Computational Aspects of Cellular Intelligence and Their Role in Artificial Intelligence

A thesis submitted to

The Robert Gordon University

in partial fulfilment of the requirements for

the degree of Doctor of Philosophy

Claire E. Gerrard

IDEAS Research Institute

School of Computing

Robert Gordon University

Aberdeen, Scotland, July 2014

Declaration

I hereby declare that this thesis is a record of work undertaken by myself. That it has not been the subject of any previous application for a degree and that all sources of information have been duly acknowledged.

Claire Gerrard, 2014

Dedication

I dedicate this thesis to Christopher who showed me why "*a person starts to truly live when they can live outside themselves*".

C. Gerrard 12:02:14

Acknowledgements

Firstly, I am indebted to my supervisory team: John McCall, George Coghill and Christopher Macleod for their help, support and encouragement throughout the project. Without their supervisory skills the project would have been much more difficult and not nearly as worthwhile.

A special note of thanks is due to Geoff MacAngus-Gerrard and to Christopher Macleod who assisted in proof reading the thesis and made many valuable suggestions to the content. Their efforts helped beyond measure to improve the quality and readability of this work.

I would also like to thank Professor Thomas Gregor at the Laboratory for the Physics for Life at Princeton University for allowing me to include screenshots of aggregating amoebae in the text.

I would like to extend my appreciation to all the members of the Robert Gordon University, School of Computing who assisted me in any way during the project. In particular, I would like to mention my appreciation to the PhD students within the Computational Intelligence Group who offered excellent advice, support and encouragement throughout the project. Finally, I would also like to thank my friends, family and my daughter Meredith for their infinite patience while travelling with me on the PhD roller-coaster ride.

C. Gerrard 12:02:14

Abbreviations

To avoid repetition of certain common terms, the following abbreviations are used in the text. These abbreviations will also be introduced before use in the text.

ABN - Artificial Biochemical Network.

AC- Artificial Chemistry

ACC - Artificial Chemical Computing

AES - Artificial Endocrine Systems

AGRN - Artificial Gene Regulatory Network

AHHS - Artificial Homeostatic Hormone System

AHoS - Artificial Hormone System

AI - Artificial Intelligence

AIS - Artificial Immune System

A-Life - Artificial Life

ANN - Artificial Neural Network

ARBN - Asynchronous RBN

ARN - Artificial Reaction Network

BN - Bayesian Network

BP - Backpropagation.

CCM - Chemical Casting Model

CHAM - Chemical Abstract Machines

CI - Cellular Intelligence

CNS - Central Nervous System

CPG - Central Pattern Generator

CRBN - Classical RBN

CSN - Cell Signalling Network.

DARBN - Deterministic asynchronous RBN

Dd; D. discoideum; Dicteosteylium discoideum

DHS - Digital Hormone System

DNA- Deoxyribonucleic acid

DOF - Degree of freedom

E. coli - Escherichia coli

EA - Evolutionary Algorithm

FF- Feedforward*; FB* -Feedback.

FPN - Functional PN

GA - Genetic Algorithm

GRN - Gene Regulatory Network

MAPK - Mitogen Activated Protein Kinase

MC - Markov chain

MLP - Multi Layer Perceptron

ODE - Ordinary Differential Equation

P. polycephalum - Physarum polycephalum

PC - Process Calculi

PDE - Partial Differential Equation

PN - Petri Net

PNS - Peripheral Nervous System

PSO - Particle Swarm Optimisation

RBN - Random Boolean Network

RNA - Ribonucleic acid.

SSA - Stochastic Simulation Algorithm

STC - Signal Transduction Cascades

TPN - Time Petri Net

UML - Unified Modelling Language

Abstract

The work presented in this thesis is concerned with an exploration of the computational aspects of the primitive intelligence associated with single-celled organisms. The main aim is to explore this “Cellular Intelligence” and its role within Artificial Intelligence.

The findings of an extensive literature search into the biological characteristics, properties and mechanisms associated with Cellular Intelligence, its underlying machinery - “Cell Signalling Networks” and the existing computational methods used to capture it are reported. The results of this search are then used to fashion the development of a versatile new connectionist representation, termed the Artificial Reaction Network (ARN). The ARN belongs to the branch of Artificial Life known as Artificial Chemistry and has properties in common with both Artificial Intelligence and Systems Biology techniques, including: Artificial Neural Networks, Artificial Biochemical Networks, Gene Regulatory Networks, Random Boolean Networks, Petri Nets, and S-Systems.

The thesis outlines the following original work: The ARN is used to model the chemotaxis pathway of *Escherichia coli* and is shown to capture emergent characteristics associated with this organism and Cellular Intelligence more generally. The computational properties of the ARN and its applications in robotic control are explored by combining functional motifs found in biochemical network to create temporal changing waveforms which control the gaits of limbed robots. This system is then extended into a complete control system by combining pattern recognition with limb control in a single ARN. The results show that the ARN can offer increased flexibility over existing methods.

Multiple distributed cell-like ARN based agents termed “Cytobots” are created. These are first used to simulate aggregating cells based on the slime mould *Dictyostelium discoideum.* The Cytobots are shown to capture emergent behaviour arising from multiple stigmergic interactions. Applications of Cytobots within swarm robotics are investigated by applying them to benchmark search problems and to the task of cleaning up a simulated oil spill. The results are compared to those of established optimization algorithms using similar cell inspired strategies, and to other robotic agent strategies.

Consideration is given to the advantages and disadvantages of the technique and suggestions are made for future work in the area.

The report concludes that the Artificial Reaction Network is a versatile and powerful technique which has application in both simulation of chemical systems, and in robotic control, where it can offer a higher degree of flexibility and computational efficiency than benchmark alternatives. Furthermore, it provides a tool which may possibly throw further light on the origins and limitations of the primitive intelligence associated with cells.

Contents

1. Introduction

This chapter outlines the background of the project and introduces its aims and objectives. It also briefly outlines the original contributions to the art made in the research - which will be described in detail in the following chapters.

1.1 Research Background

The natural world is rich in examples of elegant solutions to difficult problems. An important instance of this is found in the multitude of single-celled organisms which display an astonishing array of complex behaviours. Some can avoid light using photo-sensitive spots; some actively hunt prey; while others can even build protective shelters (*Ford 2009*). Such behaviours improved these organisms' chances of survival and arose through the process of natural selection. The behaviour of such simple organisms may be labelled as "Cellular Intelligence" (CI) (*Quevli 1916; Ford 2009*).

In recent years, a growing body of research has explored these behaviours and illuminated the remarkable capabilities of single cells to store and process information (*Bray 1995; Bhalla 2003; Ford 2009; Saigusa 2008)*. Such research is inspired by an interest in the origins of CI and the mechanisms of its biological action. The exploration of these has direct benefit to humans, for example in medicine and ecology, and is also of deeper philosophical interest.

The physical mechanisms involved are quite different from those of a digital computer. Within a cell, the current state is represented as a set of spatially distributed concentrations of chemical species. This data is processed by networks of chemical reactions, termed "Cell Signalling Networks" (CSNs). In this way, cells are able to respond to current environmental conditions, communicate with other cells, and perform internal self-maintenance operations. As a result of work into understanding these systems, several researchers have highlighted the processing capabilities of such networks (*Arkin and Ross 1994; Bray 1995; Wang et al. 2011)* and their similarities to Artificial Neural Networks (ANNs) (*Bray 1995; Bhalla 2003*). For example, it has been demonstrated that they can perform Boolean and Fuzzy Logic functions and are equivalent to a Turing Machine (*Bray 1995; Wang et al. 2011*). Furthermore, many CSNs contain topological features such as feedback loops, hierarchical and modular structures and

multi-level interconnectivity, and can thus form highly complex systems (*Bray 1995; Kholodenko 2006*).

Biological systems such as these have inspired many powerful Artificial Intelligence (AI) techniques- for example: Genetic Algorithms (GAs) (*Goldberg and Holland 1988*), Particle Swarm Optimization (PSO) (*Woolard and Fieldsend 2013),* and Ant Colony Optimization (ACO) (*Dorigo et al. 2006*). Some of these approaches are based on properties of biochemical networks - for example: Artificial Biochemical Networks (ABNs) (*Macleod and Capanni 2010*) and Artificial Immune Systems (AIS) (*Aickelin et al. 2014*).

One such bio-inspired approach involves abstracting the computational properties of chemical processing to create a model called an Artificial Chemistry. Artificial Chemistry is a subfield of Artificial Life (A-Life) (*Steels 1993*) - and in its broadest sense, it describes man-made systems, which behave similarly to real chemical systems (*Dittrich et al. 2001*).

The overall aim of this project was to investigate the modelling of Cellular Intelligence (CI) in general and more specifically its role in AI. The first stage of this was to identify the properties and mechanisms of such intelligence, and the existing computational methods used to represent it. The results of this study led to the creation of a new Artificial Chemistry (AC) representation, termed the Artificial Reaction Network (ARN). Rather than focus on micro-molecular detail, the ARN aims to capture emergent behaviours arising from a network of chemical reactions. The ARN was first validated against standard biological data, and then successfully applied to simulate CI associated with the well-characterized CSN of *Escherichia coli* chemotaxis. The computational features of the ARN, including its temporal and chaotic dynamics were explored and its operation was contrasted with relevant techniques in both Systems biology and in AI.

Having proved capable of representing the important mechanisms of CI, the ARN provided a platform to explore the role of CI in AI.

CSNs can combine sensory input, motor output, self-maintenance and adaptation in one highly sophisticated control system. Thus, one obvious potential AI application of the ARN was in robotic control. Such applications were initially examined by using a single ARN network to produce complex waveforms to control a variety of gaits in a limbed robot. This network was then extended and used to create a complete control system by combining a sensory input and motor output. The results show that the ARN offers advantages in flexibility over current connectionist AI methods. Further applications in AI were then explored by developing simulated cell-like agents termed “Cytobots”. Similarly to the control system of a cell, a

Cytobot has its own internal ARN which coordinates sensory and motor tasks. This allows it to communicate stigmergically with other Cytobots, receive environmental information which it uses to update its current state and trajectory and, in this way, generate high-level behaviour. By creating multiple distributed agents within artificial environments, it was possible to explore emergent properties similar to those of interacting cells. The ability of the Cytobots' to capture such characteristics was explored by creating a simulation of aggregating cells of the slime mould *D. discoideum.* The results demonstrated the agents' ability to capture emergent behaviour arising from such interactions. These results also highlight the potential to use Cytobots in the simulation of interacting cells.

The ability of the Cytobots to perform useful behaviours in situations similar to those in real world environments was demonstrated by a number of other simulations. Interacting groups of Cytobots were applied to the task of finding the global minima in several benchmark search problems. In each case, they found the global minima with similar capacity to that of established cell inspired optimisation algorithms. In a final simulation, the Cytobots were applied to the task of cleaning up a simulated oil spill. The results show that they have potential application in swarm robotics, where they can provide useful autonomous behaviour within real world environments.

1.2 Aims, Objectives & Research Questions

In this section the research questions, aim and objectives of the thesis are listed. The research questions were used to construct the overall aim of the research. This aim was then broken down into the six stated objectives. In the final chapter of the thesis, the accomplishment of each of these objectives is discussed.

1.2.1 Research Questions

The work reported in the thesis will address the research questions described below.

1. Can we abstract the computational aspects of Cell Signalling Networks which are responsible for generating Cell Intelligence into a simplified mathematical representation?

2. How accurately can such a model represent biological Cell Signalling Networks and how does such a model compare against standard Systems Biology approaches to modelling biological systems?

3. Is such a model capable of generating the emergent behaviour associated with Cellular Intelligence?

4. Can such a model be used to create useful bio-inspired AI?

5. Does such a method offer additional functionality or improvements to existing methods?

1.3 Overall Aim and Objectives

The research questions can be summarised by the following overall research aim.

> *To develop a representation of the computational features from which Cell Intelligence arises and to investigate its application within Artificial Intelligence.*

1.3.1 Objectives

This aim can be broken down into the objectives described below.

1. Produce a computational representation of biological Cell Signalling Networks that captures the underlying properties and mechanisms of Cell Intelligence.

2. Evaluate the biological accuracy of the new representation.

3. Explore the computational capabilities, pattern recognition and temporal dynamics of the new representation.

4. Evaluate the ability of the new representation to capture characteristics of Cell Intelligence arising from single cells.

5. Evaluate the ability of the new representation to capture Cell Intelligence arising from multiple interacting cells.

6. Identify the applications of the new representation within AI and compare them with existing methods

1.4 Original Contributions to the Art

Although researchers have created AI techniques based on aspects of CSNs before, there are several unique aspects to the approach presented here. The most important of these are listed below. These will be explored in detail in the following chapters.

1. *The creation of a new connectionist AI technique "Artificial Reaction Networks" inspired by biological Cell Signalling Networks. Outlined in chapter 4.*
2. *The production of complex waveforms for control of limbed robotic gaits by combining functional motifs found in CSNs within a rate law based connectionist system. Outlined in chapter 6.*
3. *The construction of the E. coli chemotaxis pathway using a connectionist based Artificial Chemistry. Outlined in chapter 5.*
4. *The implementation of chaotic dynamics by combining functional motifs found in CSNs within a rate law based connectionist system. Outlined in chapter 7.*
5. *The production of a complete limbed robotic control system by combining functional motifs found in CSNs within a rate law based connectionist system. Outlined in chapter 6.*
6. *The construction of multiple distributed cell-like agents by combining functional motifs found in CSNs within a rate law based connectionist system. Outlined in chapters 7 and 8.*

1.5 Chapter Overview

The thesis can be divided conceptually into three components:

1. Chapters 2-3 are introductory and contain the project background.
2. Chapters 4-8 present research, methods and experimental results.
3. Chapter 9 discusses the results and conclusions.

An overview of each chapter is given below:

Chapter 2: Cellular Intelligence

This is a background chapter which discusses the various aspects of Cell Intelligence. An in-depth overview of the properties and mechanisms of Cell Intelligence is presented. Its computational aspects are then explored and examples within the literature are highlighted. Finally, current methods of simulating Cell Intelligence are presented.

Chapter 3: Artificial Intelligence Inspired by Cell Signalling

In this background chapter, an overview of the existing AI techniques inspired by the biological mechanisms behind Cell Intelligence is provided. The chapter explores various types of methods from different sub-branches of AI including - Artificial Immune Systems, Artificial Gene Regulatory Networks, and Artificial Neural Networks.

Chapter 4: The Artificial Reaction Network

This chapter contains the first original work within the project. It presents a new model - the Artificial Reaction Network (ARN), designed specifically to represent the previously identified computational mechanisms of Cell Intelligence. This new representation is based on the previously reviewed techniques; unlike these models it abstracts only those computational features which are important in generating properties of Cell Intelligence. A complete overview of the new model is given, from its development and mathematical properties, to its strengths and weaknesses.

Chapter 5: ARN Based Simulation of E. coli Chemotaxis

This chapter contains experimental methods and results. Using biochemical detail extracted from the literature, the ARN is used to simulate the chemotaxis pathway of *Escherichia coli*. This simulation is used as a means to both verify the behaviour of the new representation against the properties of biological Cell Intelligence, and to compare it with the results of other simulation methods.

Chapter 6: Spatial & Temporal Properties of the ARN

In this chapter, further experimental methods and results are presented. These focus on examining the capacity of the ARN to perform complex spatio-temporal processing and pattern recognition. The applications of such processing are illustrated within limbed robotic control and show that the new representation has advantages over existing methods.

Chapter 7: Cytobots: ARN-Controlled Agents

This chapter introduces the concept of multiple distributed processing ARN-based agents. The agents are created by combining functional motifs found in biochemical networks and represent further novel work within the project. The ARN-agents representation and development are discussed in detail.

Chapter 8: Applications of ARN-agents

In this chapter the applications of the ARN-agents within biological simulation and in robotic control are assessed. A simulation of phases within the life-cycle of the slime mould *D. discoideum* examines the application of the agents as a tool to simulate multi-cellular systems. Another experiment, using a simulated oil-spill, is used to consider their application within swarm robotics.

Chapter 9: Summary and Conclusions

This chapter contains the overall results, summary and conclusions of the thesis. Consideration is given to the achievement of each objective, strengths and weaknesses of the project and possible future work.

1.6 Papers in Appendix 1

Research papers produced during the course of the research are included in Appendix 1. Viewed in descending chronological order, the papers provide the reader with a useful summation of the research as it developed. Results reported in earlier papers serve as the foundations for the research reported at later dates.

Appendix 1 contains the following papers:

1. Gerrard, C., McCall, J., Coghill, G. and Macleod, C. (in press) 'Applications and Design of Cooperative Multi-agent ARN based Systems', A Fusion of Foundations Methodologies and Applications, *Soft Computing*, Springer.

2. Gerrard, C., McCall, J., Coghill, G. and Macleod, C. (in press) 'Exploring aspects of Cell Intelligence with Artificial Reaction Networks', A Fusion of Foundations Methodologies and Applications, *Soft Computing*, Springer.

3. Gerrard, C, McCall, J., Coghill, G., Macleod, C. (2013) 'Artificial Reaction Network Agents', *The 12th European Conference on the Synthesis and Simulation of Living Systems (ECAL)*, Advances in Artificial Life, MIT press, Taormina, pp. 957-964.

4. Gerrard, C., McCall, J., Coghill, G., Macleod, C. (2013) 'Combining Biochemical Network Motifs within an ARN-Agent Control System', *Proceedings of the 13th Annual Workshop on Computational Intelligence (UKCI)*, IEEE, Surrey, pp. 8-15.

5. Gerrard, C., McCall, J., Coghill, G., and Macleod, C. (2013) 'Artificial Chemistry Approach to Exploring Search Spaces using Artificial Reaction Network Agents', *Congress on Evolutionary Computation (CEC)*, IEEE, Cancún, pp.1201-12.

6. Gerrard, C., McCall, J., Coghill, G. and Macleod, C. (2012) 'Adaptive Dynamic Control of Quadrupedal Robotic gaits with Artificial Reaction Networks', *Proceedings of the 19th International Conference on Neural Information Processing (ICONIP)*, Lecture Notes in Computer Science, vol. 7663, part 1, Springer, Doha, pp 280-287.

7. Gerrard, C., McCall, J., Coghill, G. and Macleod, C. (2012) 'Temporal patterns in Artificial Reaction Networks', *Proceedings of The 22nd International Conference on Artificial Neural Networks (ICANN)*, Lecture Notes in Computer Science, vol. 7552, part 1, Springer, Lausanne, pp 1-8.

8. Gerrard, C., McCall, J., Coghill, G. and Macleod, C. (2011) 'Artificial Reaction Networks', *Proceedings of the 11th UK Workshop on Computational Intelligence (UKCI)*, Manchester UK, pp 20-26.

2. Cellular Intelligence

2.1 What is Cellular Intelligence?

For many years, researchers have been interested in the complex behaviours displayed by individual cells. For example, simple unicellular eukaryotic microorganisms called protists show an astonishingly varied repertoire of seemingly intelligent behaviours. Some have sensory hairs, by means of which they can feel their way about and sense their food. While others even have locomotory appendages and stinging arrows to actively hunt and subdue their prey (*Ford 2009*). Take, for instance, the animal-like protist amoeba, *Difflugia*. Naturally found in ponds and marshlands, this heterotroph uses its pseudopodia to move around its environment and catch its prey. It is also one of several amoebozoa to build a microscopic protective shell called a "test" from siliceous sand grains in its environment, see *Fig. 2. 1* (*Ford 2004*); this task alone is of considerable complexity. It must take sensory input from its environment, correctly identify and select sand grains while coordinating movement; then intricately cement the grains together with incredible precision. With a demonstration of such complex behaviour it is easy to forget that *Difflugia* are just single-celled organisms.

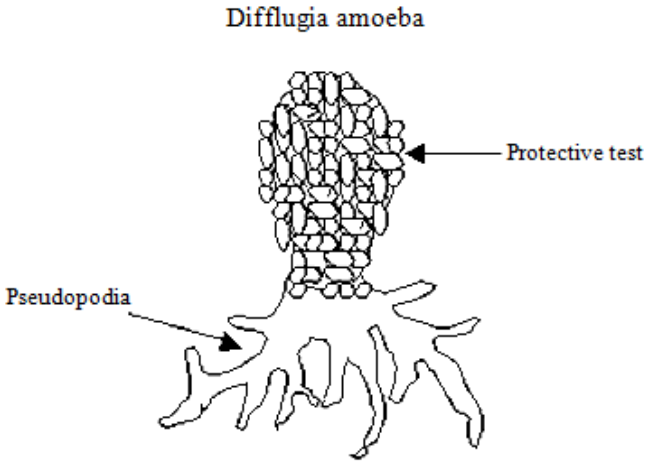


Fig. 2.1 A *Difflugia* test about 0.3 mm long. Pseudopodia can be seen extruding from the test.

In recent studies, the protist slime mould *Physarum polycephalum* gained public attention when Nakagaki et al. (*2000*) discovered that it was able to find the shortest path between two food sources within a maze. The maze was built from plastic films set on agar gel, with four possible routes of different length between the two food sources. The plasmodial slime mould was divided into pieces and placed at different positions within the maze. Initially, the pieces of plasmodium spread throughout the maze forming a single organism. As it spread, it withdrew from dead ends, until eventually it formed a thick plasmodial tube along the shortest pathway between the two food sources. This behaviour increased foraging capability, conserved energy and thereby increased its chances of survival. A further study by Saigusa et al. (*2008*) showed that, when subjected to a distinct pattern of periodic environmental changes, this simple organism was able to learn and change its behaviour in expectation of the next stimulus. The *P. polycephalum* was placed within a narrow lane where, initially, it was free to migrate under ambient conditions for a few hours. The conditions of the experiment were then modified by subjecting it to periodic 10 minute intervals of less optimal, drier and cooler conditions called a "dry simulation". This continued for a period of time, during which it was observed the organism would decrease its speed of locomotion during each dry simulation interval. Dry simulation was then discontinued, and continuous ambient conditions were reinstated. However, rather than behaving as was previously observed, the slime mould slowed down its speed at time points which coincided with those of the dry simulation. The researchers argued that this demonstration of recalling and anticipating events were indicative of a "primitive intelligence".

Unicellular organisms are by no means exceptional in demonstrating such behaviour. Research has shown that, without the direction of a nervous system, the cells of animals and plants display their own remarkable behaviour. Within living systems, these basic building blocks function autonomously, reacting to each other and their environment and cooperating to achieve tasks of considerable complexity (*Ford 2009*). For example, macrophages (a type of leukocyte) are specialised immune system cells, that function to protect the body by responding to and eliminating pathogens in a process called phagocytosis (see *Fig. 2.2*). Phagocytosis is mediated by an intricate mechanism of pattern recognition, which allows a macrophage to identify and distinguish foreign agents from host cells. Pathogens display conserved motifs on their surface called pathogen associated molecular patterns (PAMPs). These are identified by highly specific pattern-recognition receptors (PRRs) on macrophages. On recognising a pathogen, the macrophage PRR binds to the complimentary PAMP, which triggers the production of cytokines. The cytokines function to attract further leukocytes to the infected area, leading to the engulfment and destruction of the pathogen (*Aderem and Underhill 1999*).

Thus, during phagocytosis, a macrophage displays an elaborate array of behaviours, including pattern recognition, location of infected areas, coordination of movement and group attack.

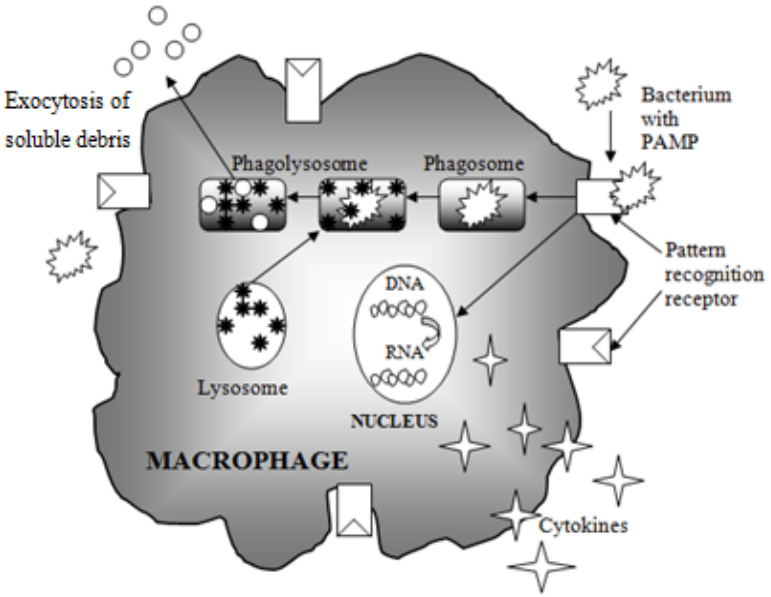


Fig. 2.2 Phagocytosis by a specialised immune response cell called a macrophage.

These illustrations provide a mere glimpse of the myriad of diverse and complex behaviours displayed by living cells. A cell has no brain or neural network with which to process information. Nevertheless these examples reveal a kind of "natural intelligence" and thus indicate that a form of information processing *is* somehow taking place. It is worth taking a moment to consider the meaning of the term "natural intelligence" and how this relates to the more anthropocentric view of intelligence. There is no agreed definition of intelligence and its defining characteristics are the subject of much debate and uncertainty (*Brooks 1999; Clark 1997; Dennett 1996*). Many view it as the ability to perform human cognitive tasks such as mathematical or logical calculations (*McCarthy 1960*), or, as Descartes argued, the ability to use language (*Descartes 1975*). Others think this view anthropocentric, and consider a more "natural" definition based on behavioural attributes which enhance an organism's chance of survival. Such behavioural attributes include: pattern recognition, classification, response, communication; learning and self-organisation. These properties, particularly in combination, portray purposeful "intelligent" high-level behaviour and are thus all the properties of natural intelligence (*Ford 2009*; *Crespi 2001; Ben-Jacob 1998*; *Dobbyn et al.* 2007; *Kauffman 1993*). All forms of life, from higher-level organisms to single-celled amoebas, display these characteristics. Therefore, the very unit of life "the cell" must have its own form of natural

intelligence. Over the past century, this type of intelligence has been referred to as “Cell Intelligence” or “Cellular Intelligence” (*Quevli 1916; Ford 2009)*.

2.2 The Machinery of Cellular Intelligence

The following section explores the properties and mechanisms behind Cell Intelligence, and discusses its origin.

In order for a cell to produce the previously discussed high-level behaviours, it must store and process information. The mechanisms involved are quite different from those of a digital computer. Within a cell’s cytoplasm, the current state is represented as a set of spatially distributed concentrations of chemical species (*Bray 1995*). This molecular “data” is processed through biochemical reactions which transform the composition of reactants and, in doing so, transform the current state of the system. The products of one reaction correspond to the reactants of another, thus forming input and output connections (*Bray 1995*). In this way, chemical data is processed by networks of reactions termed Cell Signalling Networks (CSNs). The process is best illustrated by a stylised example. Consider a highly generalised cell, as shown in *Fig. 2.3*. In this cell’s external environment are molecules of food- “protein A”. The cell-surface contains molecules of “protein B”; these are specialised transmembrane proteins called receptors, and are shaped in such a way that allows them to bind (react) with A. Bound to the intracellular component of B, is another molecule “protein C”. When molecules A and B link, as shown at step 1, this internal protein, C detaches from the receptor. The free floating C molecules diffuse within the cytoplasm as a result of normal thermodynamic forces. Eventually, protein C meets another internal protein, “protein D”, as shown in step 2. This has a complementary shape to C allowing these two molecules to bind and form a new complex - “protein E”. In this example, E is shaped so as to interact with a motor protein, “protein F”, as shown at step 3. Commonly, when stimulated in this way, motor proteins contract and allow propulsion of the cell towards a beneficial substance - for instance, toward a higher concentration of protein A. This phenomenon is called chemotaxis and is described in more detail later in the project. CSNs exist within all living organisms and in this way function to regulate all internal cellular activity such as changes in gene expression and metabolism. Furthermore, CSNs relay signals externally between local cells, distal cells, the environment, other organisms and even to themselves. They are a complex system of interconnected intra and extracellular distributed reactions, simultaneously processing streams of chemical signals, and have topological features such as feedback loops and interconnectivity (*Bray 1990; 1995*; *Kholodenko 2006*). These networks have evolved to allow an organism to adapt to current

conditions by manipulating the spatio-temporal activation levels of its intracellular proteins and thereby achieve a pattern that provides an appropriate cellular response.

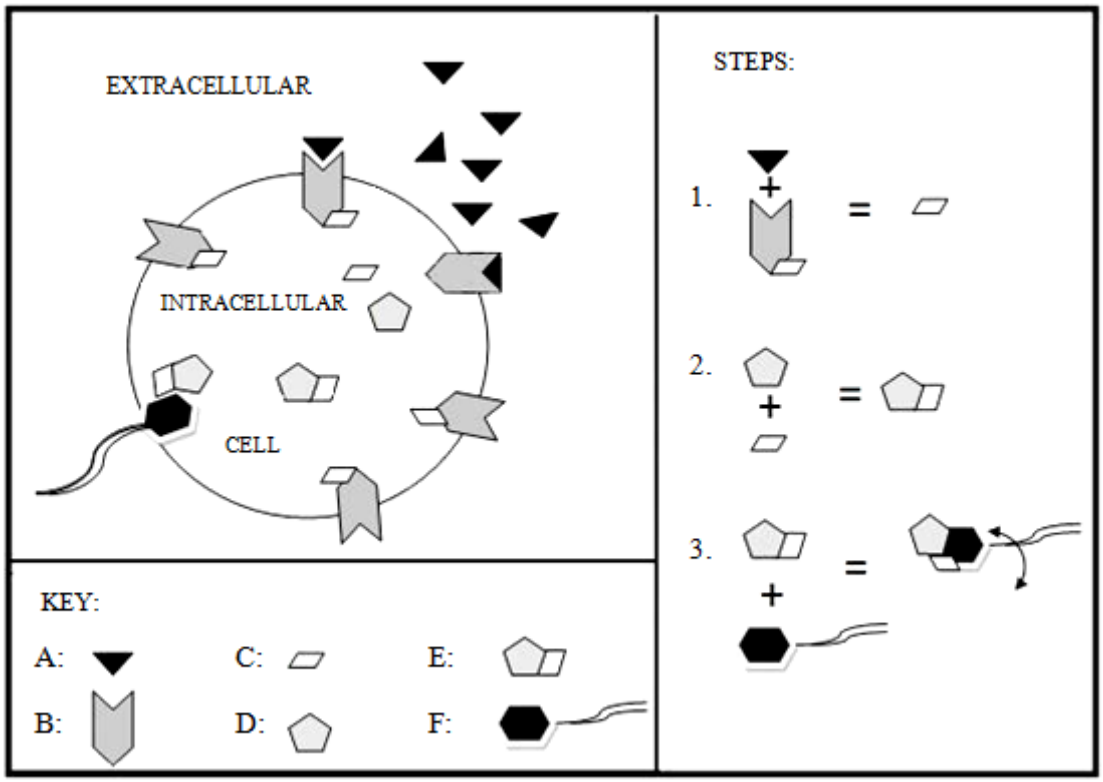


Fig. 2.3 A highly generalised example of a Cell Signalling Network

Particular pathways within CSNs, whose components regulate a particular function, are termed Signal Transduction Cascades (STC). For example, MAPK pathways utilise a three-tiered cascade of protein kinases known as a Mitogen Activated Protein Kinase module or MAPK, which communicate receptor signals to the DNA of a cell's nucleus (*Seger and Krebs 1995*). Although STCs are represented as separate pathways, they often participate in "crosstalk" where components of one pathway influence and form components of another (*Jones 2012*).

2.2.1 An Example of Cellular Intelligence

As previously mentioned, chemotaxis describes the movement of cells in response to sensed environmental chemical concentrations called chemoeffectors. Unicellular organisms swim toward chemoattractants such as amino acids and sugars, and away from chemorepellants such as fatty acids and alcohols (*Vladimirov and Sourjik 2009*). This is vital to their survival, since travelling toward chemoattractants allows them to locate food, and moving away from chemorepellants prevents their breakdown by harmful substances.

It can be shown that CSNs have the principle role in generating high-level behaviour by using the chemotaxis pathway of the bacteria *E. coli* as an example. Here, the high-level behaviour is overall movement toward more favourable conditions. The entire signalling pathway for *E. coli* chemotaxis has been identified and, as such, it represents an ideal pathway to explore the emergent properties of Cell Intelligence. *E. coli* is a rod-shaped bacterium commonly found in the lower intestine of many animals. The motile strains, as examined here, have an external tail-like structure called a flagellum that, by way of a motor complex, rotates and results in movement. Locomotion manifests itself in the form of two simple behaviours: "runs" characterised by smooth linear swims and "tumbles" where the bacteria stops and re-orientates in a random direction (*Vladimirov and Sourjik 2009*). While moving along swimming trajectories, chemoreceptors continuously signal the levels of environmental chemoeffectors (attractants or repellants). This signal is processed by six cytoplasmic proteins leading to the regulation of a motor complex responsible for the operation of the flagellar motor (*Vladimirov and Sourjik 2009*). The motor complex is attached to a bundle of helical flagellar filaments. When the motor rotates counter-clockwise (CCW), the flagellar filaments form a trailing bundle which, by pushing the cell forward, results in a "run". When the motor rotates in a clockwise direction (CW), the filaments change shape, causing the bundle to separate resulting in a "tumble" (*Vladimirov and Sourjik 2009*), as shown in *Fig. 2.4*.

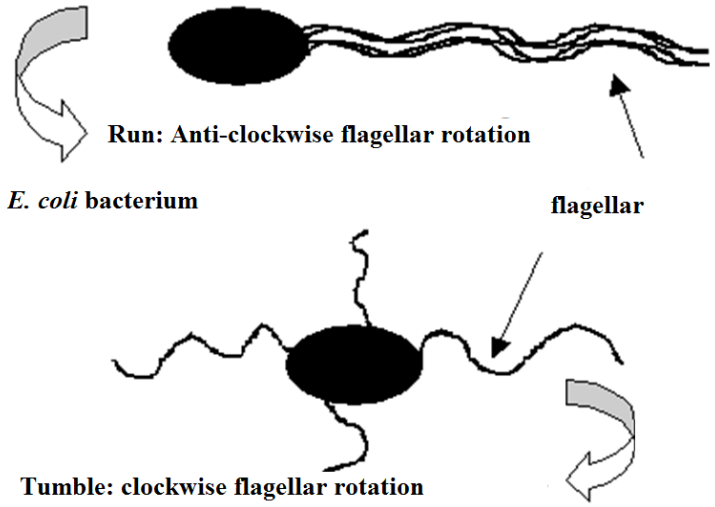


Fig. 2.4 *E. coli* movement by rotation of helical flagellar. Flagellar rotation counter-clockwise (CCW) results in a run; and clockwise (CW) results in a tumble.

The ratio of runs to tumbles is regulated by the intracellular concentrations of the proteins within the pathway (*Stock and Surrete 1996*) and results in a pattern of motion called a random biased walk. By comparing concentrations of chemoeffectors (attractants and repellants) in a

temporal fashion, the organism is able to reduce the frequency of tumbles up concentration gradients of attractants, and down gradients of repellants, thus providing the bias. The resultant high-level behaviour is overall travel toward more favourable conditions (*Vladimirov and Sourjik 2009*) as depicted in *Fig. 2.5*.

Further examples of high-level behaviour mediated by CSNs can be found in the literature (*Hellingwerf 2005*; *Nakagaki et al. 2000*; *Ben-Jacob et al. 2004*). By communicating and processing information, signalling networks are responsible for all cell activities and lie at the heart of Cellular Intelligence. The information processing units of CSNs are interacting chemicals. When connected within a network, elaborate processing capabilities emerge from the collection of low-level interactions. The next section will explore this concept by examining the computational aspects of CSNs.

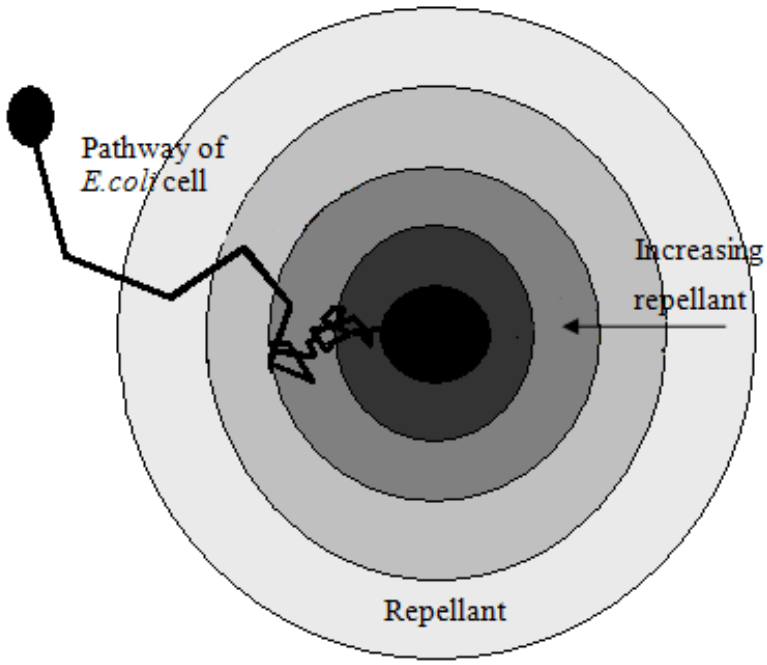


Fig 2.5 Pathway on an *E. coli* cell *through* a gradient of chemorepellant. The pattern of motion is called a biased random walk.

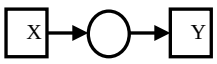
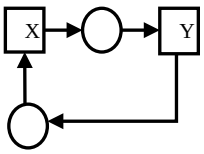
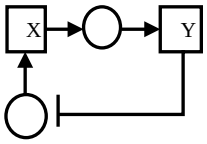
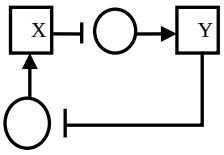
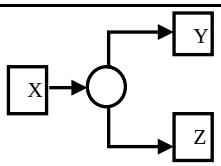
2.3 Computational Aspects of Cell Signalling Networks

As previously mentioned, within a cell, data is represented by a set of spatially distributed concentrations of chemical species. This molecular data is dynamically manipulated in response to external or internal changes with a lag time behind the determining signal. Reactions link chemicals together through productive unions (union of reactants triggers production or activation of other components) and inhibitory unions (union of two reactants inhibits production or activation). Each reaction transforms input signals into output signals, manipulating the data at each reaction step. Although signals are commonly chemical concentrations, they are not limited to this format. The input signal can take the form of temperature, light, mechanical force, or voltage. Likewise, the output signal can take the form of generated movement, light production or the formation of macromolecular structure (*Bray 1995*).

The computational properties of such signal processing within biochemical networks are well documented. It has been shown both theoretically and in wet lab experiments that such networks can perform Boolean and Fuzzy Logic functions and are equivalent to a Turing machine (*Arkin and Ross 1994*; *Bray 1995*; *Dittrich et al. 2001*). Evidence concerning the computational properties of reactions can be found in a number of independent studies. For example, Stadtman and Chock, (*1997*) demonstrated that the interconversion between phosphorylated and non-phosphorylated protein forms can act as a flexible computational unit. They showed that by modifying kinetic parameters, these units are capable of a wide range of input/output responses. Similar results were documented by Arkin and Ross (*1994*), who examined the computational properties of enzymatic reaction mechanisms and showed that the steady-states of such reactions are comparable to Boolean or Fuzzy Logic gates. It was found that sharp, digital-like, responses are achieved in cases where enzymes are saturated with their substrates; however, within a simple enzymatic reaction model, the steady-state functioned similarly to a Fuzzy Logic gate. Hjelmfelt and Ross (*1994*) demonstrated that a system of ordinary differential equations describing iodate-arsenous reactions could be used to store and recognise patterns of high and low concentrations. In connected work, Hjelmfelt et al. (*1991*) used enzymatic reaction kinetics to create a chemical implementation of a neural network. It was demonstrated that this chemical neuron was able to perform a range of Boolean functions and could be used to construct a universal Turing Machine (*Hjelmfelt et al. 1991*). In more recent studies, a number of researchers have developed these ideas; for example, Wang et al. (*2011*), constructed Boolean gates from actual biological proteins present in *E. coli*, and then, by connecting these gates together, was able to perform more complex logical operations.

Furthermore, in the field of Molecular Computing, several researchers have built on this work and created synthetic biological devices which use chains of reactions to process data, an excellent review of which is provided by de Silva and Uchiyama (*2007*).

TABLE 2.1 Part A. Structural Motifs in Biochemical Networks

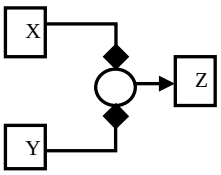
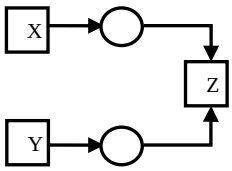
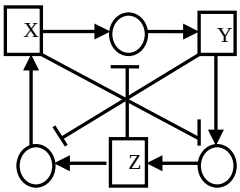
Structural Motifs Table 1 Part A		
Motif No., Name and Description	**Structure (in ARN format)**	**Biological Example**
1. Excitatory (E) The presence of X activates Y		Elementary motif common throughout most pathways. For example- Ras is a membrane associated protein that is normally activated in response to the binding of extracellular signals such as growth factors (*Tyson and Novak 2010*).
2. Inhibitory (Y) The presence of X inhibits Y. Acts as a NOT gate.		Elementary motif common throughout most pathways. For example- E-cadherin (a calcium-dependent cell–cell adhesion molecule) suppresses cellular transformation by inhibiting β-catenin (*Tyson and Novak 2010*).
3. Positive Feedback Loop (PFL) The presence of X activates Y and in turn the presence of Y activates X		One example is the pathway of caspase activation which is essential for apoptosis induction. A PFL exists between caspase-3 and caspase-9 (*Tyson and Novak 2010*).
4. Negative Feedback Loop (NFL) The presence of X activates Y and in turn the presence of Y inhibits X		For instance, the proteins Mdm2 and p53 (p53 is a tumor suppressor protein) are involved in a NFL which functions to keep the level of p53 low in the absence of p53-stabilising signals (*Tyson and Novak 2010*).
5. Double Negative Feedback (DNF) The presence of X inhibits Y and the presence of Y inhibits X		For instance, BAX is a protein which promotes apoptosis by competing with BCL. A DNF is formed between the proteins BAX and BCL *(Tyson and Novak 2010*).
6. Branch (B) The presence of X activates Y and Z		For example, the transcription factors such as E2F or P53 frequently modulate the expression of more than one gene. Enzymes often modify more than one substrate e.g. CycB-dependant kinase (*Tyson and Novak 2010*).

Key: Either inhibitory or excitatory. Excitatory. Inhibitory connection.

X/Y/Z: Chemical species. Reaction. Chemical substrate.

* Motifs may combine arbitrary numbers of components.

TABLE 2.1 Part B Structural Motifs in Biochemical Networks

Structural Motifs Table 1 Part B		
Motif No., Name and Description	**Structure (in ARN format)**	**Biological Example**
7. Logic Gate (LG1) **AND gate:** two excitatory connections from X and Y. When both X and Y are present, they activate Z **NOR gate:** two inhibitory connections from X and Y. Both X and Y must be absent for Z to be activated **SWITCH:** Excitatory connection from X and inhibitory connection from Y. The presence of X but not Y activates Z		**AND:** The protein gCam 2 kinase becomes active when both calcium ions (Ca2+) and Calmodulin (CaM) are present (*Bray 1995*). **NOR:** The activity of transcription factor E2F is a NOR function of RB and CycB where E2F is active when both RB and CycB are inactive (*Tyson and Novak 2010*). **SWITCH:** The enzyme aspartate transcarbamylase has multiple catalytic sites. It is activated by the binding of its substrates (aspartate and carbamoyl phosphate) and inactivated by cytidine triphosphate causing its substrates to dissociate (*Bray 1995*).
8. Logic Gate (LG2) OR Gate: two excitatory connections from X and Y when either X or Y are present they activate Z		Ras is a membrane associated protein that is activated by a number of different signals. E.g. in response to the binding of extracellular signals such as a number of growth factors (*Tyson and Novak 2010*).
9. Oscillator (OSC) The presence of X activates Y. In turn the presence of Y activates Z but inhibits X. The presence of Z inhibits Y and activates X.		The cyanobacteria clock protein KaiC has a well-defined closed cycle of phosphorylation and dephosphorylation states (composed of KaiA, KaiB and KaiC). In the motif shown here, all three components oscillate and each inhibits the reaction clockwise. Oscillators may have less inhibitory connections, the number of which is dependent on the mobility of the reaction species. However, the presence of all inhibitors increases stability in the presence of fluctuating environmental parameters e.g. temperature. *Note that this oscillator can also be thought of as a PFL (motif 3) combined with a system of DNFs (motif 5).

Key: Either inhibitory or excitatory. | Excitatory | Inhibitory connection

X/Y/Z: Chemical species | Reaction | Chemical substrate

* Motifs may combine arbitrary numbers of components.

A number of researchers have identified structural motifs within CSNs which perform distinct computational functions (*Tyson and Novak 2010*; *Bray 1995*). A summary of these structures and of the most common motifs is provided in *Table 1* parts A and B; a more detailed discussion is provided by Tyson and Novak (*2010*). The motifs are shown for simplicity as two or three component forms but there are larger versions with the same function; for example, an

additional component may be added to motif 9 to create a four component oscillator. Such oscillators produce a periodic change in chemical concentrations and many illustrations of these motifs can be found within the literature (*Tyson and Novak 2010*; *Bray 1995*; *Kholodenko 2006*). One important example is the universal signalling motif of a phosphorylation cycle. Here a signalling protein is interconverted by opposing enzymes (a kinase and a phosphatase) between its phosphorylated (Yp) and non-phosphorylated forms (Ys). In a multisite phosphorylation cycle, positive or negative feedback from either form can cause oscillations between stable states or render the cycle into a bistable switch, where the low and high Yp concentrations correspond to "on "and "off" states (*kholodenko 2006*). A cascade of such bistable cycles can produce multiple stable states, allowing the control of many cellular functions. For example, the cell's transition into mitosis is governed by the sequential activation or inactivation of such kinases (CDK1/Cdc2) (*Kholodenko 2006*).

Another motif (motif 7, *Table 1*), is a logic gate. For example, consider a protein with two sites that may undergo phosphorylation. If the protein is active when both sites are phosphorylated, it is similar to a Boolean AND gate. Alternatively, if this protein is active when either site is phosphorylated, it is analogous to a Boolean OR gate (motif 8, *Table 1*). The activity of the response is determined by kinetic factors such as binding affinities – or, using reaction kinetic terminology: the reaction rate, reaction order and concentration of the reacting molecules. Where conditions are highly reactive, a processing unit acts like a molecular switch giving a sharp binary on/off response - or, in cases of lower reactivity, a unit may provide a more gradual response more comparable to Fuzzy Logic.

Another important feature of CSNs is their ability to amplify a faint signal and generate a considerably greater response. Amplification occurs when there is an increase in the absolute number of output molecules compared to the number of input molecules. For example, branch motifs (motif 6, *Table 1*) allow amplification, because a single ligand activates a protein, which proceeds to trigger numerous targets. Each of these proteins then initiates a cascade of signalling events, where, at each step, there is a potential for similar amplification, thus generating a response signal on a scale much larger than the original signal (*Bray 1995*).

