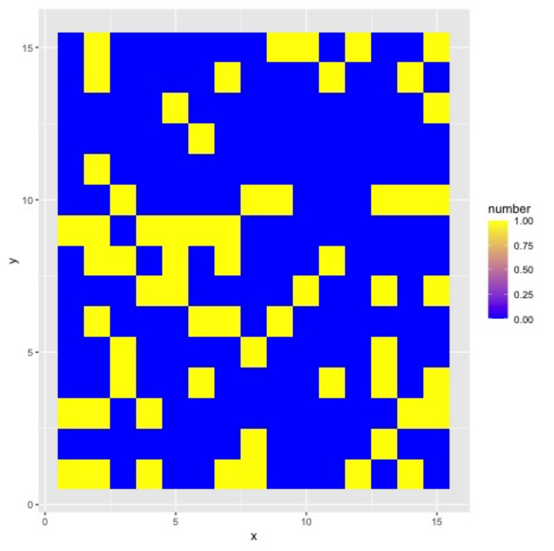
Table 7.7: Table of outstanding rates and results from 50k RRHC

Diffusion	Movement	Sink	cAMP Production	Population Density	Tag Number
0.00	139.64	25.08	20.33	3.8861813	4147
0.00	15.33	17.09	25.24	3.6061651	2387
0.00	40.83	73.87	88.85	3.4099376	3644
0.00	6.28	15.19	4.84	3.1146926	1687
159.46	118.02	43.35	0.00	1.0000000	2505
45.93	66.62	18.34	0.00	1.0000000	7572
18.92	30.45	38.07	0.00	1.0000000	3644
25.59	0.00	7.32	5.85	1.0000000	4090
14.79	0.00	21.14	94.34	1.0000000	365
29.41	0.50	15.21	0.04	0.7865301	3673
0.09	21.77	147.55	88.74	0.7739143	9503
0.11	55.29	137.23	15.38	0.7737875	4605
0.09	83.91	93.58	6.65	0.7736439	6089
0.12	45.11	106.07	22.30	0.7735341	9935

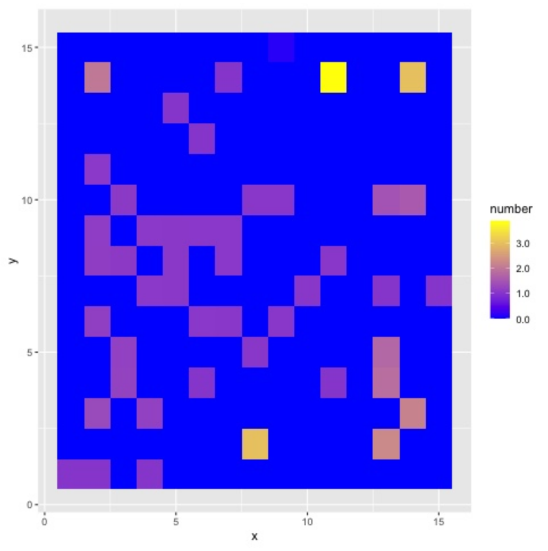
As it can be seen in Table 7.7, simulations 2505, 7572 and 3644 have no production of cAMP, hence the Dicty cells do not move at all. Once again this is an indication that the design of the model is correct as the Dicty cells are not supposed to move when cAMP is not present on the grid. The other simulations, 4090 and 365 have no movement, which provides the same result as when the rate of movement is zero.

When diffusion of cAMP is zero, in four case of simulation numbers 4147, 2387, 3644 and 1687, The population density of Dicty can go as high as 3.88. This indicates that the model is designed correctly: when there is no diffusion of the cAMP, the Dicty cells produce cAMP in their own locations. Then, Dicty cells move towards the locations with the highest concentration of cAMP. In model 4147 the first maximum location is (11,4) and in the other three with the population of higher than 3 and less than 3.88 is at (8,2). Figures 7.11a and 7.11b exhibit a heatmap of simulation 4147 in which the diffusion of cAMP is zero but the number of Dicty cells on the grid is the maximum compared to the other simulations.

Figure 7.11: Heatmap of simulation 4147, where the number of Dicty is 3.88 but the diffusion of cAMP is zero



(a) Dicty Locations in simulation 4147 at the beginning of the simulation



(b) Location and number of Dicty in the last time frame of simulation 4147

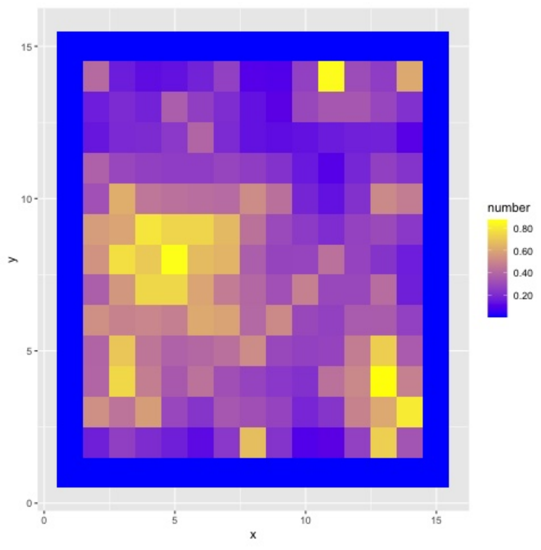
Since the number of Dicty is the highest in this simulation, a further experiment was done on 4147, by setting different numbers for diffusion. For this, scanning feature of Spike was used and four different numbers were set for diffusion of cAMP. The four numbers are 1, 0.1, 0.001, 0.0001. While the rate for diffusion changed, the other three rates remained the same as the simulation 4147 in order to observe the effect of diffusion rate on the model's behaviour.

The scanning showed that the diffusion 1 and 0.1 were too fast and the Dicty would reach maximum in the centre quickly. But 0.001 and 0.0001 showed more interesting results. The simulations were run for 5000 runs with 1000 intervals, to ensure that it is a long enough run to capture any possible patterns.

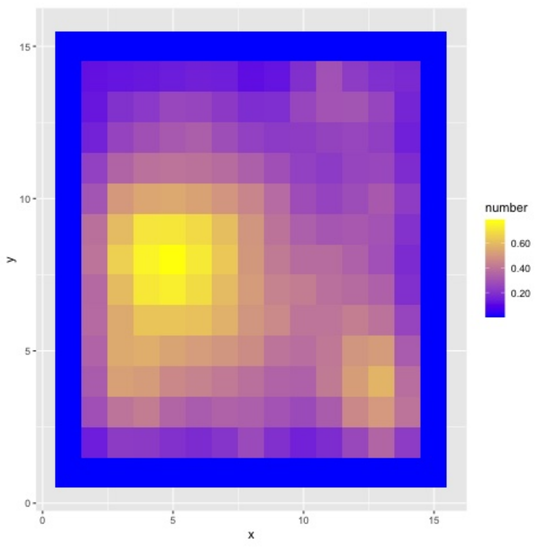
Figure 7.12a shows when the diffusion is set at 0.0001 at the time frame of 500. Three clumps of Dicty can be detected on the grid and the maximum number of the Dicty is 0.8761380. The same simulation, at the time frame 1001 is shown in Figure 7.12b. The three clusters can yet be seen but the maximum number of Dicty is 0.7904900, which is lower than before as the population is distributing on the grid slowly while cAMP is diffusing. This concludes that even though the other rates are greater than diffusion, the smallest amount of diffusion changes the behaviour of the model greatly.

On the other hand, by increasing the rate of diffusion to 0.001, the movement of Dicty is noticeably faster that there is hardly any difference between the time frame 500 and 1001 (See Figures 7.13c and 7.13d) even though the rate for the movement is the same. But looking at the time frame 100, it is noticeable that the behaviour of Dicty is exactly the same, only faster. Figure 7.13a presents the Dicty locations on the grid with three clumps at time frame 100 and Figure 7.13b shows Dicty on the grid beginning to create one clump in one corner at time frame 250. As the cAMP diffuse on the grid, the Dicty cells move less and are distributed almost everywhere on the grid, and the clump of Dicty population move towards the centre of the grid.

Figure 7.12: Heatmaps of simulation 4147 while the diffusion rate changes from 1 to 0.0001. Note that the maximum number on the bar is different, even though their colours are the same.

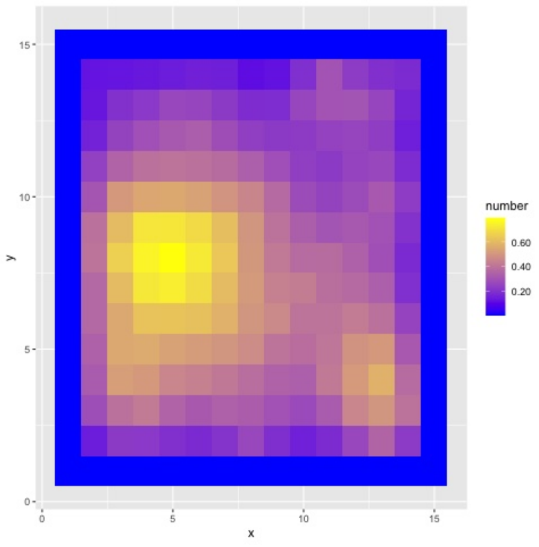


(a) Diffusion = 0.0001, time = 500

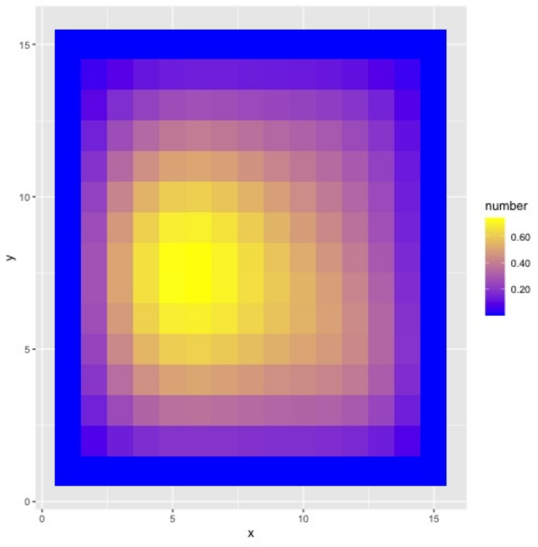


(b) Diffusion = 0.0001, time = 1001

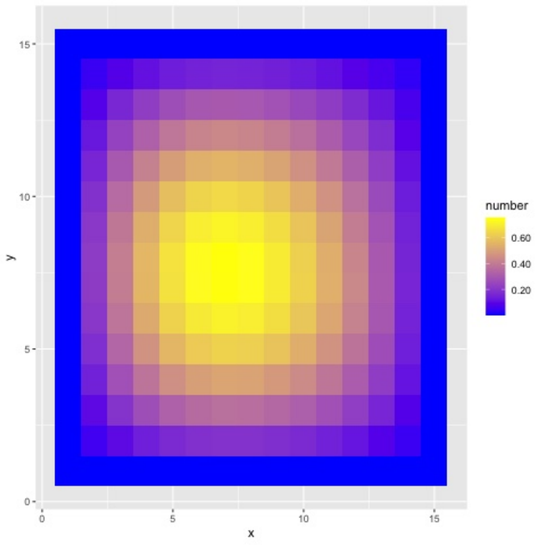
Figure 7.13: Heatmaps of simulation 4147 while the diffusion rate changes from 1 to 0.001.



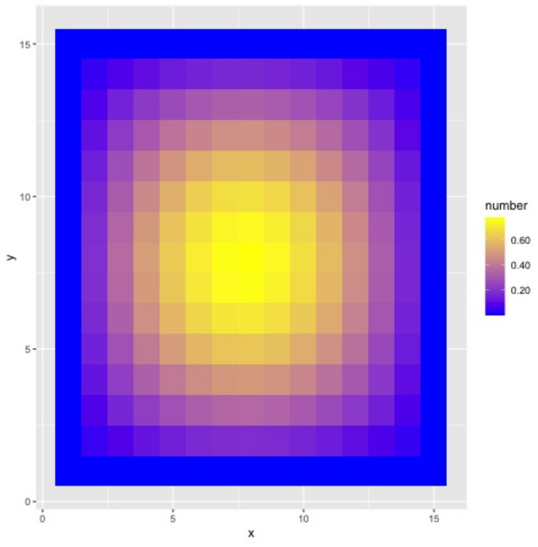
(a) Diffusion = 0.001, time = 100



(b) Diffusion = 0.001, time = 250



(c) Diffusion = 0.001, time = 500



(d) Diffusion = 0.001, time = 1001

The other interesting result of the Hill climbing is simulation 3673. In this run, the maximum number of Dicty cells on the grid is 0.7865301 on location (15,15) while none of the action rates are zero. To observe the behaviour of Dicty with these rates, the simulation was run for longer. Interestingly, in the longer run, the maximum clump of the Dicty changes its location between (1,15) and (15,15). Up to the time frame 32 the maximum number of Dicty exists on (15,15) and after that the maximum switches to (1,1) and eventually changes to other locations, while the difference between the number of Dicty is not much between two neighbouring locations. This is because of the diffusion of cAMP on the grid. As time passes the clumps of Dicty which initially were on the two corners of the gird, get closer to one another and at the end of the simulation, they accumulate at the centre of the grid. The final location of maximum number of Dicty is on location (8,7) and is 0.3691614. In other words, the Dicty cells are moving on the grid from the two corners of the grid (1,1) and (15,15) towards the centre at the cAMP is being produced and covers the whole grid. Table 7.8 provides the change in the maximum Dicty's location over time.

Table 7.8: Random Restart Hill Climbing model 3673 maximum number of Dicty that was achieved from 50k RRCH, was run for a longer simulation. in a longer run, the maximum number of Dicty increased to 0.9 and the location of the maximum changes over time as they get closer to the centre.

Population Density	Location	Time Frame
0.95252073	15,15	5
0.8959393	15,15	10
0.7956463	15,15	20
0.7516380	15,15	25
0.6963008	15,15	32
0.6896904	1,1	33
0.6019709	1,1	50
0.4512101	1,1	100
0.3594109	5,6	300
0.3600330	6,6	500
0.3691614	8,7	1001

From the results achieved from RRHC, it can be seen that the most effecting rate on the population density of Dictyis the diffusion of cAMP. The slightest change in this rate, changes the outcome of the model significantly, as it was tested above in model 4147. Using the exact same rate for the other actions, and changing diffusion slightly, proves how strong the effect of diffusion is on the behaviour of Dicty cells.

7.5.2 Visualisation of Simulated Annealing

The simulated annealing runs each contain 69 Dicty on the grid which is 30.66667% of the grid. The simulation was run for 50000 iterations and the initial temperature 100 and the final temperature at 5.925502. The calculation for these numbers are provided in Chapter 6, Algorithm 2. There are three categories in the outcome data: accepted, rejected and high AP. The target function is based on the population of the Dicty cells on the grid. If the number is higher than the previous run, then it is accepted, otherwise it might be a good enough answer to be considered, so the AP is calculated . if AP is higher than a random number between 0 and 1, it is accepted with the tag of high AP otherwise, it is completely rejected. In this section the focus of the analysis is on the accepted and high AP solutions.

Out of the 50000 runs, only 13 are accepted. This is because at temperature 99.098065 when the production of cAMP is zero, the population of Dicty remains 1.000, since they do not move. This sets this number as the best solution and the next numbers are being compared to this. Therefore, only higher numbers are accepted after this temperature. However, the only way to achieve such numbers is by setting one of the rates to zero, as it can be seen in Table 7.9.