It is not difficult to imagine how a system with such elaborate processing abilities could be configured to perform highly complex computational tasks. In fact, many researchers have argued that CSNs have many features in common with the AI technique "Artificial Neural Networks" or ANNs (*Bray 1995*; *Bhalla 2003*; *Hellingwerf 2005*). This technique was itself, inspired by a form of cell signalling, when, in 1943, McCulloch and Pitts demonstrated how a highly stylised representation of a biological neuron could perform a variety of logical

processing. The original idea has since been developed considerably, notably by Rosenblatt (*1957*), who developed the first learning algorithm, and by Werbos (*1974*) who was one of the first to propose the well-known Backpropagation training technique. It has since been shown that a layered network of artificial neurons is capable of performing any logical operation and demonstrates the ability to recognise patterns (*Rumelhart and McClelland 1986*).

ANNs have proved extremely useful in many applications involving pattern recognition and in fields such as robotic control (*Rumelhart and McClelland 1986*). The similarities drawn between ANNs and CSNs are based on computational attributes and ignore the obvious differences between their implementation. Bray provides an excellent comparison (*Bray 1995*), the main points of which are summarised here:

1. Both CSNs and ANNs are networks composed of highly-connected parallel distributed units. Each unit amplifies or attenuates and then simultaneously integrates and processes signals. In a CSN, it is a reaction which integrates and transforms a number of signals into an output, while in an ANN this is performed by the activation function of a neuron.

2. Both networks can have "hidden layers" of processing units. In a CSN, intermediary reactions function like those of the hidden units found in a neural network, where the overall output is the collective result of internal processing.

3. Both ANNs and CSNs are able to recognise patterns. In a CSN, patterns are composed of a number of input signals such as the presence of concentrations of particular chemical species within the environment.

4. Both ANNs and CSNs are robust. CSNs are able to recognise patterns, while ignoring irrelevant chemicals (noise), even when the signalling pathway has been degraded. This robustness is also found in ANNs, where the correct response to input patterns is still generated in the presence of noise and loss of units.

A significant difference between CSNs and simple traditional ANNs relates to the dimension of time. Time was not represented in the original ANN model, and its output was thus static. In a CSN, time is implicit, and a continuous response is generated. The time taken to respond to an input signal depends on numerous factors. For example, molecular diffusion rates through the cytoplasm are subject to molecular size and forces between neighbouring molecules.

The computational features discussed in this section act in concert to provide the level of processing required to generate emergent high-level behaviour in cells. Thus a CI inspired AI technique should incorporate such properties.

2.4 Representations of Cell Signalling Networks

Models of CSNs provide a means to examine complex relationships between receptor stimuli and cellular responses. They also offer insights into the mechanisms responsible for characteristics such as signal amplification, robustness, oscillations, and emergent behaviour. Furthermore, increased understanding of cell signal processing enables researchers to explore and predict the effects of disease and the outcome of pharmaceutical intervention. Traditional biology has focused on the examination of biological systems by reducing them to their individual parts, and studying them in isolation. This reductionist approach has proven unsuitable for representing the complex dynamics of CSNs, where cellular response is tightly regulated by the elaborate intracellular spatial and temporal dynamics of interconnected signalling molecules. In contrast, Systems Biology offers a holistic approach in which the entire system is examined in terms of interactions between its component parts. This approach emerged as a distinct discipline in the late 1960's, when Mihajlo Mesarovic (*1968*) described it as the use of systems theory to explain biological phenomena. However, interest was limited until the birth of functional genomics in the 1990s. This resulted in a considerable growth in availability of quality biological data and together with the exponential increase in computational power, allowed more realistic computational models to be constructed. Consequently, there are now numerous methods of representing networks of chemical reactions, ranging from the meticulously detailed quantum mechanical models to the highly abstracted discrete Boolean ones. Savageau (*1988*) provides a descriptive diagram of this spectrum, and this is summarised in *Fig. 2.6.*

INCREASING COMPLEXITY ↑

Quantum mechanical wave models	DiscreteStochastic
Potential energy models	Continuous/Deterministic
Probability distribution models	Discrete/Stochastic
Ordinary differential equation models	Continuous/Deterministic
Abstract machines	Discrete/Stochastic
Boolean functions	Discrete/Deterministic

Fig. 2.6 Spectrum of detail in representations of biochemical systems.

After figure in: Savageau M. (1988) Introduction to S-Systems and the underlying power-law formalism.

The choice of representation used depends upon the information one intends to extract from the model. For example, if the interest lies in reaction occurrence alone, simple Boolean functions would suffice. The study of Cellular Intelligence requires a representation which focuses on emergent global behaviour resulting from complex spatio-temporal interactions and network topologies. A common problem with modelling such a system, is achieving a balance between scope and detail. Too narrow a scope would result in insufficient predicative power; conversely, increasing the scope results in addition of parameters, which quickly escalate beyond manageable levels (*Aldridge et al. 2006*).

Starting from the top of this hierarchy, the next sections provide an overview of the most relevant techniques, that is, those which are able to capture any of the previously discussed properties and mechanisms of CSNs. As it is not practical to represent networks of reactions at the quantum level, techniques residing at the top of this hierarchy such as the quantum mechanical wave function will not be considered. However, particular attention is given to those techniques which incorporate both the temporal dimension and the parallel distributed processing nature of CSNs.

2.4.1 Potential Energy Models

Potential energy functions represent the thermodynamic properties of systems by considering the potential energy of their component reactions. By creating "potential energy landscapes" of CSNs it is possible to study global properties such as robustness. If the steady-state distribution of reactant species is known, a link can be drawn between the probability of a configuration state (concentration) and its associated potential energy. In a CSN, the probability of a molecular species being in a particular state will be higher if its associated energy within the landscape is lower (*Wang et al. 2006*). A similar energy technique exists that uses Gibbs free energy (*Hong et al. 2005*). Such models can provide insight into global behaviour, stability and robustness, but are limited by their inability to represent complex network topologies and temporal dynamics in a computationally efficient way. Furthermore, the use of the "energy landscape" does not facilitate their simple use in systems with conventional inputs and outputs.

2.4.2 Probability Distribution Models

Probability distribution models represent the stochastic properties of signal transduction. These use statistical models of molecular behaviours to calculate the probability distribution for each alternative behaviour occurring. There are numerous techniques available to do this, such as: Bayesian Networks (BNs); Markov Chains (MCs) and Stochastic Simulation Algorithms (SSAs).

BNs have recently drawn interest for their ability to infer CSNs from experimental data (*Woolf et al. 2005*). Here reactive species are represented as variables, each having an associated probability table. This table contains possible concentration levels (states) according to the presence of other reactant species. Such networks are disadvantageous in that they are acyclic and static and thus modelling feedback regulatory mechanisms and temporal dynamics is not viable.

SSAs are popular stochastic simulation methods based on the Chemical Master Equation (*Stoll et al. 2012*). One example is the Gillespie Algorithm which is used to predict which reaction is likely to occur and at what time (*Gillespie 1977*). The Chemical Master Equation is a linear differential equation which describes the probability that the chemical system will be in a particular state at a set time. The Gillespie Algorithm samples the probability space using a Monte Carlo step and in this way generates an exact distribution for the Chemical Master Equation. This method accounts for the randomness inherent in molecular reactions, and is useful in examining the effects of microscopic variations (*Stoll et al. 2012*). Its main disadvantage is computational cost, which can be high, particularly when elapsed time between reactions is small. Furthermore, it assumes a well-stirred reactor and thus compartmentalisation is not modelled. Finally, only few reactants can be simulated because every reaction is explicitly accounted for.

In MCs, the current state of the chain is determined by the amount of molecules present. Reactions are represented as transitions between these states (*Calder et al. 2006)*. Using MCs, one can gain information with regard to the steady-state probability distribution of a CSN; however, this is only possible in the absence of feedback mechanisms (*Decraene et al. 2007*).

2.4.3 Ordinary Differential Equation Models

A very common method used to model biological systems is ordinary differential equations (ODEs). These models are popular since they use continuous data and are able to accurately represent underlying system dynamics (*Eungdamrong and Iyengar 2004*). They also link the forward and reverse rates of a reaction with other measurable reaction parameters. The most popular of the ODE models are discussed below.

A well established and popular model is the Rate Law equation. This describes the progress of a reactant or product concentration through time. Before introducing the model, the kinetics of a typical reaction are briefly examined.

In general, the initial rate of a reaction is relatively large, and gradually decreases to zero. Typically, a reaction plot of product concentration versus time would produce a convex-up increasing graph, as shown in *Fig. 2.7*. The derivative of the curve at time *t* corresponds to the instantaneous rate of the reaction, while the average rate is given by the final product concentration divided by the total time.

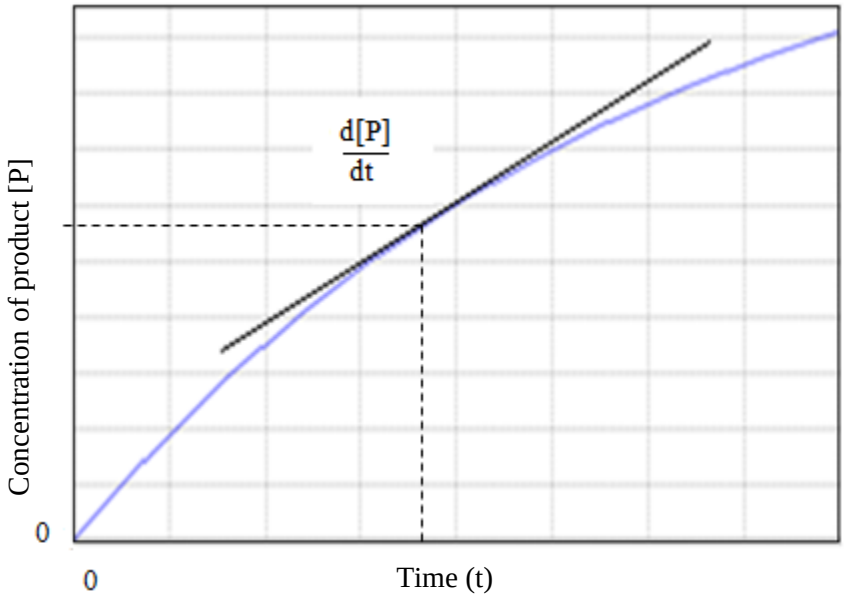


Fig 2.7 Typical reaction kinetic plot showing product concentration vs. time.

Reactions are often defined as being reversible or irreversible; a reversible reaction being one which results in a stable chemical equilibrium mixture of reactants and products. The net rate of species generation is equal to the sum of the forward and reverse rate *Eq. (2.1)* (*Morris 1974*). At equilibrium this is approximately zero.

$$rate_{net} = rate_{forward} - rate_{reverse} \tag{2.1}$$

$$[\dot{V}] = k_f[A]^g[B]^h - k_r[C]^i[D]^j \tag{2.2}$$

The Rate Law expression, given in *Eq. (2.2)*, describes the net rate of a reaction ($[\dot{V}]$) with respect to time and is equal to the difference between the two distinct functions: the forward reaction rate and the reverse reaction rate; where the first term ($k_f[A]^g[B]^h$), includes all

terms contributing to system influx the second ($k_r[C]^i[D]^j$) to decay of products. In each term, a set of concentrations ([A], [B], [C], [D]) are raised to a power which defines the order (*g*, *h*, *i*, *j*), with respect to the reactant or catalyst, and multiplied by a rate constant (k_f or k_r) (*Morris 1974*). In a non-reversible reaction between reactants A and B, the reverse term is removed from the expression. The rate constant defines the proportion of molecules that react in the particular process per unit time. In an elementary reaction (one with no intermediate steps) the order determines how many molecules are involved in the reaction for each unit of chemical concentration. Zero order, with respect to a reacting species, indicates that this concentration will not affect the rate of reaction. The overall order of the reaction is determined by totalling the orders of each chemical species (*Morris 1974*).

The Rate Law equation, like other ODE models, is able to provide an accurate representation of temporal dynamics. However, in its basic form it does not provide a means to clearly visualise a complex network of chemical reactions. Furthermore, like most ODE models, it does not account for the intracellular spatial distribution of chemical species into localised compartments - an attribute termed "compartmentalisation". Instead it assumes the cell is a well-stirred reactor (*Decraene 2007; Eungdamrong and Iyengar 2004*). This localisation restricts which molecules may react, thus affects the overall dynamics of the system. Representing the spatial distribution of chemicals allows fine-grained control over the system dynamics and thus is highly beneficial when modelling biochemical circuitry. For example, it allows the representation of flow structures such as membrane channels, transport processes and network motifs, and provides a means to explore disease pathways (*Kreyssig and Dittrich 2011*). To address this issue, compartmental ODE models have been developed which achieve coarse representations of these compartments (*Eungdamrong and Iyengar 2004*). In these methods, the same molecular species may occupy many different compartments; however each compartment is treated as though it were a separate species. Molecular exchange between compartments is modelled as a flux; this is then used to update compartments associated with the exchange. For coarse approximations, this type of model is appropriate. However, such models are inappropriate for fine spatial detail, as the number of compartments and therefore variables quickly increase beyond manageable levels (*Eungdamrong and Iyengar 2004*). Another adaptation is to use Partial Differential Equations (PDEs), where concentration of molecular species is a function of time and space. These however have the disadvantage of being more complex, computationally intensive and non-intuitive (*Eungdamrong and Iyengar 2004*).

S-Systems are based on the Rate Law and are another popular ODE representation. They have been used to model biological systems since the late 1960s (*Savageau and Voit 1987*).

Similarly, to the simple Rate Law, each ODE is composed of species concentration variables, raised to a power and multiplied by rate constants, as shown in *Eq. (2.3)*. Again, each ODE is equal to the difference between two distinct functions; the first term includes all terms contributing to system influx, the second to decay. The first half corresponds to the basic forward reaction. Where the rate of change of product j ($[P_j]$), is given by the forward reaction rate (the speed of the reaction) labeled k_f, multiplied by the product of the concentrations of the N reactants $[R_n]$, each raised to the power of its reaction order α_n. The second term represents the decomposition of product back into its original reactants. This depends on the reverse reaction coefficient k_r multiplied by the product of the concentrations of the *M* products $[P_i]$, each raised to the power of its reaction order β_i.

$$\frac{d[P_j]}{dt} = k_f \prod_{n=1}^{N} [R_n]^{\alpha_n} - k_r \prod_{i=1}^{M} [P_i]^{\beta_i} \tag{2.3}$$

To illustrate this further, consider the common reaction between two reactants labelled *A* and *B* with reaction orders (the number of molecules of each reactant used to make a molecule of product) of *q* and w respectively. These produce a single product *P*. The order of the reverse reaction is one. *Eq. (2.3)* is then reduced as described by *Eq. (2.4)*.

$$\frac{d[P]}{dt} = k_f [A]^q [B]^w - k_r [P] \tag{2.4}$$

In S-Systems, a group of rate equations are normally set up - one for each reaction. The left hand of each equation is then set to zero and they are then solved simultaneously to yield the steady-state response. If the dynamic responses are required, then numerical solution methods like Runge-Kutta are normally applied.

There is little difference between an S-System representation of a reaction and its representation as a Rate Law. Essentially the distinction between the two is that the S-System simply provides a more generalised way of representing a set of reactants and products. Thus the Rate Law and S-System representation share the same advantages and disadvantages.

Another popular ODE equation, introduced in 1913, is the Michaelis-Menten equation (*Morris 1974*). This is used to model the kinetics of enzymatic reaction kinetics. Its mechanism can be represented as shown in *Fig. 2.8*.

$$[E]+[S]\underset{k_r}{\overset{k_f}{\leftrightarrow}}[ES]\overset{k_s}{\rightarrow}[E]+[P]$$

Fig. 2.8 Michaelis-Menten mechanism of enzymatic reactions

The first reaction represents the enzyme *E* reacting reversibly (forward rate is k_f and reverse rate is k_r) with the substrate *S*, to produce the enzyme substrate complex *ES*. The second reaction, (often an allosteric change), produces the reaction product *P* (rate is k_s), and the enzyme returns to its disassociated state. The two differential equations shown in *Eq. (2.5)* and *(2.6)* give the kinetics of the above reaction.

$$\frac{d[ES]}{dt} = k_f[E][S] - (k_r + k_s)[ES] = 0 \tag{2.5}$$

$$\frac{d[P]}{dt} = k_s[ES] \tag{2.6}$$

The only essential difference between the Michaelis-Menten equations and the Rate Law is that an additional intermediate reaction which forms the enzyme substrate complex is specifically accounted for. Thus it provides a more accurate account of the specific dynamics of such enzymatic reactions. However, it means that it is restricted specifically to modelling of enzymatic reactions only. Aside from these points, the Michaelis-Menten model shares the same advantages and disadvantages as other ODE models.

Many Michaelis-Menten based models are systems of the above equations; however, others have used these equations in different ways. For example, the AB-neuron (*Eikelder et al. 2009*) is an abstract mathematical representation of phosphorylation cycles within CSNs. It has a connectionist representation where reactions form weighted links between substrates. The Michaelis-Menten equation provides the output of each reaction and represents the steady-state concentration of the product. The AB-neuron was developed as part of the ESIGNET (*2011*) project and has been used to successfully predict reaction rates of phosphorylation cycles. In its current state, it may be used to model enzymatic reactions, but does not account for transient dynamics between steady-state equilibrium.

2.4.4 Abstract Machine Models

A number of Abstract Machine models have been used to simulate chemical systems. These can be broadly classed into Algebraic Process Calculi (PC) and State Transition Systems. Abstract Machines offer discrete models and are widely used to study parallel distributed concurrent processing.

One important example of a State Transition System is the graphical and mathematical modelling tool called a Petri Net (PN). This is used extensively in several types of information processing, including modelling CSNs (*Balden et al. 2010*). In a similar way to UML (Unified Modelling Language), the graphical notation of PNs is useful as a visualisation and communication tool; however, it also provides a precise mathematical definition and theory for process execution and analysis (*Murata 1989*). The graphical representation, shown in *Fig. 2.9*, takes the form of a bipartite graph composed of two types of nodes called places (circles) and transitions (boxes/bars). Arcs, shown as directed, weighted (positive integer) lines, represent transitions (reactions) from either a place to transition or transition to place. “Input places” (representing reactants) describe the places from which an arc runs to a transition, whereas “output places” (representing products) describe the places to which an arc runs from a transition.

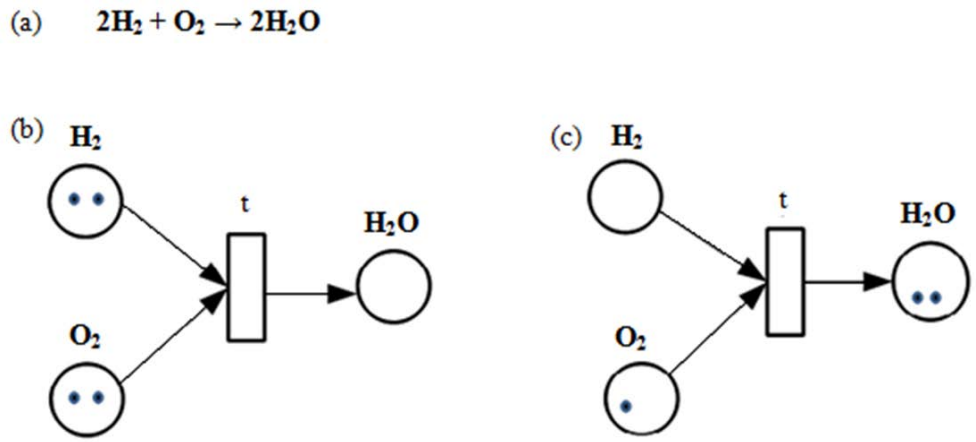


Fig.2.9 A Petri Net for the formation of water is described in (a). The Petri Net in (b) describes the state prior to formation, and (c) after products have been formed.

Each place contains 0 or a positive number of tokens (black dots). A “marking” describes the current value of tokens assigned to each place and represents the current state of the PN. If a token is present within the place then a condition has been met. Each transition represents an event, where the input place tokens represent pre-conditions and similarly the output place

tokens represent post-conditions to that event. A transition may "fire" (a reaction occurs) whenever the correct number of tokens (pre-conditions) are present in each input place. The action of firing is atomic and the tokens used within the transition are placed in the output places. For example, as shown in *Fig. 2.9b*, the required tokens are present in each input place and thus preconditions for the formation of water have been met. *Fig. 2.9c* shows the state of the PN after the transition has fired and the post-conditions have been met (*Murata 1989*).

Many extensions to the original PN have been proposed within the literature. One such example is the addition of test and inhibitor arcs (*Agerwala 1974*). A test arc verifies the presence of a token in an input place without consuming it. If a token is present in a place connected to an inhibitor arc, the transition will be prevented from firing. In biological systems, test and inhibitor arcs are particularly useful. For instance, a test arc may model the role that enzymes play in reactions (enzymes are needed for a reaction to take place, but are not consumed). Another example are Functional PNs (FPNs), which represent arc weights symbolically as functions of the number of tokens in places (*Hofestadt and Thelen 1998*). In the case of biological systems, these are used to model the way in which variations in species concentration may influence the reaction. Other PN extensions provide means to represent temporal system dynamics. For example, Time Petri nets (TPNs) (*Merlin and Farber 1976*) include a time interval associated with each transition, where each transition must fire within that time period.

PNs are designed for modelling parallel distributed processing systems, making them a powerful tool in the representation of CSNs. By separating chemical species spatially, places provide a means to model compartmentalisation and facilitate the fine-tuning of species parameters. In their basic form their limitations are similar to other discrete models, particularly their inability to represent time. Furthermore, they cannot represent reversible reactions, different chemicals (all tokens are the same) or external connections to the pathway. Extensions, such as those discussed above have been used to address these problems, although none offer ideal solutions.

Algebraic Process Calculi methods (also referred to as Term-Rewriting Systems) represent reactants as multisets of algebraic terms. Examples used to model biological systems are Chemical Abstract Machines (CHAM) (*Berry and Boudol 1992*) and AlChemy (*Fontana 1992*), which is based on λ-calculus. CHAM is an extension of the T-language introduced by Banâtre et al. (*1988*). In CHAM the reactants are represented by algebraic terms, the population of which is organised as a finite multiset. Reactions are defined by transformation rules which rewrite multisets, matching the reactant side of the rule with the product multiset. Typically

these types of model represent the reaction vessel as a well-stirred reactor, and thus do not account for spatial compartments. Algebraic Process Calculi methods are used extensively in the field of Artificial Chemistry, and are described in more detail in the next chapter.

2.4.5 Boolean Function Models

Boolean Networks, introduced by Stuart Kauffman, are an example of dynamic networks, where both time and state are discrete (*Kauffman 1969*). They were first used to describe Gene Regulatory Networks, where the state of a gene is described by a Boolean variable expressing an active (on:1) or inactive (off:0) state. This simple representation has since been used to describe the occurrence of a reaction (*Davidich 2008*). They consist of a set of n-logical nodes, each corresponding to a protein. Each node state is determined by its set of connected nodes, the states of which serve as input variables. These inputs are subject to a Boolean function representing the interaction between elements. This is used to calculate the nodes' current state and each node state is updated synchronously.

Many extensions to the original Boolean Networks have since been developed. For example, Random Boolean Networks (RBNs) have random initialisation of connections and logical functions (*Gershenson 2004*). The original Boolean Networks (which are termed Classical RBNs (CRBNs)) use a synchronous updating scheme. This has been criticised, as neither gene nor reactions change their states at the same instant (*Harvey and Bossomaier 1997*). As a result, variations in updating schemes have been developed. Harvey and Bossomaier (*1997*) introduced asynchronous RBNs (ARBNs) where a random node is selected and updated at each time-step. Unlike CRBNs, ARBNs are non-deterministic. In order to maintain asynchronicity of ARBNs and the deterministic properties of CRBNs, Gerhenson (*2002*) later developed Deterministic Asynchronous RBNs (DARBNs).

In their general form, Boolean Networks are computationally cheap due to their innate simplicity, and are focused on emergent network properties. However they discard most unit behaviour, preferring a binary switch response rather than continuous signals, and therefore cannot capture subtle system dynamics.

3. Artificial Intelligence Inspired by Cell Signalling

The previous chapter explored the emergent behaviours associated with CSNs and provided a critical analysis of the mathematical and computational methods used to model such behaviour. This chapter provides a review of the literature on AI techniques inspired by these biochemical reactions.

As already described, networks of biochemical reactions exhibit characteristics of self-organisation, robustness, fault tolerance, adaptability, and functional emergence. Such attributes are frequently desirable in computer systems. This has led researchers to abstract properties and mechanisms from these biochemical networks to create AI systems to solve current real-world problems. Many of the techniques described in the previous chapter have application as biological simulators, but also in AI. Thus, there is a broad overlap in their field of application. Generally, in AI, these techniques are used to create highly abstract models, while, when applied to Systems Biology, the models tend to be analogues of the biological system. Usually, the aims of AI and Systems Biology researchers are distinct and thus techniques are exploited in different ways for distinct purposes. However, sometimes the motivation overlaps. For example, AI researchers are also interested in the origins of intelligence and even in the mechanisms involved in the evolution of the first life forms, for example, studying the autopoiesis of cells (*McMullin 2004*). Gaining an understanding of such processes means they can be exploited to create powerful AI solutions.

The first section of this chapter provides a brief introduction to bio-inspired AI. This is followed by an overview of Artificial Chemistry, that is, man-made systems which exploit the properties and mechanisms of natural chemical systems. The most relevant of these AI models are then discussed in the remaining sections, that is, those which take into account the linkage of chemicals through interactions, and consider temporal dynamics. Only models which can be programmed on conventional computers are considered. Thus, Artificial Chemical systems implemented from organic components such as Reaction-Diffusion systems (*Adamatzky et al.*

2005), molecular devices (*Zauner and Conrad 2001*), or those implemented as electrical circuitry, for example, Reaction Diffusion Cellular Neural Networks (RDCNN) (*Arena et al. 1999*) are not discussed. The reviewed techniques are organised into six sections. These are techniques based on the: endocrine system, nervous system, immune system, metabolic system, gene regulatory system and chemical reaction systems in general.

3.1 Introduction to Bio-inspired AI

AI can be divided into two main branches: Bio-inspired Computing (also known as Natural Computing) and Symbolic AI. The focus of Bio-inspired Computing is to develop computational solutions to complex real-world problems based on ideas gained from the study of living systems. In particular, natural principles such as interaction, adaptation, selection, and emergence are used. In contrast, Symbolic AI, aims to develop computational solutions which emulate human intelligence, using symbolic expressions and rules of manipulation. Alan Turing and John Von Neumann are among the first to suggest that principles from the natural world could be used to solve computational problems (*Turing 1952*; *Neumann 1958*). Bio-inspired AI has proven extremely powerful and has expanded enormously since its birth. Many subdivisions now exist, for example: Artificial Life, Evolutionary Computing, Neural Computing, Cellular Automata, and Swarm Intelligence. Three of the most established of these techniques are Evolutionary Computing, Neural Computing and Cellular Automata. Evolutionary Computing encompasses techniques which are based on Darwinian evolutionary principles. Neural Computing techniques are inspired by the nervous system, particularly the parallel distributed nature of processing found in neural networks. Cellular Automata techniques, like many biological systems, consist of simple components which, through multiple low-level interactions give rise to emergent behaviours.

By using Genetic Algorithms (*Rajasekaran and Vijayslakshmi 2011*) as an example it can be illustrated how features seen in nature can be abstracted into an AI representation. Genetic Algorithms (GAs) belong to the Evolutionary Computing branch of bio-inspired AI. They are a type of search heuristic which use evolutionary principles such as mutation, crossover, fitness and selection to find and to optimise solutions to complex problems. The basic steps are as follows. The first stage is to randomly initialise a population of solutions, the size of which depends on the nature of the problem. Each solution is composed of a number of values such as binary strings. These values represent the chromosome or genotype within the genotype phenotype relationship. A process of fitness evaluation, selection and breeding is then repeated until a solution which meets the fitness criteria is gained. The probability of a particular

solution being selected for breeding is based on its fitness. The breeding population are then used to create the next generation. Here certain members are selected to swap chromosomal information between each other in a process called crossover and further variation is introduced by applying random mutation. The idea is that as the GA progresses, the members of each generation evolve towards higher fitness until eventually finding a solution which meets the fitness criteria (*Rajasekaran and Vijayslakshmi 2011*). The process for a highly generalised GA can be summarised in the pseudocode below:

```
Initialise the population
WHILE fitness condition not met
      Evaluate fitness
      Prune population
      Selection
      Crossover
      Mutation
END WHILE
```

3.2 Artificial Chemistry

Artificial Chemistry (AC) is a sub-branch of Artificial Life (A-Life). A-Life is the study of the organisational features and processing mechanisms found within living systems through simulations, robotics and synthetic biology. Its working hypothesis is that by following the organisational principles of a living system, it is possible to artificially generate desirable behaviours characteristic of that system. In A-Life emergence through interacting components is a central concept. In a similar way, AC exploits the organisational features, computational properties and underlying mechanisms of chemical systems to create useful artificial representations. It can be broadly described as a "man-made system which resembles a chemical system" *(Dittrich et al. 2001*). There are two main branches of AC: Molecular Computing devices, where computation is achieved using real chemicals (*Dittrich et al. 2001; de Silva and Uchiyama 2007*), or alternatively, by utilising the principles of the Chemical Metaphor to construct novel software or hardware architectures *in silico* (*Dittrich et al.* 2001). The latter approach is termed Artificial Chemistry Computing (ACC) and is the focus of this literature review. In the Chemical Metaphor, data is stored in the form of molecular species and information processing occurs through interactions (reactions) between these molecules. The result of this computation appears as emergent global behaviour (*Dittrich et al. 2001*).

ACC is used in two main applications: simulating complex systems (biological, social or ecological) and in developing novel solutions to engineering or computational problems.

Most AC systems can be defined formally by three components. The first component, C, is the set of reacting chemicals where C= c_1, ... c_n and n may be infinite. Chemicals may be written as symbols, strings, numbers, or expressions. The second component, R, is the rules governing the reactions between the chemicals and these may be written in chemical notation form, for example:

$$2H_2 + O_2 \rightarrow 2H_2O$$

The general structure of a rule is shown below:

$$s_1 + ... s_n \rightarrow p_1 + ...p_n$$

Here, the components on the left-hand side (reactant species $s_1...s_n$) react and are replaced by components on the right-hand side (product species p_1 ...p_n) (*Dittrich 2001*). The final component, "A", is an algorithm describing the reaction vessel and how these rules are applied (*Dittrich 2001*). For example, the algorithm determines when and how often to apply the rules. It also defines the global parameters such as temperature, and pressure of the vessel.

ACC approaches can be broadly categorised into microscopic or macroscopic methods (*Dittrich et al. 1997*). Microscopic methods treat each molecule explicitly, while in macroscopic methods all the molecules of one type are represented by a value signifying, for example, concentration. The dynamics depend largely on whether the ACC representation of molecules is macroscopic or microscopic. Microscopic ACCs tend to model dynamics as stochastic molecular collisions, while macroscopic ACCs frequently model the dynamics as continuous differential or discrete difference equations. Alternatives exist such as combinations of the two, or metadynamic methods where the rules of the reaction vessel change over time (*Dittrich et al. 2001*).

3.3 AI Inspired by Reactions of the Endocrine System

In mammals, the endocrine system produces and stores chemical signals called hormones. These chemicals function to regulate bodily processes and maintain a steady internal state (homeostasis), despite continuous fluctuations in external environmental conditions. Hormones regulate almost all bodily processes including growth, mood, sleep, reproduction and

metabolism. They are secreted into the bloodstream and act at a cellular level by binding to specific receptors on target cells. These target cells commence a chain of signalling events and in this way alter their activity (*Alberts et al. 1989*).

Artificial Endocrine Systems (AES) are a category of recent AI techniques which take their inspiration from the mammalian endocrine system. A typical example is the DHS (Digital Hormone System) created by Shen et al. (*2004*). The DHS has been used as a control system for a robotic swarm to facilitate a number of tasks, such as, search and seize, self-repair, and forming sensor networks. In the DHS each robot is viewed as a biological cell that communicates with other cells (robots) via artificial hormones and executes local actions via artificial receptors. A robot generates hormones and these are received by connected robots, which can then modify and propagate the hormone message. The concentration of each hormone is a function of position and time and is described by ODEs. A robot selects an action based on a probability function which considers the current state of its sensors, local topology and its received hormones. Through the low-level interactions and activities, global high-level behaviour is generated and the swarm demonstrates properties of self-organisation (*Shen et al. 2004*).

Other AES models use artificial hormones as a means of reinforcement learning, where weighted connections of neural nodes are modified allowing their activity to be regulated. For example, the Artificial Homeostatic System (*Moioli et al. 2008*) was used to create an autonomous robotic control system. It was composed of two neural modules, each of which were extended versions of the GasNet neural model (*Husbands 1998*) (discussed later), and an AES as shown in *Fig 3.1*. Each GasNet module receives information from the environment and generates a particular action. These actions were regulated by switching among the modules using the AES. In this model, the AES is a system of coupled non-linear difference equations composed of three modules: the Hormone Level Repository (HL), the Hormone Production Controller (HPC) and the Endocrine Gland (EG). The HL records the level of hormone in the system, the HPC regulates the production of hormone appropriate to current internal state and external stimulation, and the EG is responsible for secreting hormones when required. The level of hormone alters the internal system state and regulates the outputs of the GasNet modules. The outputs of the GasNet modules were directly connected to the motors of the robot, and were responsible for adjusting its velocity. Using this system, the robot successfully demonstrates both obstacle avoidance and light chasing behaviour.

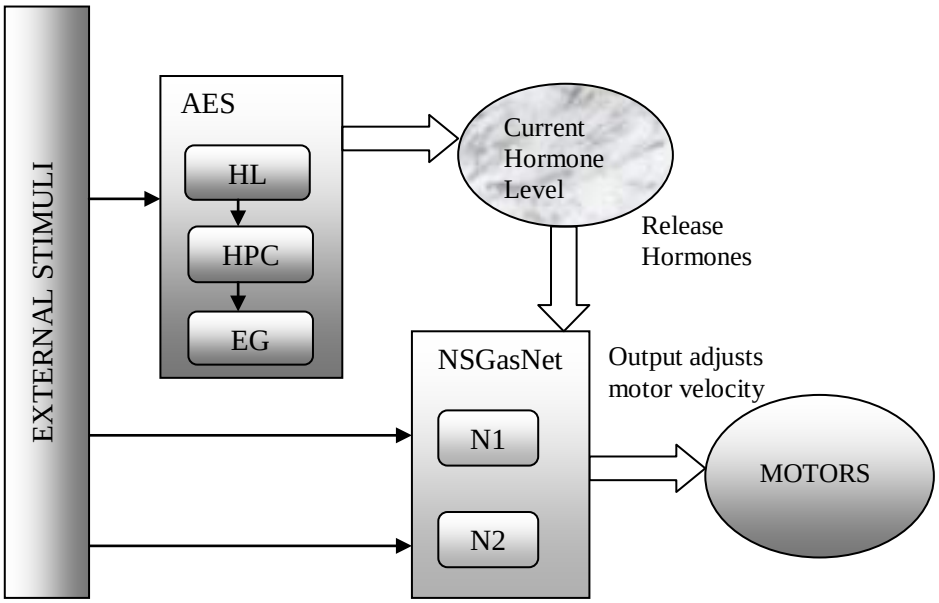


Fig. 3.1 The Artificial Homeostatic System

Adapted from Moioli et al. 2008.

Another AES robotic control paradigm was presented by Hamann et al. (*2010*) and is termed the Artificial Homeostatic Hormone System (AHHS). In this model, the inner body of a robot is segregated into virtual compartments, where each is associated with an actual physical part of the robots body (like an annelid worm). For instance, a sensor is mapped to a particular compartment. When triggered, the sensor secretes artificial hormone within the virtual compartment of the robot. Hormones diffuse between compartments through the body and result in the modulation in activity of the robotic actuators. Hormones were represented by numeric values and governed by rules where each hormone has a decay and diffusion rate, both of which were set by evolutionary computation. The dynamics of the hormone in each compartment is modelled by a system of ODEs. In Schmickl et al. (*2011*), a AHHS was used to generate motion in a simulated multi-modular reconfigurable robotic system. In this example, the modules were connected by hinges to form the robots body. The hinges were controlled by actuators which had to be coordinated to generate efficient motion. The AHHS was also used to control a wheeled robot. In this case, the wheel actuators were activated in correspondence to the level of hormone in their associated compartment, as shown in *Fig. 3.2*. For example, high levels of hormone secreted by the left proximity sensor, results in acceleration of the left wheel (for example, to implement obstacle avoidance). If hormone levels were the same in each compartment the robot drove forward in a straight line.

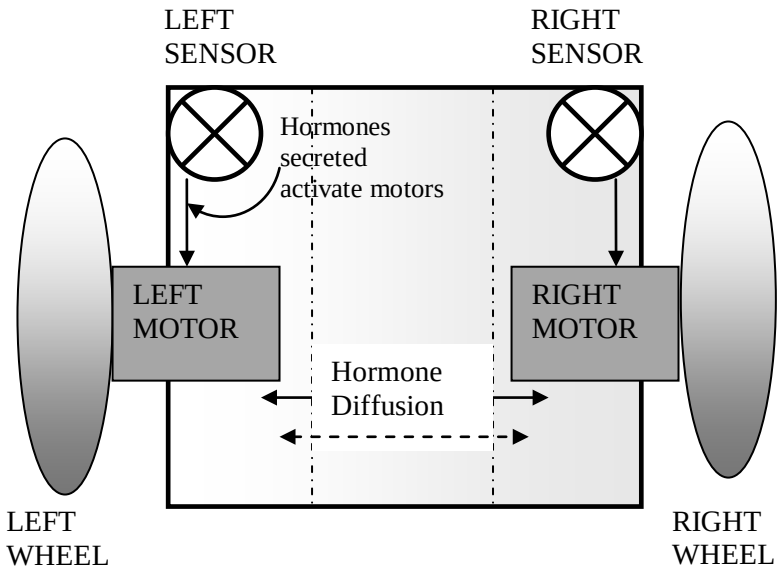


Fig. 3.2 The Artificial Homeostatic Hormone System (AHHS) for a wheeled robot.

Adapted from Schmickl et al. 2011.

Other models such as the Artificial Hormone System (AHoS) use artificial hormones to regulate task allocation, coordination, and management of heterogeneous units (*Brinkschulte et al. 2007*). This AHoS consists of three hormones: one to determine the quality of a unit to execute a particular task, one to suppress the execution of a task by a particular unit, and one to activate the execution of a task by a particular unit.

3.4 AI Inspired by Reactions of the Immune System

The immune system is responsible for defending an organism from disease by detecting and responding to foreign invaders. It is composed of molecules, cells, and organs spread throughout the body. This elaborate system must be able to distinguish between self (its tissues) from non-self (pathogens), and respond to a wide range of pathogens such as viruses, bacteria, and parasites. Typically vertebrates have two types of immunity. The first is that which is inherited at birth and is termed “innate immunity”, and the other, which is acquired over the lifetime of the organism, and is termed “acquired immunity”. On contact with a pathogen the innate immune system attempts to remove it. In most cases it is successful, however when it is not, the acquired immune system is brought into action (*Alberts et al. 1989b*).

The immune system is able to remember the antigens that it has previously been exposed to. This facilitates a faster secondary response which results in quicker removal of the infection.

Clonal Selection (*Burnet 1959*) describes the mechanisms by which the immune system remembers antigenic material. An alternative theory, proposed by Jerne (*1974*), is that of the Immune Network Theory which suggests an idiotypic network. Although this theory is not widely accepted it is often used by AI researchers as bio-inspiration for new computational paradigms. The idiotypic network involves simulation and suppression of cells via a network of communicating idiotopes. An idiotope is a marker on the antibody receptor molecule which can be recognised by receptors on other immune cells. If an idiotope is recognised by another immune cell, it connects. As more idiotopes are recognised by immune cells the network of connected cells increases. Recognition of the antigen by cell receptors results in network activation and cell proliferation, while recognition of a receptor by another cell's receptor results in network suppression. This implies that specialised immune cells communicate via a dynamic network, which continually adapts to maintain a steady-state reflective of antigen levels (*de Castro and Timmis 2002*).

Artificial Immune Systems are a class of AI techniques inspired by properties and mechanisms within the immune system. In general these techniques are either population based, for example, Clonal Selection Algorithms (*Aickelin et al. 2014*), which bare a strong resemblance to Genetic Algorithms without crossover. Or they are based on Jerne's idiotypic Immune Network Theory. The most relevant of these techniques are the latter and these are described below.

The autonomous and decentralised properties of Jerne's Immune Network Theory make it suitable for applications such as mobile-robot control, and identifying options for the configuration of communication software (*de Castro and Timmis 2002*). Many models based on this theory use the dynamics described by Farmer et al. (*1986*). While some do not use the Farmer dynamics (*Sathyanath et al. 2002*; *Opp and Sahin 2004*), those that do have far more relevance to this review as they approximate the biology very closely. Farmers' model is a differential equation which describes the change in concentration of antibodies with respect to the stimulatory and suppressive effects of the network and the natural death rate. The Farmer equation given in *Eq. (3.1)* (*Whitbrook et al. 2007*) describes the rate of change in concentration C of antibody x_i in a system of N antibodies $[x_1, x_2 \ldots x_N]$ and L antigens $[y_1, y_2 \ldots y_L]$.

$$\dot{C}(x_i) = b[A - B + C] - D \tag{3.1}$$

The first function "*A*" is a sum representing the stimulation of antibody x_i in response to all antigens. The second function "*B*" is a sum expressing the suppression of antibody x_i in

response to all other antibodies. The third function "*C*" is a sum representing the stimulation of x_i in response to the other antibodies. While the final function *D* represents the propensity for antibodies to die in the absence of interactions. The rate constant *b* models both the number of collisions at each time-step and the rate of antibody production on collision. The equation allows the determination of the concentration levels of hormones dynamically, and in this way the fitness of the robot to the current environment is calculated (*Whitbrook et al. 2007*).

Farmer based dynamics have been used in a number of behaviour arbitration systems for mobile robots described in (*Watanabe et al. 1998; Vargas et al. 2003; Luh and Liu 2004).* In each of these models, antigens represent environmental conditions, antibodies represent competence modules (i.e. simple actions or behaviours) (*see Fig 3.3*) and the dynamics were governed using the Farmer equation or variations of it.

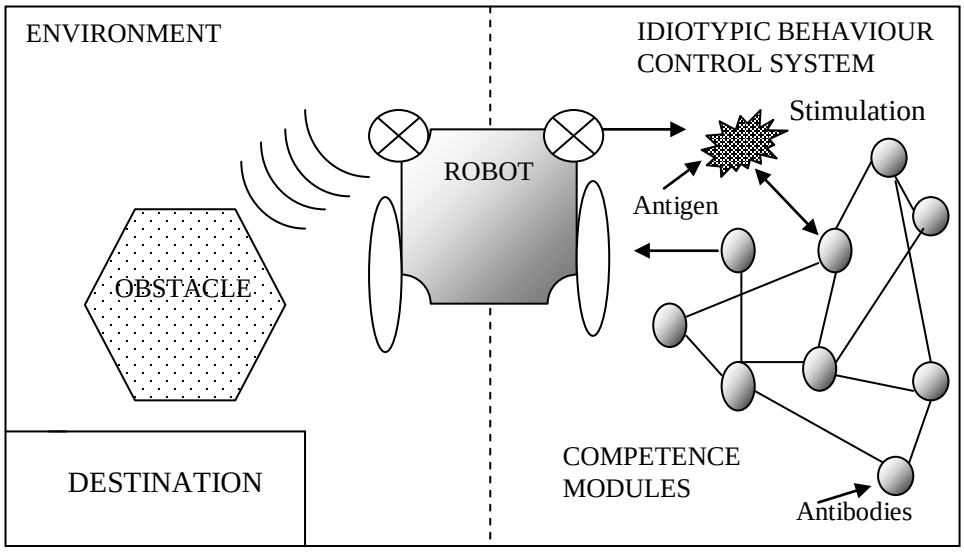


Fig. 3.3 Farmer based idiotypic network control system for mobile robot. Environmental conditions detected by sensors represent antigens and competence modules (actions) represent antibodies.

For example, Watanabe, et al. (*1998*) develops a garbage collecting robotic control system. In this model, antigens represent environmental signals detected by sensors, and antibodies represent prepared competency modules (behaviours). The basic idea is that the immune system selects a competence module (antibody) suitable for the detected current situation (antigens). Modules were stimulated and suppressed by interactions with antigens (sensor inputs) and with other modules. The concentration of each antibody at each time-step was calculated using Farmers equation. The actual module implemented at each time was chosen using roulette-wheel selection, based on the antibodies with the highest concentration after idiotypic interactions (*Watanabe et al. 1998*). Other similar idiotypic approaches were used by a number

of researchers. For instance Krautmacher and Dilger (*2004*) used a Farmer based idiotypic model to control a simulated robot navigating through a maze. While in Luh and Liu (*2004*), such a system was used to control a robot where antibodies represented steering directions and antigens a set of data, including sensory input. The idiotypic network approach has the advantage that it is self-regulating and continually adapts itself to maintain a steady-state suited to the environmental conditions. It has been shown to have benefits over reinforcement learning approaches in behaviour arbitration systems; for example, reduction in premature convergence, and a decrease in repeated behavioural patterns; however, this has not been thoroughly investigated (*Whitbrook et al. 2007*). In all the examples outlined above the idiotypic controllers were not compared with benchmark systems and thus it is not clear how idiotypic systems contribute to performance. Furthermore, in the majority of cases simulations were used, and thus it is difficult to ascertain how such a system would cope in a real-world environment (*Whitbrook et al. 2007*).

3.5 AI Inspired by Reactions of the Metabolic System

The metabolic system is responsible for the set of chemical transformations within the cells of an organism which enable it to continue to function, grow, and reproduce. There are two processes involved: catabolism and anabolism. Catabolism is the process by which complex organic constituents are broken down to release the energy which is used within the body. Anabolism is the construction of cell components such as proteins. In metabolic pathways, chemicals are transformed into others through a series of enzymatic reaction steps. Enzymes are essential to the metabolic system: they act as catalysts by allowing reactions to proceed quickly and efficiently, and can regulate the steps required in response to signals from the environment or other cells (*Alberts et al. 1989c*). Some computational models used in AI are inspired by the metabolic system and in this section the most relevant of these models are reviewed.

Zeigler and Banzaf (*2000*) present an Artificial Metabolic control system for a simulated mobile robot. This is a typical AC model, consisting of a set of molecular species, a reactor and reaction rules - as previously described in *section 2.4.4*. Each molecular species is represented by an algebraic symbol and each has a concentration value at each time-step. These species may have catalytic or inhibitory effects on other reactions, for example, by changing their reaction rate. The authors applied this model to implement a control system for a simulated Khepera robot. The robot was equipped with proximity sensors. On stimulus a sensor produced a signal and this was used to represent influx of substance. This input was connected through a network to motors, which, on reaching a particular threshold, updated the direction or speed of

wheel rotation as shown in *Fig. 3.4.* The objective of the robot was to navigate through a maze while avoiding obstacles. Genetic Programming was used to evolve the Artificial metabolic system. A population of solutions were setup consisting of a set of reaction rules arranged as a weighted directed bipartite graph. The outcome of each solution consisted of a time dependant vector of concentrations. These output concentrations were input to a fitness evaluation function.

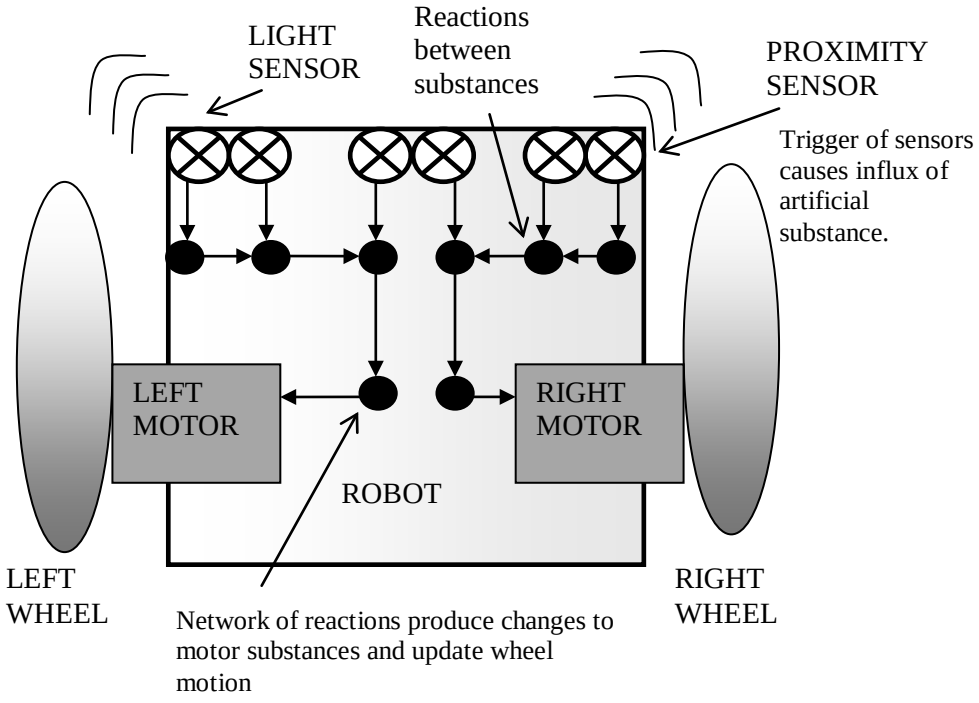


Fig. 3.4 Artificial Metabolic control system for mobile robotic control.

Another artificial metabolism model referred to as an Artificial Enzyme Substrate Chemistry (*Ziegler et al. 1998*) was used to control a physical robot with eight proximity sensors and eight light sensors in obstacle avoidance and light seeking tasks. The system was implemented in a similar manner to the previously described model. In this case, the reaction rules were implemented using ODEs. The network was manually configured rather than evolved. The artificial metabolism was shown to control the robot in the desired way, and be robust against disturbances.

The Artificial Biochemical Network (ABN), presented by Macleod and Capanni (*2010*), is a highly abstracted discrete model of a biochemical signalling network. Its inspiration was taken from metabolic and signalling networks in living systems in general. This model has been applied to robotic control to recognise external environmental stimuli and respond by

generating the appropriate robotic gait as shown in *Fig. 3.5*. It is a connectionist paradigm which consists of a number of nodes. Each node has a weighted sum activation function and Leaky Integrator. In this model, pulsed binary input signals were fed into the system. Each node consolidates these signals as inputs to the activation function and a pulsed binary output is produced. Information is encoded by the pulse on or off periods. By means of a GA the desired pattern of output pulses may be set for each specific pulsed input signal. This produces a sequence of actions in response to a single input pattern, allowing each robot actuator to be switched on or off in correspondence to a particular output signal.

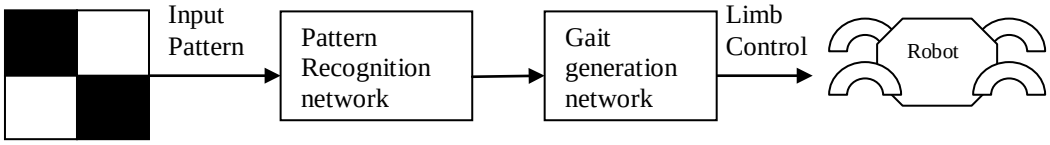


Fig. 3.5 ABN control system applied to pattern recognition and generation of temporal gait patterns.

Adapted from Macleod and Capanni (2010)

This system has advantages over benchmark alternatives in the control of limbed robots. For instance, traditional Artificial Neural Network models, as discussed later, are only able to recognise static patterns and produce a single response. While other neural models, such as spiking neural models, can recognise and produce time varying signals, but are generally more complex than the ABN.

3.6 AI Inspired by Reactions of the Nervous System

The nervous system consists of the components of an organism which allow rapid processing and response to external stimuli to coordinate voluntary and involuntary actions. Nervous systems are present in most multicellular animals and, depending on the species, exist in various levels of size and complexity. Advanced nervous systems are composed of two main components: the Central Nervous System (CNS), consisting of the brain and the spinal cord and the Peripheral Nervous System (PNS) which connects the CNS to other parts of the body. At the cellular level, the nervous system consists of two categories of cells: neurons and glial (neuroglia) cells. Glial cells function to maintain and facilitate the function of neurons. For example they supply nutrients to neurons, insulate them and remove their waste. Neurons are responsible for sensing, processing and propagating signals in response to stimulation from other cells or the outside world. Communication between cells occurs electrically via gap junctions or more commonly chemically via synaptic neurotransmitters. A neuron has a highly

specialised structure composed of the cell body, dentrites, and axon, as shown in *Fig. 3.6*. The signal, called an action potential, travels along the axon of one neuron to the dendrites of another. This, in turn, may cause the second neuron to activate or be inhibiting. Neurons are arranged into an elaborate network, which ultimately allows an organism to dynamically process information and respond to its surrounding environment (*Alberts et al.1989d*).

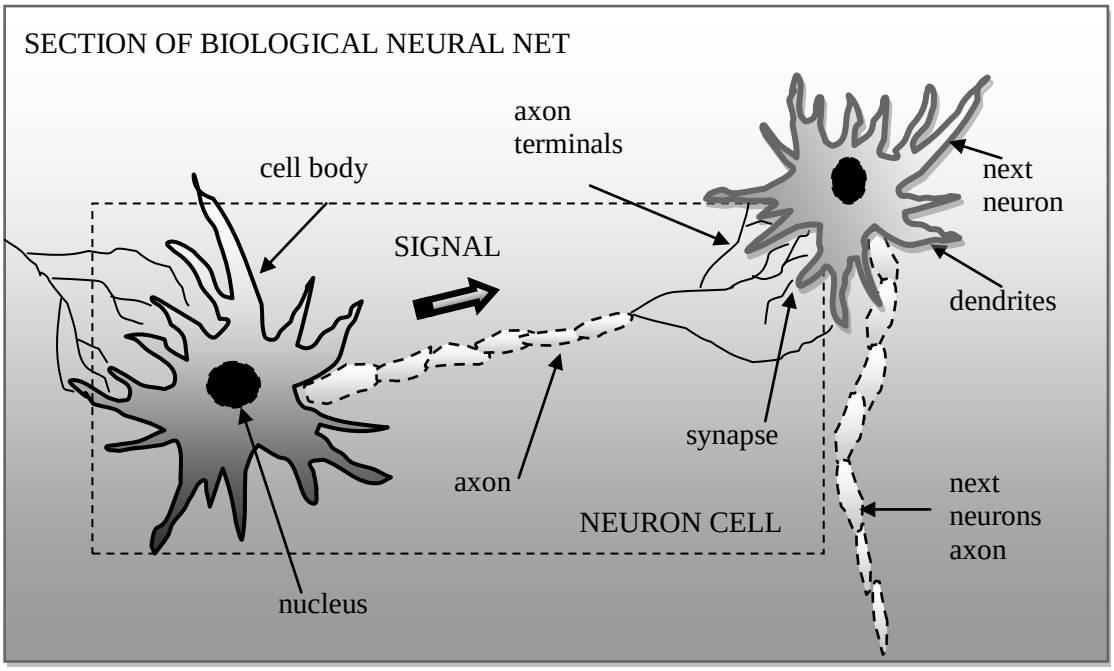


Fig. 3.6 A section of a biological neural network.

Artificial Neural Networks (ANNs) are computational methods inspired by biological neurons and their networked organisation. In general they consist of multiple artificial neurons linked together through weighted connections and, for this reason, are often referred to as "connectionist" representations. They have many applications such as classifying patterns, processing data, time series prediction, and robotics.

The first generation of ANNs were introduced in the 1940s by McCulloch and Pitts (*1943*). Their MCP neural model consists of a node which has a number of weighted incoming connections. The node summed the weighted inputs together and then passed the result through a threshold activation function. If the sum was equal to or above the fixed threshold value, the neuron produced a binary output of one. In the 1950s, Rosenblatt (*1958*) developed these ideas and introduced a more advanced threshold ANN called the "Perceptron", as shown in *Fig. 3.7*. The Perceptron was more flexible and had a learning rule to adjust the weights. Later, a more advanced learning rule was applied called Backpropagation (*Werbos 1990*).

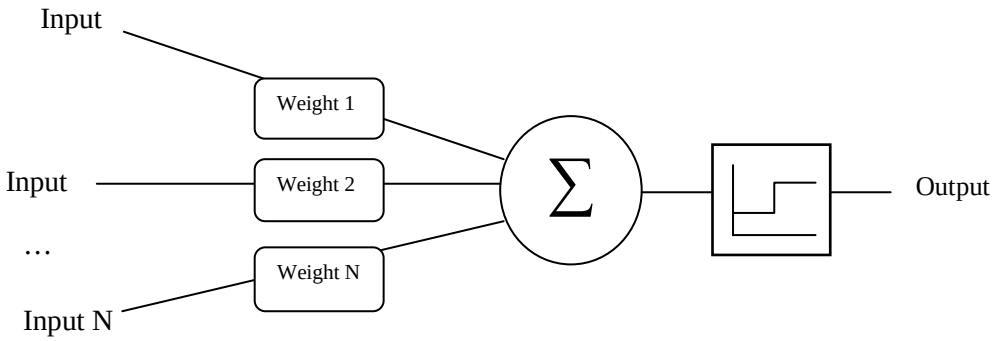


Fig. 3.7 Structural overview of the Perceptron

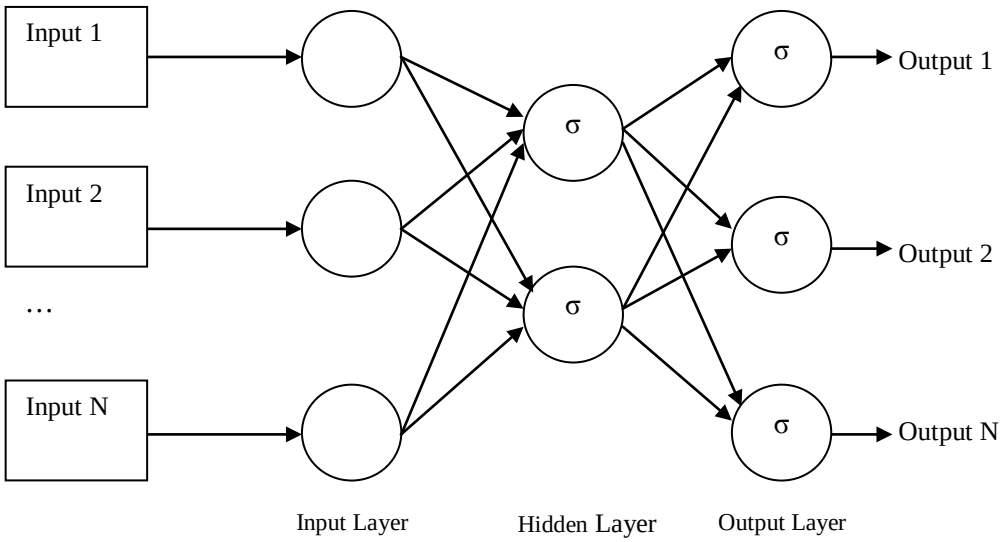


Fig. 3.8 Multi-layer Perceptron where σ represents a sigmoidal activation function

This was later extended into the Multi-Layer Perceptron (MLP) (*Block et al. 1962*) as shown in *Fig 3.8*, where multiple layers of nodes are arranged in a directed graph with each layer fully connected to the next one. This second generation of ANNs did not use a threshold activation function; instead, their activation function was continuous. This allowed them to produce analogue outputs. Commonly used examples of activation functions used in these models are the sigmoid and hyperbolic tangent. Such ANNs are more powerful then the first generation, as they, due to the invention of the Backpropagation training technique, allowed multiple layers to be connected. The ability to be arranged into layers greatly increased their computational power

(*Ghosh-Dastidar and Adeli 2009*). Many second generation ANN models have since been created; examples include the Self-organising Map (SOM) (*Kohonen 1982*), Spiking Neural Networks (*Ghosh-Dastidar and Adeli 2009*), and Recurrent Neural Networks (*Hopfield 1984*).

Real biological neurons produce a pulsing (often called "spiky") output with many action potentials. Neurological research has shown that biological neurons encode information in the spatio-temporal properties of these signals, specifically their frequency. A third generation of neural networks are called Spiking Neural models (*Ghosh-Dastidar and Adeli 2009*). These types incorporate the temporal dimension found in their biological counterpart. System information is pulse encoded by the timing between spikes (frequency). Pulse coding is a computationally powerful method in cases where temporal information needs to be processed. One of the most commonly used methods is the Integrate-and-Fire model. The output of a spiking neuron is typically modelled using a differential equation, which only fires when the current activation level reaches a threshold value. Normally the neuron is a Leaky Integrator, where the activation gradually decays over time. Learning algorithms traditionally used in first and second generation neurons such as Backpropagation are generally unsuitable due to the complex spatio-temporal dynamics of spiking neurons. Other learning methods such as unsupervised reinforcement learning algorithms are often applied.

An important ANN method is GasNets and was introduced by Husbands (*Husbands 1998*). This was inspired by synaptic modulation of nerve cells by nitric oxide (NO). A GasNet consists of a number of nodes connected by excitatory or inhibitory weighted links. The important difference between this and other traditional ANNs is that some units can emit artificial gases. These gases are able to modulate the behaviour of other units by adjusting parameters within their activation function. This is an abstraction of the reinforcement learning that is found in biological neural networks, where the network trains using either "good" or "bad" feedback signals. The nodes are modelled on a 2D plane and a gas diffusion model is used to calculate the concentration of gas generated from each emitting node. This allows the network to regulate its processing properties while it is in operation (*Husbands 1998*).

Central Pattern Generators (CPGs) are neural oscillators situated in the ganglia or spinal cord of animals which function to generate rhythmic patterns. They are essential in coordination of, for example, locomotory gaits and respiration. Their rhythmic output coordinates the physical parts of the organism involved in the activity. Sensory feedback from the environment allows CPGs to continuously adapt to the situation and provide, for example, stable motion in unpredictable terrain (*Nakada et al. 2004*). Many researchers have designed and implemented artificial CPG's, particularly for application in generating robotic gaits. A stylised CPG output for a bi-

pedal walk gait is shown in *Fig 3.9*. Most CPGs have been constructed with coupled non-linear oscillators which are composed of non-linear differential equations - such as the Wilson-Cowan neural oscillator (*Wilson and Cowan 1972*). Kimura et al. (*2001*) developed a CPG based control system using a system of coupled non-linear oscillators for a quadrupedal robot capable of adapting to irregular terrain. In other work, Billard and Ijspeert (*2000*) generated quadrupedal gait patterns for an AIBO robot using a CPG-based controller.

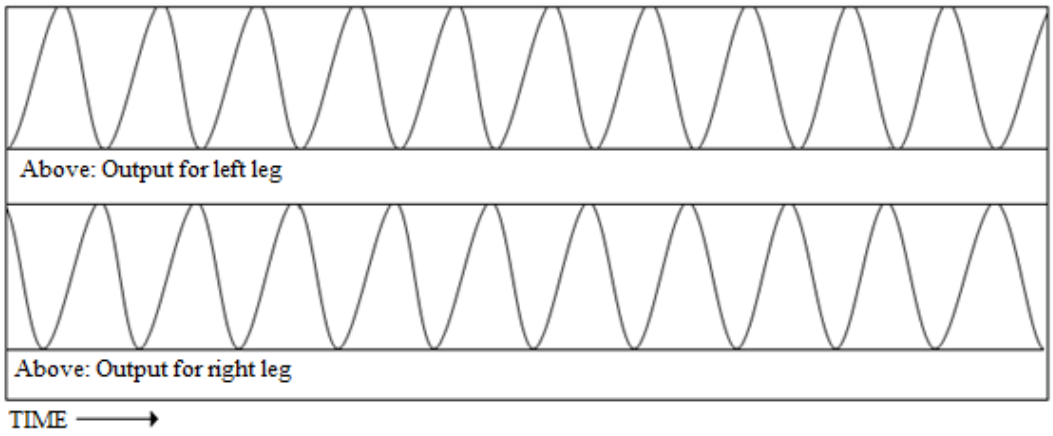


Fig. 3.9 Central Pattern Generator output for two-legged robot to control a walk gait. The generated signals are out of phase with each other.

3.7 AI Inspired by Reactions of the Gene Regulatory System

Genes are segments of DNA whose sequence of nucleic acids encode functional products - either proteins or RNA. A number of mechanisms interpret this code and synthesise these products. All cellular processes, for example, metabolism, growth and reproduction depend on genes to allow their essential molecular components to be generated. The machinery that regulates the expression levels of a gene is called a Gene Regulatory Network (GRN). Genes are able to interact indirectly with others through their products or other molecular substances in the cell, and in this way form a network of connections. In this network, the proteins, RNA and genes can be represented as nodes and their edges the reactions between them. Genes are switched on or off by elements in the GRN. The state of the set of genes at a particular time signifies the result of processing within the network at that instance. In this way, genes are stimulated or inhibited and thus their expression levels are controlled. GRNs respond to environmental signals allowing the cell to adapt its behaviour at a given time and optimise its chance of survival. This process is shown in *Fig. 3.10.*

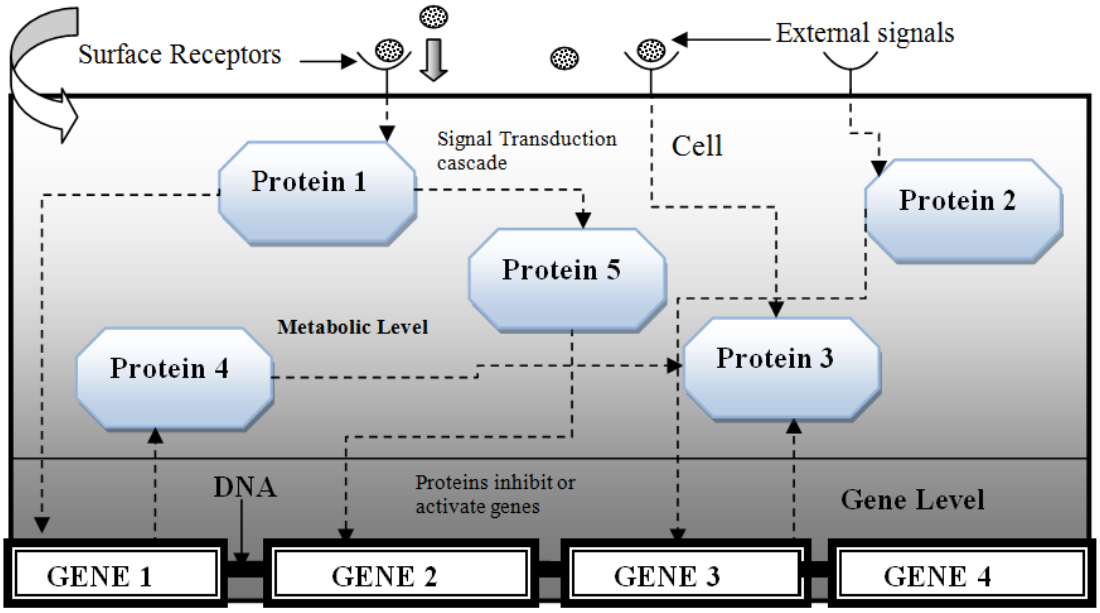


Fig. 3.10 Overview of a biological Gene Regulatory Network (GRN).

A number of AI techniques are inspired by GRNs and may be referred to as Artificial Gene Regulatory Networks (AGRNs). Examples of the most relevant of these techniques are given below.

In (*Banzaf 2004*), it was shown that an AGRN can be evolved to generate a simple series of functions such as sinusoids, exponentials and sigmoids. This model was then used to explore the idea that successful evolutionary design is best achieved through a networked system (*Banzaf 2004*). In this system, the AGRN, which is termed an Artificial Regulatory Network, consists of a genome, represented by a bit string, as shown in *Fig 3.11*. The genome is generated by randomly determining the value at each position. Certain bit patterns within the genome represent a gene which encode a particular protein. Proteins are expressed using a computational abstraction of genotype to phenotype mapping. After a protein is generated, it is able to wander around in a fixed space where it may interact with regulatory sites of the genome. If the constitute bit pattern matches at the position in which a protein and genome interact, than the protein will attach to the genome. Attachment of such proteins (termed transcription factors) at regulatory sites on the gene, have inhibitory or stimulatory effects on the expression of the particular protein which they encode - thus allowing their expression to be regulated. The time this attachment lasts depends on the quality of the match between these patterns.

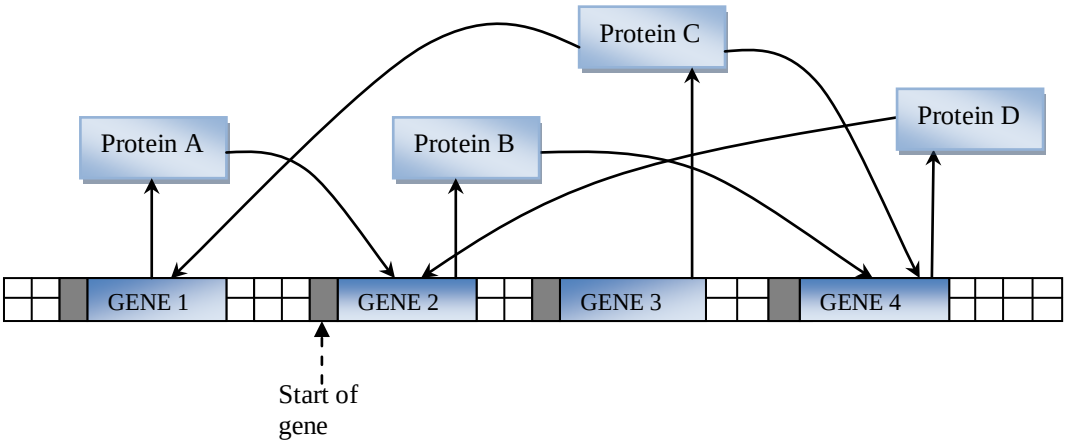


Fig. 3.11 The Artificial Regulatory Network

Adapted from Banzaf, (2004)

Another example of a GRN inspired method is the 'Distributed Gene Regulatory Network Algorithm' for multi-robot construction tasks by Guo et al. (*2009*). Using this model, simulated multiple robots were able to self-organise from random initial positions to form different pre-defined 2D geometric shapes. For example, multiple robots together may form the shape of a square. Later, moving obstacles were added to the environment, and it was shown that the robots were able to maintain their function as before, while avoiding the obstacles. In this system, each robot represented a cell and each had a genome which contained two genes- where one controlled the robots x-coordinate and the other its y-coordinate. Each gene produced a protein, which had three functions. Its first was to regulate expression of the gene that produced it, the second was to communicate its proximity to neighbouring robots allowing it to avoid collisions, and the third was to adjust the robot's position. The dynamics of the AGRN system was modelled using ODEs, which described the protein expression level for both genes and the concentration level for each protein.

3.8 AI Inspired by Generic Chemical Reactions

Obviously, systems of chemical reactions are ubiquitous in the natural world. Some AI methods are inspired by chemical processes in general rather than particular types of signalling pathways which occur in living systems. In this section the most relevant of these are reviewed.

One such method is the Chemical Casting Model (CCM) (*Kanada and Hirokawa 1994*). This is inspired by the inherent computational processing which occurs in systems of chemical

reactions. The CCM is a stochastic method and is used to solve state-space search problems. A unit of data is represented by an atom. Each atom has an internal state and may be connected to others by links to form a set of atoms called a molecule. In this way, data structures such as a tree, list or graph can be represented. Reaction rules change the state of the system and are written as production rules. The syntax of the reaction rules is (as in real chemistry):

$$\text{LHS} \rightarrow \text{RHS}$$

Where the left-hand side (LHS) and right-hand side (RHS) defines a pattern of atoms. The reaction rule may be applied when a molecule matches the LHS pattern and when a local evaluation function condition is true. When a rule is applied, the matched atoms disappear and are replaced by the RHS pattern. In this way the system moves from the initial state which represents the problem, to the final state which represents the solution. A similar method was used to solve the travelling salesman problem by Banzaf (*1990*). In this system, an initial soup of data strings resembling macromolecules was setup. Each string of data carries all the information required to create a solution to the optimisation problem. This soup of macromolecules is evolved by using specialised machines which represent enzymes which transform randomly selected molecules into new ones. In this way the system moves from one state to another until a solution is generated.

The AI techniques outlined in this chapter highlight the way in which elements of CSNs and chemical reactions in general have provided inspiration to an array of useful AI techniques. It was shown that many of the identified methods used in AI overlap with those used in biological simulation. The reviewed techniques were selected because they capture the properties and mechanisms (as identified and discussed in *chapter 2*) which contribute to high-level behaviour within cells. However, none of the methods discussed here or in *chapter 2* provide a means to capture these properties and mechanisms in one representation. In the following chapter, particular aspects of these techniques are combined in order to construct a representation capable of modelling the identified properties and mechanisms of Cell Intelligence in a single representation. This representation will later be used to explore the role of Cell Intelligence within AI.

4. The Artificial Reaction Network

This chapter presents a new Artificial Chemistry (AC) representation termed the Artificial Reaction Network (ARN). Its formulation, networked representation, computational properties advantages and disadvantages, and finally its verification are discussed. The purpose of this model is to provide a tool, well-suited to represent the "biological circuitry" responsible for generating high-level behaviour found in cells. Although representations exist for modelling chemical interactions, as discussed in *chapter 2*, none are ideal for this purpose. Thus, within the spectrum of techniques presented in *Fig. 2.6*, there is a niche for a new technique, which provides the appropriate balance between detail and scope. To achieve this, the ARN combines elements from existing AI and Systems Biology techniques, and like many AC models, as will be later discussed, it has application in both biological simulation and in solving engineering and computational problems.

4.1 Basic Formulation

As discussed in *chapter 2*, CSNs have a number of properties including compartmentalisation, temporal dynamics, and their ability to form complex networks which in turn, may function as control structures. To represent such properties elegantly and appropriately it is not practical to model each molecule individually. As previously discussed in *chapter 2*, for coarse approximations, the amount of species present may be represented by its concentration, and its temporal dynamics modelled using ODEs. The accuracy of such models may be further enhanced by setting up spatial compartments to provide a coarse approximation of the spatial distribution of chemical species. This level of abstraction incorporates the degree required to create elaborate biological circuitry while still maintaining computational efficiency and thus examine the emergent characteristics of Cellular Intelligence.

As described in *chapter 2*, rate equation models can be used to represent many different physical systems and so are very general and flexible in their applications. In the domain of chemistry, they can directly represent (or be slightly modified to represent) all the common reaction types. As described in *chapter 2*, they form the basis of S-Systems (*Voit 2000*) and so

are very well characterised in biochemical simulations. The basic rate equation is given in *Eq. (4.1)*.

$$\frac{d[P]}{dt} = k_f \prod_{n=1}^{N} [R_n]^{\alpha_n} - k_r \prod_{i=1}^{M} [P_i]^{\beta_i} \tag{4.1}$$

The equation contains two terms. The first term corresponds to the basic forward reaction. Where the rate of change of product (P), is given by the forward reaction rate (the speed of the reaction) labelled k_f, multiplied by the product of the concentrations of the N reactants $[R_n]$, each raised to the power of its reaction order α_n. The second term represents the decomposition of product back into its original reactants. This depends on the reverse reaction coefficient k_r multiplied by the product of the concentrations of the M products $[P_i]$, each raised to the power of its reaction order β_i.

To illustrate this further, consider the simple reaction between two reactants labelled A and B with reaction orders (the proportion of molecules of each reactant used to make a molecule of product) of q and s respectively. The order of the reverse reaction is one. These produce a single product P. Equation (4.1) then reduces to *Eq. (4.2)*.

$$\frac{d[P]}{dt} = k_f [A]^q [B]^s - k_r [P] \tag{4.2}$$

When used in S-Systems, a group of rate equations are normally set up - one for each reaction. The left-hand of each equation is then set to zero and they are solved simultaneously to yield the steady-state response. If the dynamic responses are required, then numerical solution methods like Runge-Kutta are normally applied (*Voit 2000*).

4.2 A Networked Representation

Clearly a large set of simultaneous ODEs written in their basic mathematical form limits the conceptualisation, visualisation and communication of complex topologies. Furthermore, in this form, each ODE term is tightly coupled, and is difficult to isolate and manage. Therefore, in order to create a networked representation with distinct biological processing units, capable of constructing complex biological circuits, the method needs to be modified. This may be done by isolating each reaction in the network to form a discrete node, which may then be modified independently of the other reactions. Such a node can be viewed as analogous to a neuron in an

ANN and has been named an Artificial Reaction Node. By analogy, networks of such nodes may be termed Artificial Reaction Networks (ARNs). Similarly to an ANN, each ARN node is a processing unit, transforming a number of inputs into an output. In an interconnected network of such units, global behaviour is determined by the connections, and unit parameters. Furthermore, by isolating each reaction like this, the individual pathways or units which make up the system can be changed, reconnected or evolved by (for instance) a Genetic Algorithm. This also allows an individual part of the network to be independently modified, and its effects studied.

Isolating the reactions in a network in this way facilitates two other important practical advantages. Firstly, visual "drag and drop" interfaces can be developed. These allow researchers to quickly change network or reaction parameters in order to study their effect. This, in turn, allows simple visualisation of the system in a graphical form which makes its conceptualisation easier. Secondly, it makes the application of object-orientated programming techniques very simple, as each node can be coded as an instance of an object.

In developing the system described, it was decided to use Euler's method in order to solve the rate equations. This offers some advantages, firstly it is simple and computationally cheap, but more importantly, it allows the whole network to run quickly in simulated real time- so that its temporal dynamics can be seen to unfold during a run. This gives the option of changing parameters in real time, so that a user can observe any resulting dynamic behaviour. Also, unlike linear multi-step methods, it does not require previous derivatives to calculate the value at each consecutive time-step, thus it contributes to stability and isolation at each node. Furthermore, as will be discussed later, the temporal output of the network could potentially be used as a control system for an "artificial cell" robot - a Cytobot.

Using the simple two input system shown in *Eq. (4.2)*, multiplying through by *dt* and changing to a discrete finite time-step Δt, the Euler approximation for each time-step is given by:

$$\Delta[P] = (k_f[A]^q[B]^s - k_r[P])\Delta t \qquad (4.3)$$

The reaction needs to be isolated from the others, so that it can form a discrete "unit". This can be done most easily by borrowing the concept of "pools" from Petri Nets (*Murata 1989*) (*see section 2.4.4*). Petri Nets pass tokens between such pools as part of their operation. In the system discussed here, the pools may hold the number of available molecules, the concentration of the reacting chemicals (for example in moles per litre) or the mass of reactants. As the

reaction proceeds, the reacting species pass from the input pools (depleting them) to the output pools (enriching them). So, in the previous example, to generate one molecule of product requires q molecules of reactant A and s molecules of reactant B. In this case, the pool containing A would get depleted by an amount ΔA given by:

$$\Delta[A] = \Delta[P]q \tag{4.4}$$

Where ΔP is the amount of product generated (which would be added to pool P). This equation works if the units used are number of molecules or moles per litre (which are not conserved quantities). However, if mass or a similarly conserved quantity is used then *Eq. (4.4)* becomes:

$$\Delta[A] = \Delta[P]\frac{q}{q+s} \tag{4.5}$$

Using more general symbols, the whole system is shown diagrammatically in *Fig. 4.1* (for a conserved quantity). It comprises a set of connected reaction nodes (circles), pools (squares), and inputs (triangles). Each pool represents the current available protein species concentration (*avail*) and each circle corresponds to a reaction unit, representing an interaction (reaction) between a number of chemicals.

While many ACs assume a well-stirred reactor, the use of pools within the ARN provides a discrete spatial structure. Inside a biological cell, concentrations of chemical species are spatially distributed into localised compartments. This compartmentalisation, as discussed previously, restricts which molecules may react together, and thus affects the overall dynamics of the system. Representing the spatial distribution of chemicals allows fine-grained control over the system dynamics and thus is highly beneficial when modelling biochemical circuitry. For example, it allows the representation of flow structures such as membrane channels, transport processes; and functional network motifs (as described in *Table 2.1*), and provides a means to explore disease pathways (*Tyson and Novak 2010*). Thus, in the ARN, each pool is represented as a well-stirred reactor and approximates a spatial compartment.

Optionally, a loss component can also be added to the pools, allowing a pool to function as a Leaky Integrator (*Pavan et al. 2013*). This could represent the destruction of reactants or products by specific or general proteases or other degradation routes as shown in *Fig. 4.1*.

Inputs (shown as triangles in *Fig. 4.1*) are a special type of pool; the only difference is that they are of fixed value and thus can be used to represent the continuous supply of environmental inputs or enzymes (in other words, a limitless resource).

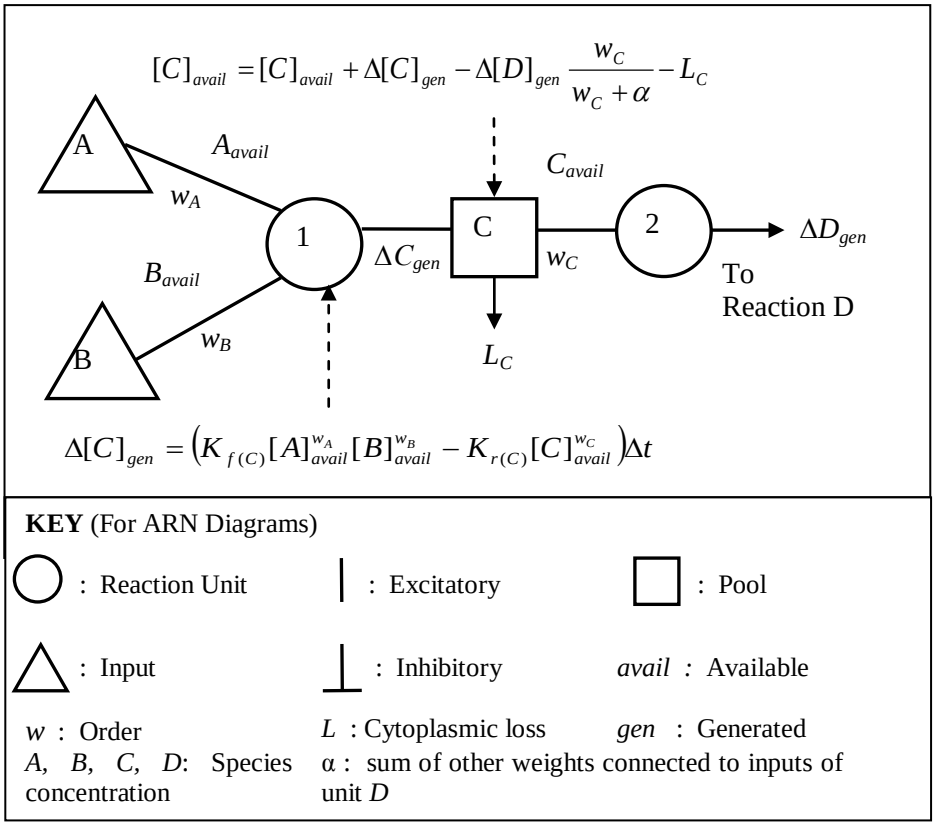


Fig. 4.1 The Artificial Reaction Network (ARN).

Connections symbolise the flow of species into and out of reaction units and their weight (*w*) corresponds to reaction order. The connections can be either excitatory, or inhibitory. A reaction with both excitatory and inhibitory connections will proceed if all connected inhibitory pools are empty and its excitatory connected pools have the required concentrations. Thus the input pools, serve as pre-conditions to the reaction, which must be met before the reaction can proceed. The inhibitory connections serve as discrete on/off switches to either the forward or reverse reaction.

The overall structure may be compared to a Perceptron, where the pools correspond to inputs, the reaction units to the weighted sum function, and these are joined together by weighted connections (*Block et al 1962*)

4.3 Computational Properties

It is fairly easy to see that the computational properties of the ARN are similar to those of the ANN. For example, consider the simple network shown in *Fig 4.2*.

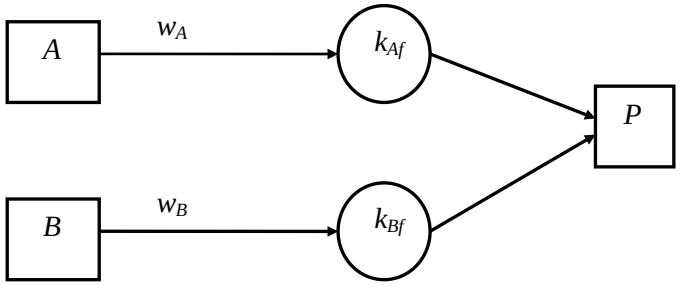


Fig. 4.2 A simple ARN network with 2 reactions.

If we assume that the orders w_A and w_B are unity and the reverse reaction rates are zero, then the rate of change of the product pool P is described by *Eq. (4.6)*.

$$[\dot{P}] = k_{Af}[A] + k_{Bf}[B] \tag{4.6}$$

$$[\dot{P}] = k_{Af}[X] + k_{Bf}[X]^2 + k_{Cf}[X]^3 \tag{4.7}$$

$$[\dot{P}] = \sum_{n=0}^{N} k_{Rnf}[R]^n \tag{4.8}$$

$$[\dot{P}] = k_{Af}[A]^{w_A}[B]^{w_B}[C]^{w_C} \tag{4.9}$$

$$[\dot{P}] = \sum_{n=0}^{N} k_{Rnf} \prod_{m=0}^{m} [R]_m^n \tag{4.10}$$

Which is the same expression as for the activity of a perceptron if A and B are the inputs and the k terms are the weights. So a network of such nodes has at least the same computational capabilities as MLPs (*Rumelhart and McClelland 1986*). In fact, the addition of non-unity orders means that effectively the node can produce non-linear separators in a similar way to

polynomial neurons (*Woo and Khor 2004*) and are rather similar to so-called "sigma-pi" units (*Gurney 1992*) - although with the added dimension of their temporal dynamic behaviour. For example, consider the ARN shown in *Fig 4.3*. Then by increasing w_B to 2 and setting w_C to 3, the first three terms of a power series are generated - as shown in *Eq. (4.7).* By adding other reactions, N further terms may be generated as shown in *Eq. (4.8).*

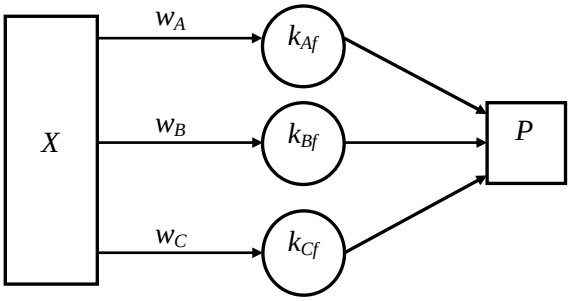


Fig. 4.3 A simple ARN network with three reactions and one input.

In the previous examples, the computational properties were examined in cases of a single input to each reaction unit. The remaining possibility is to completely deconstrain the system and allow multiple reactants to enter a single reaction unit. For example, the ARN given in *Fig. 4.4* shows one reaction with three inputs. The corresponding function for this reaction is given *in Eq. (4.9).* This simply extends the power series to multiple dimensions (in a similar way to the linear separator line of a two input MLP becoming a hyperplane for multiple inputs). The result of several such units feeding into a single pool would be described by *Eq. (4.10).*

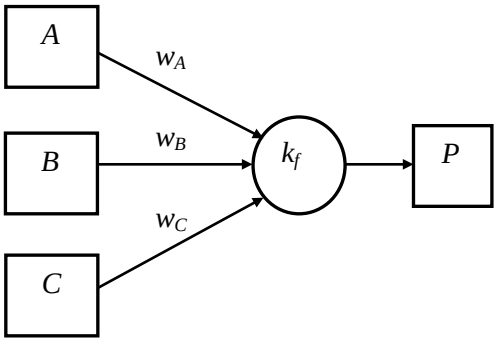


Fig. 4.4 A simple ARN network with 1 reaction and 3 inputs.

Two other models, which bears some resemblance to the ARN, exist in the literature. The first of these is the Artificial Biochemical Network (*Macleod and Capanni 2010*) introduced previously in *section 3.5*. This is a highly abstracted model of a CSN intended for robotic

control. The network consists of pulse width, position or frequency modulated units, which are useful for motor control but are not directly feasible biochemically like the ARN. The second model is the Artificial Biochemical Neuron (*Eikelder et al. 2009*) - as previously introduced in *section 2.4.3*. This model is based on the steady-state response described in the Michaelis-Menton equation. It has been used to represent phosphorylation cycles within CSNs. However, the basic idea has not been further developed in the literature.

4.4 Disadvantages

There are some potential disadvantages associated with the ARN. Firstly, in order to generate temporal dynamics, the ARN must use a method of numerical approximation, and thus it adopts the disadvantages of the chosen method. Numerical approximation methods and their associated limitations are well documented (*Uri et al. 1995*). For the reasons described in *section 4.2*, it was decided to use Eulers method throughout the experiments outlined within the thesis, although it would be possible to use other methods. As the disadvantages of Eulers method are relevant to the results achieved throughout the thesis its disadvantages are included in this section.

Eulers method has an associated cumulative error. This is because it is an iterative linear approximation to a complex function. It may be thought of as the first-order term of a Taylor expansion of the function. So, for example, if we say that the rate equation is a function of at set of reactants and products **R**, we could write an abbreviated version of *Eq. (4.1)* as *Eq. (4.11)*. The Taylor series for the Euler approximation to the third order is then given by *Eq. (4.12)*.

$$\frac{d[P]}{dt} = f(\mathbf{R}) \tag{4.11}$$

$$R_0 + f(\mathbf{R})\Delta t + \frac{\Delta t^2}{2!}\frac{d^2 f(\mathbf{R})}{dt^2} + \frac{\Delta t^3}{3!}\frac{d^3 f(\mathbf{R})}{dt^3} + \ldots \text{and so on} \tag{4.12}$$

Because the series is truncated to the linear term, the error of the approximation is the sum of the missing terms. In reality, the error contribution from successive terms is approximately Δt^2 because error from the higher order terms diminishes rapidly and is usually negligible, providing that the step-size is small (*Greenburg 1998*). The error may be of consequence if the user is trying to simulate a complex biochemical system very accurately. However, as previously discussed, this is not the main purpose of the ARN.

Other difficulties can arise using hybrid models, and detailed discussions are provided in the literature (*Kowalewski 2002*). One such issue occurs where a pool, for example S, inhibits a reaction unit by an inhibitory connection. This reaction will always be inhibited while there remains any amount of chemical in S. Meanwhile S is involved in another reaction, where the resultant flux is depleting S at each time-step. As the concentration of S decreases, so too does the flux. This leads to an infinite sequence of decreasing concentrations of S, which asymptotically approaches zero. Therefore, S will always contain a smaller but positive value, and as a result the inhibited reaction can never occur. In reality the above scenario would never occur since individual molecules would react in an individual manner. Where this is a problem, it is solved by simply setting a threshold - if a pool concentration is less than the threshold, its concentration is set to zero.

4.5 Verification of the Model

In order to verify the new ARN representation as a means to represent the properties and mechanisms of CSNs, the biological plausibility of low-level reaction kinetics needed firstly to be confirmed. This was achieved by applying varied sets of real biochemical data to a single ARN unit. The resultant time series output was compared with those recorded in literature from actual wet lab experiments, by manual calculation and by running the experiment on Berkeley Madonna (*Macey et al. 2000*) which is a reputable software designed to simulate biological reactions. *Fig. 4.5* provides typical results from one such experiment. In this example, reaction kinetic data (rate constants, reaction order) were used to create a model of the reversible isomerisation reaction between cis and trans 1-ethyl-2-methyl cyclopropane on Berkeley Madonna and on a single ARN unit. *Figure. 4.5* shows the time series results of the product concentration produced by a single ARN unit. After 2100 seconds, both methods produce the same final product concentration of 9.03×10^{-3} mol/L. The reason for their exact correspondence is due to application of Euler's approximation in both models, and, using exactly the same parameters including: step size, and starting conditions in both cases. When these results are compared with those in the literature, a slight error is seen to accumulate with each time-step, reaching a final difference of 0.11% after 2100 seconds. Such discrepancies were anticipated for two reasons. Firstly, due to the error associated with Euler's approximation, which, as previously discussed can be reduced by decreasing the time-step. Secondly, because of the error associated with gathering data from wet lab experiments - the method used to obtain the results in the literature. As the ARN is required to provide only a coarse approximation of low-level detail to represent temporal dynamics and network topologies, this error is not considered significant. Using many sets of such data, with a variety of different types of reactions,

including different forward and reverse coefficients, and reaction orders, the ARN consistently produced similarly accurate results.

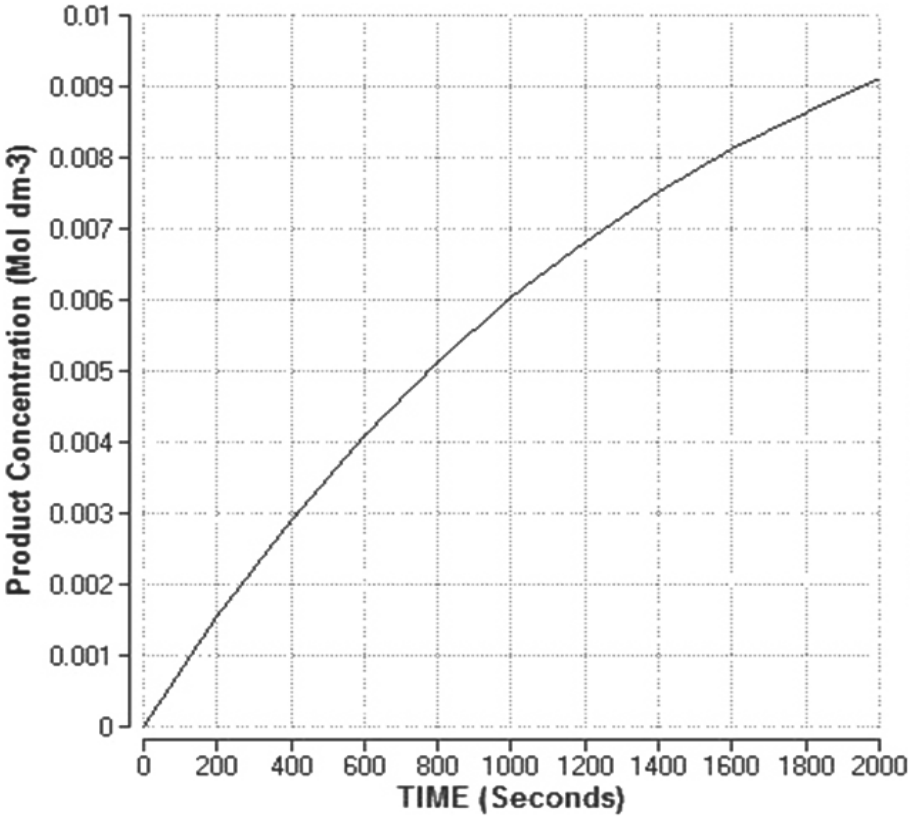


Fig.4.5 The product concentration produced by a single ARN unit.