Table 7.9: 13 runs in simulated annealing are accepted

ITER	Location	Max number of Dicty	Diffusion	Movement	Sink	cAMP Production	Temperature
5	(8,8)	0.7496059	0.36	4.73	2.07	8.32	99.976002
17	(8,8)	0.7902603	0.50	5.04	4.52	0.91	99.904043
31	(8,8)	0.8187807	0.41	0.48	5.79	3.60	99.820157
96	(8,8)	0.8621508	0.10	4.49	6.20	4.71	99.431604
138	(8,8)	0.8676832	0.06	1.71	7.04	4.65	99.181345
152	(10,1)	1.0000000	3.79	0.72	0.58	0.00	99.098065
326	(2,13)	1.6761806	0.00	4.79	1.67	5.54	98.068832
544	(2,13)	1.6981981	0.00	2.37	1.21	2.75	96.794407
899	(2,13)	1.7604976	0.00	5.83	8.23	8.80	94.754427
6935	(2,13)	1.8153252	0.00	9.22	4.98	0.45	65.964565
12156	(2,13)	1.8754337	0.00	0.21	5.82	2.90	48.223589
45721	(2,13)	1.8842095	0.00	3.67	8.36	0.21	6.435876
47098	(2,13)	1.8968641	0.00	0.02	1.99	0.88	5.925502

Table 7.10 shows the top ten high AP solutions in order of the maximum population density. 107 high AP solutions contained zero rates, which were removed and 18817 remain with no zero rates. At temperature 5.622429, close to the end of simulation, the maximum number of Dicty with no Zero rates is achieved at 0.9658049. If the Dicty is located in all the locations, there will be 0.306 in each location. So, achieving 0.9 is three times more than the spread population

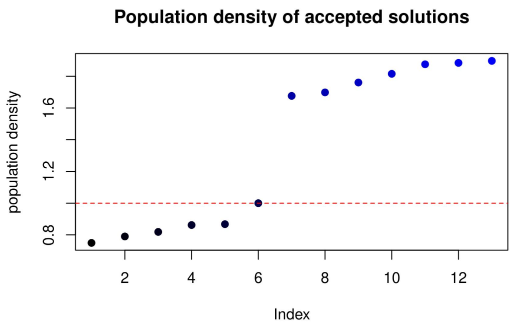
on the grid. Comparing the different rates, diffusion has a relatively low rate, compared to the results from RRHC.

Table 7.10: The top ten high AP solutions of SA sorted based on the maximum number of Dicty. the maximum is 0.9, which could have been an accepted solution is the best solution by this iteration was not 1.8.

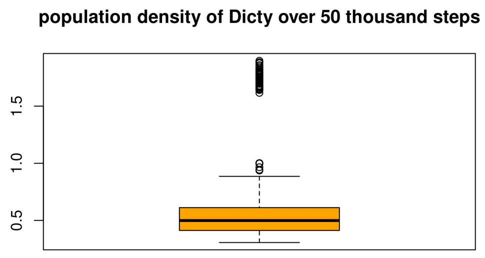
ITER	Location	Max number of Dicty	Diffusion	Movement	Sink	cAMP Production	Temperature
47973	(1,1)	0.9658049	7.65	0.01	6.43	0.10	5.622429
13404	(1,1)	0.9436496	10.74	0.02	9.19	0.12	44.744389
13157	(1,1)	0.9369829	7.83	0.01	2.43	0.07	45.412459
14950	(1,1)	0.8856991	16.73	0.03	2.21	0.04	40.780469
47452	(8,8)	0.8717055	0.09	7.86	18.43	2.45	5.800968
40094	(8,8)	0.8706830	0.08	5.67	13.33	5.93	9.020665
47574	(8,8)	0.8703896	0.12	3.19	18.61	8.83	5.758659
27417	(8,8)	0.8676441	0.05	4.84	7.78	2.66	19.301076
9721	(8,8)	0.8672616	0.05	1.89	8.2	0.34	55.809979
29207	(8,8)	0.8672304	0.06	8.06	6.57	5.59	17.335521

Simulated Annealing and Random Restart Hill Climbing are two different methods in algorithm. Therefore, the output data from these two should be treated and analysed differently. Heatmaps are not much of help in the outputs of SA as there are 13 accepted and more than 18 thousand high AP solutions. One of the important things to inspect from the result of the optimisation is to check the location of maximum population density. Figure 7.14a shows the scatter plot of the maximum population density of the accepted solutions in each simulation. The blue point are more than 1 and they have at least one zero rate. Figure 7.14b presents the box plot of the high AP solutions. As it can be seen, majority of the solutions are between 0.3 and 1.0.

Figure 7.14: The maximum population density of accepted and high AP solutions from Simulated Annealing



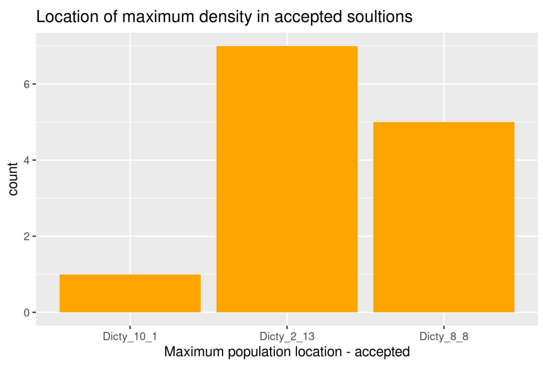
(a) The maximum population density of accepted solution. The population density higher than 1 are achieved with at least one zero rate.



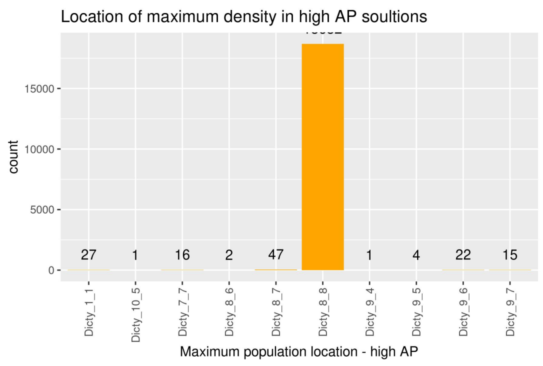
(b) Box plot of the population of high AP solutions of Simulated Annealing. As it can be seen most of the solutions are less than 1.

To check the location of maximum population density of Dicty on the 15x15 grid, both accepted and high AP solutions are visualised with bar plots which can be seen in Figures 7.15a and 7.15b respectively. In Plot 7.15a seven of the accepted solutions are located at (2,13). However, looking at the Table 7.9 it can be seen that all of these solutions have at least one zero rate. The only one at (10,1) is the solution with no cAPM production, which means the Dicty cells did not even move. Therefore, the accepted solutions with no zero rate are located at (8,8). The same behaviour could be found on the high AP solutions in Figure 7.15b, that the majority of the high AP are located at (8,8).

Figure 7.15: Bar Plot of the Location of Maximum Population Density From SA Results.



(a) The maximum population density of accepted solutions are located at (2,13) and (8,8).

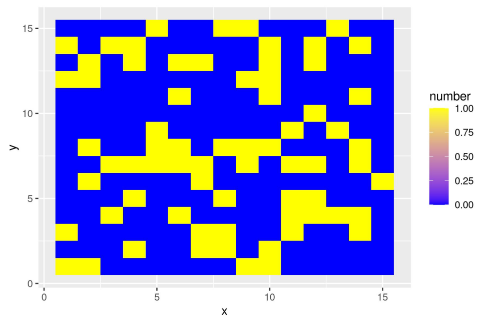


(b) The maximum population density of high AP solutions are located at (8,8)

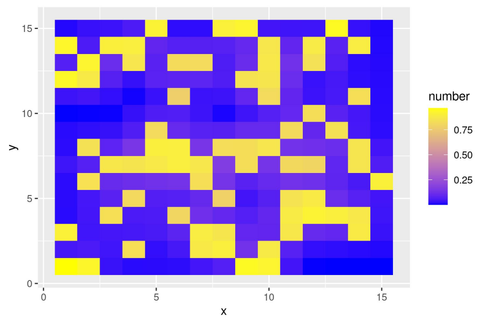
Looking at Table 7.10, it can be seen that the top high AP solutions are located at (1,1).

This is due to the very slow rate for movement. This movement rate, set between 0.01 and 0.03, is so slow that through the whole simulation the Dicty cells hardly move. This slow transition can be seen in Figures 7.16a and 7.16b.

Figure 7.16: The heatmap of the maximum high AP population density with the movement of 0.01.



(a) The initial location of Dicty on the 15x15 grid.



(b) The final location of Dicty in the final time frame.

7.5.3 Random Restart Hill Climbing Versus simulated Annealing

One of the reasons for using Random Restart Hill Climbing and Simulated annealing is to compare these two methods. These two methods are both well-known for simplicity and have been used for different studies. RRHC has a faster process, and as this process could be parallelised, this was an advantage. However, with RRHC, there is always the possibility that the algorithm is stuck in the local maxima and there is a chance that even with the added random restart feature, the best solution is not found. Also, this algorithm does not allow the worse solution to be accepted. So, if the best solution is only achievable through skipping a worse solution, this method is not the best. The number of accepted fitnesses for this method was relatively high, considering that because of the random restart, the search for the answers would start from the first step at the beginning of every random restart. The results show that at the end, the best solution out of the 50000 steps, without considering the zero rates, has been around 0.7. This might suggest that the global maxima might have reached in the 50000 steps.

On the other hand the SA simulation provides a new perspective. The highest number reached in SA is 0.9 and it is accepted because of the high AP. The reason is, the new solutions are being compared to the “best solution” rather than the previous one. By the time that the simulation reaches the temperature 5.6, the best solution is 1.89 and that is one of the results of Diffusion being zero. The algorithm allowed the rates to be zero but not less. The reason for that was to inspect the results of the zero rates as well. This is not surprising as the sum of the population of Dicty at the beginning of the simulation is 69 instead of 61. Therefore, naturally the maximum number of the population is higher. As explained before in Chapter 5 Section 5.2, The shell program works in a way that the number of occupied locations might not be the same. However, it is the same number, 61, for all RRHC rounds and the shell program was used one at the beginning of SA with 69 occupied locations. Unfortunately this number could not be checked while the simulation was running, and when after the simulation was finished, there was not enough time to re-run the simulation as it takes a month for 50000 runs.

Simulated Annealing has a much slower process. Considering the fact that there was no access to supercomputers for this thesis, this simulation took 35 days while Hill Climbing only took 5 days, minus the manual random restarts that delayed the process. Some of the restarts for HC was done manually for the results to be checked before the next steps. But after making sure that the code is working as expected, the random restart was set to automatic.

In both results it is obvious that the population density of Dicty has a relationship with the rate of diffusion, which is not unexpected considering the diffusion rate of the cAMP is the trigger for the movement and thus the accumulation of the Dicty cells. This relationship is especially visible in the “high AP” data group where the highest population achieved is 0.9658049, when none of the rates are zero. It is notable that the relationship is not linear. Among the 13 accepted results of the SA, the highest number is 0.8676832 which is at the beginning of the simulation and not yet the best solution found. The reason that 0.9658049 was not “accepted” further in the simulation was that at that temperature the best solution that the population density was being compared to was as high as 1.8968641 which was achieved with zero diffusion.

Between the two methods, RRHC is a better one for this study as it was quicker and provided the required information. This simulation can be run for more iterations in further studies.

7.6 Discussion

The components in the collection of models are only useful if they can behave as they are expected to. In this chapter one component from each level of complexity was used to be analysed and visualised as a proof of the models to be behaving as they are expected. Also, for one of the models, Dicty, the optimisation routines were run in order to find the best combination of constant rate to achieve a maximum number of Dicty as a clump on the grid as well as finding the most effective rate on the construction of fruiting body. Two methods of optimsation were used in this study: Random Restart Hill Climbing and Simulated Annealing.

The result from Random Restart Hill Climbing shows that the rate of diffusion has the most influence on the population of Dicty even if the relationship is not linear. From this it can be concluded that the smallest change in this constant rate can change the behaviour of the model completely.

Looking at the output data from SA, it can be concluded that diffusion and movement have the most effect on the population of Dicty in a cluster as well as the final locations. However, considering that there are the combination of the four rates, all of them change the behaviour. Based on the PC analysis, it was concluded that the behaviour of diffusion is the most different compared to the other rates. It seems when changing the rates, the changes on each rate should be done accordingly, meaning the three rates of cAmp production, movement and sink should be changed in one direction (for example, increase) while the changes on the diffusion rate should be on the opposite direction.

The models in this study are all continuous. Due to technical issues, such as errors from Spike and not being able to contact the developers, the stochastic models were not run for optimisation or any other parts of this thesis. However, the behaviour in stochastic models could be studied as an extension once these errors are fixed.

Chapter 8

Summary, Conclusions and Further Works

This chapter summarises this study and concludes what are the outcomes of this thesis. After that, a few suggestions for further work and moving forward are provided to imply how it is possible to use the outcomes of this study.

8.1 Summary

"Coming together is a beginning, keeping together is progress, working together is success." Henry Ford probably meant this sentence for humans. But in every level of life, from humans to bacteria, from multi-cell organisms to unicellular microorganisms communication is a must to survive. They need to communicate not only to one another, but towards the environment and even within themselves and it is essential to survive. So, even though Henry Ford was only talking about humans in a society, his quote is true about the whole biology.

Being "alive" in biology means to possess specific characteristics such as reproduction, movement and death. However, for all of these characteristics to be achieved, the communication is necessary both inside the cell between the organelles or outside between the organs and organisms. This proves the importance of communication and response in life. It could be said that the life exists because of the communication.

Biological networks inside and between the organisms are usually studied in a biological laboratory. But this method of studying them does not provide a detailed understanding of what is really happening. Hence, modelling complex and precise biological systems is a convenient method to understand, analyse and predict them wile saving time and resources and remove the human error factor from the results.

In this study a methodology was used to create a collection of model components that describe some very basic biological behaviours. The method used for creating this collection is Petri net and the chosen platform for using Petri net is Snoopy. Snoopy provides a multi-level, multi-scale and multidimensional environment which can be used for modelling complex systems in folded mannerThe reason for choosing Petri nets and Snoopy is debated in Chapter 2 after a brief comparison with other methods. Coloured Petri net is especially useful for repeated networks as it provides a folded model. This folded model can then be unfolded and reveal the whole complex system in a multidimensional setting. A protocol is also provided in this study which explains the step by step process of modelling in detail in Chapter 3

The components of this collection include: communication, response, movement, chemotaxis, reproduction and death. These are the basic behaviours that define if something is “alive”. These components could be used individually to study only one behaviour or as a combined model for more complex behaviours. After constructing the base of all models, Properties, the design of the models continues step by step from simple (Lvl0) to complex (Lvl2). Lvl0 models describe very simple behaviours such as Duplication or Diffusion ad Lvl2 models are complex networks. In all of these levels, the existence of Properties is a must, granted that the modelling is done in 2D or 3D. As a case study, these models were applied to two different microorganisms: bacteria and Dictyostelium. Each of these microorganisms belong to one of the main superkingdoms of biology which are Prokaryotes and Eukaryotes respectively. This is a hierarchical modelling method. It means that the more complex models are built from simpler components and properties. These components are explained in detail in Chapter 4 and their application to two different microorganisms in order to assemble and create a complex system is explained in Chapter 5.