In the following chapters, networks of such units are constructed and examined as a means to model high-level properties such as pattern recognition and complex temporal dynamics.

5. An ARN Based Simulation of *E. coli* Chemotaxis

One way to investigate the potential of the ARN to simulate a network of chemical reactions and capture its associated high-level behaviour is by creating a simulation of a well-characterised CSN. *Escherichia coli* chemotaxis is a good example of such a pathway and is the subject of this chapter. This simulation also uses concentrations of chemicals from actual biological systems and is therefore a good way to verify the correspondence of the ARN with its natural counterpart and also against more complex Systems Biology simulations.

5.1 Overview of *E. coli* Chemotaxis

As previously discussed (*see section 2.2*), chemotaxis describes cellular movement in response to sensed environmental chemical concentrations called chemoeffectors. The chemotaxis behaviour and CSN of *E. coli* is well characterised (*Vladimirov and Sourjik 2009; Wadhams and Armitage 2004*) and summarised in this section.

5.1.1 E. coli Random Biased Walk

As discussed in *section 2.2.1*, *E. coli* chemotaxis is described as a "biased random walk". This behaviour is the net result of intracellular dynamics between interacting proteins within its CSN. The CSN directs the bacterial flagellum (*see Fig 2.4*) to alternate between either a smooth linear swim called a "run", or a "tumble" where the bacterium stops and re-orientates itself in a random direction (*see Fig 2.5*). While moving along swimming trajectories, chemoreceptors continuously sense varying concentrations of environmental chemoeffectors (attractants and repellents). These signals are processed by the CSN and lead to the regulation of a motor complex responsible for the operation of the flagellar motor (*Vladimirov and Sourjik 2009*). As described previously, the motor complex is attached to a bundle of helical flagellar filaments. When the motor rotates counter-clockwise (CCW), the flagellar filaments form a trailing bundle which, by pushing the cell forward, results in a "run". When the motor rotates in a

clockwise direction (CW), the filaments shape change, causing the bundle to separate resulting in a "tumble". In the absence of chemoeffectors, *E. coli* alternates between runs and tumbles with runs lasting approximately 1 second and tumbles 0.1 seconds (*Vladimirov and Sourjik 2009*). In the presence of chemoeffectors tumbling frequency is increased down concentration gradients of attractants and up gradients of repellents, resulting in the biased random walk mentioned above (*Vladimirov and Sourjik 2009*). Thus, longer swim durations in response to higher attractant concentrations result in the emergence of a high-level behaviour characterised by net locomotion toward more favourable conditions. The actual quantity regulated, is the ratio of swims to tumbles, and is directly determined by the intracellular concentrations of the proteins within the pathway (*Vladimirov and Sourjik 2009*).

5.1.2 The E. coli chemotaxis CSN

This entire signal transduction pathway has been identified and is comprised of a set of protein-protein interactions. The key cytoplasmic signalling events that lead to the regulation of the motor complex are described below and summarised in *Fig 5.1*.

Cell surface receptors are responsible for detecting temporal patterns in environmental stimuli and initiating signal transduction. *E. coli* contains four types of transmembrane chemoreceptor proteins: Tsr, Trg, Tar and Tap - known as methyl-accepting chemotaxis proteins or MCP's. These MCP's are scattered over the surface of the cell membrane and each type is responsible for sensing particular chemoeffectors within the environment (*Vladimirov and Sourjik 2009*). After the signal has been internalised, it is processed by six cytoplasmic proteins (CheA, CheW, CheR, CheB, CheY, and CheZ) which transmit signals by reversible phosphorylation (*see Fig. 5.1*). CheW is an adaptor protein which forms a stable ternary signalling complex with the receptor protein and CheA. This complex, termed the "MCP complex" (*see Fig 5.1*), is responsible for initiating the chain of cytoplasmic reactions which communicate changes in levels of chemoeffectors detected by receptor molecules. In the absence of attractants, CheA (a histidine kinase) uses ATP to autophosphorylate. Phosphorylated CheA molecules ($CheA_P$) transfer phophoryl groups to aspartate residues on CheY and CheB. The cytoplasmic motor proteins: FliM, FliN, and FliG form a motor complex responsible for regulating the flagellar motor ("M" in *Fig 5.1*). Phosphorylated CheY ($CheY_P$) diffuse freely through the cytoplasm and interact with this complex by triggering clockwise (CW) motor rotation and tumbling response. As the $CheY_P$ concentration increases so too does tumbling frequency. The protein CheZ is responsible for active dephosphorylation of $CheY_P$. If receptors detect an increase in the attractant levels the autophosphorylation of CheA is inhibited. This results in a decrease in the levels of $CheY_P$ and hence decreases tumbling frequency resulting in longer run periods.

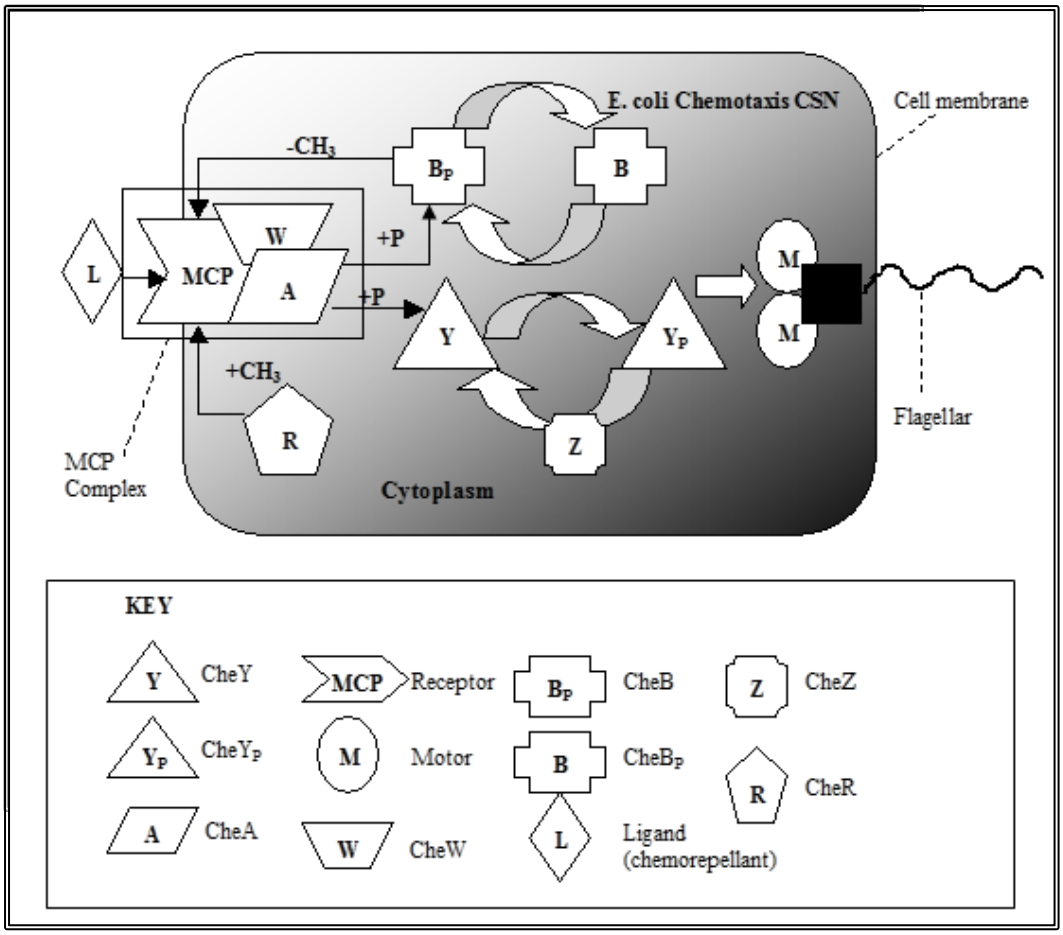


Fig 5.1 The *E. coli* chemotaxis CSN

To prevent the cell from being locked in either the swim or tumbling state, there is also a complex adaptation response (*Vladimirov and Sourjik 2009*). This response increases or decreases the sensitivity of the receptors by regulating the methylisation of the MCP complex. Methylisation provides a way to record recently detected concentration changes in chemoeffectors, thus giving the cell a primitive memory. The MCP's are in equilibrium between two states: swim and tumble, where chemoattractants bind to the run form of receptor as shown in *Fig 5.2*. By changing the receptor methylation level, the pathway is able to shift signalling complexes toward the opposing state to restore a balance between CCW and CW signal outputs. The cytoplasmic proteins CheB and CheR are responsible for regulating methylisation of the receptor proteins and hence the adaptation response. Phosphorylated CheB ($CheB_P$) acts as a methylesterase and removes methyl groups from the receptor proteins; while CheR is a methyltransferase which adds methyl groups to the receptor proteins (*Vladimirov and Sourjik 2009*). As previously mentioned, on detection of increased levels of attractant, the autophosphorylation of CheA is inhibited. This not only decreases the levels of $CheY_P$ - and hence tumbling frequency, but also decreases levels of $CheB_P$. A decrease in $CheB_P$ results in

receptors becoming more methylated due to the continuous activity of CheR. As MCP methylisation increases, the receptors shift toward the tumble form. In this form, the receptors phosphorylate CheA molecules, which then transfer phophoryl groups to CheY and CheB; hence driving the cell pathway back to its pre-stimulus levels. The initial change of the motor response is rapid in comparison to the slow adaptation response. This delay in the adaptation period allows the bacteria to perform extended runs before being set back to its original equilibrium. In this way the signalling network provides high levels of flexibility and plasticity which allow adaptation to varying environmental conditions (*Vladimirov and Sourjik 2009*).

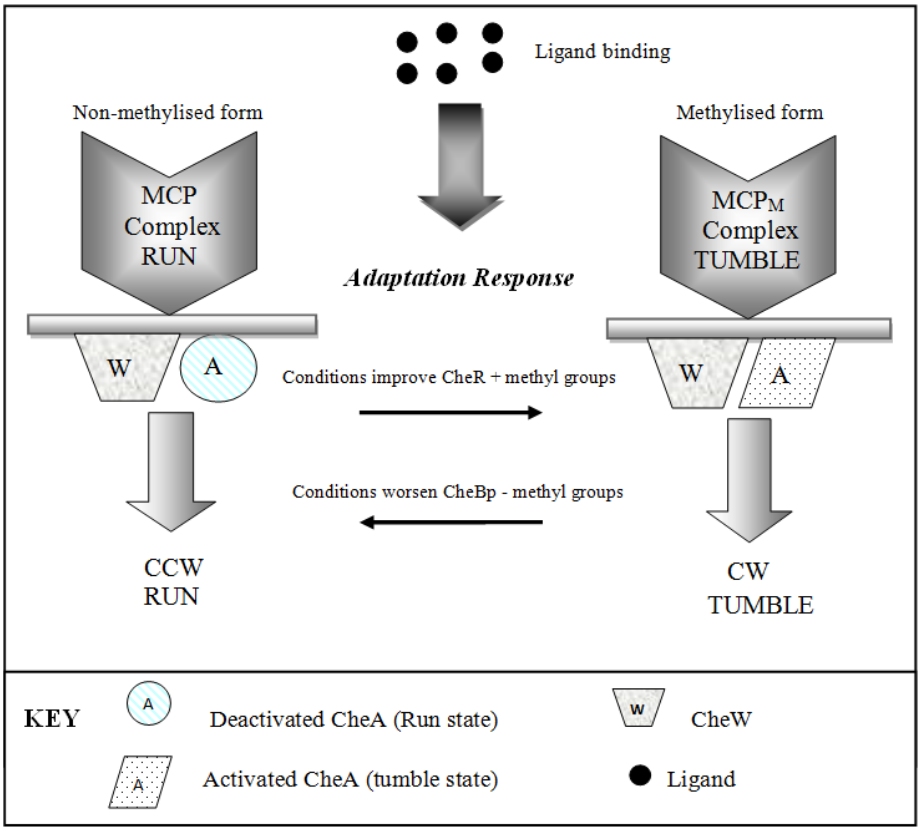


Fig 5.2 The swim and tumble states of the MCP complex.

5.2 Simulation of the *E. coli* Chemotaxis CSN

5.2.1 Structure of the Simulated Pathway

The pathway described above, was used as a basis to create a simplified ARN simulation of the chemotaxis CSN of *E. coli*.

This simulation specifically models the reactions essential in regulating the tumbling frequency and the adaptation response. Thus, only the reactions which regulate the concentration of $CheY_P$ in response to changes in environmental chemorepellents are included. This is because tumbling frequency is directly proportional to the concentration levels of $CheY_P$ (*Vladimirov and Sourjik 2009*).

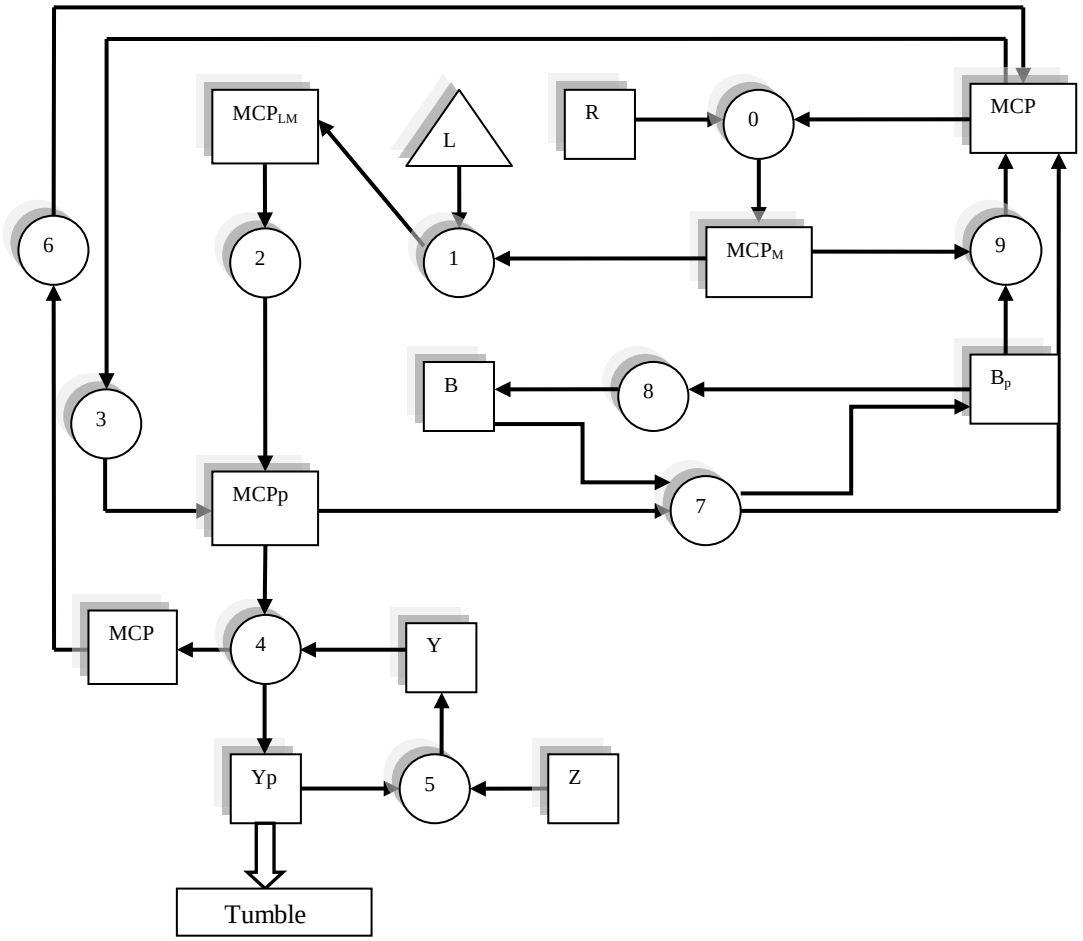
The structure of this simulation is based on the literature (*Vladimirov and* Sourjik 2009; *Bray et al. 1993*; *Wadhams and Armitage 2004*) and is shown in *Fig. 5.3*, represented in the ARN format described in *chapter 4*. It is composed of a network of 10 reaction units numbered 0-9, 11 pools of intracellular signalling proteins, a single input representing the chemorepellent (L) and arrowed lines to show the connections and also the direction of signal flow through the network. The 10 reactions modelled are given in *Table 5.1* listed 0-9. As previously mentioned, the simulation does not model reactions which are not essential in regulating the concentration level of $CheY_P$. The reactions which are not modelled are:

1. The formation of the receptor complex (MCP) which was assumed to have already taken place.

2. Y_P binding to the motor protein is not accounted for, as ultimately, it is the concentration level of $CheY_P$ that regulates the tumbling frequency, thus determining run length.

TABLE 5.1 Reactions included in *E. coli* chemotaxis CSN Simulation

Reaction	Molecules Involved	Description
0	CheR + MCP→ MCP_M +CheR	Methylisation of MCP complex by CheR (MCP complex includes CheA, CheW and receptor molecules) to produce methylated MCP_M. (MCP_M is the tumble form of the receptor.)
1	MCP_M + Ligand ⇔ MCP_{LM}	Binding of ligand (repellent) to MCP_M complex (tumble form of MCP complex) to create MCP_{LM}
2	MCP_{LM}→ MCP_{LMP}	Autophosphorylation of CheA where CheA is part of methylated MCP complex bound to ligand i.e. MCP_{LM} (tumble form of MCP complex)
3	MCP →MCP_P	Autophosphorylation of CheA. (Here CheA is part of the run form of the MCP receptor complex - this is much slower in comparison to reaction 2)
4	CheY + $CheA_P$→ $CheY_P$ + CheA	Phosphorylated CheA molecules (part of phosphorylated MCP complex) transfer phophoryl groups to CheY to create $CheY_P$
5	$CheY_P$ + CheZ→ CheY + CheZ	Dephosphorylation of $CheY_P$ by CheZ
6	MCP (various forms)→ MCP	This reaction cycles the MCP concentration back to restart the reaction cycle at reaction 0
7	CheB + $CheA_P$→ $CheB_P$ + CheA	Phosphorylation of CheB by phosphorlylated CheA ($CheA_P$) where $CheyA_P$ is part of MCP complex) to create $CheB_P$
8	$CheB_P$→ CheB	Dephosphorlyation of $CheB_P$ to create CheB
9	MCP_M+ CheBp→ MCP + $CheB_P$	Phosphorylated CheB ($CheB_P$) acts as a methylesterase and removes methyl groups from the receptor proteins converting to the run form of the receptor



Key							
MCP	non-methylated methyl accepting chemotaxis proteins (consists CheA, CheW and receptor)	B	CheB	MCP$_P$	phosphorylated CheA bound to MCP	→	Direction of signal flow
MCP$_M$	methylated methyl accepting chemotaxis protein	B$_p$	phosphorylated CheB	Z	CheZ	□	(square) chemical pools
MCP$_{LM}$	Methylated methyl accepting chemotaxis protein bound to ligand	R	CheR	L	Ligand (repellent)	○	(circles) reaction
Y$_P$	phosphorylated CheY	Y	CheY				

Fig 5.3 The structure of the ARN model of the chemotaxis CSN of *E. coli.*

5.2.2 Simulation Parameters

Network parameters, for example rate constants and protein concentrations, were set up using real biological data made available online as a central resource by the University of Cambridge (*Morton-Firth and Bourret 2011*). This data contains wet lab results which have been gathered from published work by a number of authors. It should be noted that due to limitations associated with gathering this type of data, the information serves only as an approximation to actual cellular parameters. The actual network parameters based on this data and used in the

ARN simulation are provided in *Appendix 2*. Averages were used where multiple values for the same parameter were available.

5.3 Experiments and Results

In the following experiments the simulation described in *section 5.2* is used as a means to investigate the application of the ARN to modelling biological pathways and capturing the resulting emergent behaviours.

Three sets of experiments were performed using this model. The first examines changes in the concentration of $CheY_P$ when exposing the simulation to varying levels of chemorepellents. The second investigates the ability of the model to simulate the adaptation response. The results obtained in the first experiment are then used to map the response of tumbling frequency to the repellent concentration. In the final experiment this correlation is used to model the trajectory of the random biased walk of an *E.coli* cell within an artificial environment containing a distribution of simulated chemorepellents. The results from these simulations are compared with those in the literature for both wet lab and related computational simulations.

5.3.1 Experiment 1: $CheY_P$ Levels in Varying Chemorepellent Environments

As previously described, an increase in the level of environmental chemorepellents results in a rise in the concentration of $CheY_P$ (and similarly $CheY_P$ levels fall when the cell enters more favourable conditions). In the following experiment the model is exposed to varying concentrations of chemorepellent and the steady-state concentration of $CheY_P$ is recorded. The results are used to establish if the network levels of $CheY_P$ rise and fall in response to the respective increase and decrease in chemorepellent levels as expected, and furthermore if these levels are similar to those reported in wet lab experimentation and in other computer simulations.

The ligand, shown in *Fig. 5.3*, as an input (triangle) labelled L, represents environmental chemorepellent concentration. A real *E. coli* cell travelling through its natural environment would be exposed to a continuous concentration level of chemorepellent at each passing location. Thus, in this simulation, the input L is set at a level representing a particular concentration of chemorepellent at a specific location within a physical environment. In the following experiment, different constant levels of chemorepellent were applied and the level of $CheY_P$ recorded against time. Real biochemical data was not available regarding the pathway reaction parameters in response to chemorepellent binding. However, moving toward lower

levels of chemoattractant provokes the same response as moving up gradients of chemorepellent; thus it was possible to use the available chemoattractant parameters to demonstrate the response to chemorepellents- where higher attractant concentrations were equal to lower repellent concentrations. In these experiments the range of chemoattractant levels applied is based on the sensitivity range of the receptor as given in the data (*Morton-Firth and Bourret 2011*). The levels applied were zero to 0.01 mol/L, where $1x10^{-10}$ mol/L represents a typical low concentration and 0.01 represents a high concentration. These levels are also used to represent chemorepellent, for example:

- 0 mol/L chemoattractant = 0.01 mol/L chemorepellent
- 0.01mol/L chemoattractant = 0 mol/L chemorepellent

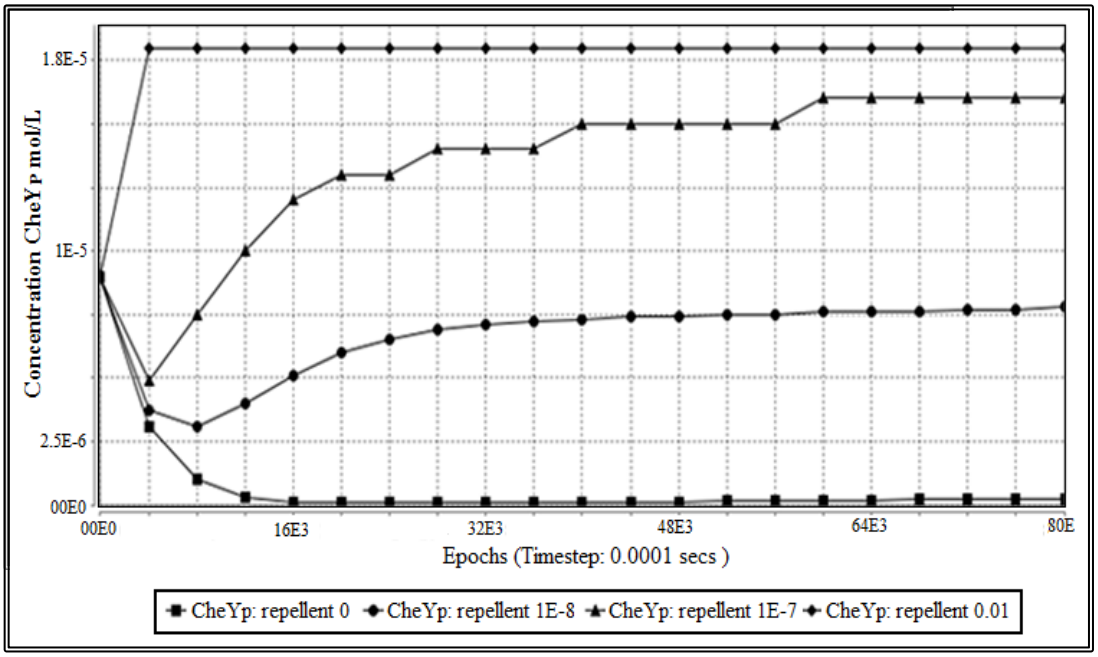


Fig. 5.4 Shows the concentration levels of $CheY_P$ in mol/L against time in seconds in response to applying the following levels of chemorepellent to the network: 0, $1x10^{-8}$, $1x10^{-7}$, and 0.01(in mols/L).

TABLE 5.2 Concentration of $CheY_P$ at steady-state

Concentration Chemorepellent	Time (secs) to reach steady-state	Concentration Yp (steady-state)	MCP_M (steady-state)
0	1.6	0	$9.65x10^{-6}$
$1E^{-8}$	8	$7.7E^{-6}$	-
$1E^{-7}$	6	$1.65E^{-5}$	-
0.01	0.4	$1.8E^{-5}$	0

The initial simulated concentration of $CheY_P$ was set to 10µm. This is based on the concentration of $CheY_P$ in the actual biological pathway at resting-state (i.e. without exposure to chemoeffectors). Every cycle of the simulation represents 0.1 ms and the flux for each reaction is calculated using *Eq. (4.3)* and used to update each connected pool. Due to its temporal properties, the network was run for a simulated time of 10 seconds to ensure that each pool had adjusted to the new level of chemorepellent and reached steady-state. The steady-state concentration levels of $CheY_P$ in mol/L generated by the ARN simulation at four different continuous concentration levels of environmental chemorepellent are shown in *Fig. 5.4* and summarised for clarity in *Table 5.2*.

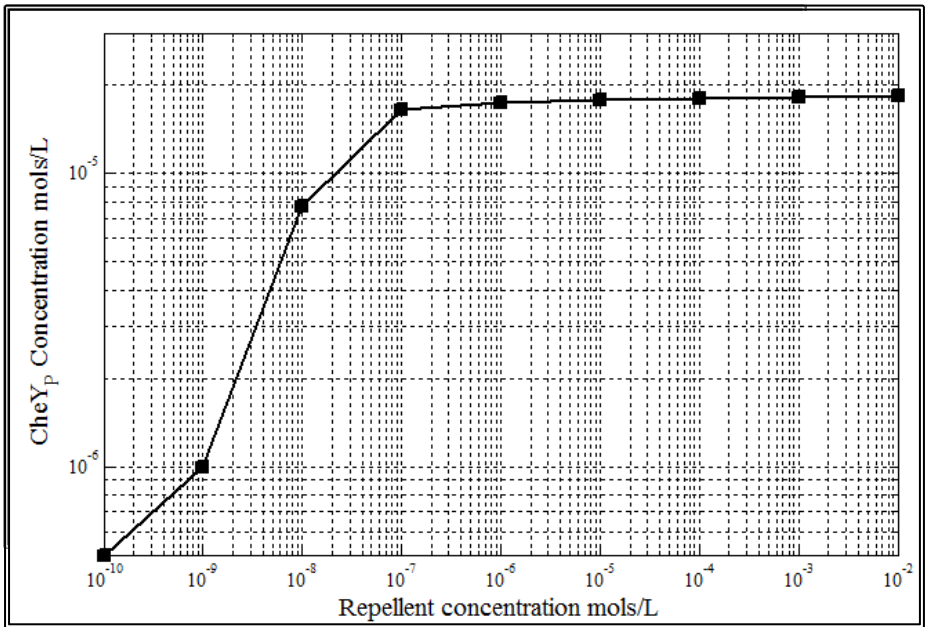


Fig. 5.5 The concentration levels of $CheY_P$ against repellent concentration in mol/L.

As can be seen, the level of $CheY_P$ begins to change within 0.4 seconds- this is comparable with that observed in the actual pathway, where the response of bacteria to a step change in attractant or repellent concentration occurs in ~0.2 seconds (*Bray et al. 2007*). A plot showing the ARN simulated concentration of repellent against $CheY_P$ concentration is shown in *Fig. 5.5*. In the actual biological pathway, increases in exposure to environmental chemorepellent concentration results in the increase of $CheY_P$ and therefore increases tumbling frequency. It can be seen clearly from *Fig. 5.5* that, like the actual organism, an increase in the level of simulated environmental chemorepellent (or decrease in chemoattractant) results in an increase of $CheY_P$ concentration. The results are in clear agreement with published data. For example, Goldman et al. (*2009*) measure and plot the concentration of active CheA ($CheA_P$) at increments of increasing concentration of aspartate (a chemoattractant). The plot of percentage aspartate concentration against percentage active CheA produces a concave down decreasing

curve and at 10% of active CheA the gradient flattens. As discussed here, an increase in the level of $CheA_P$ corresponds directly to an increase in the level of $CheY_P$. The results from this experiment (see *Fig. 5.5*) show that as the level of chemorepellent concentration increases the level of $CheY_P$ also increases producing a concave down increasing curve and at approximately 90% of the $CheY_P$ concentration the gradient flattens. Thus, bearing in mind that decreasing levels of chemoattractant are used to represent increasing chemorepellent, it can be clearly seen that the results of this experiment show the same trend.

5.3.2 Experiment 2: The Adaptation Response

To ascertain the ability of the ARN model to capture the previously discussed chemotaxis pathway adaptation response, the steady-state concentration of the methylised MCP (MCP_M) receptor complex and that of $CheY_P$ obtained by the ARN simulation was examined at the same levels of continuous environmental chemorepellent as applied in experiment 1 (*section 5.3.1)*. As discussed previously, an increase in the level of detected chemorepellent results in an increase in the level of $CheY_P$ and this, in turn, leads to a decrease in the level of MCP_M driving the receptor back to its pre-stimulus equilibrium. In this experiment, the initial concentration of $CheY_P$ is set to $1x10^{-5}$ mol/L as before. Both MCP_M and MCP are initialised to a concentration of $5x10^{-6}$ mol/L, based on the reported concentration in the actual biological pathway at resting state. The output of both MCP_M and $CheY_P$ concentrations against time for both high and low levels of chemorepellent are displayed in *Fig.5.6*. It can be seen, that applying an increase in the level of chemorepellent to the network, results in an increase of $CheY_p$ and a decrease in the level of MCP_M, thus driving the network back to its pre-stimulus equilibrium level.

In work reported by Goldman et. al (*2009*) the MCP receptors were exposed to sequential increments of the attractant aspartate. On exposure of $5x10^{-7}$mol/L aspartate the receptor methylation increased to a steady-state of approximately 35%, while exposure to $1x10^{-6}$ mol/L resulted in a steady-state of approximately 60% receptor methylation. Similarly, in the ARN simulation, the steady-state methylation increases on exposure to more favourable conditions while it decreases on exposure to less favourable conditions. Thus, the ARN simulation is able to simulate the behaviour of key proteins involved in the adaptation response, and compares well to other simulations.

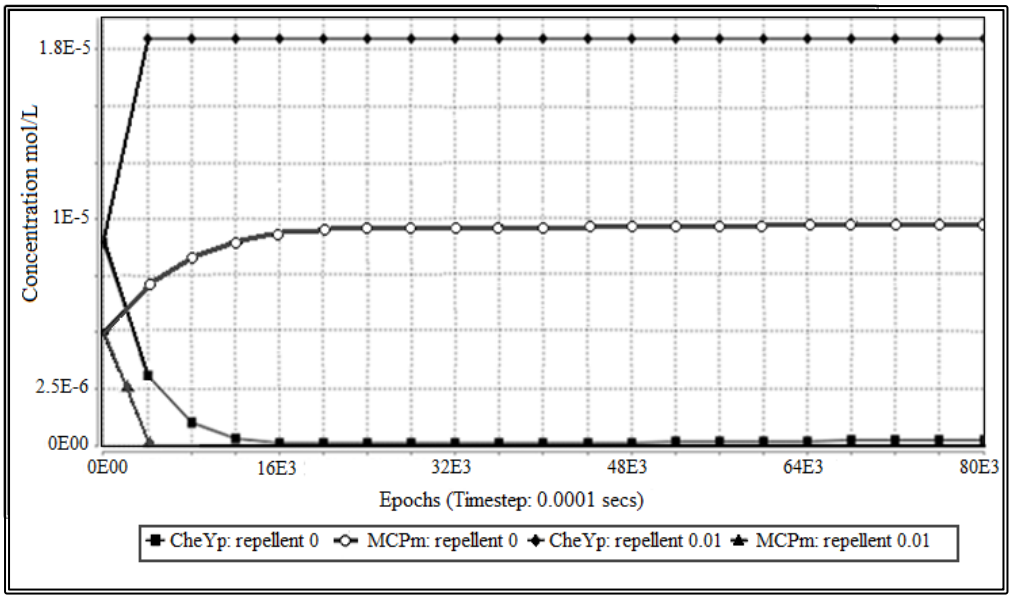


Fig 5.6 The steady-state concentration levels of CheYp and methylised MCP in mols/L recorded by the ARN when subjected to high and low chemorepellent concentration (0.01 mol/L and 0 mol/L respectively).

5.3.3 Experiment 3: Minimum Seeking Behaviour of ARN Simulated E. coli

Finally, to demonstrate the emergent behaviour of the simulated CSN, it was decided to show the chemorepellent avoiding behaviour in the context of an abstract optimisation problem. Here a simple environment of varying levels of chemorepellent was created and the chemotaxis trajectory of an artificial *E. coli* cell in response to these was recorded. The environment consisted of an inverted stepped pyramid search-space. Each step of this pyramid contained a different level of chemorepellent as used in Experiment 1 (*section 5.3.1)*. The centre of the pyramid represents the global minima of zero repellent concentration (solid black square), as shown in *Fig. 5.7*. Each progression outwards chemorepellent concentration increases logarithmically (that is 0 in the center and then $1x10^{-10}$, $1x10^{-9}$, $1x10^{-8}$ and so on), and the outermost perimeter signifies a maximum concentration of $1x10^{-2}$ mol/L. This range in chemorepellent was chosen as it includes the complete range of $CheY_P$ produced by the simulation (see *Fig. 5.5*).

In order to simulate the trajectory of a biased random walk using the model, the duration and direction of each run in response to each new chemorepellent level was calculated. The length of each run in response to the current concentration of environmental repellent was determined by mapping the resultant steady-state concentration of $CheY_P$ produced by the model to the tumbling frequencies at each level of $CheY_P$ published in the literature (*Bray et al. 2007*).

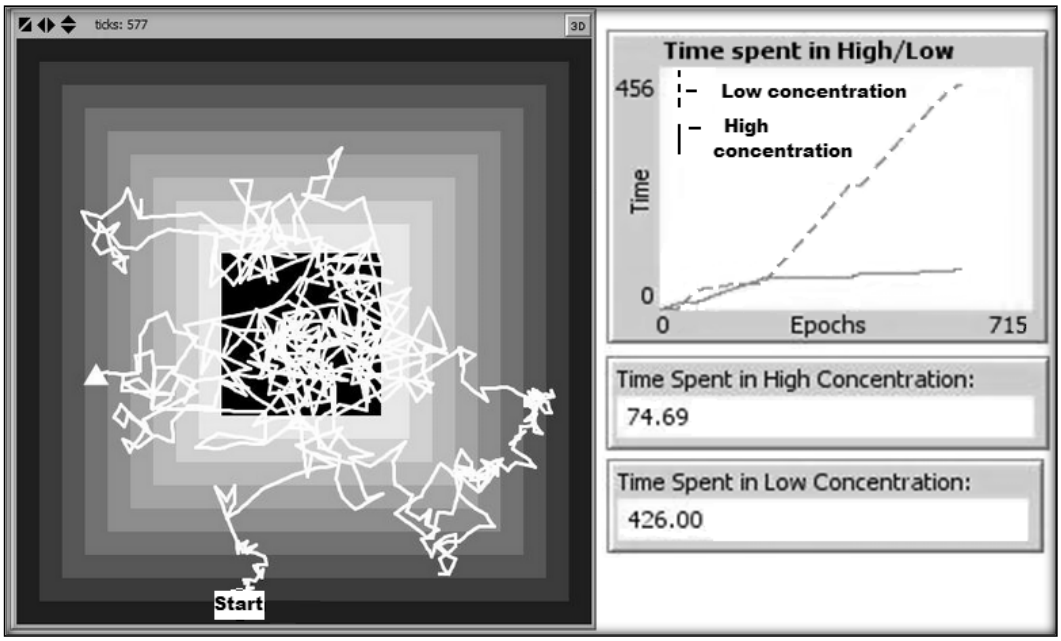


Fig. 5.7 Minimum seeking behaviour of a simulated *E. coli* cell over 500 seconds.

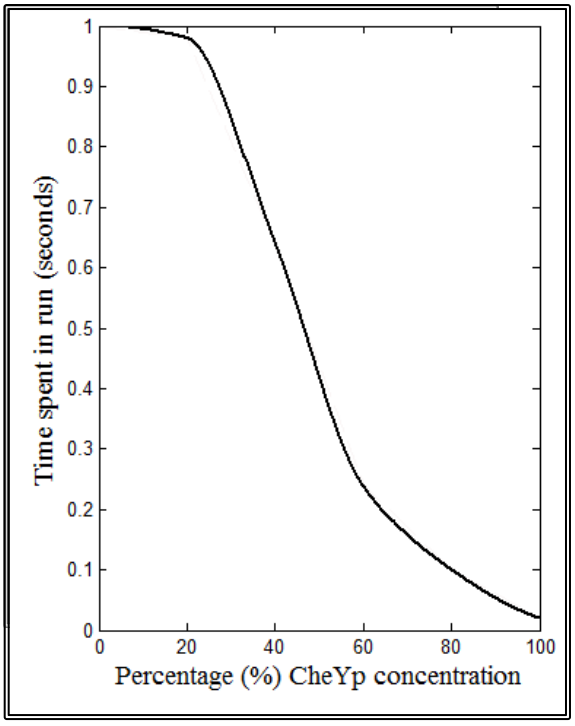


Fig. 5.8 Concentration of $CheY_P$ against run time in seconds. Adapted from Bray et al. (*1993*)

Figure 5.8 shows the concentration of $CheY_P$ against length of run and was plotted using the results reported in the literature (*Bray et al. 2007*). In an actual *E. coli* cell, a tumble causes the cell to redirect in a random direction, and therefore a simple randomised angle between 0-360 degrees provided the new direction for each successive run. Figure 5.7 displays the search

space and a typical run, where the path of the simulated *E. coli* is displayed as a white line. The results show that, over 500 seconds, the simulated *E. coli* cell remains in a high repellent concentration area (above $1x10^{-7}$ mol/L) for 74 seconds and a low concentration area (below $1x10^{-7}$ mol/L) for 426 seconds. Results were verified statistically over one-hundred runs; where the average number of seconds spent in low and high concentration was 376 and 124 respectively and the standard deviation for low was 33.9. Thus, it was found that the simulated cells were approximately three times more likely to be within a low concentration area than a high concentration area. These results correspond well with the reported behaviour of *E. coli* chemotaxis described in the literature and using other simulation methods (*Bray et al.2007*). They illustrate that the ARN can be used to accurately simulate real biology, and may have potential as a modelling tool in Systems Biology. Importantly, they also illustrate that the ARN is capable of capturing the global high-level behaviour associated with Cell Intelligence, which, in this case, is the biased movement of the simulated *E. coli* cells towards more favourable conditions.

6. Spatial & Temporal Properties of the ARN

As previously discussed in *chapter 1*, a key feature of CSNs is their ability to recognise specific patterns of chemicals within the environment and alter their spatio-temporal chemical distribution to provide an appropriate response. In this chapter, the ARN is used to explore the computational mechanisms of CSNs involved in pattern recognition and spatio-temporal pattern generation. The purpose of these experiments is to evaluate the ARN's ability to abstract these key processing features of CSNs. In doing so, applications of the ARN are explored; firstly as a means of simulating CSNs, and secondly in AI applications such as robotic control - where pattern recognition and response are key features. Furthermore, by using the ARN to investigate these properties, it may be possible to gain insight into the processing mechanisms that exist in biological CSNs.

6.1 Pattern Recognition

A previously discussed key mechanism of Cell Intelligence is the ability of a cell to recognise and respond to specific patterns of chemical signals within its environment (refer to *section 2.1*).

In the experiments outlined below, the pattern recognition capability of the ARN was tested in both the context of a general pattern recognition device and also in an abstract biological setting. In each case, four separate patterns composed of four input and four associated output mass-values were applied to the ARN. Each pattern comprised values of either 0.1, representing low concentration, or 1 corresponding to high concentration. The ARN was set up as shown in *Fig. 6.1* and consisted of seven pools, four inputs and seven reaction units organised into two layers. The network structure was based on that of standard (fully connected, feedforward) Multilayer Perceptrons (MLPs) (*Rumelhart and McClelland 1986*). This is because MLPs are a benchmark connectionist method used for pattern recognition.

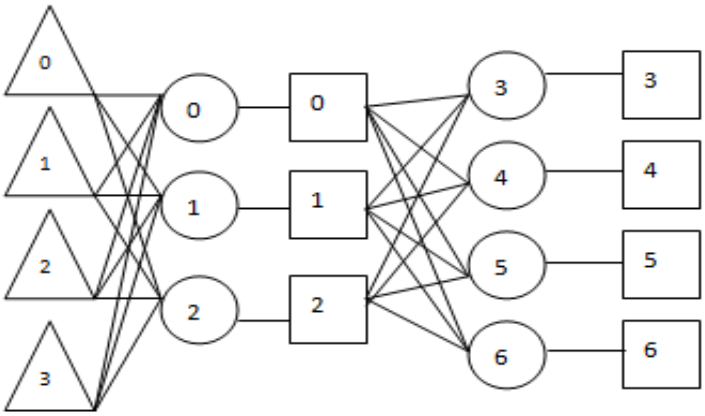


Fig. 6.1 The structure of the ARN used for pattern recognition experiments. The network consists of four inputs (triangles), seven reaction units (circles) and seven pools (squares). Each index of the input pattern array is fed into the corresponding input number. Output patterns are output at pools (squares) 3-6.

In biological CSNs, network parameters are determined by genetic factors, which are subject to evolution. To achieve a related effect within this artificial setting, an Evolutionary Algorithm (EA) (*see section 3.1*) was adopted to train the network to produce the correct output. This also allowed the combination of EAs and ARNs to be explored.

The initial value of all internal pools was 0.01, and each input value was fed into its corresponding input unit. For example, the first, second and third input value of pattern 1 is 0.1 and the fourth is 1 (*see Table 6.1*). The output values were generated by the final layer of pools (3-6). The target output values for each pattern are given under the heading "Output" in *Table 6.1*, and the actual values (to three decimal places) after training are given under "Actual Output". A population of one-hundred solutions was randomly initialised. Each solution comprised a complete set of network parameters (an array of twenty-five real numbers) including the forward and reverse rates for each unit and the weights for each connection between pools (or inputs) and units, where all the connections from a particular pool have the same weight (*see Table 6.2* for the range of these parameters).

TABLE 6.1 Patterns and results for both general and abstract biological setting experiments.

General Pattern Setting				**Abstract Biological Setting**			
Pattern	Input	Output	Actual Output	Pattern	Input	Output	Actual Output
1	0.1 0.1 0.1 1	0.1 0.1 0.1 0.1	0.1 0.1 0.1 0.1	1	1 (WR) 1 (SR) 0.1 (SA) 0.1 (WA)	1 (IS) 0.1(F) 1 (O) 0.1(DS)	1 0.1 1 0.1
2	1 0.1 1 0.1	1 1 1 0.1	1 1 1 0.1	2	0.1 (WR) 0.1 (SR) 0.1 (SA) 1 (WA)	0.1 (IS) 1 (F) 0.1 (O) 0.1(DS)	0.1 1 0.1 0.1
3	1 1 1 1	1 0.1 1 0.1	1 0.1 1 0.1	3	0.1 (WR) 1 (SR) 1 (SA) 0.1 (WA)	1 (IS) 0.1 (F) 1 (O) 0.1(DS)	1 0.1 1 0.1
4	1 0.1 1 1	1 1 1 0.1	1 1 1 0.1	4	1 (WR) 0.1 (SR) 0.1 (SA) 1 (WA)	0.1 (IS) 0.1 (F) 1 (O) 0.1(DS)	0.1 0.1 1 0.1
Key Inputs:	WR: weak repel		SR: strong repel		SA : strong attract		WA: weak attract
Key Outputs:	IS : increase speed		F: reorientation (up chemical gradient)		O: reorientation (down gradient)		DS : decrease speed

TABLE 6.2 Each solution within the population consisted of a forward and reverse rate for each unit and the weights of all the connections. The solutions were initialised with a random value between the ranges given in this table.

Range Connection Weights (W)	Forward Rate (k_f) Range	Reverse Rate (k_r) Range
$-3<=W<=3$	$0<=k_f<=6$	$0<=k_r<=6$

Due to its temporal properties, in order to obtain steady-state output values, the network was run for one-hundred cycles (a cycle ends when the complete set of pools in the network are updated once using *Eq. (4.3)* (where $\Delta t = 1$). The solution fitness (*fit*) was then calculated, where fitness was the inverse of the sum of the magnitudes of the individual errors. The individual errors were given by each individual output (O) in a pattern minus the wanted error value (target, T). The final desired error was 0.01. This is given in *Eq. (6.1)* where p is the total number of patterns, o is the number of outputs in each pattern, $T_{n,m}$ and $O_{n.m}$ are the target and output of the m^{th} output of the n^{th} pattern.

$$fit = \frac{1}{\sum_{n=1}^{p}\sum_{m=1}^{o}\left|T_{n,m} - O_{n,m}\right|} \tag{6.1}$$

The least fit half of the population was discarded and the remaining solutions were subject to mutation and crossover in order to create the new population. It was found that adjusting the crossover and mutation rate had a significant impact on the number of generations required to train the network. During training, the network frequently became fixed at error levels around 0.2, indicating that it was trapped in local minima. For this reason the mutation and crossover rates were adjusted resulting in a final setting where 40% of the solutions were subjected to single point crossover and 10% were subjected to uniformly distributed mutation. If the EA did not find a solution within one-thousand generations, it was considered not to have converged to a solution - the failure rate was approximately one in five runs. Only the runs in which the EA successfully converged on a solution were used in calculating the average and standard deviation. Out of one-hundred runs the average number of generations required to reach the target error was 387 with a standard deviation of 267.23, and the maximum number of cycles needed was 897. The large variation in the required number of cycles indicates that the EA is highly sensitive to its initial state. Approximately half the total number of EA cycles was spent decreasing the final 5% error. The parameters of a typical solution are given in *Table 6.3*. The results from the experiment are shown in *Table 6.1*. As can be seen, the ARN was able to recognise all four patterns correctly. These results show that EAs can be applied successfully to ARNs in a similar way to ANNs.