After the modelling is done, the next step is to simulate the models. Of course Snoopy provides the built-in simulation using different engines, but since Snoopy has GUI the simulation is slower for the more complicated models. For this reason, Spike is chosen for simulation of more complex systems. Spike is a command-line programme that uses ANDL and CANDL files to simulate the models. Using ANDL and CANDL files is another reason for Spike’s speed of simulation. They are human readable text files that contain the models’ entities and could be exported from any type of Petri net in Snoopy.

When a ANDL or CANDL file is exported from Snoopy, it is possible to initiate the desired changes on these files instead of the Petri net model. Spike then used these files to simulate the models with a SPC file. It is possible to amend the parameters in the SPC files instead of the original model. This will make changing the models easier as the main file that belongs to the model is unchanged, while the outputs of the simulation are based on the new introduced parameters in SPC file. Using this feature, it is also possible to scan through the parameters of the model using Spike.

The result of these simulations are then analysed and visualised using R. For each model, the result is depending on the expected behaviour which in this study this expectation is based on biological literature.

After simulating the models, it is important to see if it is possible to find the best action rates for each action in the models in order to receive the expected behaviour. This is where optimisation is useful. Using two methods of optimisation, Random Restart Hill Climbing and Simulated Annealing, these models were optimised in order to find the best possible solution. Python is used to implement these methods.

It is important to note that due to lack of time and the intensity of the python routine that was written for optimisation, only the Dicty model is chosen as an application of optimisation into a model. Also, the only models that is used for optimisation is continuous. Due to technical issues it was not possible to simulate the stochastic models. These technical issues include a bug in Spike that would not allow the stochastic simulations to run for complex systems. Due to no access to super computers, it was not possible to use Snoopy for these simulations. As mentioned before, Snoopy is much slower when it comes to simulation of complex models because of the GUI compared to Spike.

In this thesis, a step-by-step protocol is also provided to explain the process of modelling and combination of each part. This can help the future users to use the models independently.

8.2 Conclusion

In this study a methodology was introduced to create a collection of model components based on engineering principles to study biological system. These components describe movement, death, reproduction, communication and response. The modelling process is provided in this study as well as the results of the simulations and optimisation of the models.

To apply the model components into a simple biological concept, Dictyostelium and bacteria are chosen as the case studies. however, these models could be used for other (micro)organisms as well as non-biological applications due to their abstraction. While Chemotaxis demonstrates the movement towards a chemical trigger, it can also represent movement of a population towards or away from a trigger. This population could be describing anything, from bacteria to humans.

The main aim of creating this library of model component is to grant an easy way of modelling biological systems without the need of deep computational knowledge. Therefore, choosing a technique for this study was one of the most important steps. Due to its simplicity, and for other technical reasons which have been explained Chapter 2 in detail, Petri net was chosen. Using this tool, the collection is created in four levels of complexity:

1. Properties which are Functions, constants, coloursets or variables defined in the model;
2. Level-0 Basic Components are the simplest parts of a model that cannot be broken down and consist of at least one place and one transition;
3. Level-1 Simple Models which consist of One to two level-1 components and properties;
4. Level-2 which are made of more than two Level-0 or Level-1 components and properties.
5. Level-3 which is made of Level-2 models.

Among the systems, there are two complex models that their components are not included in the collection. These were the first models that were made in detail for this study. But to move forward a step towards moving the models we faced a technical issue. With the current Snoopy it is not possible to move complex system. As it has been explained before in Chapter 3, Snoopy in its current version can only move tokens around. The places exist in all the locations of the grid but are only “activated” on the marked locations. This prevents the whole model to move on a grid. Thus, to move forward, the decision was to move towards more generic and abstract models that can demonstrate movement and other behaviours without causing any problems.

The optimisation is done for finding the best combination of constant rates in the model Dicty. As discussed in Chapter 7, It turns out that among the four rates in the model, the diffusion of cAMP has the more influence on the behaviour of Dicty to the point that the slightest change, can result in the completely different number of Dicty cells. The reason for this, is because the diffusion rate of the cAMP effects directly on how fast the Dicty cells will move towards the high concentration of cAMP, while the rate for movement remains the same. This shows that even without the change in the rate of chemotaxis, it is possible to change the speed of movement with diffusion rate. Considering that there is not duplication in this model, this is an interesting outcome. The optimisation could be moved forward to stochastic models to observe the behaviour of Dicty when it only accepts integer numbers. However, due to technical issues this interesting study is not a part of this thesis.

8.3 Further Work

The focus of this study is on the construction of the collection of model components and describing basic biological behaviours. However, it also contains analysis of the output data from the models and optimising the models. Due to the limitation of time or computational issues, some steps of this thesis could not be developed any further. For example, due to the problem with Spike, it was not possible to work on stochastic models. There are many improvements that can be done to this study when the time is not limited or the problems are fixed. Here are a few suggestions for the further works using the outputs of this study:

8.3.1 Biological and Non-Biological Applications

In this study, the components of the collection of models were applied to two different biological case-studies: Dictyostelium and Bacteria. Even though originally these components were created for biological behaviour description, they can be useful in non-biological scenarios as well, since the models are abstract and generic. These models, should they be validated by biological experimental data, can be used in drug development industry.

8.3.2 Stochastic and Complex Movement

There were technical issues that prevented us from going further in some aspects of this study. The first issue is the complex movement. As explained before, the current models in Snoopy can only move the tokens on the grid but not complex models. In this study the places are located everywhere but are only activated in the defined locations. This prevents the movement the complex model to different locations. The detailed and complex model of biofilm formation faced this issue and it could not be moved.This causes a dead end when it comes to moving a whole system. As a separate funded project this issue is under study by Professors David Gilbert and Monika Heiner. But at the moment, this is not possible.

The second problem is the issue with stochastic models. On a Linux machine Spike never stops for a stochastic model and on a mac, for the same model, Spike does ends with no errors but no outputs. It seems the size of the gird as well as the complexity of the system prevents the simulation to be completed. To solve this problem, the developer's cooperation and more time for debugging Spike is required.

8.3.3 Automated Modelling

As mentioned before, ANDL and CANDL files are human-readable files exported in Snoopy from the models. To develop or amend a model, it is possible to simply edit this text file. But this changes should be done with attention, since the smallest mistake will result in errors when importing or simulating the models.

It might be possible to create a routine in a programming language that automates the combination process of the systems. A routine that chooses the correct components and combines them into a system. This way, times will be saved and human errors will be avoided. The main focus of this thesis is on creating and developing the library rather than making the process automated. Hence this suggestions cannot be taken further in this study.

8.3.4 Improving the collection

The components of the System Biofilm Formation does not exist as components in the collection of models in this thesis. This model was originally built as a detailed model. This model can be broken down into its building components and these parts can be added to the library as new components and be combined with other parts. Breaking down this model also provides this opportunity to created more detailed models from the components of the collection.

The current collection only describes simple, basic behaviours. This collection can be developed by adding more behaviours to it. There are many biological behaviours that can be modelled in abstract and from simple parts. Level-0 of this collection could include more simplistic models which can combine with other parts to create a complex system. The more Level-0 and Level-1 models are added, more combined models can be created. These models can then be applied for other case-studies, biological or non-biological.

Examples of more basic biological behaviours can include but not limited to:

1. Sexual reproduction that nrequires two different parents,
2. Physiology of movement, e.g. flagella,
3. Adding intermediate mechanisms such as breaking down the food source for energy and the consumption of that energy as two different Level0 models.

These are only a few suggestions of further modelling in biological concept. But this methodology used in this thesis could be used beyond biology.

fin.

Bibliography

[1] Martín Abadi, Bruno Blanchet, and Cédric Fournet. The Applied Pi Calculus. *Journal of the ACM*, 65(1):1–41, 2017.

[2] Julius Adler. Chemotaxis in Bacteria Motile Escherichia coli migrate in bands that are. *Advancement Of Science*, 153(3737):708–716, 1966.

[3] Eran Agmon and Ryan K. Spangler. A multi-scale approach to modeling E. coli chemotaxis. *Entropy*, 22(10):1–22, 2020.

[4] Victoria J M Allan, Maureen E Callow, Lynne E Macaskie, and Marion Paterson-Beedle. Effect of nutrient limitation on biofilm formation and phosphatase activity of a Citrobacter sp. *Microbiology*, 148(1):277–288, 2002.

[5] Ahmad Almasoud, Navam Hettiarachchy, Srinivas Rayaprolu, Dinesh Babu, Young Min Kwon, and Andy Mauromoustakos. Inhibitory effects of lactic and malic organic acids on autoinducer type 2 (AI-2) quorum sensing of Escherichia coli O157:H7 and Salmonella Typhimurium. *LWT - Food Science and Technology*, 66:560–564, 2016.

[6] G. An, B. G. Fitzpatrick, S. Christley, P. Federico, A. Kanarek, R. Miller Neilan, M. Oremland, R. Salinas, R. Laubenbacher, and S. Lenhart. Optimization and Control of Agent-Based Models in Biology: A Perspective. *Bulletin of Mathematical Biology*, 79(1):63–87, 2017.

[7] Gary An and Qi Mi. Agent-based models in translational systems biology. *Systems Biology and Medicine*, 1(2):159–171, 2009.

[8] Bassam A Annous, Pina M Fratamico, and James L Smith. Scientific status summary: Quorum sensing in biofilms: Why bacteria behave the way they do. *Journal of Food Science*, 74(1), 2009.

[9] David Artis. Epithelial-cell recognition of commensal bacteria and maintenance of immune homeostasis in the gut, 2008.

[10] George Assaf and Monika Heiner. Spatial Encoding of Systems Using Coloured Petri Nets Spatial Encoding of Systems Using Coloured Petri Nets. *Algorithms and Tools for Petri Nets,*, page 38, 2019.

[11] Fredrik Bäckhed, Ruth E Ley, Justin L Sonnenburg, Daniel A Peterson, Jeffrey I Gordon, and Fredrik Backhed. Host-Bacterial Mutualism in the Human Intestine. *Science*, 307(5717):1915–1920, 2005.

[12] Christoph Bader, Sunanda Sharma, Rachel Smtih, Jean Disset, and Neri Oxman. Viva in Silico: A position-based dynamics model for microcolony morphology simulation. *ALIFE: Artificial Life Journal*, 24(3), 2018.

[13] P. C. Baehni and Y. Takeuchi. Anti-plaque agents in the prevention of biofilm-associated oral diseases. *Oral Diseases*, 9(SUPPL. 1):23–29, 2003.

[14] Kshitij Bansal, Eric Koskinen, Thomas Wies, and Damien Zufferey. Structural counter abstraction. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*, pages 62–77. Springer, 2013.

[15] Roberta Bardini, Stefano Di Carlo, Gianfranco Politano, and Alfredo Benso. Modeling antibiotic resistance in the microbiota using multi-level Petri Nets. *BMC systems biology*, 12(Suppl 6):108, 2018.

[16] Amy L. Bauer, Catherine A.A. Beauchemin, and Alan S. Perelson. Agent-based modeling of host-pathogen systems: The successes and challenges. *Information Sciences*, 179(10):1379–1389, 2009.

[17] Falko Bause and PS Kritzinger. Stochastic Petri Nets: An introduction to the theory. *Vieweg*, pages 133–140, 2002.

[18] Jan A Bergstra, Wan Fokkink, and Alban Ponse. Process Algebra with Recursive Operations. *Handbook of Process Algebra*, pages 333–389, 2007.

[19] Francesco Bernardini, Marian Gheorghe, and Natalio Krasnogor. Quorum sensing P systems. *Theoretical Computer Science*, 371(1-2):20–33, 2007.

[20] Marco Bernardo, Pierpaolo Degano, and Gianluigi Zavattaro. *Formal Methods for Computational Systems Biology: 8th International School on Formal Methods for the Design of Computer, Communication, and Software Systems, SFM 2008 Bertinoro, Italy, June 2-7, 2008*, volume 5016. Springer, 2008.

[21] Mary Ann Blätke, Monika Heiner, and Wolfgang Marwan. BioModel Engineering with Petri Nets. In R Robeva, editor, *Algebraic and Discrete Mathematical Methods for Modern Biology*, chapter 7, pages 141–193. Elsevier Inc., mar 2015.

[22] Mary Ann Blätke and Christian Rohr. A coloured Petri net approach for spatial Biomodel Engineering based on the modular model composition framework Biomodelkit. *CEUR Workshop Proceedings*, 1373(January):37–54, 2015.

[23] Mary Ann Blätke, Christian Rohr, Monika Heiner, and Wolfgang Marwan. *A petri-net-based framework for biomodel engineering*, volume 65. Springer, 2014.

[24] Rainer Breitling, David Gilbert, Monika Heiner, and Richard Orton. A structured approach for the engineering of biochemical network models, illustrated for signalling pathways. *Briefings in Bioinformatics*, 9(5):404–421, 2008.

[25] Geo Brooks, Karen Carroll, Janet Butel, Stephen Morse, and Timothy Mietzner. *Medical Microbiology*. McGraw-Hill Education, 2015.

[26] Jonathan R Brown, Joseph Jurcisek, Vinal Lakhani, Ali Snedden, William C Ray, Elaine M Mokrzan, Lauren O Bakaletz, and Jayajit Das. In Silico Modeling of Biofilm Formation by Nontypeable Haemophilus influenzae In Vivo . *mSphere*, 4(4):1–13, 2019.

[27] Andre Gerald Buret, Jean Paul Motta, Thibault Allain, Jose Ferraz, and John Lawrence Wallace. Pathobiont release from dysbiotic gut microbiota biofilms in intestinal inflammatory diseases: A role for iron? 06 Biological Sciences 0605 Microbiology. *Journal of Biomedical Science*, 26(1):1–14, 2019.

[28] Mette Burmølle, Dawei Ren, Thomas Bjarnsholt, and Søren J. Sørensen. Interactions in multispecies biofilms: Do they actually matter? *Trends in Microbiology*, 22(2):84–91, feb 2014.

[29] Nadia Busi and Roberto Gorrieri. A Petri net semantics for π-calculus. In *International Conference on Concurrency Theory*, pages 145–159. Springer, 2012.

[30] Allyson L. Byrd, Yasmine Belkaid, and Julia A. Segre. The human skin microbiome. *Nature Reviews Microbiology*, 16(3):143–155, 2018.

[31] Huaqing Cai, Chuan Hsiang Huang, Peter N. Devreotes, and Miho Iijima. Analysis of chemotaxis in dictyostelium. *Methods in Molecular Biology*, 757:451–468, 2011.

[32] Muffy Calder and Stephen Gilmore, editors. *Computational Methods in Systems Biology*. Springer, Edinburgh, Scotland, 2007.

[33] A Carl. Petri. 1962. *Kommunikation mit automaten*, 1962.

[34] Claudine Chaouiya. Petri net modelling of biological networks. *Briefings in Bioinformatics*, 2007.

[35] Y Chen, Y Peng, and X Fu. Microbial Biofilms, Colorectal Inflammation and Cancer. *Austin Journal of Gastroenterology*, 3(1):1059, 2016.