TABLE 6.3 Resulting network parameters for one solution after training using the EA.

General Pattern Setting Parameters					
Pool	**Initial Concentration**	**Weight of Connection**	**Reaction unit**	**Forward Rate**	**Reverse Rate**
0	1st Pattern value (e.g. if pattern is no.1 input is 0.1)	2.999	0	0.723	2.816
1	2nd pattern value	-2.915	1	5.411	0.837
2	3rd pattern value	0.424	2	0.969	0.643
3	4th pattern value	-0.278	3	0.120	4.310
4	0.01	-1.714	4	1.003	1.455
5	0.01	0.750	5	0.093	0.006
6	0.01	-0.435	6	1.081	0.580
7	0.01	1.319	Note that in this case to simplify the program the hidden layer pool concentrations were updated using the unweighted flux of the product.		
8	0.01	-0.104			
9	0.01	0.501			
10	0.01	1.492			

Multilayer Perceptron ANNs (MLPs) (*Rumelhart and McClelland 1986*) have similar properties. For instance, each neuron can be approximated as either active or inactive and is comparable to the ARN whose concentration is either high or low. However, MLPs lack an explicit time dimension, whereas the ARN processes inputs over a time period. In this case, the ARN was subject to a continuous flux of inputs over one-hundred cycles, causing the pool concentrations to enter a transient phase and then stabilise at steady-state. The implications are that, unlike the MLP, where processing is discrete-time, stored patterns are recalled only if inputs are applied for a length of time greater than that required to reach steady-state. This experiment demonstrates that the ARN is an appropriate pattern recognition technique when the requirement is to establish if a set of conditions have held true over a time period.

In a further experiment, the full purpose of which will become obvious later, using the previously described network structure and set-up, the ARN was trained to recognise an additional four patterns, where the inputs were chosen to correspond to chemical signatures (for example, attractants or repellents) that trigger specific movement responses. These patterns are given in *Table 6.1*. Here, the ARN network represents a highly abstracted CSN that controls the chemotactic motion of a generalised single-celled organism. This artificial amoeba is assumed to have default slow swim behaviour and in the presence of chemoeffectors the behaviour is updated accordingly. Each input signifies an environmental chemical, where input 0 is a weak repellent (WR), 1 a strong repellent (SR), 2 a strong attractant (SA) and 3 a weak attractant (WA). Specific combinations of environmental chemicals generate specific output responses – where repellents have precedence over attractants. The presence of chemical concentration to a value approximating to 1, in an output pool, corresponds to a particular behavioural response, where output pool 0 increases speed (IS), 1 re-orientates to face up the chemical gradient (F), 2 re-orientates down the chemical gradient (O) and 3 decreases the speed (DS). Therefore, as an example, on detecting both a strong repellent and strong attractant, the cell re-orientates to face down the chemical gradient and increases speed. As can be seen in *Table 6.1*, the network generated the correct response for all the abstract biological patterns.

One property of a CSN is robustness - correct response is generated even in the presence of noise or loss of connections. In order to test this property within the ARN, random noise was introduced to the trained general pattern recognition network. Each pattern was subjected to 10% increments of uniformly distributed random noise to a total level of 60% of the input range. At each noise level outputs were obtained for all four patterns. Figure 6.2 displays noise against total error on output for all four patterns and shows that the performance of the network gently degrades with increase in noise. The total error is the difference in error (as given in *Eq. (6.1)*) between the noiseless inputs and those with noise added.

Error levels within 5% of these results are reported for both the ABN and MLP models (*Macleod and Capanni 2010*) at levels of up to 50% noise in pattern recognition tasks of the same complexity.

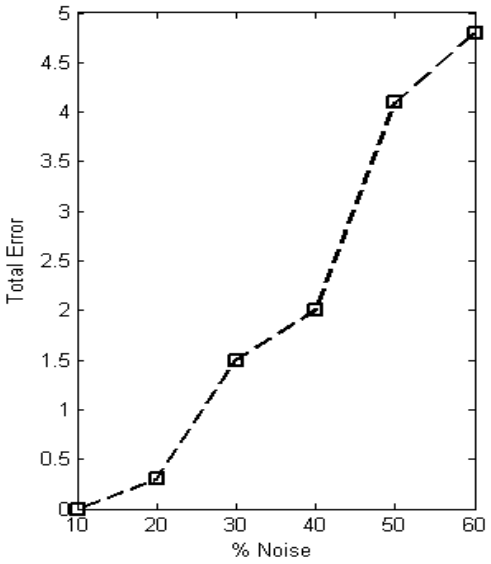


Fig. 6.2 Total error (y-axis) for all four patterns after introduction of random noise (x-axis) to patterns at 10% level increments

Similarly to an ANN, the ARN pattern recognition system is a robust connectionist network and thus provides an intuitive bridge between biology and AI. This experiment illustrates that such pattern recognition mechanisms are plausible in the CSNs of single-celled organisms, and that these mechanisms may be represented using the ARN. Furthermore, the experiment indicates that EAs have application as a means to set the unit and pool parameters. Further investigation into the role of EA within the ARN would be beneficial, specifically in finding more effective algorithms and in methods to evolve the network structure.

6.2 Spatio-Temporal Dynamics of the ARN

6.2.1 ARN-based Quadrupedal Robotic Control System

As discussed in *chapter 2*, CSNs are capable of complex computational processing. It was shown that a number of structural chemical motifs facilitate the processing of chemical signals to produce spatio-temporal dynamics suited to current conditions (*see Table 2.1*). In the experiments outlined below, these functional motifs are used to generate spatio-temporal patterns within the ARN. The purpose of this is to validate its ability to represent such patterns,

to explore their potential applications, and to gain deeper understanding of the regulatory mechanisms involved within CSNs.

One method of exploring the ARNs' ability to reproduce spatio-temporal dynamics, while investigating its potential AI applications, is to create an ARN based controller which can reproduce the patterns associated with the gaits of quadrupedal robots. Using these gaits as a means to explore the ARNs ability to generate spatio-temporal patterns has several advantages. Firstly, the gaits of quadrupeds are well characterised and the results of many other connectionist based quadruped gait control systems are available for comparison. Furthermore, successfully coordinating the motion of the four limbs requires complex spatio-temporal processing similar to that required to generate motion in biological cells. For example, the movement pattern of the unicellular slime mould *P. polycephalum* is termed shuttle streaming. This motion is characterised by a rhythmic forward and backward flow of the protoplasm. It is generated by oscillating chemical concentrations of Ca^{2+} throughout the cell, which cause internal pressure gradients and result in rhythmic contractions (*Radszuweit et al. 2013*). Another reason for investigating limbed robotic control using the ARN is that, although it has been demonstrated that reaction based ACs can be used to produce motion in animats (*Joachimczak et al. 2013*) or gaits of multi-modular robotics (*Hamann et al. 2010*), the control of limbed robots using discretised connectionist AC system, as far as the author is aware, has not yet been explored.

Terrestrial locomotion of limbed animals is achieved by multiple phase-locked patterns of limb movements known as gaits. For example, depending on speed of locomotion and terrain, quadrupeds commonly walk, trot and gallop (*Dagg 1973*). The gait phase is a value that ranges from 0 to 1 as the gait cycle proceeds. Therefore, the motion of each limb can be described relative to this. The ideal quadrupedal gaits are described by Dagg (*1973*) and others (*Hildebrand 1997*) and are used as a standard for comparison here and similarly in other studies (*Collins and Richmond 1994*). In the walk gait, each leg moves a quarter cycle out of phase and therefore four separate phases are required. In the trot gait each pair of diagonal limbs move half a cycle out of phase thus only two phases are present (see *Fig. 6.3*). Therefore, generating different gaits corresponds to generating different complexities of spatio-temporal signals, making this type of experiment ideal for investigating hierarchies of complex waveform generation.

In these experiments, an ARN controller was implemented to generate gaits of a Lynxsmotion dual-servo quadruped 2 (Q2) robot- as shown in *Fig. 6.4*. Each robotic leg is controlled by two servo motors, one for each degree of freedom (DOF). One motor raises the leg and the other

turns it. The structure of the robotic legs is shown in *Fig. 6.5*, further details of which are given by Toth and Parker (*2003*).

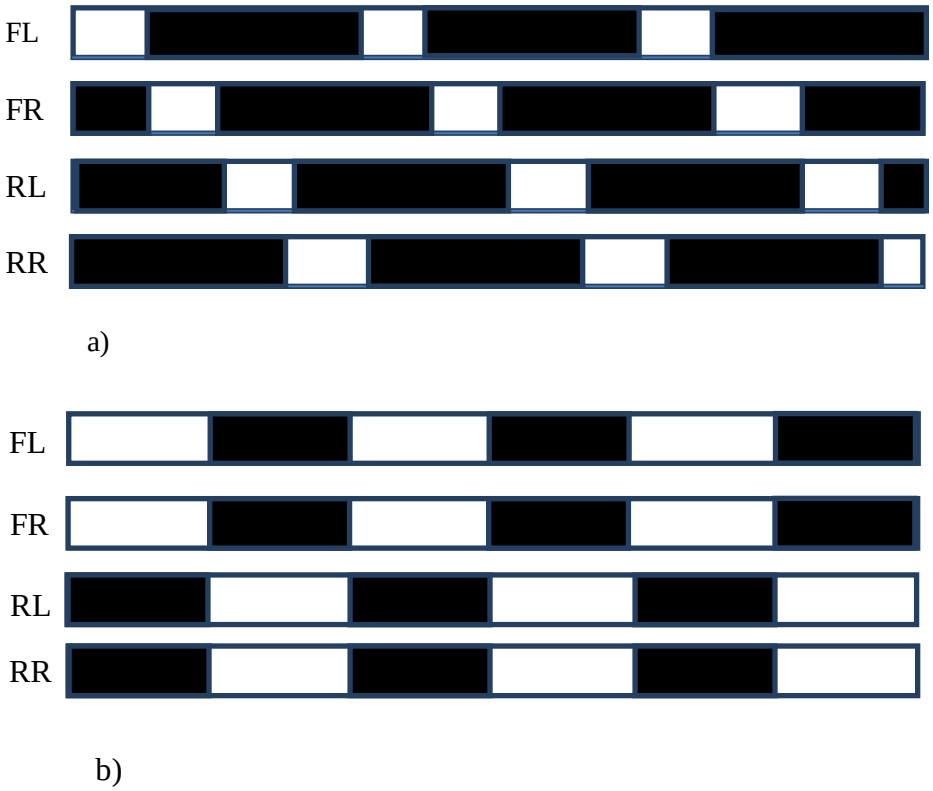


Fig. 6.3 Stepping patterns of quadrupeds showing a) walk gait b) trot gait. The black boxes show the relative length of time of step on the ground, and the white boxes show the relative length of time the leg spends raised. FL: front-left leg, FR: front-right leg, RL: rear-left leg, RR: rear-right leg.

Fig. 6.4 The Lynxsmotion dual server quadruped 2 (Q2) robot

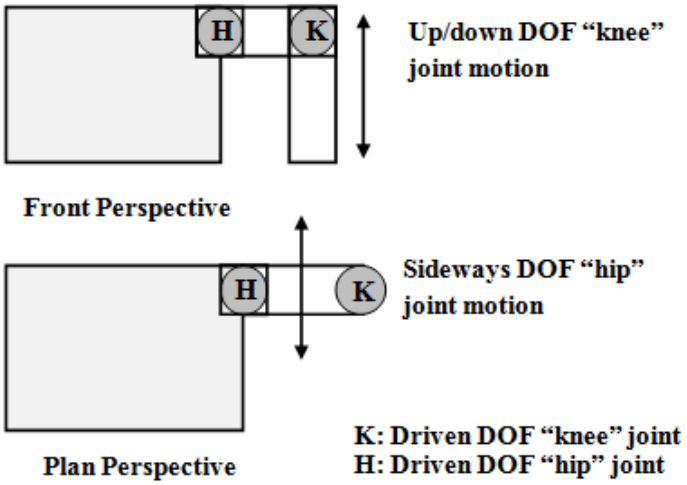


Fig. 6.5 The structure of a Lynxsmotion quadrupedal robot leg. Each leg has two DOFs and each DOF is controlled by a separate motor.

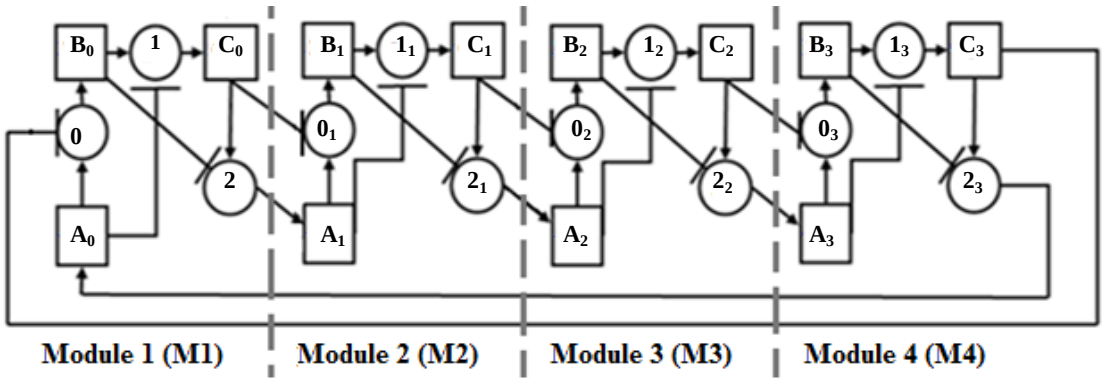


Fig. 6.6 The ARN based controller. Each module (shown separated by a dashed line) is mapped to a single leg and is responsible for controlling the two motors which generate its motion. Pool A of each module controls the up/down motor, pool B the back/forward motor and pool C the stop period for each of these motors.

Signals are sent by the ARN to each motor and control the angle of the rotor for each DOF, using a simple position to pulse width modulator interface circuit to control the servo. The structure of the ARN based controller is shown in *Fig. 6.6* and was designed by combining the previously discussed functional motifs found in CSNs (*see Table 2.1*) including inhibitory and excitatory reactions, cyclic loops, and feedback structures. The controller comprises a network of four repeating structural units or modules, where a module is separated by a dashed line. Each module controls the two motors of a separate leg, and comprises three reaction units and three pools: A, B and C. Pool A controls the up/down (U/D) motor, Pool B the back/forward

(B/F) motor and Pool C controls the off period for both motors. Pool activity is regulated by a series of excitatory and inhibitory connections between reaction units (see motif 1 and 2, *Table 2.1*). This type of connection represents the inhibitory and excitatory properties of specialised regulatory proteins common to CSNs such as enzymes. The overall network structure is based on the oscillator motif (see motif 9, *Table 2.1*) organised as a closed loop, allowing protein species to be recycled from the last to the first module and thus to generate a temporal oscillatory pattern. The structure of the ARN controller is capable of producing all the common gaits. The type of gait is easily modified by a simple adjustment of the initial pool values. For example, by initialising a C pool, a walk gait will be generated, where the C pool chosen will determine the starting leg and the value determines the angle to which the leg is raised. Similarly, a trot gait is achieved by initialising two C pools within alternate modules. In this particular design, the value to which the C pool(s) are initialised determines the DOF angle and were set specifically for the physicality of the particular robot, although it can be freely varied.

The network architecture remains fixed throughout these experiments and the network parameters are manually set. This method was employed for a number of reasons. The first reason was so that the outputs could be directly compared with other published work on similar Central Pattern Generators (CPGs) (*Billard and Ijspeert 2000*; *Collins and Richmond 1994*; *Liu et al. 2009*). The second reason was because of the associated problems with alternative methods. For example, in cases like these, where the search landscape is unknown, one effective way to set the network parameters would be to use an EA or a similar search technique, as was employed in the pattern recognition experiments. However, if such a method were used in this experiment there would be no guarantee that the functional motifs being investigated would be generated. Gradient decent algorithms also have associated problems when applied to this sort of network due to the use of feedback connections, and thus, in this application would prove difficult.

The ARN controller was considered to generate a specific gait if the relative phases of the respective oscillatory signals were within 2% of the standard gait cycle described previously. Higher values of 10% were used in other studies (*Collins and Richmond 1994*), and this was considered reasonable due to the variation found in real animal gaits (*Afelt et al.1983*). In each case, the controller first generates the U/D motor oscillation and on reaching the maximum value the B/F motor is initiated.

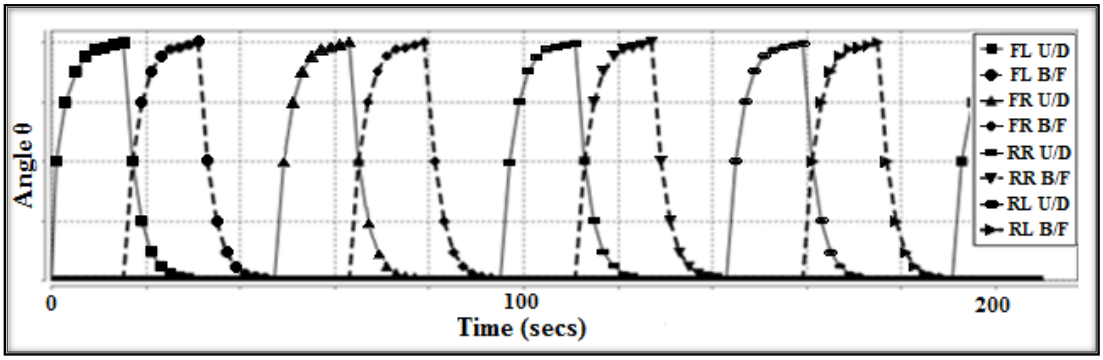


Fig. 6.7 Output generated for the walk gait. Legs are front-left (FL), front-right (FR), rear-right (RR) and rear-left (RL). The up/down (U/D) motor is displayed as a solid line and the back/forward (B/F) motor is displayed as a dashed line.

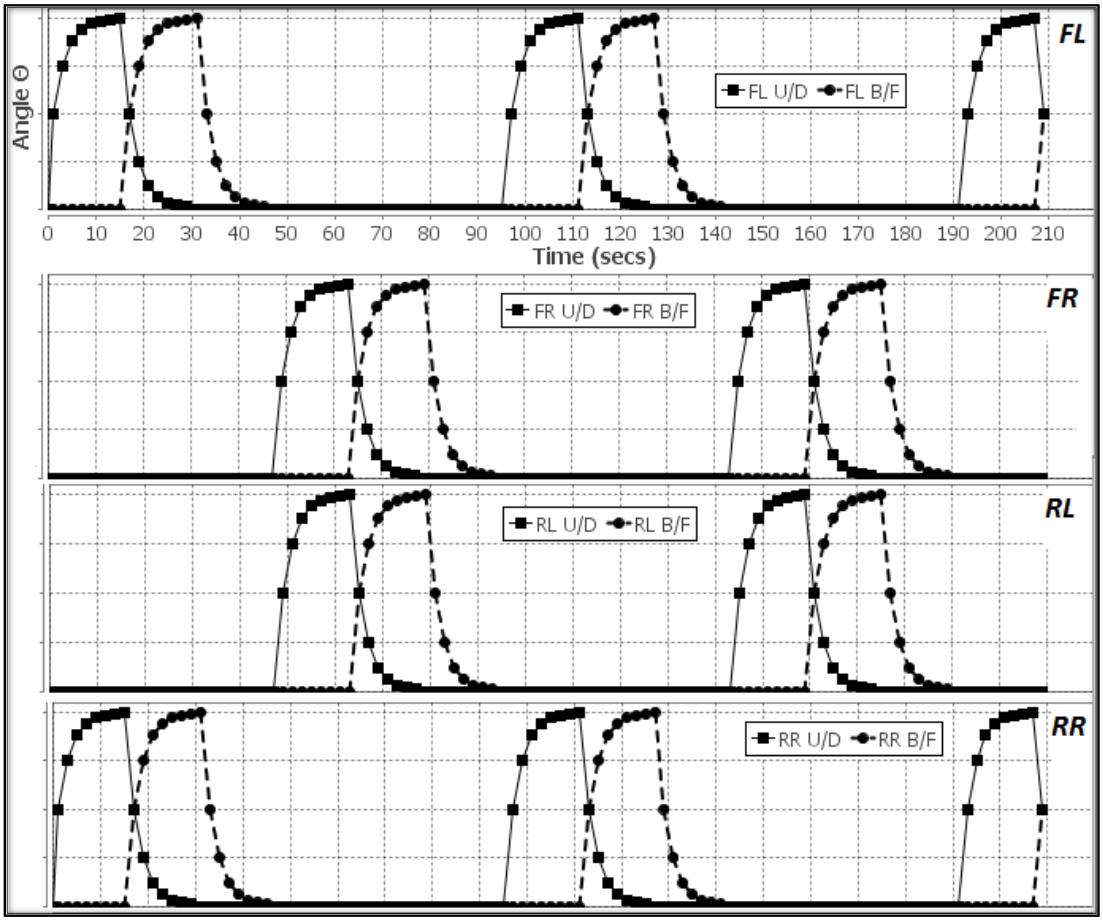


Fig. 6.8 Output generated for the trot gait. Legs and motors are labelled as before - see *Fig. 6.7*.

As can be seen in *Fig. 6.7*, the walk gait results show that the legs are a quarter cycle out of turn, with phases of 0.0, 0.25, 0.5, 0.75 between limbs in clockwise order from the FL (front-left) leg. Similarly, the trot gait results in *Fig. 6.8* show that the opposite legs are half a cycle

out of turn with phases respectively of 0.0, 0.5, 0.0, 0.5. The frequency of oscillations and therefore the gait speed is easily adjusted by applying uniform increase or decrease to k_f of each unit. Videos of the robot performing both the walk and trot gaits can be viewed online (*Gerrard 2012c*).

Both phase-locked limb patterns produced by the ARN match the standard, and compare well with other connectionist models. For example, Billard and Ijspeert (*2000*) present a CPG (Central Pattern Generator) based neural controller for a quadrupedal AIBO robot, similarly with two DOFs for each leg. The limb phases generated by this network correspond to the standard and to those produced by the ARN. Here, the network is composed of eight coupled non-linear oscillators and each oscillator consists of six leaky integrator neurons (a total of ninety-six neurons). Each neuron implements an activation function which is approximately as complex as the reaction unit function of the ARN, and therefore the complexity of the network is roughly equivalent to approximately ninety-six ARN reaction units. Similar correspondence is found in other sources. For instance, Collins explores a CPG based neural controller for a quadrupedal robot with one DOF per limb, and compares three types of activation function models, namely Stein, Van der Pol, and FitzHugh-Nagumo. The controller is composed of a network of four coupled non-linear oscillators (*Collins and Richmond 1994*), where each oscillator controls a separate limb. The Stein model consists of three first order differential equations, the Van der Paul model consists of a second order differential equation and the FitzHugh-Nagumo model consists of two first order differential equations. All these models have approximately twice the complexity of the output produced by the ARN unit. All three models also require a pulsing signal to drive the network. Generally speaking the structure of these models is less flexible then either the Billard and Ijspeert (*2000*) model or the ARN due to their rigidly fixed internal parameters. The models produced gait patterns within 10% of the standard, whereas the ARN matched the standard for both trot and walk.

Overall, the ARN has a very similar capacity to generate both walk and trot gaits as the compared controllers. However, in general, it affords a higher degree of flexibility and is less computationally complex than the compared models. Although robotic gaits might seem unconnected with cellular intelligence, the ARNs ability to produce them illustrates how cellular networks can generate the complex temporal patterns necessary in emergent behaviour. Furthermore, the results show, for the first time, as far as the author is aware, that a discretised biochemical connectionist based AC can generate the spatio-temporal patterns required to generate quadrupedal gaits.

6.2.2 ARN-based Robotic Control System Featuring Crosstalk

In a further experiment, a more complex ARN system was constructed to investigate the control of two simultaneous behaviours involving interacting pathways. Although CSN pathways are often described as separate and functionally independent, they commonly share and interact with components of other pathways, and this is referred to as "crosstalk". For example, cAMP plays a role in the regulation of cell proliferation, but its pathway also involves components in the MAPK signalling pathway (*Stork 2002*). Such interaction enables the integration of signals from different pathways and facilitates the coordination of cell activity (*Yuan et al. 2006*).

An ARN based control system may also incorporate the mechanisms of crosstalk. In the next experiment a simulated robotic system was used to illustrate this, it comprises a system with four limbs. As shown in *Fig. 6.9*, each limb has two DOFs: a simple "shoulder" joint, allowing the limb to rise and fall, and, situated at the end of the limb, a "hand" joint which enables a gripper to open and close. A typical behaviour comprises of vertical raising of the limb either independently or in parallel to other limb(s). Once the limb is fully raised, the hand opens and closes, followed by the return of the limb to its lowered position. Such a sequence might occur, for example, in an assembly robot, performing a repetitive task.

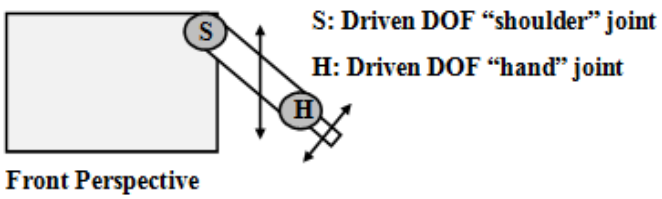


Fig.6.9 The simulated robot physical structure used for the crosstalk ARN network experiments, illustrating one of the four limbs and its two DOFs.

The ARN controller for this task comprises two distinct functional networks – pathway 1 and 2, as shown separated by a horizontal dashed line in *Fig. 6.10*. Like the previous ARN based control system, the design of the network is based on a combination of biochemical network structural motifs. Each pathway consists of four identical repeating modules labelled 1-4 (one is shown enclosed in a dashed line), and is situated at a layer (from 1-4), where each layer corresponds to the control of a separate limb.

Similarly to the previous network, this ARN controller is capable of producing a number of limb movement patterns (any number of limbs can move synchronously or independently), and the phases are easily modified by a simple adjustment of the initial pool values. Each module of

pathway 1 controls the up/down motors of a separate limb and comprises three reaction units and three pools. Pool B controls the S motor, which determines the angle to raise or lower the limb, and Pools A and C control its off period. Each module of pathway 2 comprises two reaction units and two pools. Pool E controls the H motor, which determines the angle to raise or lower the hand, and pool D controls its off period. Like the previous network, the activity of the pools in each module is regulated locally by excitatory and inhibitory connections and the modules are connected to form an overall oscillatory motif. In addition to this, each pathway is subject to feedback from the neighbouring network, which allows the two pathways to synchronise their output with each other. The results for independent limb movement are shown in *Fig. 6.11*. It can clearly be seen that each limb is out of phase. Once the S motor has raised the limb to its maximum angle (square line), the H motor then begins to extend (circle line) the hand to its maximum angle, while the limb remains raised. Both motors then simultaneously return the hand and limb to their original positions.

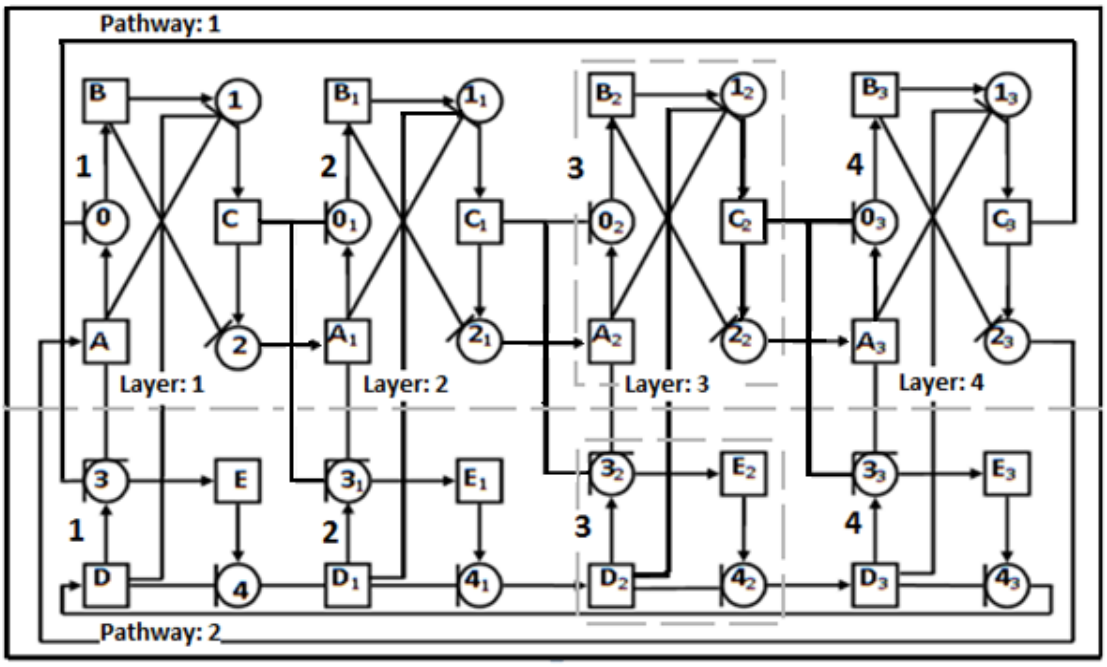


Fig. 6.10 The structure of a more complex ARN based control system involving communication or "crosstalk" between separate networks.

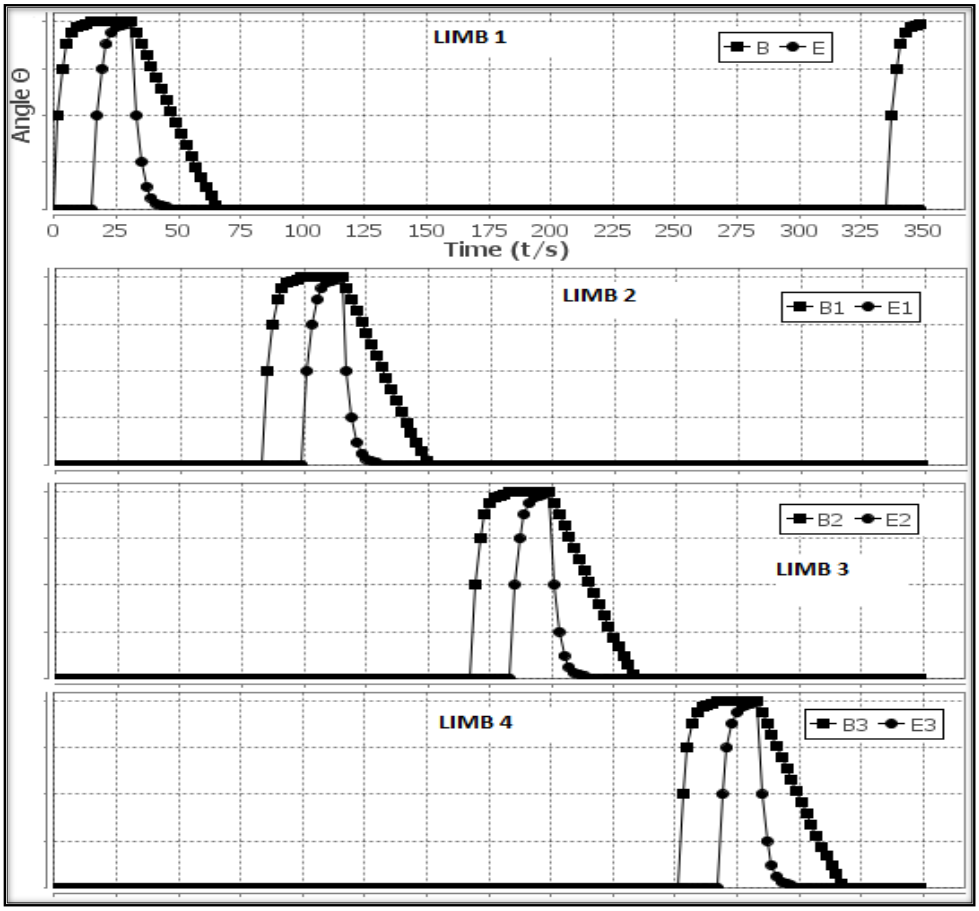


Fig. 6.11 Results of ARN based controller where elements of "crosstalk" were incorporated to control two separate behaviours.

Although this network was designed to explore elements of crosstalk, it is not difficult to imagine examples where synchronising complementary behaviours are required. In this case, the robotic hand could be equipped with a tool such as a paintbrush, screwdriver, or cutting device. More interestingly, it demonstrates for the first time, as far as the author is aware, that crosstalk can be incorporated into a discretised biochemical connectionist system. Furthermore, it demonstrates that this system can generate and synchronise multiple low-level activities enabling them to complement each other so that high-level behaviour is produced.

6.2.3 Complete Robotic Control System

It was demonstrated in *section 6.1* that an ARN can recognise patterns. Furthermore, in *section 6.2.1-6.2.2*, it was shown that such a system can generate temporal output patterns which can be used in control tasks. Of course, in the natural world, these two behaviours are linked together.

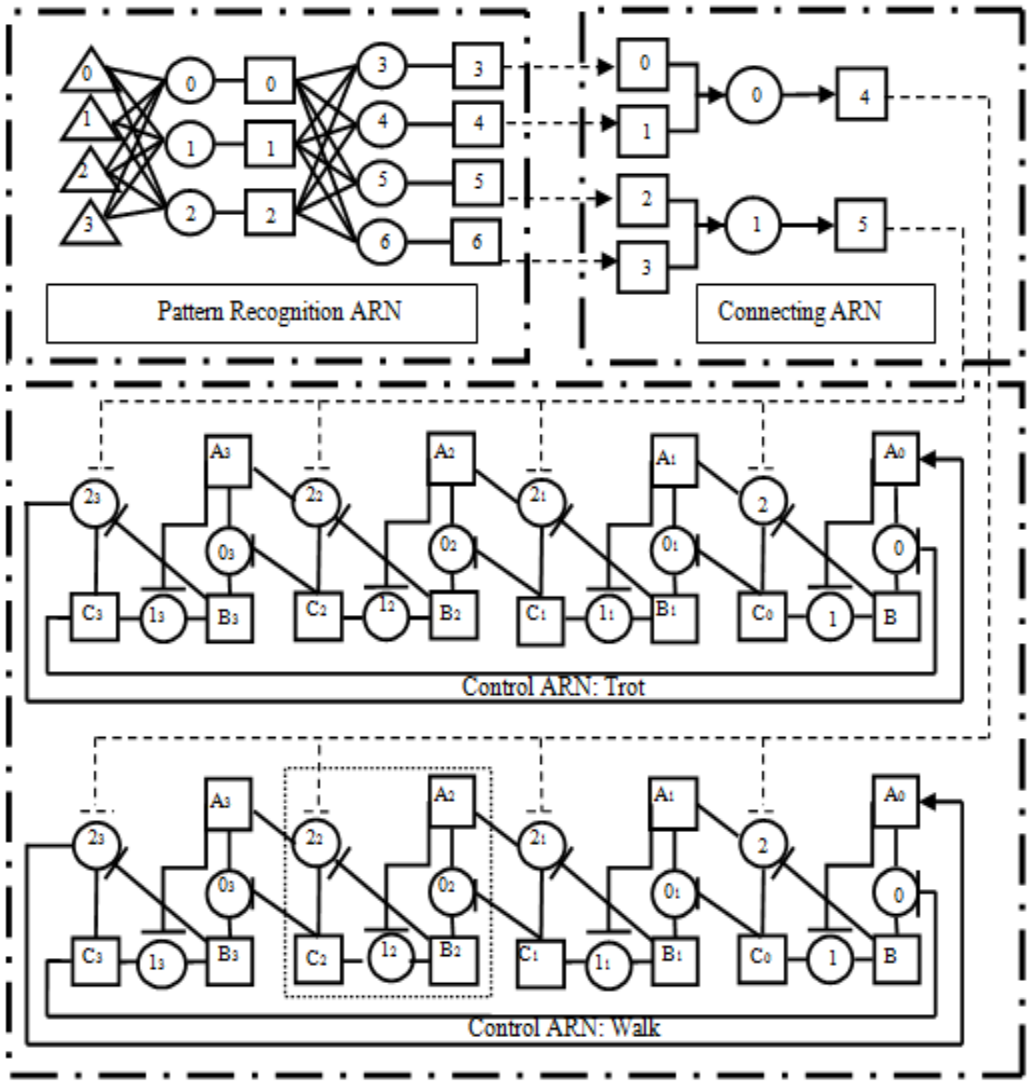


Fig. 6.12 A complete control system for a quadrupedal robot. On recognition of a particular pattern the pattern recognition ARN generates the associated output pattern. The connecting network implements two parallel Boolean AND gates which act as switches turning the walk or trot components of the control ARN off/on. The control ARN generates the required waveform which controls the robotic gait.

In the following experiment it is illustrated that both pattern recognition and control function can be combined within a single ARN-based system. Here, a more complex ARN was created to recognise specific patterns and in response, automatically generate the associated temporal gait. The ARN in this experiment reuses the pattern recognition and gait network previously described in *sections 6.1* and *6.2.1* respectively. The complete system is shown in *Fig. 6.12*. It is functionally divided into three smaller components: pattern recognition, control, and a connecting (or interfacing) network.

The structure of the pattern recognition network (PR), its implementation and training methods are identical to those described in *section 6.1*. This was the reason why a cell-signalling analogy was introduced in the pattern recognition experiments in *section 6.1*. In this case, the network was trained to recognise three separate patterns (as shown in *Table 6.4*) composed of four input and four associated output mass-values. The output pools of the PR network are equal to input pools 0, 1, 2, and 3 of the connecting network (CN). The CN comprises six pools (four inputs and two outputs) and two reaction units. Its structure is based on the motifs found in biochemical networks (see motif 7, *Table 2.1*). Essentially this component operates like two parallel Boolean AND gates, where a value of 1 at pools 0 and pool 1 will output a value of 1 at pool 4, as will a value of 1 at pools 2 and 3 output a value of 1 at pool 5. Two negative feedback connections between the CN and both control system subunits (shown as dashed-line connections) are responsible for switching between the gaits. Therefore, if a value of 1 is output at the interface network pool 4, it will inhibit all the reaction 2's of the walk subunit, thus stopping the walk gait pattern from being generated. Conversely, if a value of 0 is output at pool 4 the walk will be generated. In the same way, pool 5 of the CN controls the switching of the trot control subunit. *Table 6.4* shows the range of required behaviours in response to particular outputs generated by the CN.

TABLE 6.4 Patterns applied to the complete control system and the expected gait generated.

Pattern	PR Network Input Pool No.	PR Network Input Value	CN Network Input Pool No.	CN Input Value (& output of the PR network)	CN Output Pool No.	CN Output Value	Gait
1	0	1	0	1	4	1	Inhibit Walk
	1	0.1	1	1			
	2	1	2	0	5	0	Trot
	3	0.1	3	0			
2	0	0.1	0	0	4	0	Walk
	1	1	1	0			
	2	0.1	2	1	5	1	Inhibit Trot
	3	1	3	1			
3	0	1	0	1	4	1	Inhibit Walk
	1	0.1	1	1			
	2	0.1	2	1	5	1	Inhibit Trot
	3	1	3	1			
KEY:	PR	Pattern recognition		CN	Connecting network		

TABLE 6.5 Pattern applied to the network and expected durations of gaits.

Pattern	Walk ARN Network	Trot ARN Network	Start Time (seconds)	End Time (seconds)	Duration (seconds)
2	On	Off	0	210	210
1	Off	On	210	440	230
2	On	Off	440	560	120
1	Off	On	560	700	140
3	Off	Off	700	800	100

The control system comprises two separate subunits, both identical in structure and implementation to the ARN described in *section 6.2.1*. Each of these subunits is responsible for generating a specific temporal gait pattern: one generates walk, the other trot. The two subunits provide distinct gait patterns due to the differences in initialisation of the concentration values of C pools.

The complete system was tested to confirm its ability to both generate the correct behaviour and automatically transition between the behaviours in response to input patterns 0-3. The time periods in which patterns were applied, and the expected output states are shown in *Table 6.5*.

The results of this experiment are displayed in *Fig. 6.13*. The phases produced for each gait are exactly as described previously in *section 6.2.1*. The on/off periods of both trot and walk gaits are in agreement with the expected durations displayed in *Table 6.5*, but with a slight transitional delay. The ARN controller and gait phases produced have previously been compared with CPG models in *section 6.2.1*. The transitions between gaits generated by these models may now be compared with those of the ARN. The results given for the Billard and Ijspeert (*2000*) model, show transitions from walk to gallop in approximately four leg cycles, whereas the ARN transitions from walk to trot within two leg cycles. In both cases the transitions are very smooth. There are three models described by Collins (*1994*), and although gait graphs are provided for all these, gait transitions are only given for the Stein model. Here gaits transition quickly within approximately two leg cycles. However, in contrast to the ARN and the Billard and Ijspeert model, the leg movements during transition are very irregular.

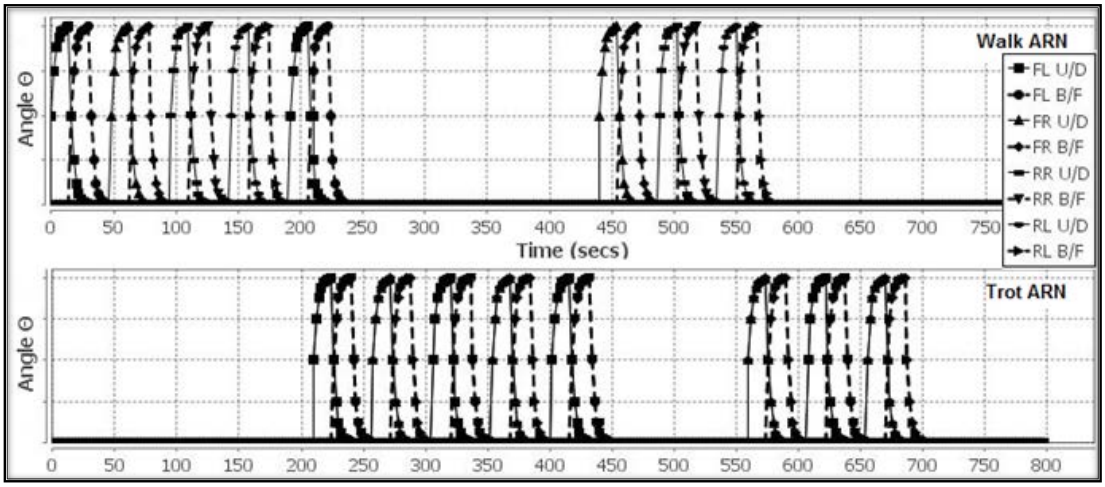


Fig. 6.13 The output of the complete ARN control system over 800 secs. Legs and motors are labelled as before - see *Fig 6.7*.

This complete control system demonstrates that the ARN, like a CSN, is capable of both recognising patterns and controlling overall behaviour in a single network. With the exception of spiking models, few ANNs can achieve this functionality. However, spiking models are often less flexible. For example, in the Integrate and Fire model, information is rate coded and all the spikes generated are uniform (*Maass 1997*). Thus, unlike the ARN, the Integrate and Fire model (*Maass 1997)* lacks the flexibility to produce pulse-width and pulse-amplitude coded information. The gait phases and transitions compared well with CPG neural controllers and showed that the ARN has application in similar robotic control tasks and can offer lower computational complexity. These experiments illustrate how a CSN might perform the complex sequencing timing and coordination processing associated with the high-level behaviours displayed by unicellular organisms. Furthermore, it shows that abstractions of both neural networks and CSNs operate in similar ways, and have comparable functionality. This also illustrates a close relationship between neural and cell intelligence.

In summary, the results presented in this chapter show that the ARN (and by extension, CSNs) are capable of performing pattern recognition in a similar way to Artificial Neural models and also of producing complex temporal dynamics reminiscent of Spiking Neural models. Additionally, as was shown, in previous chapters, the ARN can model biological reactions and simulate real CSN pathways with an accuracy matching those of standard simulation methods. This combination of attributes makes it a unique and useful tool. The ARN systems presented above show clearly that biochemical networks are quite capable of producing many of the behaviours normally ascribed to Neural Networks. This helps to illuminate the many interesting results now emerging from the behavioural biology of single cells. Of course neurons themselves have internal biochemical networks, and one future application of the ARN may be

to help unravel their more complex internal dynamics. The simplicity of the ARN makes it a potentially useful model in more practical AI and engineering systems. As demonstrated in the case of robotics, its ability to function in both input (afferent or sensory) and output (efferent or motor) networks and in the interconnection between these, gives it applications which usually only much more complex models can fulfil. This is particularly useful in the field of robotics, where such flexibility has particular application in evolutionary control networks.

7. Cytobots: ARN-Controlled Agents

In the previous chapter it was shown that an ARN can be used to model regulatory dynamics and interactions with the environment similar to those which occur in real cells. Another type of signalling dynamic that has not yet been explored within the ARN is that of cell-to-cell communication. In the natural world, cells, within for example, a colony, interact to influence each other's CSNs, which produce collective high-level behaviour. In many cases, such cells are distributed and may be mobile. The state of each cell's CSN is influenced not only by the state of the environment, but also by signals sent by other cells. *D. discoideum* is a cellular slime mould and provides an excellent example of such interaction between cells. It has a well-characterised life-cycle and, as illustrated later, exists as a group of interacting amoebae during the first stages of its life-cycle. These attributes make it an ideal model organism in which to investigate the dynamics of a system of interacting cells using the ARN.

In this chapter, the concept of mobile agents termed "Cytobots" ("cyto" from Greek for cell, and "bot" from robot) are introduced. Cytobots are autonomous cell-like agents which move around and respond to environmental information within a simulated or real environment. Each Cytobot is controlled by an internal ARN, in a similar way to which a CSN controls an amoeba. The ARN allows a Cytobot to recognise environmental patterns, communicate stigmergically with other cells and control its trajectory. Implementing Cytobots in this way provides a facility to capture the external dynamics resulting from interactions between cells and the environment, while maintaining the internal dynamics of the cell. Thus, Cytobots provide a means to explore overall global behaviour resulting from these dynamics and to investigate the properties and mechanisms involved. In the experiments outlined in *chapter 8* the Cytobots are used as a means to establish the ARNs ability to simulate such dynamics and to investigate their application in biological simulation and in robotic control.

7.1 Cytobot Behaviour

In the experiments outlined in *chapter 8*, the Cytobots are initialised and move around asynchronously within a simulated 2D environment containing a distribution of artificial

chemicals. These chemicals represent attractants, for example, food. When a Cytobot reaches a new position, it detects the surrounding level of chemical and uses the detected concentrations to set the inputs to its internal ARN. Consequently, this changes the state of the ARN, resulting in updates to the agent's trajectory. During this process, the Cytobot modifies the state of the environment by, for example, consuming the food. All the actions performed by a Cytobot fall into one of two basic behavioural modes: Foraging and Starvation. These behaviours are based on two stages of the life-cycle of *D. discoideum*: Vegetation and Aggregation.

7.1.1 Biological Counterpart: D. discoideum

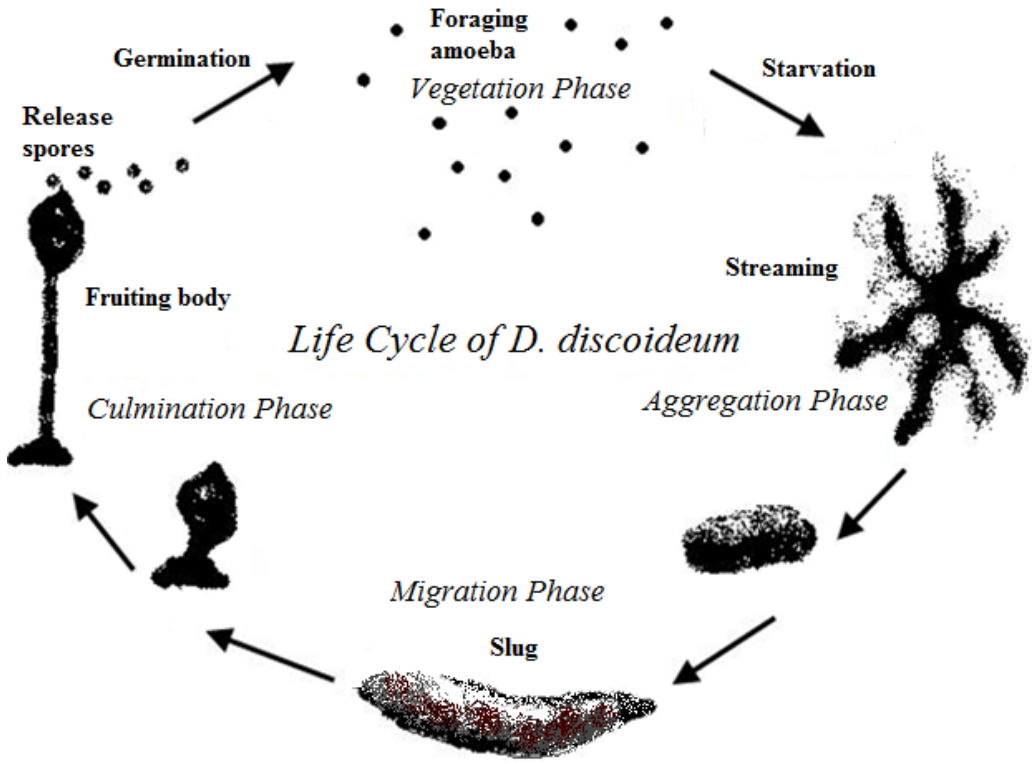
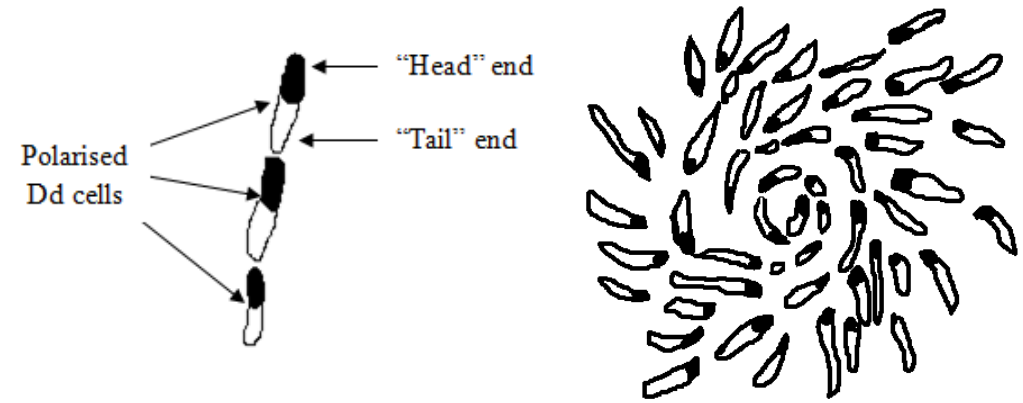


Fig. 7.1 Life-cycle of the cellular slime mould *D. discoideum*

D. discoideum (Dd) is a cellular slime mould which lives in soil and decaying leaves, feeding on bacteria. Its life-cycle contains four stages: Vegetation, Aggregation, Culmination and Migration as shown in *Fig. 7.1* and summarised below. For a more detailed description see Tyler (*2000*). During its vegetative stage Dd consists of a collection of independent amoeba (myxamoebae) which navigate toward food by moving up gradients of folic acid (FA), secreted by their bacterial prey. The trajectory of these cells is a pattern of motion similar to the random biased walk of *E. coli* discussed in *section 5.1.1*. The Dd cells extend pseudopods in a random direction; those extended toward sources of chemoattractants such as food are maintained,

while those extended toward less favourable conditions and retracted. The overall result is movement up the gradient of attractant (*Andrew and Insall 2007*).

The Dd amoebae starve when the food resource has been depleted, and begin the Aggregation phase. During Aggregation, starving cells secrete cAMP (cyclic adenosine monophosphate), which serves as a signal to attract surrounding amoebae towards each other, resulting in a densely populated aggregate of cells (*Dallon and Othmer 1997*). Aggregating cells are polarised, thus one side becomes the leading edge, which always faces in the direction of travel (*McCann et al. 2010*). Depending on parameters, such as environmental conditions and the cell population density, migrating cells can form transient emergent patterns such as streams and spirals (*McCann et al. 2010*; *Dallon and Othmer 1997*). Streaming describes a pattern of motion where cells line-up in close-order files, with the head of one following the rear of another (*McCann et al. 2010*) – and the spiral pattern describes streams of concentric cells spiralling toward the centre of the aggregate (*see Fig. 7.2*).



a) b)

Fig. 7.2 Stylised illustrations of Dd aggregation patterns. a) Shows streaming Dd cells during aggregation. This pattern is characterised by polarised cells forming a close-order file where the head on one aligns with the tail of another. b) Shows a spiral aggregation pattern, where cells form concentric spirals toward the aggregate centre.

After forming a tight aggregate, the Migration phase begins. Here, the aggregated cells form a motile pseudoplasmodium known as a "slug". The component cells coordinate their activities to allow the slug to move toward attractants such as heat and light. The slug eventually settles in a favourable position where it forms a fruiting body and enters the Culmination stage. During Culmination the fruiting body matures, and eventually releases spores which are distributed within the surrounding environment. These spores mature into cells and become the new generation of myxamoebae, and so the cycle begins anew.

7.1.2 Cytobot Foraging Mode

When in Foraging mode, a Cytobot performs a pattern of biased motion based on Dd cells during their Vegetative phase (as described previously in *section 7.1.1*). The Cytobots approximate this behaviour by performing alternate periods of forward motion, termed “runs”, and random redirections called “tumbles”. The bias is provided by reducing the tumble frequency when moving toward more favourable conditions (for example up a food gradient) - thus increasing the length of the run. Like aggregating Dd cells, each Cytobot is polarised and will always face toward the direction of travel. At each new position, P, an agent redirects itself to face a new random angle between 0 and 360 degrees (a tumble). The agent then moves forward in a straight line for a number of time-steps, based on the level of food detected at P (run). The duration of a run is longer on detection of higher levels of food and lower in less favourable conditions. The Cytobot consumes all the food (if present) at each location it passes through.

7.1.3 Cytobot Starvation Mode

The Cytobot Starvation mode is based on the pattern of motion displayed by starving cells of Dd (as previously described in *section 7.1.1*). The Cytobots enter Starvation mode if food has not been consumed within a time period. During this phase, the Cytobots respond to detected levels of environmental cAMP. Depending on the particular experiment, this chemical may already be present within the environment or it may be released by starving Cytobots. In this mode, both run and tumble behaviours differ from that in the Foraging phase. Rather than turning in a random direction, a new direction is calculated by weighting the turn direction towards the highest concentration of artificial cAMP within the surrounding area. The run period, instead of being variable, is a fixed length, which is set according to the particular experiment.

7.2 Cytobot ARN Design and Implementation

In this section, the design and implementation of the Cytobot ARN control system, which is responsible for all aspects of a Cytobot’s Foraging and Starvation behaviour, is discussed. Each Cytobot operates asynchronously with respect to the other agents - enabling it to react independently to situated environmental patterns, communicate with others and contribute to high-level collective function. The simulated environment in which the Cytobots are placed contains a distribution of artificial chemicals. These chemicals represent the attractants of either food or cAMP. When a Cytobot moves to a new position, the surrounding level of chemical is

used to set the inputs to its ARN. Consequently this changes the internal state of the network and updates the agent's trajectory. During this process, the agent modifies the state of the environment by consuming food or releasing cAMP. As shown later in the experiments detailed in *chapter 8*, representing the environmental chemicals in different ways enables the Cytobots to produce a range of interesting behaviours.

As previously discussed, the Foraging and Starvation modes are based on those of chemotaxing Dd cells during their Vegetation and Aggregation phases respectively. It was decided not to base the Cytobot ARN on the specific biochemical detail of the Dd chemotaxis CSN. Instead, a highly abstract approach was adopted. This decision was made because, unlike the *E. coli* chemotaxis CSN discussed in *section 5.1.2*, there is no general agreement about the specific protein interactions within the Dd chemotaxis pathway to enable accurate representations to be developed (*Manahan et al. 2004*). When modelling such a network, researchers often adopt a modular approach, where related signalling events are grouped into functional units (*Manahan et al. 2004*). Simulating the pathway in this way allows it to be modelled when some of the biochemical detail is unknown. In a similar way, the Cytobot ARN was designed by dividing function into modular units. In a further step toward abstraction, none of the actual biochemical reactions are represented. Instead, the Cytobot ARN modules are constructed by combining the functional motifs of real biochemical networks discussed in *section 2.3*. The advantage of this is that these mechanisms are universally found in other pathways, and thus show that the ARN is capable of potentially modelling any pathway. Similarly to the ARNs employed in *chapter 6*, the functional motifs and parameters were set manually. Although an EA could have been used, it would not have guaranteed that the structural motifs would be generated within the system.

The Cytobot ARN is composed of six subnetworks as shown in *Fig. 7.3* (a larger version of the diagram is given in *Appendix 3* for clarity). These are: the Master Oscillator; the Run Length Network; the Food Network; the Signalling Network; the Chaotic Network and the Weighted Direction Network. Each subnetwork contributes a functional aspect to either (or both) Starvation and Foraging behavioural modes. The Run Length Network controls the length of forward motion; the Signalling Network acts as a switch between the Foraging and Starvation modes; the Chaotic Network is used in Foraging mode to determine the angle to redirect. While in Starvation mode, the Weighted Direction Network determines this angle. Finally, the Master Oscillator synchronises each network to determine the overall output. Each of these subnetworks is discussed in detail in the following sections. A complete listing of all the network parameters including the input and output pools of each reaction, the connection weights, forward and reverse rates of each reaction, and the initial value of each pool is

provided in *Appendix 4*. Note, that in all the Cytobot experiments outlined in this work, the ARN uses conserved mass values (see *Eq. (5) section 1.2* for calculations for conserved mass).

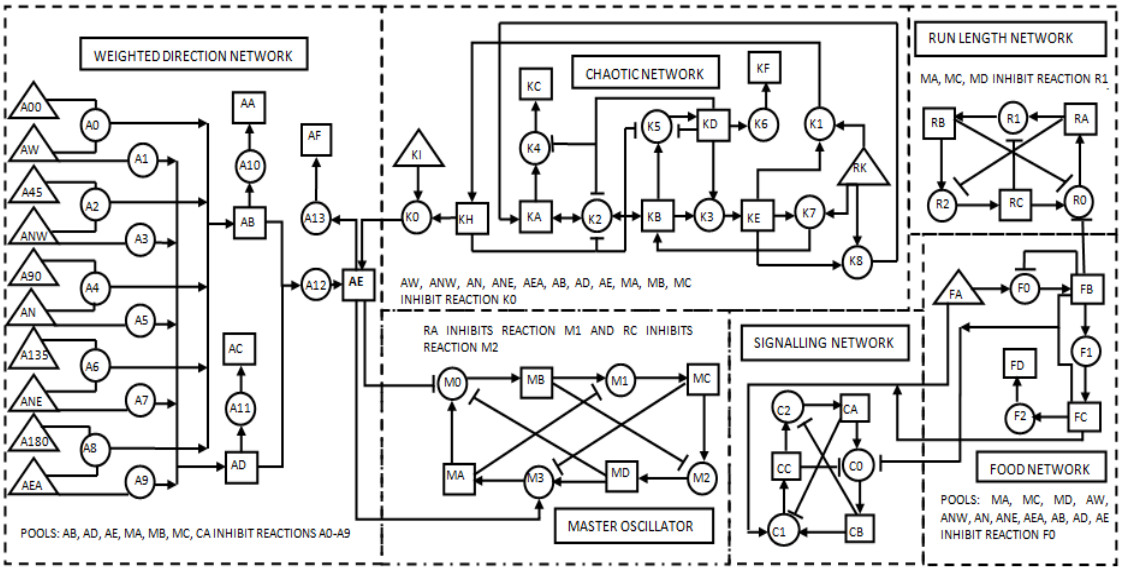


Fig. 7.3 The Cytobot ARN network. Each Cytobot is controlled by an instance of this network and thus has an independent state at time *t*. The network is composed of six subnetworks.*Pools are considered empty when the value of its component chemical is $\leq 1x10^{-3}$

7.2.1 The Master Oscillator

The Master Oscillator functions to synchronise the outputs from all the other subnetworks together, and provides the motor response at each time-step. It is a four component oscillator (*Table 2.1, motif 9*) with a token unit of chemical cycling around it. It consists of four reaction units: M0, M1, M2, and M3 (all with a reaction rate of 1), four pools MA, MB, MC and MD and generates a pulsed-width-modulated waveform. Each pool is associated with one of three behaviours. At every time-step, if a particular pool contains the token unit, then its corresponding behaviour is performed. Pool MA activates *turn*, MC activates *run* and MB and MD activates *stop*. Thus, if pool MC contains a chemical for a period of ten time-steps, the agent will move forward for ten time-steps. The other subnetworks inhibit (*Table 2.1, motif 2*) or excite (*Table 2.1, motif 1*) the reaction units of the Master Oscillator to allow or prevent chemical flow.

Note, that this oscillator motif allows the Cytobot ARN to function easily as the control system for the motor actuators of a wheeled robot. Here, MC would switch on all the wheel motors, while MA would switch on left-wheel motors only, thus turning the robot. The remaining pools would act as off switches.

7.2.2 The Food and Run Length Network

The Food Network interfaces with the environment at pool FA using an excitatory connection (*Table 2.1, motif 1*), and inhibits the Run Length network in accordance with the level of detected food. The forward rate of reaction at node F0 is 1, thus the content of FA is transferred to pool FB in a single time-step. The presence of chemical FB inhibits (*Table 2.1, motif 2*) R0 for a number of time-steps, according to the level of food (by setting forward rate of unit F1 to 1 and weight to 0; this can be an exact correlation). The Run Length network is a three component oscillator (*Table 2.1, motif 9*). While reaction R0 is inhibited, it prevents pool RC from emptying. RC inhibits reaction M2 (*Table 2.1, motif 2*) of the Master Oscillator thus preventing pool MC from emptying for the same number of time-steps. As discussed previously, the number of time-steps which pool MC contains the token unit, represents the number of time-steps to move forward.

7.2.3 The Signalling Network

The Signalling Network functions as a switch between Starvation and Foraging mode. Low food levels trigger the starvation response and allows the Weighted Direction Network to control each new angle. Sufficient food will switch off the Weighted Direction Network and allows the Chaotic Network to control each new angle. It is a three component oscillator (*Table 2.1, motif 9*) with a token unit of chemical flowing around it. Pool CA acts as the switch between Foraging and Starvation modes. Here the presence of chemical in CA inhibits the Weighted Direction Network (*Table 2.1, motif 2*), while its absence switches on the Weighted Direction Network; this in turn inhibits the Chaotic Network, as shown in *Fig. 7.3*. In this network, all reaction units have a forward rate of 0.5. This produces a continuously oscillating waveform and ensures a minimum number of time-steps for each behaviour. A NOR gate (*Table 2.1, motif 7*) activates pool CB in the absence of food chemical in both pools FB and FC of the food network, thus allowing pool CB to empty. An AND gate (*Table 2.1, motif 7*) will lead pool CA to eventually refill by activating pool CC, only when food is present in input FA and pool FC of the Food Network.

7.2.4 The Weighted Direction Network

The Weighted Direction Network senses cAMP within the agent's immediate environment and calculates a tumble angle which is weighted toward higher levels. This network interfaces with the environment via a number of receptor pools (AW, ANW, AN, ANE, AEA) which sense the level of food around the Cytobot. These pools represent receptors and are positioned at points

around the front of its perimeter (as shown in *Fig. 7.4*), allowing the agent to travel in a similar way to that of an aggregating Dd cell.

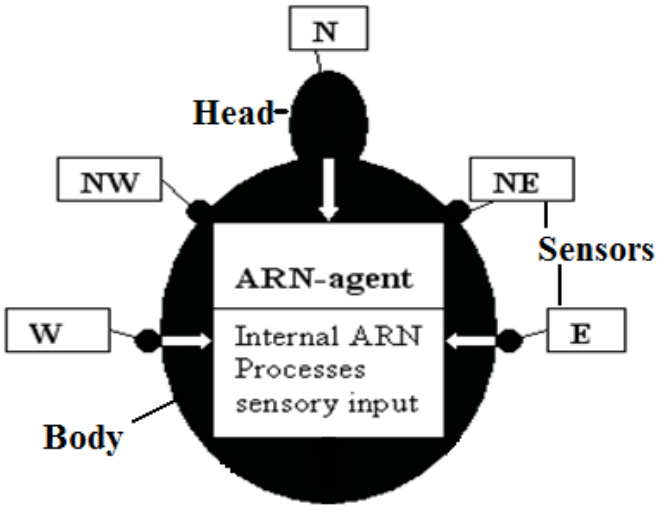


Fig. 7.4 Location of the Cytobot sensors around its perimeter.

Each receptor input pool forms one input of an AND gate (*Table 2.1*, motif 7); the other input is a static pool, containing a fixed level of chemical, corresponding to its direction. The direction pools start from AW (west) with a corresponding static pool of value 0 (A00) and progress in 45 degree steps through each direction to AEA (east) with a corresponding static pool of value 180 (A180). As the receptor positions around the agent are fixed, directions are always relative to the direction in which the agent is facing. All connections have a weight of 1 with the exception of the connection between pool AD and reaction A12 which has a weight of -1. This negative connection weight raises the sum of food detected in AD to -1, which multiplied by AB, allows an average angle to be calculated. Detected signals are classed as being in one of the following cardinal and ordinal directions: W, NW, N, NE, and E. Thus signals are detected from all directions above the horizontal plane. The calculated angle interfaces with the remaining subnetworks at pool AE. Pool AE is the output of an OR gate (*Table 2.1, motif 8*), and its inputs are activated by either the Weighted Direction Network or the Chaotic Network. AE also forms the inhibitory input of a SWITCH (*Table 2.1, motif 7*), where the presence of chemical in MA and absence in AE activates pool MB of the Master Oscillator. In the actual organism, receptors are set around the cell perimeter and direct movement appropriately. In this simulation, for simplicity, a count of the number of time-steps "*n*" that MA contains the token unit, is processed to gain the new heading "*h*" relative to the agents' current heading "*c*" using *Eq. (7.1)* and then applying *statement 1*:

$$h \equiv (n - 90) + c \tag{7.1}$$

Statement 1

```
IF (h > 360)THEN h = h – 360
IF (h < 0)  THEN h = h + 360
```

Thus, if the number of time-steps is 120 and the agent is facing north, then the current heading would equal 0 and the new heading would equal 30.

7.2.5 The Chaotic Network

The Chaotic Network, shown in *Fig. 7.3*, is responsible for generating the pseudo-random angles which agents use to perform each Foraging mode tumble. It is a networked implementation of a Logistic Map, given by *Eq. (7.2)*, where X_n is a state variable of value $0 < X_n < 1$ and λ is a system parameter of value $1 \leq \lambda \leq 4$:

$$X_{n+1} = \lambda X_n (1 - X_n) \tag{7.2}$$

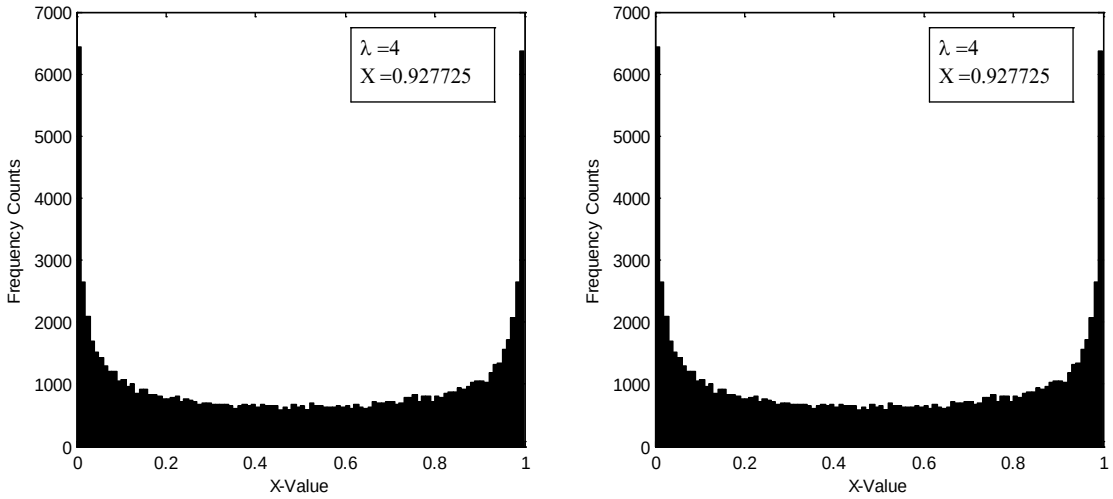
Without prior knowledge of the initial conditions, the output of the Logistic Map is not predictable; whereas, with prior knowledge it is deterministic. Therefore, the resulting series cannot be described as truly random, but as pseudo-random and its output has long been proposed as a pseudo-random number generator. Ulam and von Neumann (*1947*) were the first to examine this, and it has been successfully used in that capacity by several researchers (*Patidar et al. 2009*; *Phatak and Rao 1995*). The probability density distribution of the Logistic Map, as given by *Eq. (7.3)* (where P(X) is the probability of X occurring at any time-step), is non-uniform (*Patidar et al., 2009*):

$$P(X) = \frac{1}{\pi\sqrt{X(1 - X)}} \tag{7.3}$$

When λ=4, the distribution is "U" shaped, with a higher probability of values closer to the minima and maxima of X and a symmetric distribution around the midpoint.

To implement the Logistic Map, a number of motifs are combined, including multiple branch motifs (*Table 2.1, motif 6* - KB activates KD and KE), PFLs (*Table 2.1, motif 3* - a multi-component PFL exists where KA leads to activation of KE, which results in the activation of

KA) and NFLs (*Table 2.1, motif 4* - KA activates KD which in turn inhibits KA). At the start of the simulation, pools KA and KB are initialised to the same random value (a unique number for each Cytobot), between 0 and 1 (to 5 decimal places). This value represents the initial value of X of *Eq. (7.2).* All the other pools are initialised to 0, with the exception of the static pools KI and RK, whose initial values are 360 and 1 respectively. Reaction K2 is responsible for generating each new value of X and has a forward and reverse rate of 4 (the Logistic Map exhibits chaotic behaviour when λ is 4). The connection between KA and K2 has a weight of 1 and that between K2 and KB has a weight of 2. The remaining series of reactions function to copy the value of X three times; where two copies serve as the new initial values of KA and KB, and one participates in the final output of the network at KH. KI has a fixed value of 360 which allows the network to convert the pseudo-random number at KH to an angle value between 0 and 360 at reaction K0. However, reaction K0 cannot proceed until all eleven pools that inhibit it are empty. These inhibitory connections (*Table 2.1, motif 2*) ensure that random angles are not output while the agent is in starvation mode, and that pool AE is empty before adding more chemical. Appendix 5 provides a more detailed account of the operation of the Chaotic Network.



a) b)

Fig. 7.5 Frequency distribution for each value of X when X is initalised to 0.9277725 and λ=4 resulting from: a) the Chaotic Network b) Recursive relation given by *Eq. (7.2)* obtained using Matlab.

The ARN implementation of the Logistic Map was verified against the recursive relation shown in *Eq. (7.2)* using Matlab, where λ=4, initial X = 0.927725, and iterated for $1x10^5$ steps. The complete range of state-variables between 0 and 1 were divided into one-hundred equal subintervals and the frequency of occurrence of each subinterval interval was plotted. Similarly, the Chaotic Network component of the ARN was run for $1x10^5$ cycles, using the same

parameters of X (initial value) and λ. These results were processed in the same way and are *shown in Fig. 7.5*. The frequency distribution gained from the ARN is identical to that obtained using Matlab and by other researchers using the same parameters (*Patidar et al. 2009*). The same comparison was repeated one-hundred times at different values of X, and the ARN consistently produced the same values as *Eq. (7.2).*

In the following chapter swarms of Cytobots are used to investigate the dynamics of multi-agent chemical based systems and their applications in biological simulation and in robotics.

8. Cytobot Experiments and Results

In this chapter the dynamics, emergent behaviours and practical applications of multi-agent Cytobot systems are investigated in a number of experiments. These include: the simulation of *D. discoideum* amoebae during their Foraging and Aggregation life-cycle phases, solving benchmark search problems, and finally a simulated oil-spill clean-up operation.

8.1 Cytobot *D. discoideum* Simulation

The experiments outlined in this section examine the application of Cytobots in the simulation of emergent behaviours in interacting groups of biological cells. A number of Cytobots are instantiated within an artificial environment and used to simulate a population of *D. discoideum* (Dd) cells, where each Cytobot represents one cell. Two phases of the life-cycle of Dd are modelled: Vegetation and Aggregation (and their transition). The results are compared with the behaviour of real Dd cells and with other simulations.

8.1.1 Cytobot D. discoideum Simulation: Methodology

The experiments are grouped into two sets: Aggregation (AG1-10 of *Table 8.1*), which models the Aggregation phase only, and Foraging to Aggregation (AGF3 and AGF8 of *Table 8.1*), where both the Vegetative and Aggregative phases are simulated including the transition between them. Each experiment is performed at varying population densities of Cytobots (p) and different ranges of detection of cAMP (r). The experiments AGF3 and AGF8 are performed with the same p and r as experiments AG3 and AG8 respectively, to compare the effect of the Foraging phase on the number of mounds formed and length of time to complete the Aggregation phase. The emergent patterns, numbers of mounds, and length of time to complete phases is examined and compared in both sets of experiments and with the literature. In each experiment the Cytobots move within a 2D simulated environment, which represents a physical area of 5.06 mm^2- approximately half the maximum aggregation territory reported in the literature (*Dallon and Othmer 1997*). A screen output shows the position of the Cytobots in real-time and is a grid of 500 × 500 pixels, where each is represented by a square of side 4.5

μm. In nature, aggregating Dd cell densities are typically 250 to 1000 per mm^2 (*Dallon and Othmer 1997*). Due to the demands on computational resources required to manage a population of Cytobots within the upper range, a cell density at the lower biological range of 250 agents per mm^2 (1250 Cytobots) and another at 150 per mm^2 (750 Cytobots) were chosen.

TABLE 8.1 *D. discoideum* Simulation Results

Setup			**Results**			
No.	Cytobots per mm^2 (p)	Range (r) in mm	Mean No. of mounds; (σ)	Minimum no. of mounds	Maximum no. of mounds	Aggregation Phase Mean time in Hours; (σ)
AG1	150	5	1 (0)	1	1	8.98 (0.09)
AG2	150	2.5	4 (0.31)	3	4	9.63 (0.17)
AGF3	150	1	5.9 (1.06)	5	7	9.47 (0.65)
AG3	150	1	5.2 (0.82)	4	6	9.92 (0.34)
AG4	150	0.5	8.4 (1.19)	7	9	10.23 (0.59)
AG5	150	0.1	14.2 (2.36)	12	16	10.6 (1.82)
AG6	250	5	1 (0)	1	1	8.95 (0.11)
AG7	250	2.5	1 (0)	1	1	9.6 (0.20)
AGF8	250	1	6.8 (1.81)	6	9	9.71 (0.87)
AG8	250	1	4.3 (0.37)	4	5	10.05 (0.58)
AG9	250	0.5	6.7 (1.62)	5	8	12.65 (1.94)
AG10	250	0.1	-	-	-	-

In both sets of experiments, the Cytobots are initialised at random positions in Foraging mode within the simulated environment. In the AGF experiments, the environment is initialised with a radial outwardly-decreasing gradient of food (*z*), as described by *Eq. (8.1)*, where *x* and *y* are Cartesian coordinates on the horizontal plane:

$$z = \sqrt{x^2 + y^2} \tag{8.1}$$

The Cytobots begin the experiment in the previously described Foraging mode and remain in this mode until the food resource is depleted and Starvation mode is triggered. In a real environment, food is non-uniformly distributed, may be regenerated and can move (in the case of bacterial prey).

```
FOR each Cytobot
    Get current agents' facing direction C_F
    Assign a value to direction C_F using statement 1

        FOR each (index n) detected cAMP signal
            Get detected signal incoming direction C_A
            Assign a value to direction C_A using statement 1
            IF CA = C_F THEN k_n = 3
                    ELSE IF C_A = C_F-1 OR C_A = C_F+1 THEN k_n=2
                ELSE IF C_A = C_F-2 OR C_A = C_F +2 THEN k_n=1
              ELSE k_n=0
             END IF
            Calculate distance d_n
            Store each C_A with k_n and d_n
        END FOR

    Calculate W_A for current agent using Eq. (8.2)
END FOR

Statement 1: East = 1; North East = 2; North = 3; North West =4; West = 5


Where:
W_A= total weight of direction A
N= total number of agents within range of detection
d_n= distance of current agent from agent n
C_A = direction of incoming signal detected by current agent
C_F = the current agents facing direction
k_n = value of cAMP signal from agent n
```

Fig. 8.1 The strength of signal for each cardinal or ordinal direction above the horizontal plane of a Cytobot is calculated using this pseudocode. The result for each direction is used to set the corresponding direction input pool of the ARN Weighted Direction network.

$$W_A = \sum_{n=1}^{N} \frac{k_n}{d_n} \tag{8.2}$$

Thus, this setup is simplified, but is comparable to other simulations (*Becker 2010*). If a Cytobot does not detect food for a period of approximately five time-steps it will enter Starvation mode. The exact number depends on the level of food detected in the recent past, because higher levels take longer to flow through the network. Cytobots in Starvation mode emit a cAMP signal into their environment at equal strength in all directions around their

circumference. Each Cytobot in Starvation mode detects the cAMP signal of all other starving agents within a radius r. The total value for each direction is calculated using the pseudocode given in *Fig. 8.1* and these totals are used to set the Weighted Direction network input (receptor) pools.

A range of r values were explored, including that of real Dd cells: 1, 0.5, and 0.1 mm (*McCann et al. 2010*). The cAMP signal degrades linearly with increasing distance from the emitting cell. Each cycle represents 1 minute of time. In this time an aggregating Cytobot moves 9μm - a distance which corresponds to that of actual aggregating Dd cells (*Rifkin and Goldberg 2006*). Therefore, after 1 hour of motion, a Cytobot travels a distance of 540μm. In this simulation, just as in biology, there are always remaining cells that do not aggregate, and thus the simulation runs until 95% of agents are at a distance of less than 0.1mm (100μm) from their nearest neighbour.

8.1.2 Cytobot D. discoideum Simulation: Results and Discussion

Each experiment was performed one-hundred times. The results for each experiment, given in *Table 8.1*, show the minimum and maximum total number of mounds formed, the mean and standard deviation (σ) of the number of mounds formed and similarly the mean and standard deviation of the number of hours taken to complete aggregation. The results were found to have a normal distribution. In the AG experiments an increase in p by 100 per mm^2 resulted in a decrease in the number of mounds formed at each value of r, with the exception of experiment AG6. This is not surprising, as denser populations have more chance of interacting, and thus form fewer clusters, each having a higher number of agents. Similarly, decreasing r results in a general increase in the number of mounds formed at both values of p. The likely reason for this is that as r decreases the Cytobots' area of influence becomes increasingly smaller, and thus the number of isolated stable clusters with fewer agents increases. In the AGF experiments, agents generally focus on consuming food in each of the remaining areas of highest concentration (*see Fig. 8.2 K-L*). Having consumed almost all the food, agents begin switching to Starvation mode (*Fig. 8.2 M*). In these experiments the number and location of resulting mounds differs from that of the AG experiments at the same values of r and p. For example, experiment AG8 results in an average of 4.3 mounds within the test space (*Fig. 8.2 E*) while AGF8 results in an average of 6.8 mounds and a general shift in mound formation further away from the centre of the environment (*as shown in Fig. 8.2 O*). The likely explanation is that, at the time of switching to aggregation, the majority of cells were forced outward toward the next remaining highest concentration of food. Emergent behaviours and clustering patterns similar to the biological organism were also observed.

In experiments AG8-10 and AGF8, the value of r and p are within the ranges for real Dd cells. These experiments are used to compare the behaviours and aggregation time with the values for real Dd in the literature. In experiments AG8-9 and AFG8, mound formation completes within the range reported for the actual organism of 9-13 hours (*Cotter et al. 1992*; *Becker 2010*). These results are comparable with other work. For instance, Becker (*2010*) reports an aggregation time of 11.6 hours for a simulated population of Dd with a cell density of 200mm^2. In experiment AG10, the population never satisfied the criteria for completion of mound formation and instead the agents appeared to move in a fashion reminiscent of Brownian motion. The likely explanation is firstly, because the simulation does take into account glycoprotein's which allow aggregating cells to attach together on contact; secondly, because r is small, fewer agents are detected by each Cytobot. Thus, momentarily larger clusters with higher attraction strength go undetected and quickly dissipate - an effect that would not occur if agents stayed together.

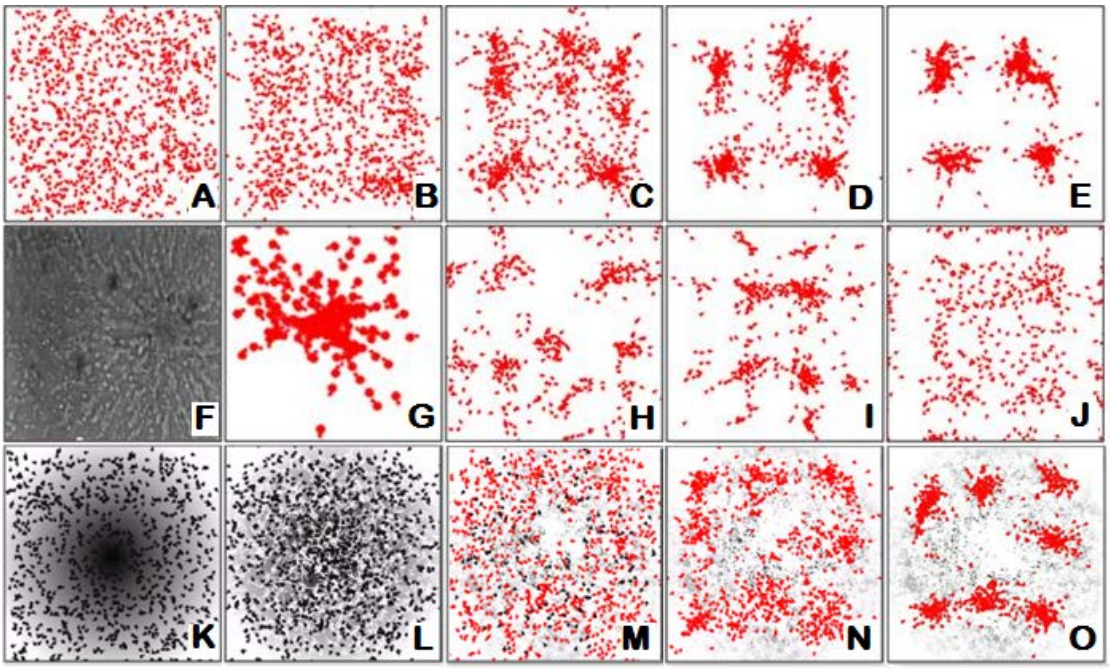


Fig. 8.2 Screenshots of the Dd simulation. Dots represent the Cytobots (black - vegetative and red - aggregative), and greyscale colour represents the food distribution. Screenshots A-E show experiment AG8 at A- 1hr, B- 2hr, C-5hr, D- 8hr, E- 10hr; Image F shows real Dd cells aggregating; Screenshot G shows streaming behaviour (close-up lower right-hand C); Screenshot H-J shows pattern formation; Screenshot K-O shows experiment AGF8 where: K- 0hr vegetation, L- 4hr vegetation, M-transition to aggregation, N- 5hrs aggregation, O- 10hr aggregation.

Diagram F courtesy of T, Gregor, Laboratory for the Physics of Life, Princeton University, 2013 Used with permission.

As previously discussed, the Cytobots are polarised. Implementing the agents in this way allowed us to observe whether or not the previously described streaming behaviour occurs. A close-up of the right-hand corner of screenshot C is shown in *Fig. 8.2 G* and shows the agents beginning to form a cluster. The protruding head of each agent can be seen clearly, where each lines-up its head to the rear of another agent and forms a stream. As can be seen in *Fig. 8.2 F*, this is very similar to the streaming behaviour in real cells of Dd. Other emergent patterns occurred during different experiments including spirals (*Fig. 8.2 H*), symmetric patterns (*Fig. 8.2 I*), and waves (*Fig. 8.2 J*).

These results show that the Cytobots are able to simulate behaviour of individual unicellular organisms, and the emergent behaviours arising from their interaction. It highlights a potential use as a means to simulate groups of interacting cells - for example, a bacterial colony or tissue component within a multicellular organism. Applications include the modelling of the effects of disease (e.g. faulty gene expression) and pharmaceuticals on global behaviour. The results demonstrate the parallels between ARN agents and their biological counterparts; like amoebae, their internal network of spatially distributed dynamic chemical species allows them to autonomously coordinate and direct their movement, recognise and respond to patterns in the environment, and produce high-level behaviour.

8.2 Cytobot Optimisation Experiments

In the following experiments, the application of Cytobots to swarm robotics is investigated by applying them to search tasks within a variety of simulated landscapes. The aim is to show that these agents can perform a range of useful search behaviours in a variety of situations, and that their search strategy can compare to that of established optimisation algorithms using similar cell inspired strategies. The Cytobot design and behavioural set remain exactly the same as in the previous Dd simulation, but the environment is modified to represent each search problem.

8.2.1 Cytobot Optimisation Experiments: Methodology

The Cytobots task is to find the global minima in a number of 2D benchmark optimisation problems. These problems were chosen as they reflect a variety of real-world terrains and thus allow the limits of the Cytobot search capacity to be explored. These are the following functions: Rosenbrock, Peaks, Inverted sinc, and Bowl (*see Table 8.2* for formulae, domains, and minima).

TABLE 8.2 Formulae, domain and minima for the Rosenbrock, Bowl, Peaks, and Inverted sinc functions used in the search space experiments.

Function	Formulae	Domain	f(x*) = f*
Rosenbrock	$f(x) = 100 \cdot (x_2 - x_1^2)^2 + (1 - x_1)^2$	$x_1, x_2 \in [-2,2]$	$f(1,1) = 0$
Rosenbrock	As above	As above	As above
Rosenbrock	As above	As above	As above
Bowl	$f(x) = \sqrt{x_1^{\,2} + x_2^{\,2}}$	$x_1, x_2 \in [-1,1]$	$f(0,0) = 0$
Peaks	$f(x) = 3*(1 - x_1)^2 \cdot \exp(-(x_1^{\,2}) - (x_2 + 1)^2)$ $-10 \cdot (x_1/5 - x_1^{\,3} - x_2^{\,5}) \cdot \exp(-x_1^{\,2} - x_2^{\,2})$ $-1/3 \cdot \exp(-(x_1 + 1)^2 - x_2^{\,2})$	$x_1, x_2 \in [-3,3]$	$f(0.23,-1.63)$ $= -6.55$
Inverted sinc	$f(x) = -1 \cdot \left(\frac{\sin\left(\sqrt{x_1^2 + x_2^2}\right)}{\sqrt{x_1^2 + x_2^2}} \right)$	$x_1, x_2 \in [-10,10]$	$f(0,0) = -1$

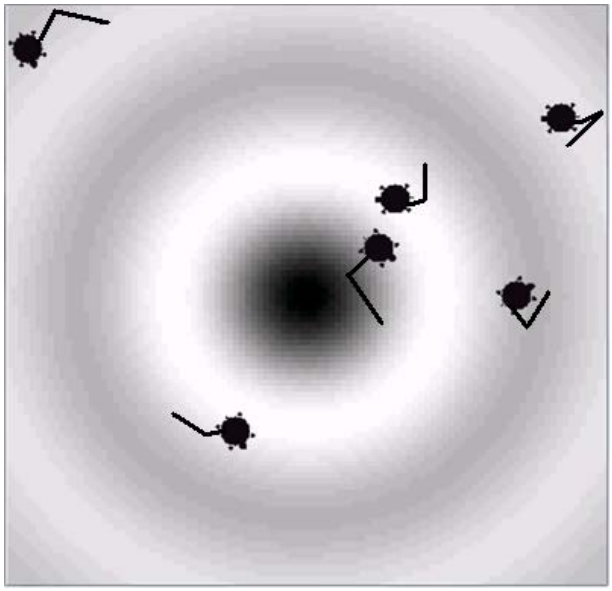


Fig. 8.3 A screenshot of the simulation showing six Cytobots in the Inverted sinc search space. The greyscale colour represents the food distribution.

Three experiments were performed for each function, where each uses either 1, 3 or 6 Cytobots and was performed one-hundred times. The task of the agents was to find the minima of the functions within as few evaluations (reading value of food at current x, y-coordinate) as possible. The range of output values for each of the functions represents the concentration of food (also the fitness of an agent at that point) within a simulated environment. Values approaching the minima represent higher food levels, and those approaching the maxima represent lower food levels. The simulated environment consists of a 2D area of 400 x 400 pixels. A scaling factor was used to map the domain to the actual dimensions of the simulation,

e.g. Rosenbrock domain of [-2, 2] mapped to a simulation space of [-200, 200] by a scaling factor of 100. For display purposes, a corresponding grayscale colour was used to show the distribution of food within the environment as displayed in the screenshot of the simulation in *Fig. 8.3* (see *Appendix 6* for screenshots of the other functions). A token marks the current position of each agent (*see Fig. 8.3*) and each agent has an instance of an ARN network. At the start of each experimental run, the agent's ARN network is initialised as described previously and it is positioned at random x and y coordinates within the search space. The agents undergo alternating phases of "searching" and "repositioning" for a number of cycles until one reaches a position of within 0.04 of the global minima of the function. This value was chosen as it is within 1% of the global minima for all the functions used. The high-level pseudocode describing the searching and repositioning phases is provided in *Fig. 8.4*.

```
INITIALISE Cytobots
WHILE (best fitness > 0.04 of global minima)
    START search phase

        WHILE (more searching phase moves)
            FOR each agent start searching phase
                Turn agent
                Set receptor pools of Food Network
                Set receptor pools of Weighted Direction Network
                Move agent forward
                IF (new food level > previous food level)
                    Record fitness
                    Record current position
                END IF
            END FOR
        ENDWHILE
    END search phase

    START reposition phase
        CALCULATE new central point P to reposition
        INITIALISE agents at new position
    END reposition phase
END WHILE
```

Fig. 8.4 High level pseudocode for each experiment

Searching is characterised by the two ARN controlled behavioural modes - Foraging and Starvation, as described in *section 7.1*. During each search phase an agent performs a total of three moves (three evaluations of the environment). The length of a run corresponds to the number of pixels a Cytobot moves forward and is subject to the output from the Run Length network. After each tumble, and before moving forward, the food level at the current position is input into the ARN network as described previously. The agents travel at a speed of one pixel per time-step; thus the number of time-steps produced by the Run Length network corresponds directly to the number of pixels the agent moves forward. As a Cytobot travels, the food at each

passing position is consumed and its path within the simulation is represented in black (as shown in *Fig. 8.3*). During the search phase, a central control unit, external to all Cytobot agents, keeps track of each Cytobots best fitness and the coordinates of that value.

Let $A = \{a_O, \cdots, a_N\}$ equal the set of all agents

$$f_{tot} = \sum_{a=0}^{N} f_a \tag{8.3}$$

Then $\forall a \in A \setminus \{a^*\}$

$$f_{ra} = \frac{f_a}{f_{tot}} \tag{8.4}$$

$$d_{ax} = x_a - x_{a^*} \tag{8.5}$$

$$d_{ay} = y_a - y_{a^*} \tag{8.6}$$

$$p_x = x_{a^*} + \left(\sum_{a=0}^{N} d_{ax} \times f_{ra} \right) \tag{8.7}$$

$$p_y = y_{a^*} + \left(\sum_{a=0}^{N} d_{ay} \times f_{ra} \right) \tag{8.8}$$

$$x_a = rand(r) + p_x \tag{8.9}$$

$$x_a = rand(r) + p_y \tag{8.10}$$

Where:

a*= agent with highest fitness

f_a= fitness of agent a_n

f_{tot}= total fitness of all agents

f_{ra}= ratio of agent a_n fitness to f_{tot}

x_{a*}= the x-coordinate of a*

y_{a*}= the y-coordinate of a*

x_a= agent a_n x-coordinate

y_a = agent a_n y-coordinate

d_{ax}=difference between x_a and x_{a*}

d_{ay}= difference between y_a and y_{a*}

p_x=total of all d_{ax}

p_y= total of all d_{ay}

rand(r)= random value within a defined radius *r*

After completing the searching phase, agents switch to the repositioning phase. This phase is used to focus searching toward areas containing higher food levels and represents the movement of a Dd slug during the migration stage of its life-cycle (see *section 7.1.1*). When the Cytobots enter the repositioning phase, the central control unit processes each agent's best fitness position to compute a new central point P, weighted in favour of higher fitness, as described by *Eq. (8.3-8.10)*. Agents are then repositioned randomly within an area of radius *r* from point P to begin the next search phase. For the purposes of this simulation, travelling to the new position was not modelled, as this does not affect overall behaviour and would only occur if the Cytobots were applied to real-world environments.