[36] Jacek Chodak and Monika Heiner. Spike - a command line tool for continuous , stochastic & hybrid simulation of (coloured) Petri nets. In *Proc. 21th German Workshop on Algorithms and Tools for Petri Nets (AWPN 2018).*, pages 1–6. University of Augsburg, 2018.

[37] David L. Chopp, Mary Jo Kirisits, Brian Moran, and Matthew R. Parsek. A mathematical model of quorum sensing in a growing bacterial biofilm. In *Journal of Industrial Microbiology and Biotechnology*, 2002.

[38] Federica Ciocchetta. Bio-PEPA with events. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 5750 LNBI:45–68, 2009.

[39] Federica Ciocchetta, Adam Duguid, Stephen Gilmore, Maria Luisa Guerriero, and Jane Hillston. The Bio-PEPA tool suite. *QEST - 6th International Conference on the Quantitative Evaluation of Systems*, pages 309–310, 2009.

[40] Federica Ciocchetta and Maria Luisa Guerriero. Modelling Biological Compartments in Bio-PEPA. *Electronic Notes in Theoretical Computer Science*, 227(C):77–95, 2009.

[41] Federica Ciocchetta and Jane Hillston. Bio-PEPA: An Extension of the Process Algebra PEPA for Biochemical Networks. *Electronic Notes in Theoretical Computer Science*, 194(3):103–117, 2008.

[42] Federica Ciocchetta and Jane Hillston. Bio-PEPA: A framework for the modelling and analysis of biological systems. *Theoretical Computer Science*, 410(33-34):3065–3084, 2009.

[43] Marcie B. Clarke, David T. Hughes, Chengru Zhu, Edgar C. Boedeker, and Vanessa Sperandio. The QseC sensor kinase: A bacterial adrenergic receptor. *Proceedings of the National Academy of Sciences of the United States of America*, 103(27):10420–10425, 2006.

[44] Michael Clerx, Michael T. Cooling, Jonathan Cooper, Alan Garny, Keri Moyle, David P. Nickerson, Poul M. F. Nielsen, and Hugh Sorby. Cellml 2.0. *Journal of Integrative Bioinformatics*, 17(2-3):20200021, 2020.

[45] José-Manuel Colom and Jörg Desel. Application and theory of petri nets and concurrency. In *Proceedings of the 34th International Conference Lecture Notes in Computer Science*. Springer, 2013.

[46] Elizabeth K. Costello, Keaton Stagaman, Les Dethlefsen, Brendan J.M. Bohannan, and David A. Relman. The application of ecological theory toward an understanding of the human microbiome, 2012.

[47] Silvia Daun, Jonathan Rubin, Yoram Vodovotz, and Gilles Clermont. Equation-based models of dynamic biological systems. *Journal of critical care*, 63(8):1–18, 2008.

[48] Rene David and Alla Hassane. Petri Nets for Modeling of Dynamic Systems A Survey. *Automatica*, 30(2):175–202, 1994.

[49] Anna De Breij, Lenie Dijkshoorn, Ellen Lagendijk, Joke Van Der Meer, Abraham Koster, Guido Bloemberg, Ron Wolterbeek, Peterhans Van Den Broek, and Peter Nibbering. Do Biofilm Formation and Interactions with Human Cells Explain the Clinical Success of Acinetobacter baumannii? *PLoS ONE*, 5(5), 2010.

[50] Sigrid C J De Keersmaecker, Kathleen Sonck, and Jos Vanderleyden. Let LuxS speak up in AI-2 signaling. *Trends in Microbiology*, 14(3):114–119, 2006.

[51] V de Lorenzo and Antoine Danchin. Synthetic biology: discovering new worlds and new words. *EMBO reports*, 9:822–827, 2008.

[52] Michael A. Deakin. Modelling biological systems. *Dynamics of Complex Interconnected Biological Systems*, pages 2–16, 1990.

[53] Christine M. Dejea, Payam Fathi, John M. Craig, Annemarie Boleij, Rahwa Taddese, Abby L. Geis, Xinqun Wu, Christina E. DeStefano Shields, Elizabeth M. Hechenbleikner, David L. Huso, Robert A. Anders, Francis M. Giardiello, Elizabeth C. Wick, Hao Wang, Shaoguang Wu, Drew M. Pardoll, Franck Housseau, and Cynthia L. Sears. Patients with familial adenomatous polyposis harbor colonic biofilms containing tumorigenic bacteria. *Science*, 359(6375):592–597, 2018.

[54] Matthew P Delisa, James J Valdes, and William E Bentley. Mapping Stress-Induced Changes in Autoinducer AI-2 Production in Chemostat-Cultivated Escherichia coli K-12

Mapping Stress-Induced Changes in Autoinducer AI-2 Production in Chemostat-Cultivated Escherichia coli K-12. *Journal of Bacteriology*, 183(9):2918–2928, 2001.

[55] Rui Dilão and Marcus J.B. Hauser. Chemotaxis with directional sensing during Dictyostelium aggregation. *Comptes Rendus - Biologies*, 336(11-12):565–571, 2013.

[56] Eleonora Distrutti, Lorenzo Monaldi, Patrizia Ricci, and Stefano Fiorucci. Gut microbiota role in irritable bowel syndrome: New therapeutic strategies, 2016.

[57] Jack D. Dockery and James P. Keener. A mathematical model for quorum sensing in Pseudomonas aeruginosa. *Bulletin of Mathematical Biology*, 63:95–116, 2001.

[58] Vinayak Doraiswamy, Martin Buist, and Andrew B. Goryachev. Computational modelling of chemotaxis in cooperative phenomena in bacterial populations. *BMC Systems Biology*, 1(S1):1–2, 2007.

[59] Qingyou Du, Yoshinori Kawabe, Christina Schilde, Zhi Hui Chen, and Pauline Schaap. The Evolution of Aggregative Multicellularity and Cell-Cell Communication in the Dictyostelia. *Journal of Molecular Biology*, 427(23):3722–3733, 2015.

[60] Ravindra Duddu, David L. Chopp, and Brian Moran. A two-dimensional continuum model of biofilm growth incorporating fluid flow and shear stress based detachment. *Biotechnology and Bioengineering*, 103(1):92–104, 2009.

[61] Hermann J. Eberl, David F. Parker, and Mark C.M. Vanloosdrecht. A new deterministic spatio-temporal continuum model for biofilm development. *Journal of Theoretical Medicine*, 3(3):161–175, 2001.

[62] Matthew Paul Edgington. *Mathematical Modelling of Bacterial Chemotaxis Signalling Pathways*. PhD thesis, University of Reading, 2015.

[63] Zahra Eidi, Farshid Mohammad-Rafiee, Mohammad Khorrami, and Azam Gholami. Modelling of Dictyostelium discoideum movement in a linear gradient of chemoattractant. *Soft Matter*, 13(44):8209–8222, 2017.

[64] Sivan Elias and Ehud Banin. Multi-species biofilms: Living with friendly neighbors. *FEMS Microbiology Reviews*, 36(5):990–1004, 2012.

[65] Blessing O. Emerenini, Burkhard A. Hense, Christina Kuttler, and Hermann J. Eberl. A mathematical model of quorum sensing induced biofilm detachment. *PLoS ONE*, 10(7):1–25, 2015.

[66] Hindley J. EngineRoger and Jonathan P. Seldin. Lambda-calculus and combinators, an introduction. *Lambda-Calculus and Combinators, an Introduction*, 9780521898:1–345, 2008.

[67] Jasmin Fisher and David Harel. On statecharts for biology. *Symbolic Systems Biology: Theory and Methods*, 2010.

[68] Hans Curt Flemming and Jost Wingender. The biofilm matrix. *Nature Reviews Microbiology*, 8(9):623–633, 2010.

[69] Mallory R. Frederick, Christina Kuttler, Burkhard A. Hense, and Hermann J. Eberl. A mathematical model of quorum sensing regulated EPS production in biofilm communities. *Theoretical Biology and Medical Modelling*, 2011.

[70] Rudolph Freund, Gheorghe Paun, Grzegorz Rozenberg, and Arto Salomaa. *Membrane Computing*, volume 3850. Springer, Vienna, 2005.

[71] W Claiborne Fuqua, Stephen C. Winans, and E Peter Greenberg. MINIREVIEW Quorum Sensing in Bacteria : the LuxR-LuxI Family of Cell Density-Responsive Transcriptional Regulatorst. *Journal of bacteriology*, 176(2):269–275, 1994.

[72] Meng Gao, Huizhen Zheng, Ying Ren, Ruyun Lou, Fan Wu, Weiting Yu, Xiudong Liu, and Xiaojun Ma. A crucial role for spatial distribution in bacterial quorum sensing. *Scientific Reports*, 6, 2016.

[73] Qian Gao, Fei Liu, David Tree, and David Gilbert. Multi-cell Modelling Using Coloured Petri Nets Applied to Planar Cell Polarity. *Proceedings of the 2nd International Workshop on Biological Processes & Petri Nets (BioPPN2011)*, pages 135–150, 2011.

[74] J. Garcia-Ojalvo, M. B. Elowitz, and S. H. Strogatz. Modeling a synthetic multicellular clock: Repressilators coupled by quorum sensing. *Proceedings of the National Academy of Sciences*, 101(30):10955–10960, 2004.

[75] Alan Garny. Opencor. *Nature Precedings*, pages 1–1, 2011.

[76] David Gilbert, Monika Heiner, and Leila Ghanbar. Personalised models for human – gut microbiota interaction. *PeerJ preprints*, 3267(e3267v1):1–4, 2017.

[77] David Gilbert, Monika Heiner, Leila Ghanbar, and Jacek Chodak. Spatial quorum sensing modelling using coloured hybrid Petri nets and simulative model checking. *BMC Bioinformatics*, 20(4)(173):1–23, 2019.

[78] David Gilbert, Monika Heiner, and Sebastian Lehrack. A unifying framework for modelling and analysing biochemical pathways using Petri nets. In *International Conference on Computational Methods in Systems Biology*, pages 200–216. Springer, 2007.

[79] Randy Goebel, Yuzuru Tanaka, and Wolfang Wahlster. *Multi-Agent Systems and Agreement Technologies*. Springer, 2020.

[80] Laurent Golé, Charlotte Rivière, Yoshinori Hayakawa, and Jean Paul Rieu. A quorum-sensing factor in vegetative Dictyostelium Discoideum cells revealed by quantitative migration analysis. *PLoS ONE*, 6(11):1–9, 2011.

[81] Stavros Gonidakis and Valter D Longo. Assessing Chronological Aging in Bacteria Stavros Gonidakis and Valter D. Longo Abstract. In *Cell Senescence: Methods and Protocols, Methods in Molecular Biology*, volume 965, chapter 28, pages 421–437. Springer Science+Business Media, 2013.

[82] Andres F. Gonzalez Barrios and Luke E.K. Achenie. Escherichia coli autoinducer-2 uptake network does not display hysteretic behavior but AI-2 synthesis rate controls transient bifurcation. *BioSystems*, 99(1):17–26, 2010.

[83] Thomas E. Gorochowski. Agent-based modelling in synthetic biology. *Essays In Biochemistry*, 60(4):325–336, 2016.

[84] Andrew B. Goryachev. Understanding bacterial cell-cell communication with computational modeling. *Chemical Reviews*, 111(1):238–250, 2011.

[85] Caitriona M Guinane and Paul D Cotter. Role of the gut microbiota in health and chronic gastrointestinal disease: Understanding a hidden metabolic organ. *Therapeutic Advances in Gastroenterology*, 6(4):295–308, 2013.

[86] Warren Hedley, Melanie Nelson, David Bullivaant, and Paul Nielsen. A Short Introduction to CellML. *The Royal Society*, 359:1073–1089, 2001.

[87] Monika Heiner, Robin Donaldson, and David Gilbert. Petri Nets for Systems Biology. *Symbolic Systems Biology: Theory and Methods*, pages 61–97, 2010.

[88] Monika Heiner and David Gilbert. BioModel engineering for multiscale Systems Biology. *Progress in Biophysics and Molecular Biology*, 111(2-3):119–128, 2013.

[89] Monika Heiner, Ronny Richter, and Martin Schwarick. Snoopy: a tool to design and animate/simulate graph-based formalisms. *Proceedings of the 1st international conference*

on Simulation tools and techniques for communications, networks and systems & workshops, pages 1–10, 2008.

[90] Burkhard A Hense and Martin Schuster. Core Principles of Bacterial Autoinducer Systems. *Microbiology and Molecular Biology Reviews*, 2015.

[91] Moshe Herzberg, Ian K Kaye, Wolfgang Peti, and Thomas K Wood. YdgG (TqsA) Controls Biofilm Formation in Escherichia coli K-12 through Autoinducer 2 Transport. *Journal of Bacteriology*, 188(2):587–598, 2006.

[92] Thomas Höfer and Philip K. Maini. Streaming instability of slime mold amoebae: An analytical model. *Physical Review E - Statistical Physics, Plasmas, Fluids, and Related Interdisciplinary Topics*, 56(2):2074–2080, 1997.

[93] Ralf Hofestädt. Advantages of Petri-Net Modeling and Simulation for Biological Networks. *International Journal of Bioscience, Biochemistry and Bioinformatics*, 7(4):221–229, 2017.

[94] Alina Maria Holban, Coralia Bleotu, Mariana Carmen Chifiriuc, and Veronica Lazar. Control of bacterial virulence by cell-to-cell signalling molecules. *Microbial pathogens and strategies for combating them: science, technology and education*, pages 311–321, 2013.

[95] Sara Hooshangi and William E Bentley. LsrR quorum sensing "switch" is revealed by a bottom-up approach. *PLoS Computational Biology*, 7(9), 2011.

[96] M Hucka, A Finney, H M Sauro, H Bolouri, J C Doyle, H Kitano, A P Arkin, B J Bornstein, D Bray, A Cornish-Bowden, A A Cuellar, S Dronov, E D Gilles, G Ginkel, V Gor, I I Goryanin, W J Hedley, T C Hodgman, J . H Hofmeyr, P J Hunter, N S Juty, J L Kasberger, A Kremling, U Kummer, N Le Novère, L M Loew, D Lucio, P Mendes, E Minch, and E D Mjolsness. The Systems Biology Markup Language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics*, 19, 2003.

[97] Niall P. Hyland and John F. Cryan. Microbe-host interactions: Influence of the gut microbiota on the enteric nervous system. *Developmental Biology*, 417(2):182–187, sep 2016.

[98] Anisa S Ismail, Julie S Valastyan, and Bonnie L Bassler. A Host-Produced Autoinducer-2Mimic Activates Bacterial Quorum Sensing. *Cell Host & Microbe*, 19(4):1–11, 2016.

[99] Sorin Istrail, P Pevzner, and M Waterman. *Lecture Notes in Bioinformatics*. Springer, 2012.

[100] Sheldon H. Jacobson. Analyzing the performance of local search algorithms using generalized hill climbing algorithms. *Operations Research/ Computer Science Interfaces Series*, 15:441–467, 2002.

[101] Sally James, Patric Nilsson, Geoffrey James, Staffan Kjelleberg, and Torbjörn Fagerström. Luminescence control in the marine bacterium Vibrio fischeri: an analysis of the dynamics of lux regulation. *Journal of Molecular Biology*, 296(4):1127–1137, mar 2000.