8.2.2 Cytobot Optimisation Experiments: Results and Discussion

The experimental results are displayed in *Table 8.3*. For each experiment, the average best, f_a, and best solution, f_b, for one-hundred independent runs are presented. The average number of evaluations and the standard deviation for all agents is displayed as "Avg Eval for all agents" and "Std Dev" respectively (the results are normally distributed). The average number of relocations for each agent is presented in the final column as "Avg Reloc per agent". In all experiments the Cytobots were able to find the global minima. Cytobots performed best in Bowl and Rosenbrock functions, where, using 6 Cytobots, the average number of total evaluations and relocations per agent respectively for Bowl was 56.4 and 2.1 and for Rosenbrock was 79.8 and 3.4. The Cytobots performed least well in the Inverted sinc search space, where the lowest number of total evaluations was 94.8 using 6 Cytobots. In all the experiments, a slight increase in the number of Cytobots generally results in a significant reduction in the total number of evaluations performed. This is most significant for Peaks where using 3 and 6 Cytobots results in approximately 30% and 60% respective reductions in the total number of evaluations when compared to the results for 1 Cytobot. The Mann Whitney U test was used to determine any significant (95% confidence) statistical difference in the total number of evaluations (for all agents) between experiments using 1 & 3 and 3 & 6 Cytobots. In all experiments there was a significant difference between 1 & 3 agents, with the exception of the Inverted sinc function. In Peaks there was a significant difference in all experiments, while in the Inverted sinc there was no significant difference found. Thus increasing the number of Cytobots from 1 to 3 both reduces the time to find the global minima and the number of evaluations, but this effect can be quickly reversed if too many Cytobots are added.

TABLE 8.3 Cytobot optimisation experiment results. The average best (f_a) and best solution (f_b) for one-hundred independent runs are presented. Average number of evaluations (Avg Eval) and standard deviation (*Std Dev* σ) for all agents is given. The average number of relocations for each agent is given as "Avg Reloc per agent".

Function	No. of Agents	Avg Best (fa) and Best (fb) Results	Avg Eval for all agents (*Std Dev σ)*	Avg Reloc per agent
Rosenbrock	1	$f_a = 0.04$ $f_b = 0$	98.5 (16.3)	32
Rosenbrock	3	$f_a = 0.03$ $f_b = 0$	84.6 (9.5)	8.4
Rosenbrock	6	$f_a = 0.01$ $f_b = 0$	79.8 (4.8)	3.4
Bowl	1	$f_a = 0.03$ $f_b = 0$	81.9 (14.8)	26.3
Bowl	3	$f_a = 0.02$ $f_b = 0$	64.2 (7.6)	6.1
Bowl	6	$f_a = 0.02$ $f_b = 0$	56.4 (5.8)	2.1
Peaks	1	$f_a = -6.51$ $f_b = -6.55$	151.7 (59.2)	49.6
Peaks	3	$f_a = -6.51$ $f_b = -6.55$	108.9 (17.4)	11.1
Peaks	6	$f_a = -6.52$ $f_b = -6.55$	64.8 (5.7)	2.6
Inverted sinc	1	$f_a = -1.04$ $f_b = -1$	163.5 (63.4)	53.5
Inverted sinc	3	$f_a = -1.03$ $f_b = -1$	109.5 (30.1)	11.2
Inverted sinc	6	$f_a = -1.03$ $f_b = -1$	94.8 (10.1)	4.2

The paths of agents through the search space indicate reasons for variation in the results. In simple landscapes such as Bowl, agents descend steadily toward the minima, as shown in *Fig. 8.5*. Similarly in Rosenbrock, agents quickly descend to the narrow valley and are forced to steadily move along it by moving up the nutrient gradient created by the consumption of food, until finding the global minima. In Peaks, agents move from their initial positions and search many parts of the domain. Figure 8.6 shows the agents' trajectories, one can see that peaks are avoided and troughs are pursued. However, if fewer agents are used they may quickly become trapped in local minima causing a significant rise in the number of evaluations. Increasing the number of agents by a small amount expands the amount of search space explored per cycle, and increases the chance of finding better solutions and (or) leaving local minima. Another possibility is to increase the number of moves for each searching phase, thus allowing an agent

to travel a sufficient distance to escape local minima. Similar solutions could be adopted in the Inverted sinc search space.

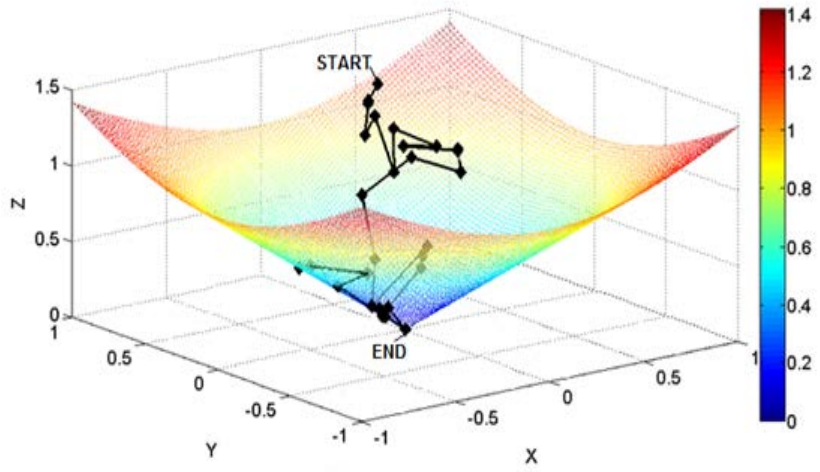


Fig. 8.5 Typical path of a Cytobot in the Bowl search space

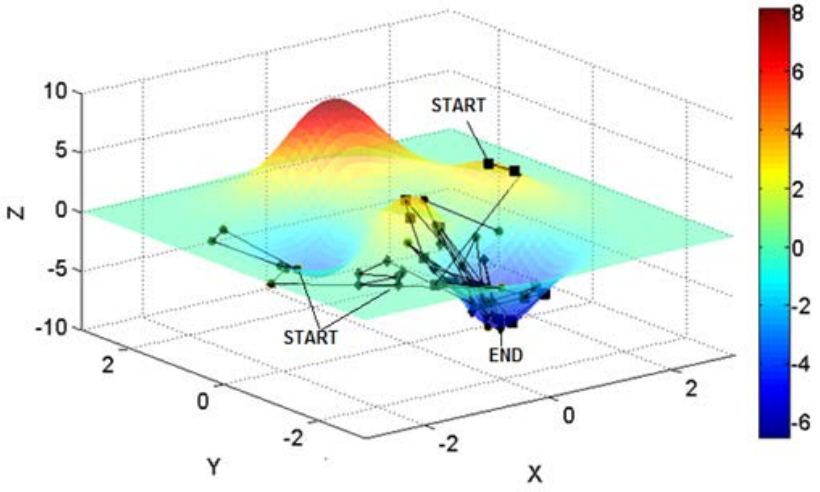


Fig. 8.6 Typical path of three Cytobots in the Peaks search space

These results were compared with other optimisation algorithms inspired by behaviours of single-celled organisms. For example, Passino developed the Bacterial Foraging Optimisation Algorithm (BFOA), inspired by foraging behaviours, reproduction and dispersal events in the life-cycle of *E. coli* (*Passino 2002*). Like the foraging behaviour of the Cytobots, movement is modelled as a biased random walk, where, after each random redirection, the cell moves forward a length according to current food levels. In a nutrient hill-climbing experiment

(without swarming effects), 50 cells are initialised at random starting positions within a 2D search space. This search space is similar to Peaks but with five troughs and a domain of [30, 30]. Similarly to the Cytobots the cells tend toward valleys and avoid peaks. After four generations (four reproductive steps), and moving one-hundred chemotactic steps (moves) between generations, the cells find the global minima.

Similarly in other work, Chen et al. (*2009*) applied BFOA using 6 cells to the 2D Bowl function with domain [-5,5], and the global minima was found within 50 chemotactic steps. In our experiments, 6 Cytobots find the global minima after an average of 9.4 evaluations, which is the equivalent to 9.4 moves (or 9.4 chemotactic steps in the terminology of *Chen et al*). After adjusting for the difference in domain size, the numbers of moves are highly consistent for Cytobots and the cells in BFOA. In other related work, Monismith and Mayfield (*2008*) created the Slime Mould Optimisation Algorithm inspired by the life-cycle of Dd. The state space is represented as a sparse mesh which cells populate and make modifications to, for example, deposit attractant. Using a combination of behavioural states inspired by the life-cycle of Dd, artificial cells perform local searches, and move to positions in favour of their personal best and the best fitness of their neighbourhood. The Slime Mould Optimisation Algorithm, like the Cytobots, finds the global minima of the 2D Rosenbrock function.

The results presented above show that the agents are able to find best fitness solutions in all problems, and match the performance of cell inspired optimisation algorithms in similar search spaces. Increasing the number of agents by small increments (two or three), can halve the number of function evaluations required to find the global minima. These experiments serve as a preliminary to implementing ARN systems to control real-world distributed autonomous robotic agents. Such agents could be applied to search problems in real-world environments. For example, oil-spill clean-up operations - as illustrated in the next experiment, where the objective is to travel to higher concentrations of oil, while consuming it at each passing location. The search behaviour results from the internal ARN network, but is enhanced by the collective activities and stigmergic interaction of the agents. The Cytobots obviously do not compare directly with conventional optimisation techniques like Genetic Algorithms, since they have a complex internal structure. However, this is not their purpose and they may be much more effectively utilised as independent control systems in autonomous robot-like agents.

8.3 Cytobots Oil-Spill Clean-up Simulation

To illustrate a practical application, the Cytobots were used to tackle a simplified oil-spill clean-up simulation. In these experiments, the same ARN used previously is used to perform different behaviours by altering its interface with the environment.

8.3.1 Cytobots Oil-Spill Clean-up Simulation: Methodology

The Cytobots move within a 2D environment containing an oil-spill on water. This oil is analogous to a distribution of food within a nutrient landscape. The task of the Cytobots is to clean-up the spill as quickly as possible by consuming oil at each location. The agents move through the environment by switching between the two previously described behavioural modes - Foraging and Starvation. To enable the Cytobots to behave differently, rather than modify the network, the interface between the Cytobots and the environment was altered. To achieve this, the concentration of oil surrounding the agents was used to represent both food and cAMP attractants. Thus, the amount of oil at each new position was fed into both the receptor pools of the Weighted Direction network and those of the Food network. At the start of each experiment, the Cytobots were distributed randomly within the environment, and the ARN network was initialised as previously described. The agents start the simulation in Foraging mode but during the simulation alternate between Foraging and Starvation modes. Starvation behaviour is triggered when the last positions visited (minimum of two) contained zero food. In Starvation mode, instead of turning in a random direction, the new direction is weighted toward higher concentrations of food within its surrounding area as previously described. This behaviour forces exploration of unexplored search space, because previously visited positions have a food level of zero. Consumption of environmental food therefore acts as a stigmergic signal, where agents are inclined to move up the nutrient gradient created by their foraging activities. On consuming a sufficient amount of food, the Cytobot switches back to Foraging mode, repeating this behaviour until 95% of the oil is consumed. Here, we model the spillage of 100 tonnes of Statfjord crude oil at 15^0C under a wind speed of $5ms^{-1}$ The oil is distributed over a 2D sea surface of 300m by 200m, an area of $60000m^2$, where two pixels corresponds to 1m, as shown in *Fig. 8.7A*. The particular oil type, simulation variables and parameter set were chosen in order to compare directly with work by Kakalis and Ventikos (*2008*) who present a robotic swarm concept for oil-spill confrontation. For this reason, we account for an initial response time of (a virtual) 14 hours. Based on the mathematical models found in Kakalis and Ventikos (*2008*), which account for the main factors of short term changes in oil characterisation, the volume of oil after 14 hours is reduced to $150m^3$. Beyond this starting state, the volume is only influenced by the Cytobots. The speed of each agent is $0.5ms^{-1}$ and is

based on other robotic agents in oil cleaning scenarios (*Kakalis and Ventikos 2008*); thus the Cytobots move one pixel (0.5m) for every time-step. The actual cleaning surface is 1m, thus the Cytobots clean a two pixel wide area in each time-step.

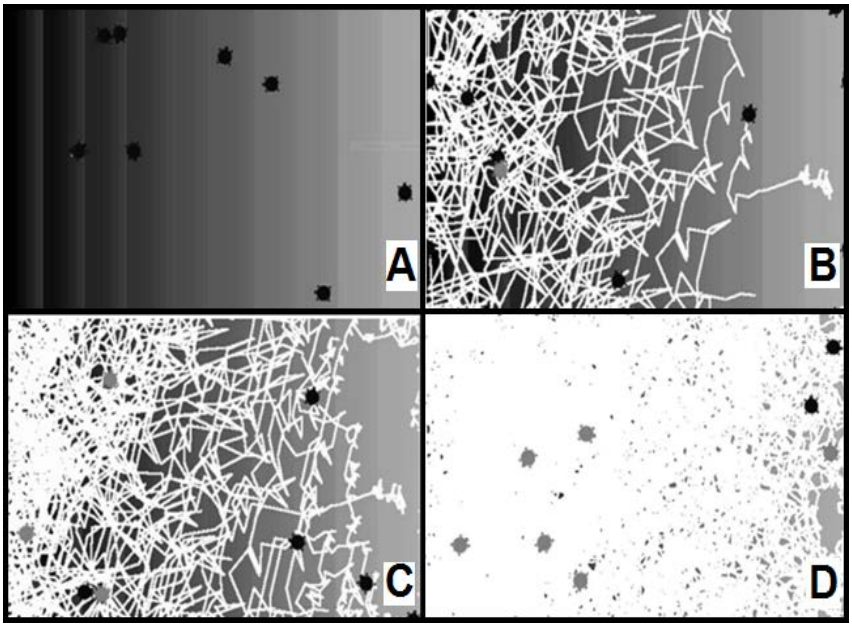


Fig. 8.7 Oil simulation using 8 Cytobots at A- 0 hours, B- 2 hours, C- 4 hours and D- 9.6 hours

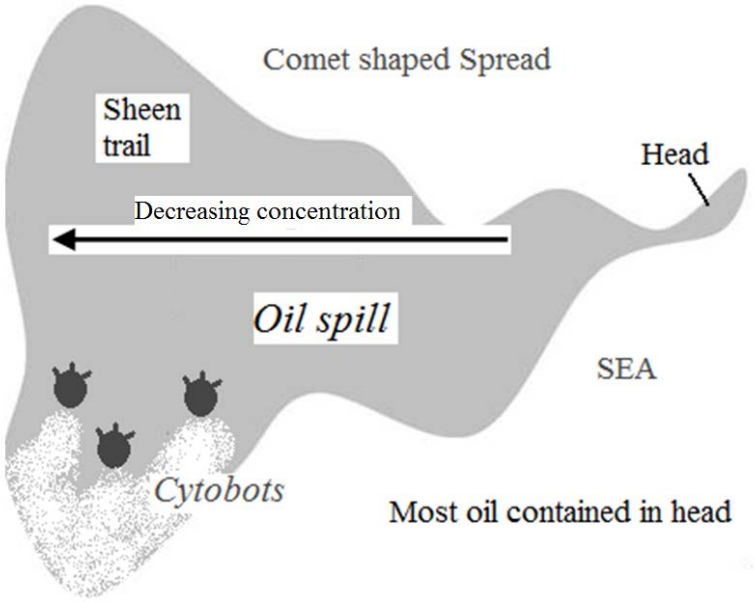


Fig. 8.8 Typical "comet" shaped oil-spill shape as seen from above. Most oil is contained in the narrow head section closest to the release point. Environmental factors such as wind or current spread the oil to create a trail.

Mathematical modelling of oil-spill dynamics is non-trivial and at best can offer a crude approximation of its actual trajectory. Most oil-spills quickly form a comet shape (*see Fig. 8.8*) with most of the oil within the head, and a trail of sheen (*Wang and Stout 2007*). To represent a simplified version of the comet-shaped spread, the simulation area is divided into one-hundred 3m x 200m segments. The first segment contains 0.015 tonnes of oil, and each subsequent segment increases by 0.03 tonnes from right to left.

8.3.2 Cytobots Oil-Spill Clean-up Simulation: Results and Discussion

In each experiment, a different number of Cytobots was deployed - 3, 5, 8 and 15 and the recovery rate achieved by each group was compared. The simulation time was measured from deployment of the Cytobots at 0 hours (14 hours after the oil was spilled) and stopped when the Cytobots had collectively removed 95% of the $150m^3$ of oil. Each experiment was run one-hundred times, and the average volume of oil consumed at 6 minute intervals was calculated. Figure 8.9 presents the average volume of oil consumed by the group of Cytobots against time. The results for each experiment were found to be normally distributed. Table 8.4 provides the average length of time taken to clean 95% of the oil (Avg. time) and standard deviation (σ) for each experiment. By adding 2 additional agents to the group of 3, the length of time is reduced by 3.7 hours, thus 1.85 hour average difference per extra Cytobot. This difference decreases 1.12 hours per Cytobot for 8 agents, then to 0.76 per agent for 15. The variation can be accounted for by examining the agents' paths through the oil. Rates are much faster at the beginning of the experiments, where Cytobots move toward the oil-rich left-side of the environment. This can be seen in the series of screenshots of a typical experiment shown in *Fig. 8.7*, where A shows the starting position at time 0, and B shows that after 2 hours the Cytobots have moved toward the left-hand side, focusing mainly on highly concentrated areas (consumed oil is shown in white). Initially, the rate of oil removal is high because Cytobots focus on the highest concentration areas and as the landscape remains largely unexplored they are unlikely to go over their path, thus each new location results in consumption of oil. However, as time progresses, large patches become cleaned and a higher probability exists for the Cytobots to revisit previously cleaned areas. The consumption of oil in *Fig. 8.7 C-D* at 4 and 9.6 hours respectively shows more clearly that Cytobots focus cleaning efforts on the area of highest concentration first, and are gradually forced to move toward the next highest concentration by the gradient created by their foraging activities. Figure 8.7 D shows the state of the oil at the end of the simulation, where only small patches remain mainly in areas of low oil concentration.

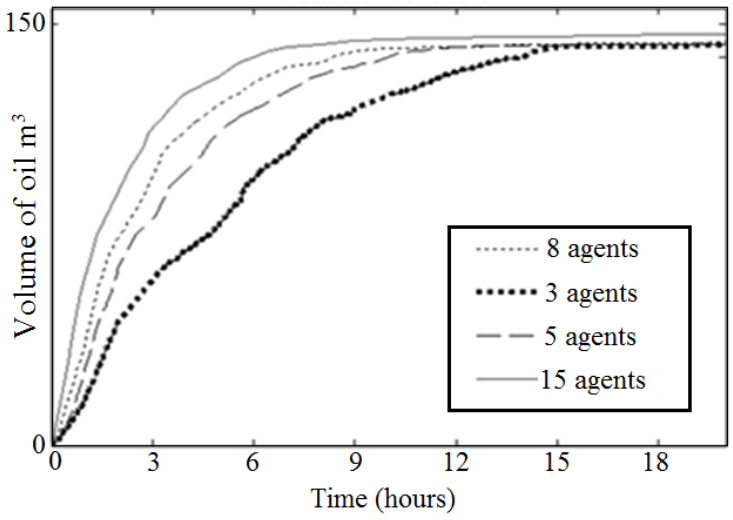


Fig. 8.9 Average volume of oil cleaned against time for each group of Cytobots

TABLE 8.4 Average length of time taken to clean 95% of the oil-spill for each group of Cytobots

No. of Cytobots	Avg. finish times	Standard deviation (σ)
3	15.2	3.4
5	11.5	2.7
8	9.6	2.8
15	6.1	3.1

These results can be compared to the simulation by Kakalis and Ventikos. Here, varying numbers of simulated EU-MOP robots are deployed to tackle $150m^3$ of Statfjord oil over $60000m^3$ (as before). In this case, the robots have a slightly faster speed of 0.54m/s but have the same 1m skimming face. Each EU-MOP robot has a storage capacity of $2m^3$ and a transit speed of $2.1ms^{-1}$. The times taken for 3, 5, 8, and 15 EU-MOPS are 54, 32, 20 and 10 hours respectively. For comparison, the results of our simulation can be adjusted to include unloading of the oil at a servicing vessel. Using the same storage capacity and transit speed and assuming the distance to the ship and back is two times 300m and that each Cytobot fills the same amount simultaneously, then the new times are 17.2, 12.7, 10.3 and 6.5 for 3, 5, 8 and 15 Cytobots respectively. The Kakalis and Ventikos simulation has several differences to the one reported here, particularly in the distribution of the oil. Also, some key parameters are missing from their paper, for example, the distance to the boat. Despite these differences, our results are very similar. For example, the reported simulation time for 15 EU-MOPS is 10 hours and in our simulation 5 and 8 Cytobots took 12.7 and 10.3 hours respectively. Given the differences in the simulation and differences in operation of the robots, the resulting clean-up times are

comparable, suggesting that the Cytobots have potential application as distributed robotic agents in real-world environments. This application demands an internal control system which can function without reference to other agents within the environment which are operating in parallel. By modifying the environment, (which in this case was consumption of food), the agents can stigmergically communicate and display emergent behaviour.

The Cytobots offer a unique range of abilities. Like cells, their internal network of spatially distributed dynamic chemical species allows them to autonomously coordinate and direct their movement, recognise and respond to patterns in the environment, and produce high-level behaviour.

9. Summary & Conclusions

9.1 Introduction

The purpose of this final chapter is to provide an overall summary of the project. The first section revisits the original objectives in order to evaluate how well these were met within the course of the research. This is followed by a summary of original contributions to the art, suggestions for future work and final remarks.

9.2 The Project Objectives Revisited

9.2.1 Evaluation of Objective Attainment

The objectives set out at the beginning of the project developed significantly in the first year. Objective five was added later after the literature review highlighted the importance of interaction between multiple cells in the generation of Cell Intelligence. At the start of the project the objectives were merely ideas and, although preliminary research had identified a niche for development of CSN inspired AI, little was known about how they would be met.

The project objectives (as given in *chapter 1*) are listed below together with a discussion of their attainment.

1. *Produce a computational representation of biological Cell Signalling Networks that captures the underlying properties and mechanisms of Cell Intelligence.*

   The completion of this first objective involved a number of tasks. The first of these was to identify the key features of Cell Intelligence and its mechanisms. This started with a literature review on the mechanisms and capabilities of information processing within cells, leading in turn to a complete overview of CSNs (*chapter 2*). The second task involved investigating the current methods used to represent the properties and mechanisms of CSNs and the applications of such models in both Systems Biology (*chapter 2*) and AI (*chapter*

3). This served two purposes: firstly, to learn from existing AI and biological simulation methods, and secondly to ensure the originality of the new representation. At this point, a new representation was developed and this resulted in the ARN, as described in *chapter 4*. The ARN combines features from existing methods which facilitate modelling of the identified properties and mechanisms of Cell Intelligence. The ability of the ARN to capture high-level behaviour of both single and multiple cells was later verified throughout the project *(chapters 5, 6, 7, 8*). This work clearly demonstrates the attainment of this objective.

2. *Evaluate the biological accuracy of the new representation*

   To evaluate the biological accuracy of the ARN representation, a number of experiments were performed. The ability of the ARN to represent the dynamics of individual biochemical reactions was tested and the results were shown to compare to those of standard modelling tools (*chapter 4*). Later, by simulating the *E. coli* chemotaxis CSN (*chapter 5*), it was demonstrated that the ARN could be used to model an entire signalling pathway. The accuracy of the ARN, in simulating both the dynamics of individual biochemical reactions and whole CSNs, compared to that of other standard Systems Biology tools. These results thus demonstrate the attainment of this objective.

3. *Explore the computational capabilities, pattern recognition and temporal dynamics of the new representation.*

   The computational capabilities of the ARN were firstly evaluated by constructing small networks which could perform operations such as: Boolean and Fuzzy Logic, switching and oscillation. For each of these networks, examples from the literature were used as a basis of their structural design and also to demonstrate their occurrence within biological CSNs (*chapter 2*). In *chapter 4* the ARN was demonstrated to have at least the same processing capability as an MLP. Its pattern recognition functionality was explored in *chapter 6*, where a fully-connected, layered ARN was successfully trained using an EA to recognise simple patterns. Using the gaits of qudrupedal robots as a test bed, the ARN was then shown to be capable of generating a range of complex temporal signals (*chapter 6*). These experiments also showed that ARN could automatically transition between these signals in response to environmental stimuli. Thus, the demonstration of the ARNs capacity for pattern recognition and the generation of complex temporal waveforms illustrate that this objective was clearly met.

4. *Evaluate the ability of the new representation to capture characteristics of Cell Intelligence arising from single cells.*

   Having evaluated the ARN's ability to accurately model and capture the computational properties and mechanisms of CSNs, its ability to represent the characteristics of Cell Intelligence demonstrated by a unicellular organism was examined (*chapter 5*). Here, an ARN-based simulation of the *E. coli* chemotaxis CSN demonstrated the temporal transition of concentrations of key chemotaxis proteins in response to different levels of environmental chemorepellant. Ultimately, the resulting pattern of chemical concentrations indicated chemotaxis down the chemorepellant gradient, and thus demonstrated the high-level behaviour of movement toward more favourable conditions. This work illustrates the achievement of this objective.

5. *Evaluate the ability of the new representation to capture Cell Intelligence arising from multiple interacting cells.*

   It was shown that the ARN could capture complex emergent behaviour arising from groups of multiple interacting cells using ARN-agents termed Cytobots (*chapter 7*). Cytobots were shown to capture emergent behaviour of the slime mould *D. discoideum* (*chapter 8*). In this simulation, each Cytobot represented a *D. discoideum* cell, and it was demonstrated that these responded autonomously to the surrounding environment and to other Cytobots using stigmergic communication. Thus the work in chapters 7 and 8 demonstrate the achievement of this objective.

6. *Identify the applications of the new representation within AI and compare with existing methods*

Many applications of the ARN were demonstrated throughout the project, and thus this objective was met. These were:

1) Simulation of an individual CSN (*chapter 5*)
2) Simulation of multiple interacting CSNs (*chapter 8*)
3) Simulation of emergent behaviour in both individual and multiple interacting cells (*chapter 5 and 8*)
4) Limbed robotic control and pattern recognition (*chapter 6*)
5) Robotic control in general (for example, in the oil-spill experiment described in *chapter 8*)

Existing methods were contrasted with the ARN throughout the project, particularly within the results section for each experiment.

9.3 Original Contributions to the Art

The original contributions, as given in *chapter 1* are listed below. Each is followed by the location of the original work within this thesis and a justification of its originality.

1. *The creation of a new connectionist AI technique - the "Artificial Reaction Network" inspired by biological Cell Signalling Networks (chapter 4).*

   As discussed in *chapter 3 and 4*, there are techniques similar to the ARN. However, the ARN has a number of unique characteristics that distinguish its operation. These are:

   - A distinct graphical notation associated with its mathematical definition
   - The use of inhibitory connections, which combine discrete switches with continuous reaction dynamics
   - The specific use of Euler's approximation to generate time-domain response
   - The use of "pools"
   - The isolation of component parts within a connectionist representation

   Furthermore, the ARN has a number of advantages that were discussed throughout the project. In summary these are:

   - It is highly flexible, individual reaction parameters (units/pools/connections) can easily be modified independently of other reactions.
   - Distinct graphical system and associated mathematical definition facilitate simple building and conceptualisation of pathways. This in turn facilitates the development of drag-and-drop user interfaces.
   - Ability to easily represent and visualise structural motifs found in CSNs and study emergent properties of the network.
   - The temporal dynamics of entire networks and (or) individual reactions can be viewed in simulated real-time. The effects of modifications to the network can be viewed as they occur.
   - The ARN can both process input patterns, and generate associated output patterns.
   - Direct outputs from the ARN can be used to control robotic actuators.

- It enables both intracellular dynamics and extracellular dynamics to be modelled and resulting emergent properties studied in one system.

2. *The production of complex waveforms for control of limbed robotic gaits by combining functional motifs found in CSNs within a rate law based connectionist system (chapter 6).*

   Although differential equation reaction models have been used to generate motion of soft-bodied robots (as discussed in *chapter 3*), none that use functional motifs found in real biochemical networks or that employ rate law equations have been used to generate the gaits of limbed robots.

3. *The construction of the E. coli chemotaxis pathway using a connectionist based Artificial Chemistry (chapter 5).*

   To the best of my knowledge, this is the first time that a connectionist-based Artificial Chemistry was used to simulate the chemotaxis pathway of *E. coli* and model its high-level behaviour.

4. *The implementation of chaotic dynamics by combining functional motifs found in CSNs within a rate law based connectionist system (chapter 7).*

   Chaotic dynamics have been shown in real biochemical reactions, for example the Belousov-Zhabotinsky reaction (*Winfree 1984*) and computational models of such dynamics have been created. However, as far as the author is aware, this is the first time such dynamics have been generated by combining functional motifs within a connectionist-based Artificial Chemistry.

5. *The production of a complete limbed robotic control system by combining functional motifs found in CSNs within a rate law based connectionist system (chapter 6).*

   As far as the author is aware , this is the first time that a complete limbed robotic control system has been created by combining functional motifs found in CSNs within a rate law based connectionist Artificial Chemistry. It was shown that such a system can combine pattern recognition and response within a single network and automatically generate and transition between different quadrupedal gaits in response to a range of environmental patterns.

6. *The construction of multiple distributed cell-like agents by combining functional motifs found in CSNs within a rate law based connectionist system (chapters 7 and 8).*

   There are other agent-based systems that have complex internal networks as discussed in *chapter 3* (for example GRNs). However, these examples do not use functional motifs found in CSNs nor do they use the rate law equation to model internal dynamics. Furthermore these do not show the ability to perform multiple operations. For example, move forward, turn, adjust length of motion, recognise environmental patterns, and change phase (as was demonstrated by the Cytobots in the oil-spill clean-up simulation in *chapter 8*); or parallel the collective high-level behaviour found in groups of interacting unicellular organisms.

9.4 Suggestions for Future Work

Overall considerations for future work can be divided broadly into two categories: 1) improvements to the model and 2) further applications. An overview of those considered the most important are presented below.

9.4.1 Improvements to Model

- Currently programming a large ARN network is complex. Increasing the number of reaction units increases the number of connections and pools, thus increasing the size of the program. Managing such a system is cumbersome. For practical use, it would be highly convenient to develop a drag-and-drop graphical user interface allowing connections, pools and units to be simply arranged, connected and modified as desired. This would make the ARN accessible to wider research communities, for example, as a modelling tool in the field of Systems Biology.

- The accuracy of the model could also be improved by, for example, examining the use of second or higher-order methods of approximating the differential equations. The current method of approximation – Euler's method, was deemed suitable for the purposes of this project. However, providing a higher degree of accuracy may be useful in particular modelling scenarios.

- An important aspect of future work, which would make the ARN more versatile and accessible particularly within the AI community, is the ability to:

1) Evolve entire ARNs autonomously.

2) To facilitate autonomous adaptation and learning within the ARN representation.

- The first point refers to the ability to automatically evolve not only the network parameters, as was demonstrated in the pattern recognition experiments (*chapter 5*), but also to evolve the topology of the network. The second point refers to the ability of the network to continuously and autonomously adapt its topology and network parameters while in operation – thus allowing it to maintain or improve its current functioning.

9.4.2 Applications

- An interesting property of Cell Intelligence is the ability to store and recall information. As discussed in *chapter 2*, such memory is of interest because it illustrates that intelligence is present in organisms without a nervous system. Such knowledge sheds light on the nature of intelligence in a more general sense and on the limitations of cellular behaviour. However, the mechanisms involved are poorly understood. The ARN is a technique suited to investigating such mechanisms and properties of CSNs which could facilitate this primitive memory. Such insights would be important within both biological science and AI.

- As was demonstrated in the *D. discoideum* simulation (*chapter 8*) Cytobots provide a powerful tool in the simulation of interacting cells. They allow CSN dynamics at multiple levels (intracellular and extracellular) to be modelled in parallel. This could be particularly useful in examining the emergent properties of interacting cells. For example, the effect of mutant or diseased cells on a colony can be studied by introducing ARN-agents with altered pathways to a population of ARN-agents with wild-type pathways. Another potential field of application is within environmental science. For example, models of micro-ecologies containing heterogeneous species can be created and used to predict potential effects on the environment or on each other.

9.5 Concluding Remarks

In conclusion, the author would like to make the following personal remarks with regard to the undertaking of the work. The project has been a great success and all the original objectives have been fulfilled. The work has resulted in six conference papers (most in leading international conferences) and two journal publications both in Soft Computing (Springer). There have been many challenges over the course of the project and these have been

successfully overcome. The project has been highly interesting, extremely worthwhile and provides an extensive report on the computational aspects of CSNs.

References

Adamatzky, A., de Lacy Costello, B. and Asai, T. (2005) *Reaction-diffusion computers*, London: Elsevier Science.

Aderem, A. and Underhill, D. (1999) 'Mechanisms of phagocytosis in macrophages', Annual Review Immunology, vol.17, pp. 593–623.

Afelt, Z., Blaszczyk, J. and Dobrzecka, C. (1983) 'Speed control in animal locomotion: transitions between symmetrical and nonsymmetrical gaits in the dog', Acta Neurobiologiae Experimentalis, vol.43, pp. 235-250.

Agerwala, T. (1974) A complete model for representing the coordination of asynchronous processes. In: *Hopkins Computer Research*, Report 32, John Hopkins University.

Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K. and Watson, J. (1994) Cell Signalling. In: *Molecular biology of the cell*, 3rd ed., New York: Garland Science, pp. 721-734.

Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K. and Watson J. (1989) Cell Signaling. In: *Molecular biology of the cell*, 2nd ed., New York: Garland Science, pp. 682-690.

Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K. and Watson, J. (1989b) The Immune System. In: *Molecular biology of the cell*, 2nd ed., New York: Garland Science, pp. 1002-1011.

Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K. and Watson, J. (1989c) Small Molecules, Energy and Biosynthesis. In: *Molecular biology of the cell*, 2nd ed., New York: Garland Science, pp. 41-58.

Alberts, B., Bray, D., Lewis, J., Raff, M., Roberts, K. and Watson, J. (1989d) The Nervous System. In: *Molecular biology of the cell*, 2nd ed., New York: Garland Science, pp. 1059-1065.

Aldridge, B., Burke, J., Lauffenburger, D. and Sorger, P. (2006) 'Physicochemical modeling of cell signaling pathways', Nature Cell Biology, vol. 8, pp. 1195–1203.

Andrew, N. and Insall, R. (2007) 'Chemotaxis in shallow gradients is mediated independently of PtdIns 3-kinase by biased choices between random protrusions', Nature Cell Biology, vol. 9, no. 2, pp. 193-200.

Ankers, J., Spiller, D., White, M. and Harper, C. (2008) 'Spatio-temporal protein dynamics in single living cells', Current Opinion Biotechnology, vol. 19, pp. 1-6.

Arena, P., Fortuna, L. and Branciforte, M. (1999) 'Reaction-diffusion CNN algorithms to generate and control artificial locomotion', Circuits and Systems I: Fundamental Theory and Applications, IEEE Transactions, vol. 46, no. 2, pp. 253-260.

Arkin, A. and Ross, J. (1994) 'Computational functions in biochemical reaction networks', Biophysical Journal, vol. 67, pp. 560-578.

Baldan, P., Cocco, N., Marin, A. and Simeoni, M. (2010) 'Petri Nets for Modeling Metabolic Pathways: A Survey', Natural computing, vol. 9, no. 4, pp. 955-989.

Banâtre, J., Coutant, A. and Le Métayer, D. (1988) 'A parallel machine for multiset transformation and its programming style', Future Generation Computer Systems, vol. 4, pp. 133–144.

Banzhaf, W. (1990). 'The "molecular" traveling salesman', Biological Cybernetics, vol. 64, no. 1, pp. 7-14.

Banzhaf, W. (2004) 'On evolutionary design, embodiment, and artificial regulatory networks', *International Seminar Embodied Artificial Intelligence*, Lecture Notes Computer Science, vol. 3139, Dagstuhl Castle, pp. 284-292.

Banzhaf, W., Dittrich, P., and Rauhe, H. (1996) 'Emergent computation by catalytic reactions', Nanotechnology, vol. 7, no. 4, pp. 307.

Becker, M. (2010) 'Simulation model for the whole life cycle of slime mould *Dictyostelium Discoideum*', *Proceedings of the 24th European Conference on Modeling and Simulation (ECMS)*, Kuala Lumpur, pp. 247-253.

Winfree, A. (1984) 'The Prehistory of the Belousov-Zhabotinsky Oscillator', Journal of Chemical Education, vol. 61, pp. 661-663.

Ben-Jacob, E. (1998) 'Bacterial wisdom, Gödel's theorem and creative genomic webs', Physica, vol. 248, pp. 57-76.

Ben-Jacob, E., Becker, I., Shapira, Y. and Levine, H. (2004) 'Bacterial linguistic communication and social intelligence', Trends in Microbiology, vol. 12, no. 8, pp. 366-371.

Berry, G. and Boudol, G. (1992) 'The chemical abstract machine', Theoretical computer science, vol. 96, no. 1, pp. 217-248.

Bhalla, U. (2003) 'Understanding complex signalling networks through models and metaphors', Progress in Biophysics and Molecular Biology, vol. 81, no. 1, pp. 45-46.

Billard, A. and Ijspeert, A. (2000) 'Biologically inspired neural controllers for motor control in quadruped robot', *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN)*, Como, pp 637-641.

Block, H., Knight Jr, B. and Rosenblatt, F. (1962) 'Analysis of a four-layer series-coupled perceptron', Reviews of Modern Physics, vol. 34, no. 1, pp. 135.

Bray, D. (1995) 'Protein molecules as computational elements in living cells', Nature, vol. 376, no. 6538, pp. 307-12.

Bray, D. (2011) Biochemical data (1998-2010) for Bacterial chemotaxis in silico, [Online], Available: http://www.pdn.cam.ac.uk/groups/comp-cell/Data.html [Last Accessed 10 May 2011].

Bray, D. (1990) 'Intracellular Signaling as a Parallel Distributed Process', Journal of theoretical biology, vol. 143, pp. 215-231.

Bray, D., Bourret, R. and Simon, M. (1993) 'Computer simulation of the phosphorylation cascade controlling bacterial chemotaxis', Molecular Biology of the Cell, vol. 4, no. 5, pp. 469.

Bray, D., Levin, M. and Lipkow, K. (2007) 'The chemotactic behavior of computer-based surrogate bacteria', Current Biology, vol. 17, no. 1, pp. 12-19.

Brinkschulte, U., Pacher, M. and von Renteln, A. (2007) 'Towards an artificial hormone system for self-organizing real-time task allocation', *International Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS)*, Lecture Notes in Computer Science, vol. 4761, Springer, Santorini Island pp. 339-347.

Brock, D., Douglas, T., Queller, D. and Strassmann, J. (2011) 'Primitive agriculture in a social amoeba', Nature, vol. 469, no. 7330, pp. 393-396.

Brooks, R. (1999) Cambrian intelligence. In: *The Early History of the New AI*. Cambridge Massachusetts: MIT Press:

Burnet, F. (1959) *The clonal selection theory of acquired immunity*, Nashville Tennessee: Vanderbilt University Press.

Calder, M., Vyshemirsky, V., Gilbert, D. and Orton, R. (2006) 'Analysis of Signaling Pathways Using Continuous Time Markov Chains', Transactions on Computational Systems Biology VI, Springer, vol. 4220, pp. 44-67.

Chen, H., Zhu, Y. and Hu, K. (2009) 'Cooperative bacterial foraging optimization', Discrete dynamics in nature and society, vol. 2, no. 1, pp. 501-517.

Clark, A. (1997) *Being there: putting brain, body, and world together again*, Cambridge, Massachusetts: MIT Press.

Collins, J. and Richmond, S. (1994) 'Hard-wired central pattern generators for quadrupedal robots', Biological Cybernetics, vol. 71, pp. 375-385

Cotter, D., Sands, T., Virdy, K., North, M., Klein, G. and Satre, M. (1992) 'Patterning of development in *Dictyostelium discoideum*: factors regulating growth, differentiation, spore dormancy and germination', Biochemistry and Cell Biology, vol. 70, no. 10-11, pp. 892-919.

Crespi, B. (2001) 'The evolution of social behavior in microorganisms', Trends in Ecology and Evolution, vol. 16, no. 4, pp. 178-183.

Dagg, A. (1973) 'Gaits in mammals', Mammal Review, vol. 3, pp. 135-154

Dallon, J. and Othmer, H. (1997) ‘A discrete cell model with adaptive signaling for aggregation of *Dictyostelium discoideum*’, Philosophical Transactions of the Royal Society B: Biological Sciences, vol. 352, no. 1351, pp. 391–417.

Davidich, M. and Bornholdt, S. (2008) ‘Boolean network model predicts cell cycle sequence of fission yeast’, Plos One, vol. 3, no. 2, pp. 1672.

de Castro, L. and Timmis, J. (2002) Artificial Immune Systems: A Novel Approach to Pattern Recognition. In: Corchado, J. and Alonso, L. and Fyfe, C. (eds.), *Artificial Neural Networks in Pattern Recognition*, University of Paisley, pp. 67-84.

de Silva, A. and Uchiyama, S. (2007) ‘Molecular logic and computing’, Nature Nanotechnology, vol. 2, no. 7, pp. 399-410.

Decraene, J., Mitchell, G. and McMullin, B. (2007), Evolving artificial cell signaling networks: perspectives and methods. In: Dressler, F. and Carerras, I. (eds.), *Advances in Biologically Inspired Information Systems*, Studies in Computational Intelligence, vol. 69, Springer, pp. 164-184.

Dennett, D. (1996) *Kinds of Minds. Toward an Understanding of Consciousness*, New York: Basic Books.

Descartes, R. Discourse of the Method, Translated by Veitch, J. (1975) *Meditations on the First Philosophy, Principles of Phislosophy*, London: Dent.

Devreotes, P. (1989) ‘*Dictyostelium discoideum*: a model system for cell-cell interactions in development’, Science, vol. 245, no. 4922, pp. 1054-1058.

Dittrich, P. (2005) Chemical computing. In: *Unconventional programming paradigms*, Berlin Heidelberg: Springer, pp. 19-32.

Dittrich, P., Banzhaf, W., Rauhe, H., and Ziegler, J. (1997) ‘Macroscopic and microscopic computation in an artificial chemistry’, *Proceedings of the Second German Workshop on Artificial Life (GWAL)*, pp. 19-22.

Dittrich, P., Zeigler, J. and Banzhaf, W. (2001) ‘Artificial chemistries- a review’, Artificial Life, vol. 7, no. 3, pp. 225-275.

Dobbyn, C., Mustafa, A., Hirst, T., Richards, M., Smith, N. and Wong, P. (2007) Natural and Artificial Intelligence, *In: Artificial Life II (Block 1 Module M366)*, The Open University, pp.159-210.

Eikelder, T., Crigins, S., Steijaert, M., Liekens, A. and Hilbers, P. (2009) 'Computing with feedforward networks of artificial biochemical neurons', *The 4th International Workshop on Natural Computing, Proceedings in Information and Communications Technology (IWNC)*, vol. 1, Himeji, pp. 38-47.

ESIGNET project. (2011) Evolving cell signaling networks *in silico*, [Online], Available: http://www.esignet.net/index.php?menuid=2 [Last Accessed 12 February 2011]

Eungdamrong, N. and Iyengar, R. (2004) 'Modeling cell signaling networks', Biology of the Cell, vol. 96, pp. 355–362.

Farmer, J., Packard, N. and Perelson, A. (1986) 'The immune system, adaptation, and machine learning', Physica D: Non-linear Phenomena, vol. 2, no.1-3, pp. 187-204.

Ferrell, J. (2004) 'Self perpetuating states in signal transduction: positive feedback, double negative feedback and bistability', Current Opinion in Cell Biology, vol. 14, no. 2, pp. 142-148.

Fontana, W. (1991) Algorithmic chemistry. In: Langton, C., Taylor, C., Farmer, J. and Rasmussen, S. (eds.), *Artificial Life II*, Addison-Wesley, pp. 159–209.

Ford, B. (2009) 'On intelligence in cells: The case for whole cell biology', Interdisciplinary Science Reviews, vol. 34, no. 4, pp. 350-365.

Ford, B. (2004) 'Are cells Ingenious?', The Microscope, vol. 52, pp. 135-144.

Gerrard, C., McCall, J., Coghill, G. and Macleod, C. (2011) 'Artificial Reaction Networks', *Proceedings of the 11th UK Workshop on Computational Intelligence (UKCI)*, Manchester UK, pp 20-26.

Gerrard, C., McCall, J., Coghill, G. and Macleod, C. (2012a) 'Temporal patterns in Artificial Reaction Networks', *Proceedings of The 22nd International Conference on Artificial Neural Networks (ICANN)*, Lecture Notes in Computer Science, vol. 7552, part 1, Springer, Lausanne, pp 1-8.

Gerrard, C., McCall, J., Coghill, G. and Macleod, C. (2012b) 'Adaptive Dynamic Control of Quadrupedal Robotic gaits with Artificial Reaction Networks', *Proceedings of the 19th International Conference on Neural Information Processing (ICONIP)*, Lecture Notes in Computer Science, vol. 7663, part 1, Springer, Doha, pp 280-287.

Gerrard, C., (2012c) Lynxsmotion Q2 quadrupdal robot walk and trot gaits. [Online], Available:https://drive.google.com/folderview?id=0BxGVfJFH9UmZlJTV1pROFFRb00&usp=sharing

Gerrard, C, McCall, J., Coghill, G., Macleod, C. (2013a) 'Artificial Reaction Network Agents', *The 12th European Conference on the Synthesis and Simulation of Living Systems (ECAL)*, Advances in Artificial Life, MIT press, Taormina, pp. 957-964.

Gerrard, C., McCall, J., Coghill, G., Macleod, C. (2013b) Combining Biochemical Network Motifs within an ARN-Agent Control System', *Proceedings of the 13th Annual Workshop on Computational Intelligence (UKCI)*, IEEE, Surrey, pp. 8-15.

Gerrard, C., McCall, J., Coghill, G., and Macleod, C. (2013c) 'Artificial chemistry Approach to Exploring Search Spaces using Artificial Reaction Network Agents', Congress on Evolutionary Computation (CEC)', IEEE, Cancún, pp.1201-12.

Gerrard, C., McCall, J., Coghill, G. and Macleod, C. (in press) 'Applications and Design of Cooperative Multi-agent ARN based Systems', *Soft Computing*, Springer.

Gerrard, C., McCall, J., Coghill, G. and Macleod, C. (in press) 'Exploring aspects of cell intelligence with artificial reaction networks', A Fusion of Foundations Methodologies and Applications, *Soft Computing*, Springer.

Gershenson, C. (2004) 'Classification of random Boolean networks', *Proceedings of the 8th International Conference on Artificial Life*, Sydney, MIT Press, pp1-8.

Gershenson, C. (2004) 'Introduction to Random Boolean Networks', Workshop and Tutorial *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems*, Boston, pp. 160–173.

Ghosh-Dastidar, S. and Adeli, H. (2009) 'Spiking neural networks', International Journal of Neural Systems, vol. 19, no. 4, pp. 295-308.

Gillespie, D. (1977) ‘Exact stochastic simulation of coupled chemical reactions’, Journal of physical Chemistry, vol. 81, no. 25, pp. 2340-2361.

Goldman, J., Levin, M., and Bray, D. (2009) ‘Signal amplification in a lattice of coupled protein kinases’, Molecular BioSystems, vol. 5, no. 12, pp. 1853-1859.

Greenburg, M. (1998) Quantitative Methods: Numerical Solution of Differential Equations. In: *Advanced Engineering Mathematics*, 2nd edition, India: Pearson Education, pp. 296.

Greensmith, J., Whitbrook, A. and Aickelin, U. (2010) Artificial immune systems. In: Gendreau, M. and Potvin, J. (eds.), International Series in Operations Research and Management Science, *Handbook of Metaheuristics*, 2nd edition, vol. 146, pp. 421-448.