[102] Caroline H. Johnson, Christine M. Dejea, David Edler, Linh T. Hoang, Antonio F. Santidrian, Brunhilde H. Felding, Julijana Ivanisevic, Kevin Cho, Elizabeth C. Wick, Elizabeth M. Hechenbleikner, Winnie Uritboonthai, Laura Goetz, Robert A. Casero, Drew M. Pardoll, James R. White, Gary J. Patti, Cynthia L. Sears, and Gary Siuzdak. Metabolism links bacterial biofilms and colon carcinogenesis. *Cell Metabolism*, 2015.

[103] Amandeep Kaur, Neena Capalash, and Prince Sharma. Quorum sensing in thermophiles: Prevalence of autoinducer-2 system. *BMC Microbiology*, 18(1):1–16, 2018.

[104] Paul Kirk, Thomas Thorne, and Michael P.H. Stumpf. Model selection in systems and synthetic biology. *Current Opinion in Biotechnology*, 24(4):767–774, 2013.

[105] Marc W. Kirschner. The meaning of systems biology. *Cell*, 121(4):503–504, 2005.

[106] Gavin Kistner and Chris Nuernberger. Developing user interfaces using scxml statecharts. In *Proceedings of the 1st EICS Workshop on Engineering Interactive Computer Systems with SCXML*, pages 5–11, 2014.

[107] Hiroaki Kitano. Computational systems biology, 2002.

[108] Naoki Kobayashi, Benjamin C. Pierce, and David N. Turner. Linearity and the pi-calculus. *ACM Transactions on Programming Languages and Systems*, 21(5):914–947, 1999.

[109] Natallia Kokash. An introduction to heuristic algorithms. *Department of Informatics and Telecommunications*, pages 1–8, 2005.

[110] Paul E Kolenbrander, Robert J Palmer, Saravanan Periasamy, and Nicholas S Jakubovics. Oral multispecies biofilm development and the key role of cell-cell distance, 2010.

[111] Alexander Kurilshikov, Cisca Wijmenga, Jingyuan Fu, and Alexandra Zhernakova. Host Genetics and Gut Microbiome: Challenges and Perspectives. *Trends in Immunology*, 38(9):633–647, 2017.

[112] Chang Ro Lee, Jung Hun Lee, Moonhee Park, Kwang Seung Park, Il Kwon Bae, Young Bae Kim, Chang Jun Cha, Byeong Chul Jeong, and Sang Hee Lee. Biology of Acinetobacter baumannii: Pathogenesis, antibiotic resistance mechanisms, and prospective treatment options. *Frontiers in Cellular and Infection Microbiology*, 7(MAR), 2017.

[113] Jun Li, Can Attila, Liang Wang, Thomas K. Wood, James J. Valdes, and William E. Bentley. Quorum sensing in Escherichia coli is signaled by AI-2/LsrR: Effects on small RNA and biofilm architecture. *Journal of Bacteriology*, 2007.

[114] Jun Li, Liang Wang, Yoshifumi Hashimoto, Chen-Yu Tsao, Thomas K. Wood, James J. Valdes, Evanghelos Zafiriou, and William E. Bentley. A stochastic model of Escherichia coli AI-2 quorum signal circuit reveals alternative synthesis pathways. *Molecular systems biology*, 2(1):67, 2006.

[115] Shan Li, Sergey R. Konstantinov, Ron Smits, and Maikel P. Peppelenbosch. Bacterial Biofilms in Colorectal Cancer Initiation and Progression, 2017.

[116] Yung Hua Li and Xiao Lin Tian. Quorum Sensing and Bacterial Social Interactions in Biofilms: Bacterial Cooperation and Competition. *Stress and Environmental Regulation of Gene Expression and Adaptation in Bacteria*, 2:1197–1205, 2016.

[117] Fei Liu and Monika Heiner. Modeling membrane systems using colored stochastic Petri nets. *Natural Computing*, 12(4):617–629, 2013.

[118] Fei Liu, Monika Heiner, and David Gilbert. Coloured Petri nets for multilevel, multiscale and multidimensional modelling of biological systems. *Briefings in Bioinformatics*, pages 1–10, 2017.

[119] Catherine M Lloyd, James R Lawson, Peter J Hunter, and Poul F Nielsen. The cellml model repository. *Bioinformatics*, 24(18):2122–2123, 2008.

[120] Leslie M Loew and James C Schaff. The virtual cell: a software environment for computational cell biology. *TRENDS in Biotechnology*, 19(10):401–406, 2001.

[121] Laurence Loewe and Jane Hillston. Computational models in systems biology. *Genome Biol.*, 9(12):328, 2008.

[122] William F. Loomis. Cell signaling during development of dictyostelium2014. *Developmental Biology*, 391(1):1–16, 2014.

[123] Daniel López, Hera Vlamakis, and Roberto Kolter. Biofilms., 2010.

[124] T. K. Lu and J. J. Collins. Dispersing biofilms with engineered enzymatic bacteriophage. *Proceedings of the National Academy of Sciences*, 104(27):11197–11202, 2007.

[125] Markus Lumpe. *A Pi-Calculus Based Approach to Software Composition*. PhD thesis, Citeseer, 1999.

[126] Charles Macal and Michael North. Tutorial on agent-based modelling and simulation. *InProceedings of the Winter Simulation Conference*, 4(3):14–pp, 2005.

[127] Charles Macal and Michael North. Introductory tutorial: Agent-based modeling and simulation. *Proceedings - Winter Simulation Conference*, pages 6–20, 2014.

[128] Stefanía Magnúsdóttir, Almut Heinken, Laura Kutt, Dmitry A Ravcheev, Eugen Bauer, Alberto Noronha, Kacy Greenhalgh, Christian Jäger, Joanna Baginska, Paul Wilmes, Ronan M T Fleming, and Ines Thiele. Generation of genome-scale metabolic reconstructions for 773 members of the human gut microbiota. *Nature Biotechnology*, 2017.

[129] Michele A Maltz, Brian L Weiss, Michelle O'neill, Yineng Wu, and Serap Aksoy. OmpA-mediated biofilm formation is essential for the commensal bacterium Sodalis glossinidius to colonize the tsetse fly gut. *Applied and Environmental Microbiology*, 78(21):7760–7768, 2012.

[130] Julian R. Marchesi, David H. Adams, Francesca Fava, Gerben D.A. Hermes, Gideon M. Hirschfield, Georgina L. Hold, Mohammed Nabil Quraishi, James Kinross, Hauke Smidt, Kieran M. Tuohy, Linda V. Thomas, Erwin G. Zoetendal, and Ailsa Hart. The gut microbiota and host health: A new clinical frontier. *Gut*, 65(2):330–339, 2016.

[131] Adil Mardinoglu, Saeed Shoaie, Mattias Bergentall, Pouyan Ghaffari, Cheng Zhang, Erik Larsson, Fredrik Bäckhed, and Jens Nielsen. The gut microbiota modulates host amino acid and glutathione metabolism in mice. *Mol Syst Biol*, 11, 2015.

[132] Clair R Martin, Vadim Osadchiy, Amir Kalani, and Emeran A Mayer. The Brain-Gut-Microbiome Axis. *Cellular and Molecular Gastroenterology and Hepatology*, 2018.

[133] Kendle M. Maslowski and Charles R. MacKay. Diet, gut microbiota and immune responses. *Nature Immunology*, 12(1):5–9, 2011.

[134] Wayne Materi and David S. Wishart. Computational systems biology in drug discovery and development: methods and applications. *Drug Discovery Today*, 12(7-8):295–303, 2007.

[135] I S Mian and C Rose. Communication theory and multicellular biology. *Integrative Biology*, 3(4):350–367, 2011.

[136] C. Milho, M. Andrade, D. Vilas Boas, D. Alves, and S. Sillankorva. Antimicrobial assessment of phage therapy using a porcine model of biofilm infection. *International Journal of Pharmaceutics*, 557:112–123, feb 2019.

[137] Sedlacek Mj and Walker C. Antibiotic resistance in an in vitro subgingival biofilm model. *Oral Microbiol Immunol*, 22:333–339, 2007.

[138] Eugenio Moggi. *Computational lambda-calculus and monads.* University of Edinburgh, Department of Computer Science, Laboratory for∼. . . , 1989.

[139] Johannes Müller, Christina Kuttler, Burkard A Hense, Michael Rothballer, and Anton Hartmann. Cell–cell communication by quorum sensing and dimension-reduction. *Journal of mathematical biology*, 53(4):672–702, 2006.

[140] Johannes Müller, Christina Kuttler, and Burkhard A. Hense. Sensitivity of the quorum sensing system is achieved by low pass filtering. *Biosystems*, 92(1):76–81, apr 2008.

[141] Carey D. Nadell, Joao B. Xavier, and Kevin R. Foster. The sociobiology of biofilms, 2009.

[142] Masoud Najafi, Ramine Nikoukhah, and Domaine De Voluceau. Modeling and simulation of differential equations in Scicos. *The Modelica Association*, pages 177–185, 2006.

[143] Sergiu Netotea, Iris Bertani, Laura Steindler, Adam Kerenyi, Vittorio Venturi, and Sandor Pongor. A simple model for the early events of quorum sensing in Pseudomonas aeruginosa: Modeling bacterial swarming as the movement of an activation zone. *Biology Direct*, 4, 2009.

[144] John M.E. Nichols, Douwe Veltman, and Robert R. Kay. Chemotaxis of a model organism: Progress with Dictyostelium. *Current Opinion in Cell Biology*, 36:7–12, 2015.

[145] Jeremy K. Nicholson, Elaine Holmes, James Kinross, Remy Burcelin, Glenn Gibson, Wei Jia, and Sven Pettersson. Host-Gut Microbiota Metabolic Interactions. *science*, 336(6086):1262–1267, 2012.

[146] Patric Nilsson, Anna Olofsson, Magnus Fagerlind, Torbjörn Fagerström, Scott Rice, Staffan Kjelleberg, and Peter Steinberg. Kinetics of the AHL Regulatory System in a Model Biofilm System: How Many Bacteria Constitute a "Quorum"? *Journal of Molecular Biology*, 309(3):631–640, jun 2001.

[147] Ben Niu, Hong Wang, Qiqi Duan, and Li Li. Biomimicry of quorum sensing using bacterial lifecycle model. *BMC Bioinformatics*, 14(SUPPL8), 2013.

[148] Elizabeth A Novak, HanJuan Shao, Carlo Amorin Daep, and Donald R Demuth. Autoinducer-2 and QseC control biofilm formation and in vivo virulence of Aggregatibacter actinomycetemcomitans. *Infection and immunity*, 78(7):2919–2926, 2010.

[149] Peter J Olver. *Introduction to partial differential equations*. Springer, 2014.

[150] Alline R Pacheco and Vanessa Sperandio. Inter-kingdom signaling: chemical language between bacteria and host. *Current Opinion in Microbiology*, 12(2):192–198, 2009.

[151] Joachim Parrow. An Introduction to Pi Calculus. In *Handbook of process algebra*, pages 479–545. Elsevier, 2001.

[152] Marco Patrignani, Eric Mark Martin, and Dominique Devriese. On the semantic expressiveness of recursive types. *Proceedings of the ACM on Programming Languages*, 5(POPL):1–29, 2021.

[153] Mor Peleg, Iwei Yeh, and Russ B. Altman. Modelling biological processes using workflow and petri net models. *Bioinformatics*, 18(6):825–837, 2002.

[154] Ignacio Pérez Hurtado de Mendoza, David Orellana Martín, Miguel Ángel Martínez del Amor, Luis Valencia Cabrera, Agustín Riscos Núñez, and Mario de Jesús Pérez Jiménez. 11 years of p-lingua: A backward glanc. In *CMC20: 20th International Conference on Membrane Computing (2019)*. Editura BIBLIOSTAR, 2019.

[155] Judith Pérez-Velázquez, Meltem Gölgeli, and Rodolfo García-Contreras. Mathematical Modelling of Bacterial Quorum Sensing: A Review. *Bulletin of Mathematical Biology*, 78(8):1585–1639, 2016.

[156] Andrew Phillips, Luca Cardelli, and Giuseppe Castagna. A Graphical Representation for Biological Processes in the Stochastic pi-Calculus. In *InTransactions on Computational Systems Biology VII*, pages 123–152. Springer, 2006.

[157] Gordon Plotkin and Corrado Priami. *Transactions on Computational Systems Biology VI*. Springer Verlag, 2006.

[158] Susan L. Prescott. History of medicine: Origin of the term microbiome and why it matters. *Human Microbiome Journal*, 4:24–25, 2017.

[159] Corrado Priami. Stochastic Pi calculus. *The Computer Journal*, 38(7):578–589, 1995.

[160] H M Probert and G R Gibson. Bacterial biofilms in the human gastrointestinal tract. *Issues Intest. Microbiol*, 3:23–27, 2002.

[161] Frank Puhlmann and Mathias Weske. Using the *pi*-Calculus for Formalizing Workflow Patterns. *Lecture Notes in Computer Science*, pages 153–168, 2005.

[162] Andrei Păun and Gheorghe Păun. The power of communication: P systems with symport/antiport. *New Generation Computing*, 20(3):295–305, 2002.

[163] R. Randal Bollinger, Mary Lou Everett, Daniel Palestrant, Stephanie D. Love, Shu S. Lin, and William Parker. Human secretory immunoglobulin A may contribute to biofilm formation in the gut. *Immunology*, 2003.

[164] R Rojas. A Tutorial Introduction to the Lambda Calculus. *Blackwell*, 2015.

[165] Francisco José Romero-Campero and Mario J Pérez-Jiménez. Modelling gene expression control using P systems: The Lac Operon, a case study. *BioSystems*, 91(3):438–457, 2008.

[166] Elio Rossi, Annika Cimdins, Petra Lüthje, Annelie Brauner, Åsa Sjöling, Paolo Landini, and Ute Römling. "It's a gut feeling"–Escherichia coli biofilm formation in the gastrointestinal tract environment. *Critical Reviews in Microbiology*, pages 1–30, 2017.

[167] Timothy J Rudge, Paul J Steiner, Andrew Phillips, and Jim Haseloff. Computational modeling of synthetic microbial biofilms. *ACS Synthetic Biology*, 1(8):345–352, 2012.

[168] Michael A Ruggiero, Dennis P Gordon, Thomas M Orrell, Nicolas Bailly, Thierry Bourgoin, Richard C Brusca, Thomas Cavalier-Smith, Michael D Guiry, and Paul M Kirk. A higher level classification of all living organisms. *PLoS ONE*, 10(4):1–60, 2015.

[169] Nazanin Saeidi, Mohamed Arshath, Matthew Wook Chang, and Chueh Loo Poh. Characterization of a quorum sensing device for synthetic biology design: Experimental and modeling validation. *Chemical Engineering Science*, 103:91–99, 2013.

[170] Nadin S. Schaadt, Anke Steinbach, Rolf W. Hartmann, and Volkhard Helms. Rule-based regulatory and metabolic model for Quorum sensing in P. aeruginosa. *BMC Systems Biology*, 7(81):1–14, 2013.

[171] Filip Scheperjans, Velma Aho, Pedro A B Pereira, Kaisa Koskinen, Lars Paulin, Eero Pekkonen, Elena Haapaniemi, Seppo Kaakkola, Johanna Eerola-Rautio, Marjatta Pohja, Esko Kinnunen, Kari Murros, and Petri Auvinen. Gut microbiota are related to Parkinson's disease and clinical phenotype. *Movement Disorders*, 30(3):350–358, 2015.

[172] Inna Sekirov, Sl Russell, and Lcm Antunes. Gut microbiota in health and disease. *Physiological Reviews*, 90(3):859–904, 2010.

[173] Bart Selman and Carla P. Gomes. Hill-climbing Search. *Encyclopedia of cognitive science 81*, 82:333–336, 2006.

[174] Ron Sender, Shai Fuchs, and Ron Milo. Revised Estimates for the Number of Human and Bacteria Cells in the Body. *PLoS Biology*, 2016.

[175] Chaminda Jayampath Seneviratne, Cheng Fei Zhang, and Lakshman Perera Samaranayake. Dental plaque biofilm in oral health and disease. *The Chinese journal of dental research : the official journal of the Scientific Section of the Chinese Stomatological Association (CSA)*, 14(2):87–94, 2011.

[176] Gamini Seneviratne, M.L.M.A.W. Weerasekara, K.A.C.N. Seneviratne, J.S. Zavahir, M.L. Kecske, and I.R. Kennedy. Importance of Biofilm Formation in Plant Growth Promoting Rhizobacterial Action. In *Plant growth and health promoting bacteria*, pages 81–95. Springer, 2010.

[177] oliver James Shaw. *Modelling Bacterial Regulatory Networks With Petri Nets.* PhD thesis, University of Newcastle upon Tyne, 2007.

[178] Mohammad Soheilypour and Mohammad R.K. Mofrad. Agent-Based Modeling in Molecular Systems Biology. *BioEssays*, 40(7), 2018.

[179] Felix Sommer and Fredrik Bäckhed. The gut microbiota-masters of host development and physiology. *Nature Reviews Microbiology*, 11(4):227–238, 2013.

[180] Vanessa Sperandio, Alfredo G. Torres, and James B. Kaper. Quorum sensing Escherichia coli regulators B and C (QseBC): a novel two-component regulatory system involved in the regulation of flagella and motility by quorum sensing in E. coli. *Molecular Microbiology*, 43(3):809–821, 2002.

[181] Guy-Bart Stan. Modelling in Biology. *Imperial College London*, 8.8(September):1–25, 2020.

[182] Philip S. Stewart. Biophysics of biofilm infection. *Pathogens and Disease*, 70(3):212–218, 2014.

[183] Jibin Sun, Rolf Daniel, Irene Wagner-Döbler, and An Ping Zeng. Is autoinducer-2 a universal signal for interspecies communication: A comparative genomic and phylogenetic analysis of the synthesis and signal transduction pathways. *BMC Evolutionary Biology*, 4:1–11, 2004.

[184] Michael G. Surette, Melissa B. Miller, and Bonnie L. Bassler. Quorum sensing in Escherichia coli, Salmonella typhimurium, and Vibrio harveyi: A new family of genes responsible for autoinducer production. *Microbiology*, 96:1639–1644, 1999.

[185] Stephen Swift, Allan Tucker, Veronica Vinciotti, Nigel Martin, Christine Orengo, Xiaohui Liu, and Paul Kellam. Consensus clustering and functional interpretation of gene-expression data. *Genome biology*, 5(11):1–16, 2004.

[186] Kouichi Takahashi, Satya Nanda Vel Arjunan, and Masaru Tomita. Space in systems biology of signaling pathways - Towards intracellular molecular crowding in silico. *FEBS Letters*, 579(8):1783–1788, 2005.

[187] C. Täubner, B. Mathiak, A. Kupfer, N. Fleischer, and S. Eckstein. Modelling and simulation of the TLR4 pathway with coloured petri nets. In *Annual International Conference of the IEEE Engineering in Medicine and Biology - Proceedings*, pages 2009–2012, 2006.

[188] Elizabeth Thursby and Nathalie Juge. Introduction to the human gut microbiota. *The Biochemical journal*, 474(11):1823–1836, 2017.

[189] Helena Tlaskalová-Hogenová, Renata Tpánková, Hana Kozáková, Tomáš Hudcovic, Luca Vannucci, Ludmila Tuková, Pavel Rossmann, Tomá Hrní, Miloslav Kverka, Zuzana Zákostelská, Klára Klimeová, Jaroslava Pibylová, Jiina Bártová, Daniel Sanchez, Petra Fundová, Dana Borovská, Dagmar Rtková, Zdenk Zídek, Martin Schwarzer, Pavel Drastich, and David P Funda. The role of gut microbiota (commensal bacteria) and the mucosal barrier in the pathogenesis of inflammatory and autoimmune diseases and cancer: Contribution of germ-free and gnotobiotic animal models of human diseases, 2011.

[190] Valentina Tremaroli and Fredrik Bäckhed. Functional interactions between the gut microbiota and host metabolism, 2012.

[191] Nouha Bakaraki Turan, Dotse Selali Chormey, Çağdaş Büyükpınar, Güleda Onkal Engin, and Sezgin Bakirdere. Quorum sensing: Little talks for an effective bacterial coordination, 2017.

[192] Peter J Turnbaugh, Ruth E Ley, Micah Hamady, Claire M Fraser-Liggett, Rob Knight, and Jeffrey I Gordon. The Human Microbiome Project. *Nature*, 449:804, oct 2007.

[193] Conway Tyrrell and Paul S Cohen. Commensal and Pathogenic Escherichia coli Metabolism in the Gut. *Microbiol Spectr.*, 3(3), 2015.

[194] Luke K Ursell, Jessica L Metcalf, Laura Wegener Parfrey, and Rob Knight. Defining the human microbiome. *Nutrition Reviews*, 70(suppl_1):S38–S44, aug 2012.

[195] Luis Valencia-Cabrera, Miguel Á Martínez-del Amor, and Ignacio Pérez-Hurtado. A simulation workflow for membrane computing: From mecosim to pmcgpu through p-lingua. In *Enjoying Natural Computing*, pages 291–303. Springer, 2018.

[196] Han Van de Waterbeemd and Eric Gifford. ADMET in silico modelling: Towards prediction paradise? *Nature Reviews Drug Discovery*, 2(3):192–204, 2003.

[197] Wil MP Van der Aalst. Pi calculus versus Petri nets: Let us eat "humble pie" rather than further inflate the "Pi hype". *BPTrends*, 3(5):1–11, 2005.

[198] Benjamin L. Vaughan, Bryan G. Smith, and David L. Chopp. The Influence of Fluid Flow on Modeling Quorum Sensing in Bacterial Biofilms. *Bulletin of Mathematical Biology*, 2010.

[199] Björn Victor and Faron Moller. The Mobility Workbench—a tool for the π-calculus. In *International Conference on Computer Aided Verification*, pages 428–440. Springer, 1994.

[200] Estefania Vidal-Henriquez and Azam Gholami. Spontaneous center formation in Dictyostelium discoideum. *Scientific Reports*, 9(1):1–11, 2019.

[201] Alessandro Usseglio Viretta and Martin Fussenegger. Modeling the quorum sensing regulatory network of human-pathogenic Pseudomonas aeruginosa. *Biotechnology Progress*, 20(3):670–678, 2004.

[202] Nikita Vladimirov, Linda Løvdok, Dirk Lebiedz, and Victor Sourjik. Dependence of bacterial chemotaxis on gradient shape and adaptation rate. *PLoS Computational Biology*, 4(12), 2008.

[203] Matthew Walters and Vanessa Sperandio. Quorum sensing in Escherichia coli and Salmonella. *International Journal of Medical Microbiology*, 296(SUPPL. 1):125–128, 2006.

[204] Liang Wang, Yoshifumi Hashimoto, Chen-Yu Tsao, James J Valdes, and William E Bentley. Cyclic amp (camp) and camp receptor protein influence both synthesis and uptake of extracellular autoinducer 2 in escherichia coli. *Journal of bacteriology*, 187(6):2066–2076, 2005.

[205] J. P. Ward and J. R. King. Thin-film modelling of biofilm growth and quorum sensing. *Journal of Engineering Mathematics*, 2012.

[206] J. P. Ward, J. R. King, a. J. Koerber, J. M. Croft, R. E. Sockett, and P. Williams. Early development and quorum sensing in bacterial biofilms. *Journal of mathematical biology*, 47:23–55, 2003.

[207] John P Ward, John R King, Adrian J Koerber, Julie M Croft, R Elizabeth Sockett, and Paul Williams. Digital Object Identifier (Mathematical Biology Early development and quorum sensing in bacterial biofilms. *J. Math. Biol*, 47:23–55, 2003.

[208] Jumpei Washio and Nobuhiro Takahashi. Metabolomic studies of oral biofilm, oral cancer, and beyond. *International Journal of Molecular Sciences*, 17(6), 2016.

[209] Hans V Westerhoff and Bernhard O Palsson. The evolution of molecular biology into systems biology. *Nature Biotechnology*, 22(10):1249–1252, 2004.

[210] Thomas Wright and Ian Stark. The bond-calculus: A process algebra for complex biological interaction dynamics. *arXiv*, pages 1–18, 2018.

[211] Karina B Xavier and Bonnie L Bassler. Regulation of uptake and processing of the quorum-sensing autoinducer AI-2 in Escherichia coli. *Journal of bacteriology*, 187(1):238–248, 2005.

[212] Jing Yan, Andrew G Sharo, Howard A Stone, Ned S Wingreen, and Bonnie L Bassler. Vibrio cholerae biofilm growth program and architecture revealed by single-cell live imaging . *Proceedings of the National Academy of Sciences*, 113(36):E5337–E5343, 2016.

[213] Jessica S. Yu and Neda Bagheri. Agent-Based Models Predict Emergent Behavior of Heterogeneous Cell Populations in Dynamic Microenvironments. *Frontiers in Bioengineering and Biotechnology*, 8(June):1–22, 2020.

[214] Le Zhang, Zhihui Wang, Jonathan A Sagotsky, and Thomas S Deisboeck. Multiscale agent-based cancer modeling. *Journal of Mathematical Biology*, 58(4-5):545–559, 2009.

[215] Tianyu Zhang, Al Parker, Ross P Carlson, Phil S Stewart, and Isaac Klapper. Flux-Balance Based Modeling of Biofilm Communities. *bioRxiv*, page 441311, 2018.

[216] Kai Zhao, Mingzhu Liu, and Richard R. Burgess. Adaptation in bacterial flagellar and motility systems: From regulon members to 'foraging'-like behavior in E. coli. *Nucleic Acids Research*, 35(13):4441–4452, 2007.

[217] Jin Zhou, Yihua Lyu, Mindy Richlen, Donald M. Anderson, and Zhonghua Cai. Quorum sensing is a language of chemical signals and plays an ecological role in algal-bacterial interactions. *Critical reviews in plant sciences*, 35(2):81–105, 2017.

[218] Vincent Zijnge, M. Barbara M. Van Leeuwen, John E. Degener, Frank Abbas, Thomas Thurnheer, Rudolf Gmür, and Hermie J.M. Harmsen. Oral biofilm architecture on natural teeth. *PLoS ONE*, 5(2):1–9, 2010.

Appendix A

Acinetobacter baumannii

One of the most interesting bacteria in human body is *Acinetobacter baumannii*. This bacteria lives inside the body naturally but is an opportunist and may cause diseases if the circumstances allows it. As an experiment for detailed modelling, the process of quorum sensing and biofilm formation specifically in this bacteria was modelled in detail. However, since this model was never developed further and was not a part of the collection of models, it was not published as a part of the main body of this thesis. The model was built in Snoopy. The model can be found in Figure A.1.

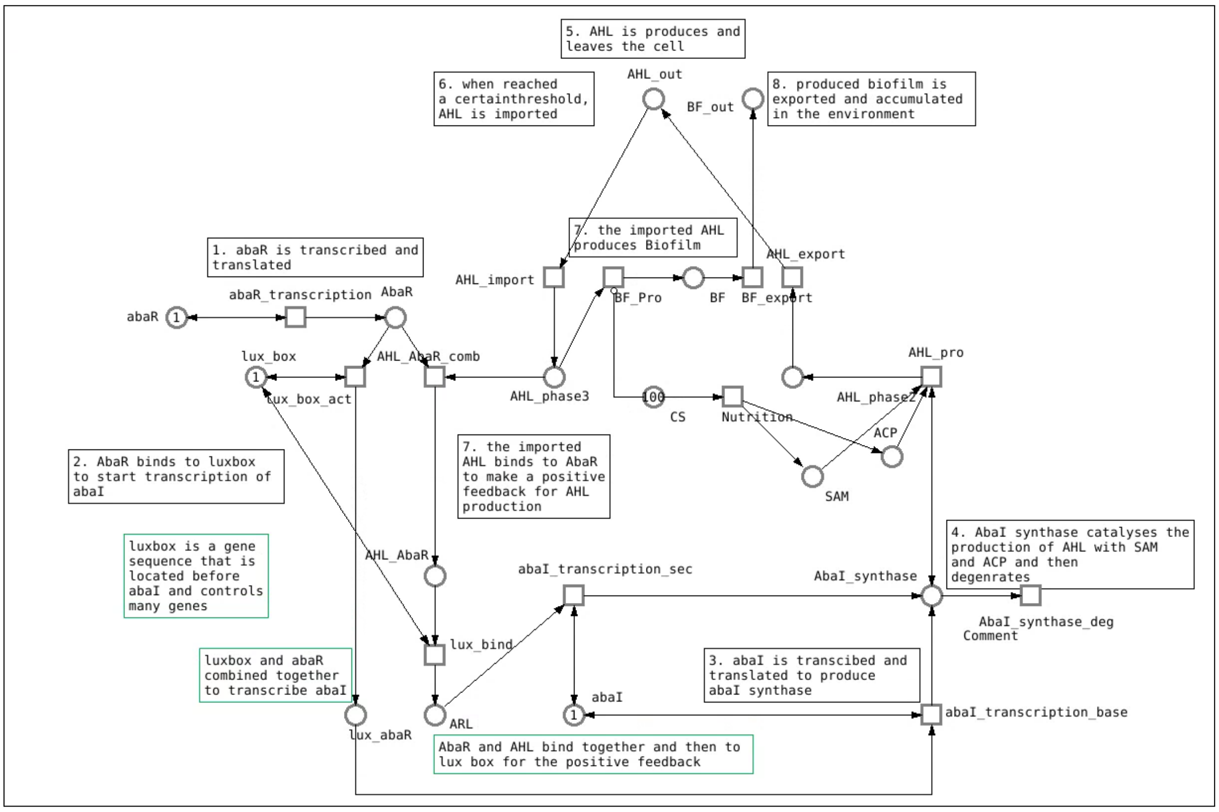

5. AHL is produces and leaves the cell
6. when reached a certainthreshold, AHL is imported
AHL_out
BF_out
8. produced biofilm is exported and accumulated in the environment
7. the imported AHL produces Biofilm
1. abaR is transcribed and translated
AHL_import
AHL_export
BF_Pro
BF
BF_export
abaR_transcription
AbaR
abaR
1
lux_box
AHL_AbaR_comb
AHL_pro
1
lux_box_act
AHL_phase3
100
AHL_phase2
CS
Nutrition
ACP
2. AbaR binds to luxbox to start transcription of abaI
7. the imported AHL binds to AbaR to make a positive feedback for AHL production
SAM
4. AbaI synthase catalyses the production of AHL with SAM and ACP and then degenrates
luxbox is a gene sequence that is located before abaI and controls many genes
AHL_AbaR
abaI_transcription_sec
AbaI_synthase
AbaI_synthase_deg
Comment
lux_bind
luxbox and abaR combined together to transcribe abaI
3. abaI is transcibed and translated to produce abaI synthase
abaI
1
lux_abaR
ARL
abaI_transcription_base
AbaR and AHL bind together and then to lux box for the positive feedback

Figure A.1: The detailed model of *Acinetobacter baumannii..* that describes the quorum sensing and biofilm formation specifically in this bacteria

Appendix B

Random Restart Hill Climbing Code

The code for Random Restart Hill Climbing developed over time. The final code is proved below:

```python
#!/usr/bin/python
import random
import glob
import os
import subprocess
import pandas as pd
import shutil
import subprocess
import csv
import numpy as np
import matplotlib.pyplot as plt
import sys
import pylab
from PIL import Image


outer_loop = str(50)
m = int(outer_loop)
inner_loop = str(10000)
n = int(inner_loop)

for j in range(0, m):
    print(j, 'the outer loop starts here')
    #create a csv file to save all the useful information
    with open('Values_index.csv', mode='w') as Values_index:
```

```
                values_writer = csv.writer(Values_index, delimiter=',',
                quotechar='"', quoting=csv.QUOTE_MINIMAL)
                values_writer.writerow(['number of itereation','status',
                'total Dicty', 'k_diff', 'k_mvmnt',
                'k_sink', 'k_pro', 'max location', 'max amount new',
                'max amount old', 'max col first row','precentage increase'])
    #create a tag for folder name
    random_f = str(random.randint(0000,9999))
    folder_name = ('HCV4.2.4' + '_' + inner_loop + '_' + random_f)

    #set the directory in a local path
    path = os.chdir(r'.')
    read_file = os.chdir(r'.') #read the files in the set directory
    #os.system('./randomspc_py.sh')
    os.system('~/randomspc_py.sh')
    myFiles_old = glob.glob('*ter.spc') #read the latest spc file
    mf = str(myFiles_old[0])
    #create new and new2 files for the next steps
    mf_new = ('new_'+mf)
    mf_new2 = ('new2_'+mf)
    cp_cmnd = ('cp' + ' ' + mf + ' ' + mf_new)
    cp_cmnd2 = ('cp' + ' ' + mf + ' ' + mf_new2)
    os.system(cp_cmnd)
    os.system(cp_cmnd2)

    #creat random numbers between 0 and 1, up to 2 numbers decimal
    rnd_num1  = str('{:.2f}'.format(round(random.uniform(0, 20),2)))
    rnd_num2  = str('{:.2f}'.format(round(random.uniform(0, 20),2)))
    rnd_num3  = str('{:.2f}'.format(round(random.uniform(0, 20),2)))
    rnd_num4  = str('{:.2f}'.format(round(random.uniform(0, 20),2)))
    print(rnd_num1, rnd_num2, rnd_num3, rnd_num4) #create and print 4 random
   numbers

    #read the new_*.spc file
    read_file = os.chdir(r'.')
    #read the new_spc file
    myFiles_old = glob.glob('new_*.spc')
    latest_file = max(myFiles_old, key=os.path.getctime)
    fin_old = open(latest_file, "r")
    fin_old_lines = fin_old.readlines()
    line17 = fin_old_lines[17]
    line18 = fin_old_lines[18]
    line19 = fin_old_lines[19]
    line20 = fin_old_lines[20]
    k_diff = line17[19:]
    k_mvmnt = line18[21:]
    k_sink = line19[19:]
    k_pro = line20[27:]
    k_diff = ''.join(k_diff.split("[["))
    k_diff = ''.join(k_diff.split(']];}'))
    k_diff = k_diff.replace('\n','')
    k_mvmnt = ''.join(k_mvmnt.split("[["))
    k_mvmnt = ''.join(k_mvmnt.split(']];}'))
    k_mvmnt = k_mvmnt.replace('\n','')
    k_sink = ''.join(k_sink.split("[["))
    k_sink = ''.join(k_sink.split(']];}'))
    k_sink = k_sink.replace('\n','')
    k_pro = ''.join(k_pro.split("[["))
    k_pro = ''.join(k_pro.split(']];}'))
    k_pro = k_pro.replace('\n','')
    print("k_strings", k_diff, k_mvmnt, k_sink, k_pro)
    #replace the Ks with the random numbers that were created earleir

    myFiles_old = glob.glob('new_*.spc')
    mf_new = myFiles_old[0]
    NewLine17 = line17.replace(k_diff, rnd_num1)
    NewLine18 = line18.replace(k_mvmnt, rnd_num2)
    NewLine19 = line19.replace(k_sink, rnd_num3)
    NewLine20 = line20.replace(k_pro, rnd_num4)
    fout = open('new_conf-sim-DictyV13_parameter.spc', "w")
    fin_old_lines[17] = NewLine17
    fin_old_lines[18] = NewLine18
    fin_old_lines[19] = NewLine19
    fin_old_lines[20] = NewLine20
    fout.writelines(fin_old_lines)
    fout.close()
    #copy new_*.spc to new2_.spc so that the files are the same.
    #new2 is changing and new 1 only changes when the rates are accepeted
    myFiles_old = glob.glob('new*.spc')
    mf_new = myFiles_old[0]
    mf_new2 = myFiles_old[1]
    cp_cmnd = ('cp' + ' ' + mf_new + ' ' + mf_new2)
    os.system(cp_cmnd)

    port = str(random.uniform(1111, 9999))
    #call spike
    #spike = ('/usr/local/bin/spike-1.6.0rc1' + ' '+ 'exe' + ' '+ '-f='+ mf_new
   + ' ' + '-port=' + port)
    spike = ('spike' + ' '+ 'exe' + ' '+ '-f='+ mf_new + ' ' + '-port=' + port)
    spike = os.system(spike) #call spike
    list_output = glob.glob('*_.csv') #list the csv files in the path
    latest_output = max(list_output, key=os.path.getctime)
    #print(latest_output)

    output = pd.read_csv(latest_output)#read the latest csv file
    output = output.loc[:, ~output.columns.str.startswith('cAMP')]
    output = output.loc[:, ~output.columns.str.startswith('Time')]
    num_di = output.iloc[0,0]
    output = output.drop(output.columns[0], axis=1)

    row100 = output.iloc[-1]
    max100 = max(row100, key=lambda x:float(x))
    maxValueIndex = row100.idxmax()
    max_colr1 = output[maxValueIndex].iloc[0]
    max_col = output[maxValueIndex]
    print('max_col is:', max_colr1)
    print('max100 is:', max100)
    pc_increase = ((max100 - max_colr1)/max_colr1)*100
    length = len(output)
    label = ("random simulation" , maxValueIndex)
    plt.plot(max_col, label = label)
    plt.xlabel('Time')
    plt.ylabel('Dicty number')
    plt.legend(loc="lower right")
    nx = str(n)
    first_run = 'first run'

    with open('Values_index.csv', mode='a') as Values_index:
        values_writer = csv.writer(Values_index, delimiter=',',
        quotechar='"', quoting=csv.QUOTE_MINIMAL)
        values_writer.writerow([nx, first_run, num_di,
        NewLine17, NewLine18, NewLine19, NewLine20,
        maxValueIndex, max100, max100, max_colr1, pc_increase])


    #Create a random spc file
    #zip up the csv files
    zipname = (folder_name + ".zip")
    zip_command = ('zip -rm' + ' ' + zipname + ' ' + '*_.csv' )
    os.system(zip_command)


    for i in range(0, n):
        print(i, 'the inner loop starts here')
        fin_new = open('new2_conf-sim-DictyV13_parameter.spc', 'r') #open new2
    spc file
        fin_new_lines = fin_new.readlines()
        ni = str(i)
        line17 = fin_new_lines[17]
        line18 = fin_new_lines[18]
        line19 = fin_new_lines[19]
        line20 = fin_new_lines[20]

        k_diff_new = line17[19:]
        k_mvmnt_new = line18[21:]
        k_sink_new = line19[19:]
        k_pro_new = line20[27:]
        k_diff_new = ''.join(k_diff_new.split("[["))
        k_diff_new = ''.join(k_diff_new.split(']];}'))
        k_diff_new = k_diff_new.replace('\n','')
        k_mvmnt_new = ''.join(k_mvmnt_new.split("[["))
        k_mvmnt_new = ''.join(k_mvmnt_new.split(']];}'))
        k_mvmnt_new = k_mvmnt_new.replace('\n','')
        k_sink_new = ''.join(k_sink_new.split("[["))
        k_sink_new = ''.join(k_sink_new.split(']];}'))
        k_sink_new = k_sink_new.replace('\n','')
        k_pro_new = ''.join(k_pro_new.split("[["))
        k_pro_new = ''.join(k_pro_new.split(']];}'))
        k_pro_new = k_pro_new.replace('\n','')

        diff_float = float(k_diff_new)
        mvmnt_float = float(k_mvmnt_new)
        sink_float = float(k_sink_new)
        pro_float = float(k_pro_new)

        MaxDelta = 10
        g = 5
        #get a random number up to 2 decimals between 0 and MaxDelta
```

```python
    delta1 = '{:.2f}'.format(round(random.uniform(0, MaxDelta),2) - g)
    delta2 = '{:.2f}'.format(round(random.uniform(0, MaxDelta),2) - g)
    delta3 = '{:.2f}'.format(round(random.uniform(0, MaxDelta),2) - g)
    delta4 = '{:.2f}'.format(round(random.uniform(0, MaxDelta),2) - g)

    #adding delta to the current values and round the numbers to 2 decimals
    sml_chng1 = '{:.2f}'.format(eval(str((round(diff_float, 2))) + "+" + delta1))
    sml_chng2 = '{:.2f}'.format(eval(str((round(mvmnt_float, 2))) + "+" + delta2))
    sml_chng3 = '{:.2f}'.format(eval(str((round(sink_float, 2))) + "+" + delta3))
    sml_chng4 = '{:.2f}'.format(eval(str((round(pro_float, 2))) + "+" + delta4))

    while str(sml_chng1) <= str(0):
        delta1 = '{:.2f}'.format(round(random.uniform(0, MaxDelta),2) - g)
        sml_chng1 = '{:.2f}'.format(eval(str((round(diff_float, 2))) + "+" + delta1))
    while str(sml_chng2) <= str(0):
        delta2 = '{:.2f}'.format(round(random.uniform(0, MaxDelta),2) - g)
        sml_chng2 = '{:.2f}'.format(eval(str((round(mvmnt_float, 2))) + "+" + delta2))
    while str(sml_chng3) <= str(0):
        delta3 = '{:.2f}'.format(round(random.uniform(0, MaxDelta),2) - g)
        sml_chng3 = '{:.2f}'.format(eval(str((round(sink_float, 2))) + "+" + delta3))
    while str(sml_chng4) <= str(0):
        delta4 = '{:.2f}'.format(round(random.uniform(0, MaxDelta),2) - g)
        sml_chng4 = '{:.2f}'.format(eval(str((round(pro_float, 2))) + "+" + delta4))

    NewLine17 = line17.replace(k_diff_new, sml_chng1)
    NewLine18 = line18.replace(k_mvmnt_new, sml_chng2)
    NewLine19 = line19.replace(k_sink_new, sml_chng3)
    NewLine20 = line20.replace(k_pro_new, sml_chng4)
    NewLines_new = (NewLine17, NewLine18, NewLine19, NewLine20)

    fin_new = glob.glob('new2_*.spc') #read the latest spc file
    latest_file = max(fin_new, key=os.path.getctime) #get the latest file in the fol
    fout_new = open(latest_file, "w")
```

```
        fin_new_lines[17] = NewLine17 #write the new files in new2
        fin_new_lines[18] = NewLine18
        fin_new_lines[19] = NewLine19
        fin_new_lines[20] = NewLine20
        fout_new.writelines(fin_new_lines)
        fout_new.close()

        #call spike for the new file
        myFiles = glob.glob('new2*.spc')
        mf_new2 = myFiles[0]
        #port = str(random.uniform(1111, 9999))

        #spike = ('/usr/local/bin/spike-1.6.0rc1' + ' '+ 'exe' + ' '+ '-f='+
    mf_new2 + ' ' + '-port='+ port)
        spike = ('spike' + ' '+ 'exe' + ' '+ '-f='+ mf_new + ' ' + '-port=' +
    port)
        spike2 = os.system(spike)

        list_output2 = glob.glob('*_.csv')
        latest_output2 = max(list_output2, key=os.path.getctime)
        output2 = pd.read_csv(latest_output2)
        output2 = output2.loc[:, ~output2.columns.str.startswith('cAMP')]
        output2 = output2.loc[:, ~output2.columns.str.startswith('Time')]
        num_di = output2.iloc[0,0]
        output2 = output2.drop(output2.columns[0], axis=1)
        row100_new = output2.iloc[-1]## Extract the 100th row
        print(row100_new)
        max100_new = max(row100_new, key=lambda x:float(x))
        print(max100_new)
        maxValueIndex_new = row100_new.idxmax()
        max_colr1_new = output[maxValueIndex_new].iloc[0]
        max_col_new = output[maxValueIndex_new]
        pc_increase_new = ((max100_new - max_colr1_new)/max_colr1_new)*100
        length = len(output2)


        if max100 < max100_new:
            read_file = os.chdir(r'.')
            #copy the new file to the old file
            original = r'new2_conf-sim-DictyV13_parameter.spc'
            target = r'new_conf-sim-DictyV13_parameter.spc'
            shutil.copy(original, target) #copy the spc file into *new_spc
            with open('Values_index.csv', mode='a') as Values_index:
                    values_writer = csv.writer(Values_index, delimiter=',',
                    quotechar='"', quoting=csv.QUOTE_MINIMAL)
                    values_writer.writerow([ni, 'accepted', num_di, sml_chng1,
                    sml_chng2, sml_chng3, sml_chng4,
                    maxValueIndex_new, max100_new, max100, max_colr1_new, pc_increase_new])
            max100 = max100_new #add the information into the csv file


        elif max100 >= max100_new:
            with open('Values_index.csv', mode='a') as Values_index:
                    values_writer = csv.writer(Values_index, delimiter=',',
                    quotechar='"', quoting=csv.QUOTE_MINIMAL)
                    values_writer.writerow([ni, 'rejected', num_di,
                    sml_chng1, sml_chng2, sml_chng3, sml_chng4,
                    maxValueIndex_new, max100_new, max100,
                    max_colr1_new, pc_increase_new]) #add the information to the csv file
                    pass #reject the new rates and continue with the previous rates
        zipname = (folder_name + ".zip")
        zip_command = ('zip -rm' + ' ' + zipname + ' ' + '*_.csv' )
        os.system(zip_command)


    mkdir = ("mkdir" + ' ' + folder_name)
    os.system(mkdir)
    mv_zip = ("mv" + ' ' + zipname + ' ' + folder_name + '/')
    os.system(mv_zip)
    mv_index = ("mv" + ' ' + 'Values_index.csv' + ' ' + folder_name + '/')
    os.system(mv_index)
    mv_jpg = ("mv" + ' ' + '*.jpg' + ' ' + folder_name + '/')
    os.system(mv_jpg)
    #mv_spc = ("mv" + ' ' + 'new*.spc' + ' ' + folder_name + '/')

sys.exit()
```

Appendix C

Simulated Annealing Code

The code used for Simulated Annealing is as below. The main reference for this code is [185]:

```
#!/usr/bin/python
import random
import glob
import os
import subprocess
import pandas as pd
import shutil
import subprocess
import csv
import numpy as np
import matplotlib.pyplot as plt
import sys
import pylab
from PIL import Image
import math

m = int(50000)
t0 = 100
c = 0.99994
Titer = t0 * (c)**m

Temp = str(t0)

folder_name = ('V6' + '_' + Runs + '_' + Temp)

```

```
T = t0

with open('SA_ValuesIndex.csv', mode='w') as Values_index:
            values_writer = csv.writer(Values_index, delimiter=',',
            quotechar='"', quoting=csv.QUOTE_MINIMAL)
            values_writer.writerow(['number of iteration','status',
            'k_diff', 'k_mvmnt', 'k_sink', 'k_pro',
                'max location', 'max amount new', 'max amount old',
                'best solution', 'max col first row','precentage increase',
                'Dicty on Grid', 'Temperature', 'Fitness diff', 'PA', 'Rand'])

mkdir = ("mkdir" + ' ' + folder_name)
os.system(mkdir)

#set the directory in a local path
path = os.chdir(r'.')
#Create a random spc file
os.system('~/randomspc_py.sh')
#Read the newly created spc file = mother file
myFiles_old = glob.glob('*.spc')
mf = str(myFiles_old[0])
#create new and new2 files for the next steps
mf_new = ('new_'+mf)
mf_new2 = ('new2_'+mf)
cp_cmnd = ('cp' + ' ' + mf + ' ' + mf_new)
cp_cmnd2 = ('cp' + ' ' + mf + ' ' + mf_new2)
os.system(cp_cmnd)
os.system(cp_cmnd2)

#creat random numbers between 0 and 20, up to 2 numbers decimal
rnd_num1  = str('{:.2f}'.format(round(random.uniform(0, 20),2)))
rnd_num2  = str('{:.2f}'.format(round(random.uniform(0, 20),2)))
rnd_num3  = str('{:.2f}'.format(round(random.uniform(0, 20),2)))
rnd_num4  = str('{:.2f}'.format(round(random.uniform(0, 20),2)))
print(rnd_num1, rnd_num2, rnd_num3, rnd_num4)

#read the new_*.spc file
read_file = os.chdir(r'.')
#read the new_spc file
myFiles_old = glob.glob('new_*.spc')
latest_file = max(myFiles_old, key=os.path.getctime)
fin_old = open(latest_file, "r")
```

```
fin_old_lines = fin_old.readlines() #

#Read the required lines: 17, 18, 19, 20
line17 = fin_old_lines[17]
line18 = fin_old_lines[18]
line19 = fin_old_lines[19]
line20 = fin_old_lines[20]
#read the required parts of the line [Ks]
k_diff = line17[19:]
k_mvmnt = line18[21:]
k_sink = line19[19:]
k_pro = line20[27:]
k_diff = ''.join(k_diff.split("[[")) #finde the beginning of number
k_diff = ''.join(k_diff.split(']];}')) #take the unwanted chr away
k_diff = k_diff.replace('\n','')
k_mvmnt = ''.join(k_mvmnt.split("[["))
k_mvmnt = ''.join(k_mvmnt.split(']];}'))
k_mvmnt = k_mvmnt.replace('\n','')
k_sink = ''.join(k_sink.split("[["))
k_sink = ''.join(k_sink.split(']];}'))
k_sink = k_sink.replace('\n','')
k_pro = ''.join(k_pro.split("[["))
k_pro = ''.join(k_pro.split(']];}'))
k_pro = k_pro.replace('\n','')
#print("k_strings", k_diff, k_mvmnt, k_sink, k_pro) #read the lines

#replace the Ks with the random numbers that were created earleir
myFiles_old = glob.glob('new_*.spc')
mf_new = myFiles_old[0]
NewLine17 = line17.replace(k_diff, rnd_num1)
NewLine18 = line18.replace(k_mvmnt, rnd_num2)
NewLine19 = line19.replace(k_sink, rnd_num3)
NewLine20 = line20.replace(k_pro, rnd_num4)

#write the new lines in the new)*.spc file
fout = open(mf_new, "w")
fin_old_lines[17] = NewLine17
fin_old_lines[18] = NewLine18
fin_old_lines[19] = NewLine19
fin_old_lines[20] = NewLine20
fout.writelines(fin_old_lines)
fout.close()
```

```

#copy new_*.spc to new2_.spc so that the files are the same.
#new2 is changing and new 1 only changes when the rates are accepeted
myFiles_old = glob.glob('new*.spc')
mf_new = myFiles_old[0]
mf_new2 = myFiles_old[1]
cp_cmnd = ('cp' + ' ' + mf_new + ' ' + mf_new2)
os.system(cp_cmnd)
port = str(random.uniform(1111, 9999))
#call spike
spike = ('spike' + ' '+ 'exe' + ' '+ '-f='+ mf_new + ' ' + '-port=' + port)
spike = os.system(spike) #call spike

#list the csv files in the path
list_output = glob.glob('*.csv')
#get the latest csv file
latest_output = max(list_output, key=os.path.getctime)

#read the csv file, remove extra unwanted cols
output = pd.read_csv(latest_output)
output = output.loc[:, ~output.columns.str.startswith('cAMP')]
output = output.loc[:, ~output.columns.str.startswith('Time')]
num_di = output.iloc[0,0]
output = output.drop(output.columns[0], axis=1)

## Extract the last row
row100 = output.iloc[-1]
#find the max in the last row
max100 = max(row100, key=lambda x:float(x))
#find the col in which maximum exists
maxValueIndex = row100.idxmax()
max_colr1 = output[maxValueIndex].iloc[0]
max_col = output[maxValueIndex]

#set the first solution as the best solution
best_sol = max100

#check the percentage increase in the max col
pc_increase = ((max100 - max_colr1)/max_colr1)*100
length = len(output)

#zip up the csv files
```

```
zipname = (folder_name + ".zip")
zip_command = ('zip -rm' + ' ' + zipname + ' ' + '*_.csv' )
os.system(zip_command)


#update the Values index file
with open('SA_ValuesIndex.csv', mode='a') as Values_index:
    values_writer = csv.writer(Values_index, delimiter=',',
    quotechar='"', quoting=csv.QUOTE_MINIMAL)
    values_writer.writerow(['first', 'new_sim', k_diff, k_mvmnt,
    k_sink, k_pro, maxValueIndex, max100,
        max100, best_sol ,max_colr1, pc_increase, num_di, T])

for x in range (1, m):
    #read new2_.spc file
    fin_new = glob.glob('new2_*.spc')
    latest_file = max(fin_new, key=os.path.getctime)
    fin_new = open(latest_file, "r")
    fin_new_lines = fin_new.readlines()
    line17 = fin_new_lines[17]
    line18 = fin_new_lines[18]
    line19 = fin_new_lines[19]
    line20 = fin_new_lines[20]
    k_diff_new = line17[19:]
    k_mvmnt_new = line18[21:]
    k_sink_new = line19[19:]
    k_pro_new = line20[27:]
    k_diff_new = ''.join(k_diff_new.split("[["))
    k_diff_new = ''.join(k_diff_new.split(']];}'))
    k_diff_new = k_diff_new.replace('\n','')
    k_mvmnt_new = ''.join(k_mvmnt_new.split("[["))
    k_mvmnt_new = ''.join(k_mvmnt_new.split(']];}'))
    k_mvmnt_new = k_mvmnt_new.replace('\n','')
    k_sink_new = ''.join(k_sink_new.split("[["))
    k_sink_new = ''.join(k_sink_new.split(']];}'))
    k_sink_new = k_sink_new.replace('\n','')
    k_pro_new = ''.join(k_pro_new.split("[["))
    k_pro_new = ''.join(k_pro_new.split(']];}'))
    k_pro_new = k_pro_new.replace('\n','')
    diff_float = float(k_diff_new)
    mvmnt_float = float(k_mvmnt_new)
    sink_float = float(k_sink_new)
    pro_float = float(k_pro_new)

    mx = str(x)

    #get a random number up to 2 decimals between 0 and MaxDelta
    MaxDelta = 10
    g = 5
    delta1 = '{:.2f}'.format(round(random.uniform(0, MaxDelta),2) - g)
    delta2 = '{:.2f}'.format(round(random.uniform(0, MaxDelta),2) - g)
    delta3 = '{:.2f}'.format(round(random.uniform(0, MaxDelta),2) - g)
    delta4 = '{:.2f}'.format(round(random.uniform(0, MaxDelta),2) - g)

    #small change
    sml_chng1 = '{:.2f}'.format(eval(str((round(diff_float, 2))) + "+" + delta1))
    #adding delta to the current values if smlchange is less than zero
    while str(sml_chng1) <= str(0):
        delta1 = '{:.2f}'.format(round(random.uniform(0, MaxDelta),2) - g)
        sml_chng1 = '{:.2f}'.format(eval(str((round(diff_float, 2))) + "+" + delta1))

    sml_chng2 = '{:.2f}'.format(eval(str((round(mvmnt_float, 2))) + "+" + delta2))
    #and round the numbers so there is no more than 2 decimals
    while str(sml_chng2) <= str(0):
        delta2 = '{:.2f}'.format(round(random.uniform(0, MaxDelta),2) - g)
        sml_chng2 = '{:.2f}'.format(eval(str((round(mvmnt_float, 2))) + "+" + delta2))

    sml_chng3 = '{:.2f}'.format(eval(str((round(sink_float, 2))) + "+" + delta3))
    while str(sml_chng3) <= str(0):
        delta3 = '{:.2f}'.format(round(random.uniform(0, MaxDelta),2) - g)
        sml_chng3 = '{:.2f}'.format(eval(str((round(sink_float, 2))) + "+" + delta3))

    sml_chng4 = '{:.2f}'.format(eval(str((round(pro_float, 2))) + "+" + delta4))
    while str(sml_chng4) <= str(0):
        delta4 = '{:.2f}'.format(round(random.uniform(0, MaxDelta),2) - g)
        sml_chng4 = '{:.2f}'.format(eval(str((round(pro_float, 2))) + "+" + delta4))

    #write the new lines
    NewLine17 = line17.replace(k_diff_new, sml_chng1)
    NewLine18 = line18.replace(k_mvmnt_new, sml_chng2)
    NewLine19 = line19.replace(k_sink_new, sml_chng3)
    NewLine20 = line20.replace(k_pro_new, sml_chng4)

    fin_new = glob.glob('new2_*.spc')
    latest_file = max(fin_new, key=os.path.getctime)
    fout_new = open(latest_file, "w")
    fin_new_lines[17] = NewLine17
    fin_new_lines[18] = NewLine18
    fin_new_lines[19] = NewLine19
    fin_new_lines[20] = NewLine20
    fout_new.writelines(fin_new_lines)
    fout_new.close()

    #call spike for the new file
    myFiles = glob.glob('new2*.spc')
    mf_new2 = myFiles[0]
    spike = ('spike' + ' '+ 'exe' + ' '+ '-f='+ mf_new2 + ' ' + '-port='+ port)
    spike2 = os.system(spike)

    #check the max from the new csv file
    list_output2 = glob.glob('*.csv')
    latest_output2 = max(list_output2, key=os.path.getctime)
    output2 = pd.read_csv(latest_output2)
    output2 = output2.loc[:, ~output2.columns.str.startswith('cAMP')]
    output2 = output2.loc[:, ~output2.columns.str.startswith('Time')]
    num_di = output2.iloc[0,0]
    output2 = output2.drop(output2.columns[0], axis=1)
    row100_new = output2.iloc[-1]## Extract the last row
    max100_new = max(row100_new, key=lambda x:float(x))
    maxValueIndex_new = row100_new.idxmax()
    max_colr1_new = output[maxValueIndex_new].iloc[0]
    max_col_new = output[maxValueIndex_new]
    pc_increase_new = ((max100_new - max_colr1_new)/max_colr1_new)*100
    length = len(output2)
    os.system(zip_command)

    #fitness
    s = max100
    s1 = max100_new
    def fitness(s):
        R = max100 * math.exp(max100)
        res = abs(s * math.exp(s) - R)
        return(res)
    f_old = fitness(s)
    f_new = fitness(s1)


    df = abs(f_old - f_new)
    print('df = ', df)

    print(x)

    if T > Titer and best_sol < max100_new:
        myFiles_old = glob.glob('new*.spc')
        mf_new = myFiles_old[0]
        mf_new2 = myFiles_old[1]
        cp_cmnd = ('cp' + ' ' + mf_new2 + ' ' + mf_new)
        os.system(cp_cmnd)
        best_sol = max100_new
        max100 = best_sol

        #copy the spc file into *new_spc
        with open('SA_ValuesIndex.csv', mode='a') as Values_index:
                values_writer = csv.writer(Values_index, delimiter=',',
                quotechar='"', quoting=csv.QUOTE_MINIMAL)
                values_writer.writerow([mx, 'accepted', sml_chng1,
                sml_chng2, sml_chng3, sml_chng4,
                    maxValueIndex_new, max100_new, max100, best_sol,
max_colr1_new,
                    pc_increase_new, num_di, T, df])


    elif T > Titer and best_sol >= max100_new:
        #lower T, Higher P
        e = 2.71828
        p = e ** -(math.exp(-df/T))
        print('p=', p)
        PA = p
        rand = random.random()
        #print('random possibility=', rand)
        #rand = 0.9
        RA = rand
        if p >= rand:
            myFiles_old = glob.glob('new*.spc')
            mf_new = myFiles_old[0]
            mf_new2 = myFiles_old[1]
            cp_cmnd = ('cp' + ' ' + mf_new2 + ' ' + mf_new)
            os.system(cp_cmnd)
            max100 = max100_new

            #copy the spc file into *new_spc
            with open('SA_ValuesIndex.csv', mode='a') as Values_index:
                values_writer = csv.writer(Values_index, delimiter=',',
                quotechar='"', quoting=csv.QUOTE_MINIMAL)
                values_writer.writerow([mx, 'AP high', sml_chng1,
                sml_chng2, sml_chng3, sml_chng4,
                    maxValueIndex_new, max100_new, max100, best_sol,
    max_colr1_new,
                    pc_increase_new, num_di, T, df, PA, RA])
            max100 = max100_new


        else:
           with open('SA_ValuesIndex.csv', mode='a') as Values_index:
                values_writer = csv.writer(Values_index, delimiter=',',
                quotechar='"', quoting=csv.QUOTE_MINIMAL)
                values_writer.writerow([mx, 'rejected', sml_chng1, sml_chng2,
                sml_chng3, sml_chng4,
                    maxValueIndex_new, max100_new, max100, best_sol,
                    max_colr1_new, pc_increase_new, num_di, T, df, PA, RA])
                #add the information into the csv file
           pass

    if T <= Titer:
        print("Break of the simulation: Reached the minimum Teperature: ", Titer
    )
        break

    T = c*T
    print('new temperature=', T)


mv_zip = ("mv" + ' ' + zipname + ' ' + folder_name + '/')
```

```
os.system(mv_zip)
mv_index = ("mv" + ' ' + 'SA_ValuesIndex.csv' + ' ' + folder_name + '/')
os.system(mv_index)



sys.exit()
```