Guo, H., Meng, Y., and Jin, Y. (2009) ‘A cellular mechanism for multi-robot construction via evolutionary multi-objective optimization of a gene regulatory network’, BioSystems, vol. 98, no. 3, pp. 193-203.

Gurney, K. (1992) ‘Training nets of hardware realizable sigma-pi units’, Neural Networks, vol. 5, pp. 289-303.

Hamann, H., Stradner, J., Schmickl, T. and Crailsheim, K. (2010) ‘A hormone-based controller for evolutionary multi modular robotics: From single modules to gait learning’, *Congress on Evolutionary Computation (CEC)*, Barcelona, pp. 1-8.

Harvey, I. and Bossomaier, T. (1997) ‘Time out of joint: Attractors in asynchronous random Boolean networks’, *Proceedings of the Fourth European Conference on the Synthesis and Simulation of Living Systems (ECAL)*, Complex Adaptive Systems, MIT Press, Brighton UK, pp. 67-75.

Helikar, T., Konvalina, J., Heidel, J. and Rogers, J. (2008) ‘Emergent decision making in biological signal transduction’, Proceedings of the National Acadamy of Sciences of the United States of America, vol. 104, pp. 1913–1918.

Hellingwerf, K. (2005) ‘Bacterial observations: a rudimentary form of intelligence’, Trends in Microbiology, vol. 13, pp. 152-158.

Hild, W., Pollinger, K., Caporale, A., Cabrele, C., Keller, M., Pluym, N., Buschauer, A., Rachel, R., Tessmar, J., Breunig, M., et al. (2010) ‘G protein-coupled receptors function

as logic gates for nanoparticle binding and cell uptake', Proceedings of the National Acadamy of Sciences of the United States of America, vol. 107, pp. 10667-10672.

Hildebrand, M. (1997) 'Analysis of asymmetrical gaits', Journal of Mammalogy, vol. 58, pp. 131-156.

Hjelmfelt, A. and Ross, J. (1994) 'Pattern recognition, chaos, and multiplicity in neural networks of excitable systems', Proceedings of the National Acadamy of Sciences of the United States of America, vol. 91, pp. 63-67.

Hjelmfelt, A., Weinberger, E. and Ross, J. (1991) 'Chemical implementation of neural networks and Turing machines', Proceedings of the National Academy of Sciences, vol. 88, no. 24, pp. 10983-10987.

Hofestadt, R. and Thelen, S. (1998) 'Quantitative modeling of biochemical networks', In Silico Biology, vol. 1, no. 1, pp. 39-53.

Hong, Q. and Reluga, T. (2005) 'Nonequilibrium Thermodynamics and Nonlinear Kinetics in a Cellular Signaling Switch', Physical Review Letters, vol. 94, no. 2.

Hopfied, J. (1984) 'Neural networks with graded response have collective computational properties like those of two-stage neurons', Proceedings of the National Academy of Sciences of the United States of America, vol. 81, pp. 3088-3092.

Husbands, P. (1998) 'Evolving robot behaviours with diffusing gas networks', *First European Workshop Evolutionary Robotics*, Evolutionary Robotics, Lecture Notes in Computer Science, vol. 1468, Springer, Paris, pp. 71-86.

Husbands, P., Smith, T., Jakobi, N. and O'Shea, M. (1998) 'Better living through chemistry: Evolving GasNets for robot control', Connection Science, vol. 10, no. 3-4, pp. 185-210.

Jerne, N. (1974) 'Towards a network theory of the immune system', Annales de l'Institut Pasteur Immunology,vol. 25, pp. 373–389.

Joachimczak, M., Kowaliw, T., Doursat, R. and Wrobel, B. (2013) 'Controlling development and chemotaxis of soft-bodied multicellular animats with the same gene regulatory network', *European Conference on the Synthesis and Simulation of Living Systems (ECAL)*, Advances in Artificial Life, MIT press, vol. 12, Taormina, pp. 454-461.

Jones, B. (2012) 'Cell signalling: Mediating hormonal crosstalk', Nature Reviews Molecular Cell Biology, vol. 13, no. 11, pp. 685-685.

Kakalis, N. and Ventikos, Y. (2008) 'Robotic swarm concept for efficient oil spill confrontation', Journal of Hazardous Materials, vol. 154, no. 1-3, pp. 880-7.

Kanada, Y., and Hirokawa, M. (1994) 'Stochastic problem solving by local computation based on self-organization paradigm', *Proceedings of the Twenty-Seventh International Conference on System Sciences*, vol. 3, IEEE, Hawaii, pp. 82-91.

Kauffman, S. (1969) 'Metabolic stability and epigenesis in randomly constructed genetic nets', Journal of Theoretical Biology, vol. 22, no. 3, pp. 437-467.

Kauffman, S. (1993) *The origins of order: Self organization and selection in evolution*, New York: Oxford University press.

Kessin, R. (2003) 'Making Streams', Nature, vol. 422, pp. 481-482.

Kholodenko, B. (2006) 'Cell signaling dynamics in time and space', Nature review molecular cell biology, vol. 27, no. 3, pp. 165-176.

Kimura, H., Fukuoka, Y. and Konaga, K. (2001) 'Adaptive dynamic walking of a quadruped robot by using neural system model', Advanced Robotics, vol. 15, no. 8, pp. 859-876.

Kohonen, T. (1982) 'Self-organized formation of topologically correct feature maps', Biological Cybernetics, vol. 43, no. 1, pp. 59-69.

Kondo, T., Ishiguro, A., Watanabe, Y., Shirai, Y. and Uchikawa, Y. (1998) 'Evolutionary construction of an immune network-based behavior arbitration mechanism for autonomous mobile robots', Electrical Engineering in Japan, vol. 123, no. 3, pp. 1-10.

Kowalewski, S. (2002) 'Introduction to the Analysis and Verification of Hybrid Systems.Modeling', In: Engell, S., Frehse, G. and Scheieder, E. (eds.), *Analysis and Design of Hybrid Systems*, Lecture Notes in Control and Information Science, Springer, vol. 279, pp. 153-171.

Krautmacher, M. and Dilger, W. (2004) 'AIS based robot navigation in a rescue scenario', *Proceedings of the 3rd International Conference in Artificial Immune Systems (ICARIS)*,

Artificial Immune Systems, Lecture Notes Computer Science, vol. 3239, Catania, pp. 106-118.

Kreyssig, P. and Dittrich, P. (2011) 'Reaction flow artificial chemistries', *11th European Conference on the Synthesis and Simulation of Living Systems (ECAL), Advances in Artifical Life*, MIT press, Paris, pp. 431-437.

Lee, W. and Yang, K. (2008) 'Applying Intelligent Computing Techniques to Modeling Biological Networks from Expression Data', Genomics, Proteomics & Bioinformatics, vol. 6, no. 2, pp. 111-120.

Liu, C., Chen, Y., Zhang, J. and Chen, Q. (2009) 'CPG driven locomotion control of quadruped robot', IEEE International conference on Systems Man and Cybernetics, San Antonio, pp 2368-2373.

Luh, G. and Liu, W. (2004) 'Reactive immune network based mobile robot navigation', *Proceedings of the 3rd International Conference on Artificial Immune Systems (ICARIS)*, Lecture Notes in Computer Science, vol. 3239, Catania, pp. 119-132.

Lyon, P. (2006) 'The biogenic approach to cognition', Cognitive processes, vol. 7, pp. 11–29.

Maass, W. (1997) 'Networks of spiking neurons: the third generation of neural network models', Neural networks, vol. 10, no. 9, pp. 1659-1671.

Macey, R., Oster, G., and Zahnley, T. (2000) Berkeley Madonna version 5 and User's Guide 2000. [Online], Available: http://www.berkeleymadonna.com [Last Accessed 1 May 2011].

MacLeod, C. and Capanni, N. (2010) 'Artificial biochemical networks: a different connectionist paradigm', Artificial Intelligence Review, vol. 33, no. 1-2, pp. 124-135.

Manahan, C., Iglesias, P., Long, Y. and Devreotes, P. (2004) 'Chemoattractant signaling in *Dictyostelium discoideum*', Annual Review of Cell and Developmental Biology, vol. 20, pp. 223–253.

McCann, C., Kriebel, P., Parent, C. and Losert, W. (2010) 'Cell speed, persistence and information transmission during signal relay and collective migration', Journal of Cell Science, vol. 123, pp. 1724-1731.

McCarthy, J. (1960) 'Recursive functions of symbolic expressions and their computation by machine', Communications of the ACM, vol. 3, no. 4, pp. 184-195.

McCulloch, W. and Pitts, W. (1943) 'A logical calculus of the ideas immanent in nervous activity', The Bulletin of Mathematical Biophysics, vol. 5, no. 4, pp. 115-133.

McMullin, B. (2004) 'Thirty years of computational autopoiesis: A review', Artificial Life, vol. 10, no. 3, pp. 277-295.

Merlin, P. and Farber, D. (1976) 'Recoverability of communication protocols- implications of a theoretical study', IEEE Transactions on Communications, vol. 24, no. 9, pp. 1036–1043.

Mesarović, M. (1968) *Systems theory and biology- view of a theoretician*, Berlin Heidelberg: Springer.

Moioli, R., Vargas, P., Von Zuben, F. and Husbands, P. (2008) 'Towards the evolution of an artificial homeostatic system', IEEE *World Congress on Evolutionary Computation (CEC)*, Hong-Kong, pp. 4023-4030.

Monismith, D. and Mayfield, B. (2008) 'Slime mold as a model for numerical optimization', *IEEE Swarm Intelligence Symposium (SIS)*, St Louis USA, pp. 21-23.

Morris, J. (1974) *A Biologist's Physical Chemistry*, Barrington, E. and Willis, A. (eds.), Great Britain: Edward Arnold.

Morton-Firth, C. and Bourret, R. (2011) Experimental data in bacterial chemotaxis, [Online], Available: www.pdn.cam.ac.uk/groups/comp-cell/Exp_data/exptdata.doc [Last Accessed 10 May 2011].

Morton-Firth, C., Shimizu, T. and Bray, D. (1999) 'A free-energy-based stochastic simulation of the tar receptor complex', Journal of Molecular Biology, vol. 286, no. 4, pp. 1059-1074.

Murata, T. (1989) 'Petri Nets: Properties analysis and applications' Proceedings of the IEEE, vol. 77, no. 4, pp. 541-580.

Nakada, K., Asai, T. and Amemiya, Y. (2004) 'Design of an artificial central pattern generator with feedback controller', Intelligent Automation and Soft Computing, vol. 10, no. 2, pp.185-192.

Nakagaki, T., Yamada, H. and Toth, A. (2000) 'Maze-solving by an amoeboid organism', Nature, vol. 407, no. 6803, pp. 470-470.

Opp, W. and Sahin, F. (2004) 'An Artificial Immune System Approach to mobile sensor networks and mine detection', *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, vol. 1, pp. 947-952.

Passino, K. (2002) 'Biomimicry of bacterial foraging for distributed optimization and control', IEEE Control Systems, vol. 22, no. 3, pp.52–67.

Patidar, V., Sud, K. and Pareek, N. (2009) 'A pseudo random generator based on chaotic logistic map and its statistical testing', Informatica, vol. 33, pp. 441-452.

Phatak, S., and Rao, S. (1995) 'Logistic map: A possible random-number generator', Physical review E, vol. 51, no. 4, pp. 3670.

Quevli, Nels. (1916) *Cell intelligence, the cause of evolution*. Minneapolis: Colwell Press.

Rifkin, J. and Goldberg, R. (2006) 'Effects of chemoattractant pteridines upon speed of *D. discoideum* vegetative amoeba', Cell Motility and the Cytoskeleton, vol. 63, no. 1, pp. 1-5.

Rajasekaran, S. and Vijayslakshmi, P. (2011) *Neural networks, Fuzzy logic and Genetic algorithms*. PHI Learning Private Limited.

Rosenblatt, F. (1958) 'The perceptron', Psychological Review, vol. 65, no. 6, pp. 386-408.

Rosenblatt, F. (1957) *The Perceptron- a percieving and recognizing automaton*. Report 85-460-1, Cornell Aeronautical Laboratory.

Rumelhart, D., and McClelland, J. and the PDP research group. (1986) *Parallel distributed processing: Explorations in the microstructure of cognition*, vol. 1, Cambridge Massachusetts: MIT Press.

Saigusa, T., Tero, A., Nakagaki, T. and Kuramoto, Y. (2008) 'Amoebae Anticipate Periodic Events', Physical Review,vol. 100, no. 1, pp. 1-4.

Sathyanath, S. and Sahin, A. (2002) 'AISIMAM- An AIS based intelligent multi agent model and its application to a mine detection problem', *Proceedings of the IEEE International Conference on Systems, Man*, and Cybernetics, vol. 3, Tunisia.

Savageau, M. and Voit, E. (1987) 'Recasting Nonlinear Differential Equations as S-Systems: A Canonical Nonlinear Form', Mathematical Biosciences, vol. 87, pp. 83–115.

Savageau, M. (1988) 'Introduction to S-systems and the underlying power-law formalism', Mathematical and Computer Modeling, vol. 11, pp. 546-551.

Schmickl, T., Hamann, H. and Crailsheim, K. (2011) 'Modelling a hormone-inspired controller for individual-and multi modular robotic systems', Mathematical and Computer Modelling of Dynamical Systems, vol. 17, no. 3, pp 221-242.

Seger, R. and Krebs, E. (1995) 'The MAPK signaling cascade', The FASEB Journal, vol. 9, no. 9, pp. 726-735.

Shen, W., Will, P., Galstyan, A. and Chuong, C. (2004) 'Hormone-inspired self-organization and distributed control of robotic swarms', Autonomous Robots, vol. 17, no. 1, pp. 93-105.

Stadtman, E. and Chock, P. (1997) 'Superiority of interconvertible enzyme cascades in metabolic regulation: analysis of multicyclic systems', Proceedings of the National Academy of Sciences of the United States of America, vol. 74, pp. 2766–2770.

Stock, J. and Surrete, M. (1996) 'Chemotaxis', Escherichia coli and Salmonella typhimurium: Cellular and Molecular Biology', Ingraham, J., (ed.). Washington DC, American society for microbiology, pp. 1103-1129.

Stoll, G., Viara, E., Barillot, E. and Calzone, L. (2012) 'Continuous time boolean modeling for biological signaling: application of Gillespie algorithm', BMC Systems Biology, vol. 6, no. 1, pp. 116.

Stork, P. (2002) 'Crosstalk between cAMP and MAP kinase signaling in the regulation of cell proliferation', Trends in cell biology, vol.12, no. 6, pp. 258-266.

ten Eikelder, T., Crigins, S., Steijaert, M., Liekens, A. and Hilbers, P. (2009). 'Computing with feedforward networks of artificial biochemical neurons', *Proceedings of the 2nd International Workshop on Natural Computation*, vol 1, Nagoya, pp 38-47.

Toth, D. and Parker, G. (2003) 'Evolving Gaits for the Lynx motion Hexapod II Robot', *Proceedings of the 7th World Multiconference on Systems, Cybernetics, and Informatics*, Orlando USA, pp 229-234.

Turing, A. (1952) 'The chemical basis of morphogenesis', Philosophical Transactions of the Royal Society of London B, vol. 327, pp. 37–72.

Tyler, M. (2000) *Developmental Biology: A guide for experimental study*, 2nd ed, Sunderland Massachusetts: Sinauer, p. 31-34.

Tyson, J and Novák, B. (2010) 'Functional motifs in biochemical reaction networks', Annual Review of Physical Chemistry, vol. 61, pp. 219-240.

Ulam, S. and von Neumann, J. (1947) 'On combinations of stochastic and deterministic processes', Bulletin of the American Mathematical Society, vol. 53, pp. 1120.

Uri, A., Mattheij, R., and Russell, R. (1995) *Numerical solution of boundary value problems for ordinary differential equations*, vol. 13, Society for Industrial and Applied Mathematics.

Vargas, P., de Castro, L. and Michelan, R. (2003) 'An immune learning classifier network for autonomous navigation', *Proceedings of the 2nd International Conference on Artifiical Immune Systems (ICARUS)*, Lecture Notes Computer Science, vol. 2787, Edinburgh, pp. 69-80.

Vladimirov, N. and Sourjik, V. (2009) 'Chemotaxis: how bacteria use memory', The Journal of Biological Chemistry, vol. 390, no. 11, pp. 1097-1104.

Voit, E. (2000) *Computational analysis of biochemical systems: a practical guide for biochemists and molecular biologists*. Cambridge New York: Cambridge University Press, p71-75.

von Neumann, J. (1958) *The computer and the brain*. New Haven Connecticut: Yale University Press.

Wadhams, G. and Armitage, J. (2004) 'Making sense of it all: Bacterial Chemotaxis', Nature Reviews Molecular Cell Biology, vol. 5, pp. 1024-1037.

Wang, B., Kitney, R. Joly, N. and Buck, M. (2011) 'Engineering Modular and Orthogonal genetic logic gates for robust digital-like synthetic biology', Nature Communications, vol. 2, pp. 508.

Wang, J., Huang, B., Xia, X. and Sun, Z. (2006) 'Funneled Landscape Leads to Robustness of Cell Networks: Yeast Cell Cycle', Biophysical journal, vol. 91, no. 5, pp. 54–56.

Wang, Z. and Stout, S. (2007). *Oil spill environmental forensics: fingerprinting and source identification.* Academic Press: Burlington Massachusetts.

Watanabe, Y., Ishiguro, A., Shirai, Y. and Uchikawa, Y. (1998) 'Emergent construction of behavior arbitration mechanism based on the immune system', *Proceedings of the IEEE International Conference on Evolutionary Computation (ICEC)*, Anchorage USA, pp. 481 486.

Werbos, P. (1990) 'Backpropagation through time: what it does and how to do it', Proceedings of the IEEE, vol. 78, no. 10, pp. 1550-1560.

Werbos, P. (1974) Beyond Regression: *New Tools for Prediction and Analysis in the Behavioral Sciences*, PhD thesis, Harvard University.

West, S., Diggle, S., Buckling, A., Gardner, A. and Griffin, A. (2007) 'The social lives of microbes', Annual Review of Ecology, Evolution and Systematics, vol. 38, pp. 53-77.

Whitbrook, A., Aickelin, U. and Garibaldi, J. (2007) 'Idiotypic immune networks in mobile-robot control', IEEE Transactions on Systems, Man, and Cybernetics, Part B, vol. 37, no. 6, pp. 1581-1598.

Wilson, H. and Cowan, J. (1972) 'Excitatory and Inhibitory interactions in localized populations of model neurons', Biophysical Journal, vol. 12, pp. 1-24.

Wolkenhauer, O., Ullah, M., Kolch, W. and Cho, K. (2004) 'Modeling and simulation of intracellular dynamics: Choosing an appropriate framework', IEEE Transaction on Nanobioscience, vol. 3, no. 3, pp. 200-207.

Woo, W., and Khor, L. (2004) 'Blind restoration of nonlinearly mixed signals using multilayer polynomial neural network', Vision, Image and Signal Processing, IEEE Proceedings, vol. 151, no. 1, pp. 51–61.

Woolf, P., Prudhomme, W., Daheron, L., Daley, G. and Lauffenburger, D. (2005) 'Bayesian analysis of signaling networks governing embryonic stem cell fate decisions', Bioinformatics, vol. 21, pp. 741-753.

Wright, M. (2005). *Introduction to Chemical Kinetics*, Chichester UK: Wiley.

Yuan, Y., Zhang,W., Cheng, H. and Liu, J. (2006) 'Crosstalk between calcium and reactive oxygen species signaling', Acta Pharmacologica Sinica, vol. 27, no. 7, pp. 821-826.

Zauner, K., and Conrad, M. (2001) 'Molecular approach to informal computing', Soft Computing, vol. 5, no. 1, pp. 39-44.

Ziegler, J. and Banzhaf, W. (2000) 'Evolving a "nose" for a robot', *Workshop Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, Portland USA.

Ziegler, J., Dittrich, P. and Banzhaf, W. (1998) Towards a metabolic robot control system. In: Holcombe, M. and Paton, R. (eds.), *Information processing in cells and tissues*, New York: US Springer, pp. 305-317.

Appendix 1

Papers produced during the course of the project

The eight papers produced during the course of the project are contained within this appendix.

Paper 1: ‘Applications and Design of Cooperative Multi-agent ARN based Systems’

Reference:

Gerrard, C., McCall, J., Coghill, G. and Macleod, C. (in press) ‘Applications and Design of Cooperative Multi-agent ARN based Systems’, A Fusion of Foundations Methodologies and Applications, *Soft Computing*, Springer.

Applications and Design of Cooperative Multi-agent ARN based Systems

Claire E. Gerrard • John McCall • Christopher Macleod • George M. Coghill

Claire. E. Gerrard • John McCall
IDEAS Research Institute, Robert Gordon University, Aberdeen, AB10 7GJ, UK
email: c.e.gerrard@rgu.ac.uk

John McCall
email: j.mccall@rgu.ac.uk

Geroge M. Coghill
School of Computing Science, University of Aberdeen, Aberdeen, AB24 3FX, UK.
email: g.coghill@abdn.ac.uk

Christopher Macleod
School of Engineering, Robert Gordon University, Aberdeen, AB10 7GJ, UK.
email: chris.macleod@rgu.ac.uk

Keywords Artificial Biochemical Network (ABN), Artificial Chemistry, Artificial Neural Network (ANN), Swarm Robotics

Abstract The Artificial Reaction Network (ARN) is an Artificial Chemistry inspired by Cell Signalling Networks (CSNs). Its purpose is to represent chemical circuitry and to explore the computational properties responsible for generating emergent high-level behaviour. In previous work, the ARN was applied to the simulation of the chemotaxis pathway of *E. coli* and to the control of quadrupedal robotic gaits. In this paper, the design and application of ARN-based cell-like agents termed Cytobots are explored. Such agents provide a facility to explore the dynamics and emergent properties of multicellular systems. The Cytobot ARN is constructed by combining functional motifs found in real biochemical networks. By instantiating this ARN, multiple Cytobots are created, each of which is capable of recognizing environmental patterns, stigmergic communication with others and controlling its own trajectory. Applications in biological simulation and robotics are investigated by first applying the agents to model the life-cycle phases of the cellular slime mould *D. discoideum* and then to simulate an oil-spill clean-up operation. The results demonstrate that an ARN based approach provides a powerful tool for modelling multi-agent biological systems and also has application in swarm robotics.

1 Introduction

In recent years, researchers have become increasingly interested in the complex behaviours displayed by individual cells (Ford 2009; West et al. 2007). For example, the cellular slime mould *D. discoideum* (Dd), starts life as a collection of solitary amoebae which actively hunt bacterial prey. But on starvation these cells secrete a cAMP (cyclic adenosine monophosphate) signal resulting in a complex aggregation response and the formation of a travelling multicellular "slug". Dd also has a symbiotic relationship with its bacterial prey using a primitive form of "farming" to ensure sufficient food availability within a new environment (Brock, 2011).

In order to generate this emergent high-level behaviour, a cell must be able to store and process information. This is accomplished by Cell Signalling Networks (CSNs) which function as the cell's internal processing machinery. They do this by manipulating chemical data within elaborate networked hierarchical control structures which connect chemical species together in productive or inhibitory unions. In this way, cells are able to respond to changes within their environment, communicate with other cells, and perform internal self-maintenance operations (Bray 1995). Several researchers have highlighted the processing capabilities of these networks (Bray 1995; Arkin and Ross 1994; Bhalla 2003) and their similarities to Artificial Neural Networks (ANNs) (Bray 1995; Bhalla 2003). As discussed later, some have identified structural motifs common to many CSNs which form basic computational processing units.

The Artificial Reaction Network (ARN) is an Artificial Chemistry technique inspired by biological CSNs. The design, computational properties, mathematical formalism (Gerrard et al. 2013) and validation (Gerrard et. al 2011) of the ARN have already been discussed in detail. The ARN was previously used to simulate the chemotaxis pathway of *E. coli* (Gerrard et al, 2011), in pattern recognition and to generate complex temporal waveforms to control limbed robots (Gerrard et al. 2012a, b; 2013). Previous work focused on exploring the properties and mechanisms which lead to high-level behaviour in individual cells. The focus of this work is to explore those which result from groups of interacting cells using a new technique termed the "Cytobot". Cytobots are cell-like agents which, under direction of their internal ARN, autonomously move within and respond to their environment. Like other Artificial Chemistry approaches (Joanchimczak et al. 2013; Shen et al. 2004; Guo et al. 2009), a Cytobot system is composed of multiple cell-like components which communicate with each other and control their local actions via artificial chemicals. The specific objectives of the results presented here are as follows: Firstly, to explore the mechanisms and computational properties that lead to emergent high-level behaviour within and between groups of interacting cells. Secondly, to investigate applications of the ARN technique in biological simulation, and finally as a distributed robotic control system.

The following novel work is presented: 1) A complete overview of the design of Cytobots including their biological background and computational properties; and 2) a Cytobot based simulation of the life phases of Dd; and 3) a simulated oil-spill clean-up operation using a Cytobot swarm.

The paper is structured as follows: Section 2 briefly summarises the ARN representation. Section 3 discusses the biological background and behaviour of the Cytobots. Section 4 presents a complete overview of the Cytobot ARN design and discusses the biological functional motifs of which the network is composed. The experiments in section 5 explore the applications and properties of Cytobot systems. The first experiment (section 5.1) applies the Cytobots to the simulation of the Foraging and Aggregation phases of Dd. The phase times and emergent behaviours are compared with the literature. The results show that Cytobots are able to accurately model the behaviour of individual unicellular organisms, and that arising from interactions among such groups. They also demonstrate a high-level of flexibility where, for example, the pathway within an individual cell may be modified and its effects on high-level behaviour of the entire cell group viewed over time. In a further experiment (section 5.2), robotic swarm applications are investigated and a Cytobot swarm is applied to a simulated oil-spill clean-up operation. The results compare well with other related methods and show that Cytobots may have practical applications within the real-world as a physical robotic swarm.

2 The Artificial Reaction Network

A brief summary of the ARN is provided here; a full account can be found in our previous paper published in this journal (Gerrard et. al, 2013).

The ARN focuses on the inherent networked properties of CSNs and is specifically designed to represent “biological circuitry”; it consists of linked processing units connected together via weighted connections and for this reason may be described as “connectionist”. It is a networked representation similar to other ACC models (Zeigler and Banzaf 2000; Eikelder et al. 2009). As shown in Fig. 1, the ARN comprises a set of networked reaction nodes (circles), pools (squares), and inputs (triangles) and is depicted as a directed weighted graph. Each pool stores the current available amount of a particular chemical species (*avail*); thus, the complete set of pool concentrations at time t, corresponds to the current state of the system. While many ACs assume a well-stirred reactor, the use of pools approximates a chemical compartment, allowing a representation of the spatial compartmentalisation which occurs within cells. This also provides a means to represent flow structures such as membrane channels and transport processes. Inputs are a special type of pool which are of fixed value and thus can be used to represent the continuous flow of environmental inputs or enzymes. Data is processed by reaction units which transform incoming pool values to connected outgoing pool values.

Connections symbolise the flow of chemical into and out of reaction units and their weight (*w*) corresponds to reaction order. Connections provide a means to create complex control structures by combining inhibitory or excitatory unions.

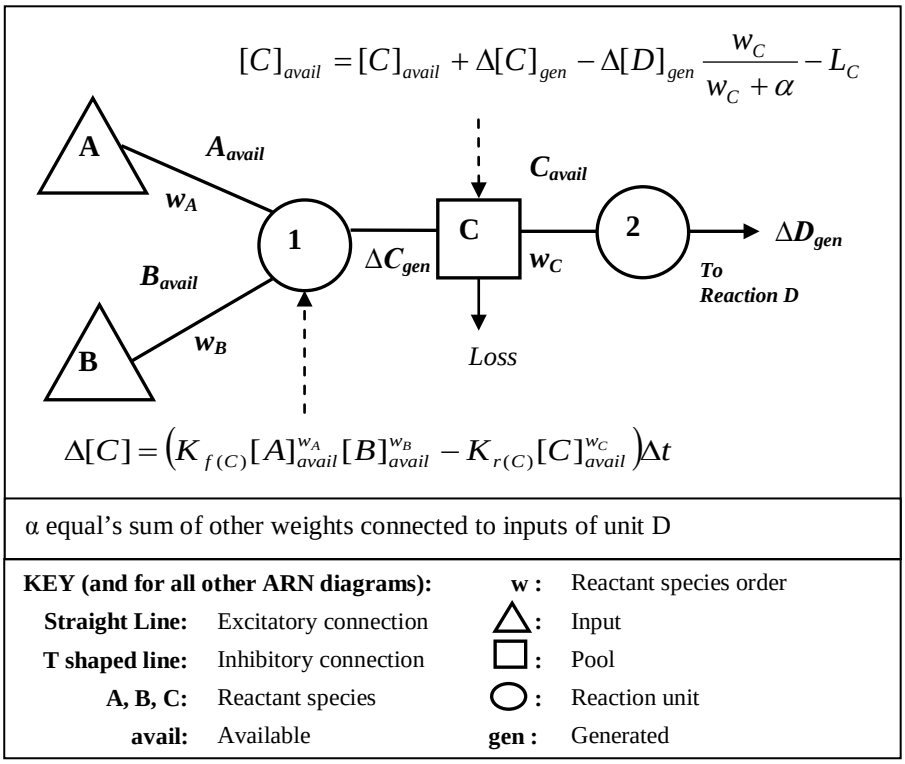


Fig. 1 Schematic diagram of a simplified Artificial Reaction Network (ARN). Reactant chemicals A and B react at unit 1. The rate of the reaction at unit 1 at time t is given by Eq. (1). The current concentration in pool C is updated using Eq. (2).

Figure 1 shows the reaction between species A and B to produce species C. The result of applying Euler's method to the differential rate equation is given by Eq. (1) and this is used to calculate each reaction unit's temporal flux value over the time interval Δt.

$$\Delta[C] = \left(K_{f(C)}[A]^{w_A}_{avail}[B]^{w_B}_{avail} - K_{r(C)}[C]^{w_C}_{avail}\right)\Delta t \quad (1)$$

This result is then used to update the current concentration of each reaction's connecting pools. As mentioned previously, conserved mass values are used throughout the experiments detailed in this work and thus each pool is updated at each time interval for example at pool C.

$$[C]_{avail} = [C]_{avail} + \Delta[C]_{gen} - \Delta[D]_{gen} \frac{w_C}{w_C + \alpha} - L_C \quad (2)$$

Non-conserved values may also be modelled and the mathematics for this is given in our previous work (Gerrard et. al 2013).

3 Cytobot Behaviour

A Cytobot has 2 behavioural modes which are based on the chemotaxis behaviour of *D. discoideum* amoebae. These modes and their biological basis are described below.

3.1 Biological Basis of Cytobot Behaviour: Chemotaxis of *D. discoideum* Amoebae

The Dd life cycle has 4 stages: Vegetation, Aggregation, Culmination and Migration, a detailed description of the biology is given by Devreotes (1989). During its Vegetative stage the organism consists of a collection of amoebae which navigate toward food by moving up gradients of folic acid secreted by their bacterial prey. The trajectory of these cells is a pattern of motion similar to a random biased walk. Dd cells extend pseudopods in a random direction; those extended toward sources of chemoattractants, such as food, are maintained; while those extended toward less favourable conditions and retracted. The overall result is movement up the gradient of attractant (Andrew and Insall 2007).

When the food resource has been depleted, the amoebae begin to stave and enter the Aggregation phase. Starving cells secrete cAMP (cyclic adenosine monophosphate), which serves as a signal to attract surrounding amoebae towards each other, resulting in a densely populated aggregate (Devreotes 1989) commonly referred to as a “mound”. Aggregating cells are polarized, thus one side becomes the leading edge, which always faces in the direction of travel (McCann et al. 2010). Depending on parameters such as environmental conditions and the cell population density, migrating cells can form transient emergent patterns such as streams and spirals (McCann et al. 2010; Dallon and Othmer 1997). Streaming describes a pattern of motion where cells line up in close-order files, with the head of one following the rear of another (McCann et al. 2010) and the spiral pattern describes streams of concentric cells spiralling toward the centre of the aggregate.

3.2 Cytobot Foraging Mode

In Foraging mode, a Cytobot performs a pattern of motion based on the previously described chemotaxis of Dd cells during their Vegetative phase. The Cytobots approximate this behaviour as a random biased walk by performing alternate periods of forward motion termed “runs” and random redirections called “tumbles”. The bias is provided by reducing the tumble frequency when moving toward more favourable conditions (for example up food gradient), thus increasing the length of the run. Like aggregating Dd cells, each Cytobot is polarised, and will always face toward the direction of travel. At each new position P, an agent redirects itself to face a new random angle between 0 and 360 degrees (a tumble). The agent then moves forward in a straight line for a number of time-steps, based on the level of detected food at P (a run). The Cytobot consumes all the food (if present) at each location it passes through.

3.3 Cytobot Starvation Mode

The Cytobot Starvation mode is based on the pattern of motion displayed by starving cells of Dd. The Cytobot enters Starvation mode if it has not consumed food within a fixed time period. During this phase, the Cytobots respond to detected levels of environmental cAMP. Depending on the particular experiment, this chemical may already be present within the environment or it may be released by the starving Cytobots. In this mode, both run and tumble behaviours differ from that in the Foraging phase. Rather than turning in a random direction, a new direction is calculated by weighting the turn toward the highest concentration of artificial cAMP within the surrounding area. The run period, instead of being variable, is a fixed length, which is set according to the particular experiment.

4 Cytobots: Design and Implementation

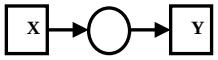
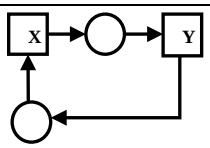
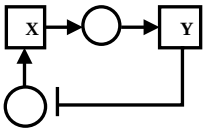
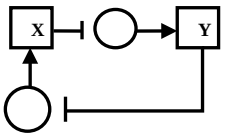
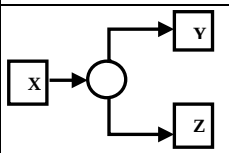
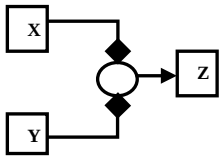
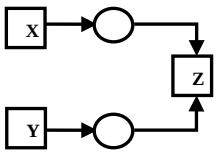
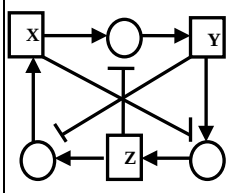
In previous work it was shown that the ARN can be used to model the reactions of specific proteins involved in signalling pathways (Gerrard et al. 2011). To enable the Cytobot ARN to produce the behaviour of chemotaxing Dd amoeba, rather than simulating specific protein interactions, a more abstract method was employed. There are a number of reasons for this approach. For instance, there are significant gaps in our current knowledge of the chemical interactions involved within this pathway (Manahan et al. 2004), and thus it is not possible to create an accurate representation. When modelling such a network researchers often adopt a modular approach- where related signalling events are grouped into functional units (Manahan et al. 2004). In this way, the Cytobot ARN was designed by dividing functions into modular units. The functional modules are constructed by combining “structural motifs” from real biochemical networks. Such motifs, each of which perform distinct computational functions, have been identified by a number of researchers (Tyson and Novak 2010; Bray 1995; kholodenko 2006). There are a number of advantages in this approach. Firstly, these motifs are universally found in other pathways, and thus show that the ARN is capable of potentially modelling any pathway. Furthermore, by creating entire systems composed of these motifs illustrates how biologically plausible motifs can be combined and cooperate together to produce functionally distinct pathways and how such pathways cooperate (a feature of crosstalk) to produce overall cellular behaviour. These functional motifs and the manner in which they are combined within the Cytobot ARN control system is discussed below.

A summary of common structural motifs, their computational function, structure (in the previously defined ARN format), and biological examples of each is provided in Table 1. Note that these motifs are shown for simplicity as 2 or 3 component forms but there are larger versions with the same function; for example, an additional component may be added to motif 9

to create a 4 component oscillator. One important biological example is the universal signalling motif of a phosphorylation cycle. Here a signalling protein is interconverted by opposing enzymes (a kinase and a phosphatase) between its phosphorylated (Yp) and non-phosphorylated forms (Ys). In a multisite phosphorylation cycle, feedback from either form can cause oscillations between stable states or render the cycle into a bistable switch, where the low and high Yp concentrations correspond to “on “and “off” states (kholodenko 2006). A cascade of such bistable cycles can produce multiple stable states, allowing the complex interdependent control of many cellular functions. For example, the cell’s transition into mitosis is governed by the sequential activation or inactivation of such kinases (CDK1/Cdc2) (kholodenko 2006). Space prevents a detailed discussion of each motif but an in-depth account including the biological mechanism, structure and examples is provided by Tyson and Novak (2010).

4.1 Functional Motifs in Biochemical Networks

Table 1 Functional Motifs in Biochemical Networks

Motif No., Name and Description	Structure (in ARN format)	Biological Example
1. Excitatory (E) The presence of X activates Y		Elementary motif common throughout most pathways. E.g. Ras is a membrane associated protein that is normally activated in response to the binding of extracellular signals such as growth factors (Tyson and Novak 2010).
2. Inhibitory (Y) The presense of X inhibits Y. Acts as a NOT gate.		Elementary motif common throughout most pathways. E.g. E-cadherin (a calcium-dependent cell–cell adhesion molecule) suppresses cellular transformation by inhibiting β-catenin (Tyson and Novak 2010).
3. Positive Feedback Loop (PFL) The presence of X activates Y and in turn the presence of Y activates X		The pathway of caspase activation is essential for apoptosis induction. A PFL exists between caspase-3 and caspase-9 (Tyson and Novak 2010).
4. Negative Feedback Loop (NFL) The presence of X activates Y and in turn the presence of Y inhibits X		The proteins Mdm2 and p53 (p53 is a tumour suppressor protein) are involved in a NFL which functions to keep the level of p53 low in the absence of p53-stabilizing signals (Tyson and Novak 2010).
5. Double-negative Feedback (DNF) The presence of X inhibits Y and the presence of Y inhibits X		BAX is protein which promotes apoptosis by competing with BCL. A DNF is formed between the proteins BAX and BCL (Tyson and Novak 2010).
6. Branch (B) The presence of X activates Y and Z		The transcription factors such as E2F or P53 frequently modulate the expression of more than one gene. Enzymes often modify more than one substrate e.g. CycB-dependant kinase (Tyson and Novak, 2010).
7. Logic Gate (LG1) **AND gate**: 2 excitatory connections from X and Y when both X and Y are present they activate Z. **NOR gate**: two inhibitory connections from X and Y. Both X and Y must be absent for Z to be activated**.** **SWITCH:** Excitatory connection from X and inhibitory connection from Y. The presence of X but not Y activates Z		**AND**: The protein gCam 2 kinase becomes active when both calcium ions (Ca^{2+}) and Calmodulin (CaM) are present (Bray, 1995).**NOR**: The activity of transcription factor E2F is a NOR function of RB and CycB where E2F is active when both RB and CycB are inactive (Tyson and Novak, 2010).**SWITCH**: The enzyme aspartate transcarbamylase has multiple catalytic sites. It is activated by binding of its substrates (aspartate and carbamoyl phosphate) and inactivated by cytidine triphosphate causing its substrates to dissociate (Bray 1995).
8. Logic Gate (LG2) **OR Gate:** : 2 excitatory connections from X and Y when either X or Y are present they activate Z		For instance, Ras is a membrane associated protein that is activated by a number of different signals. E.g. in response to the binding of extracellular signals such as a number of growth factors (Tyson and Novak 2010).
9. Oscillator (OSC) The presence of X activates Y. In turn the presence of Y activates Z but inhibits X. The presence of Z inhibits Y and activates X.		The cyanobacteria clock protein KaiC has a closed cycle of phosphorylation and dephosphorylation states (composed of KaiA, KaiB and KaiC). In the structure shown left, all 3 chemicals oscillate and each inhibits the reaction clockwise left. Oscillators may have less inhibitory connections, the number of which is dependent on the mobility of the reaction species. However, the presence of all inhibitors increases stability in the presence of fluctuating environmental parameters e.g. temperature.

Key: Either inhibitory or excitatory. X/Y/Z: Chemical species

Excitatory connection Reaction

Inhibitory connection Chemical substrate

* Motifs may combine arbitrary numbers of components.

4.2 The Cytobot ARN

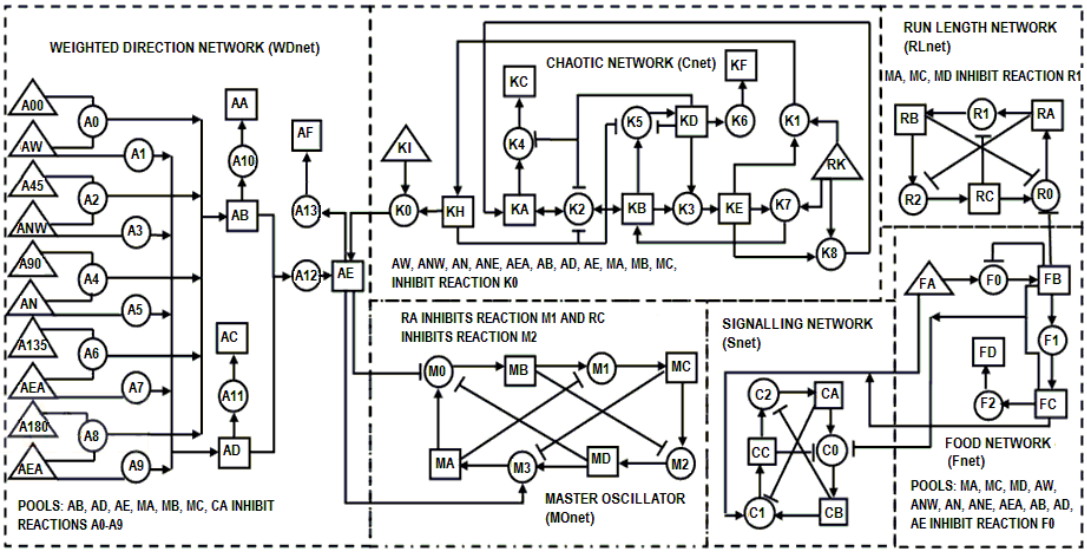


Fig. 2 The Cytobot ARN network. Each Cytobot is controlled by an instance of this network and thus has an independent state at time t. The network is composed of 6 subnetworks. * Note that in these experiments pools are considered empty when the value of its component chemical is $\leq 1x10^{-3}$.

Each Cytobot maintains its own copy of an ARN which enables it to operate asynchronously with respect to the other agents. In turn, this allows each Cytobot to react independently to situated environmental patterns, communicate with others and contribute to higher-level collective function. The Cytobots are placed within a simulated environment containing a distribution of chemicals. These chemicals represent the attractants of either food or cAMP. When a Cytobot moves to a new position, the surrounding level of chemical is used to set the inputs to its ARN. Consequently this changes the state of the network and updates its trajectory. During this process, the agent modifies the state of the environment by consuming food or releasing cAMP.

The Cytobot ARN is composed of 6 subnetworks as shown in Fig. 2. Each subnetwork contributes a functional aspect to either (or both) Starvation and Foraging behavioural modes. Each of these subnetworks is discussed below in detail.

4.2.1 The Master Oscillator

The Master Oscillator (MOnet) functions to synchronize the outputs from all the other subnetworks together and is what the individual Cytobot references at each time-step to ascertain its overall behavior. It is a 4 component oscillator (Table 1, motif 9), with a token unit of chemical cycling around it. It consists of 4 reaction units: M0, M1, M2, and M3 (all with a reaction rate of 1), 4 pools MA, MB, MC and MD and generates a pulsed-width-modulated waveform. Each pool is associated with 1 of 3 behaviors. At every time-step, if a particular pool contains the token unit, then its corresponding behavior is performed. Pool MA activates turn,

MC activates run and MB and MD activates stop. Thus, if pool MC contains a chemical for 10 time-steps, the agent will move forward for 10 time-steps. The other subnetworks inhibit (Table 1, motif 2) or excite (Table 1, motif 1) the reaction units of the MOnet, to allow or prevent chemical flow.

Note that this oscillator motif allows the Cytobot ARN to function easily as the control system for the motor actuators of a wheeled robot. Here, MC would switch on all wheel motors, while MA would switch on left-wheel motors only, thus turning the robot. The remaining pools would act as off switches.

4.2.2 The Food and Run Length Network

The Food Network (FNet) interfaces with the environment at pool FA, using an excitatory connection (Table 1, motif 1) and inhibits the Run Length network (RLnet) in accordance with the level of detected food. The forward rate of reaction at node F0 is 1, thus the content of FA is transferred to pool FB in a single time-step. The presence of chemical FB inhibits (Table 1, motif 2) R0 for a number of time-steps, according to the level of food (by setting forward rate of unit F1 to 1 and weight to 0, this can be an exact correlation). The RLnet is a 3 component oscillator (Table 1, motif 9). While reaction R0 is inhibited, it prevents pool RC from emptying. RC inhibits reaction M2 (Table 1, motif 2) of the MOnet thus preventing pool MC from emptying for the same number of time-steps. As discussed previously, the number of time-steps which pool MC contains the token unit represents the number of time-steps to move forward.

4.2.3 The Signalling Network

The Signalling Network (Snet) functions as a switch between Starvation and Foraging mode. A low food level triggers the starvation response and allows the Weighted Direction Network (WDnet) to control each new angle. Sufficient food will switch off the WDnet and allows the Chaotic Network (Cnet) to control each new angle. It is a 3 component oscillator (Table 1, motif 9) with a token unit of chemical flowing around it. Pool CA acts as the switch between Foraging and Starvation modes. Here the presence of chemical in CA inhibits the WDnet (Table 1, motif 2), while its absence switches on the WDnet; this in turn inhibits the Cnet, as shown in Fig. 2. In this oscillatory network, all reaction units have a forward rate of 0.5. This produces a continuously oscillating waveform and ensures a minimum number of time-steps for each behaviour. A NOR gate (Table 1, motif 7) activates pool CB in the absence of food chemical in both pools FB and FC of the Fnet, thus allowing pool CB to empty. An AND gate (Table 1, motif 7) will lead pool CA to eventually refill by activating pool CC, only when food is present in input FA and pool FC of the Fnet.

4.2.4 The Weighted Direction Network

The Weighted Direction Network (WDnet) senses cAMP within the agent's immediate environment and calculates a tumble angle which is weighted toward higher levels. This network interfaces with the environment via a number of receptor pools (AW, ANW, AN, ANE, AEA) which sense the level of food around the Cytobot. These pools represent receptors and are positioned at points around the front of its perimeter (as shown in Fig. 3), allowing the agent to travel in a similar way to that of a aggregating Dd cell. Each receptor input pool forms one input of an AND gate (Table 1, motif 7); the other input is a static pool containing a fixed level of chemical in correspondence to its direction. Directions start from AW (west) with a corresponding numeric value of 0 (A00) and progress in 45 degree steps through each direction to east (thus, the maximum value is 180). As the receptor positions around the agent are fixed, directions are always relative to that in which the agent is facing. All connections have a weight of 1 with the exception of the connection between pool AD and reaction A12 which has a weight of -1. This negative connection weight raises the sum of food detected in AD to -1, which multiplied by AB, allows an average angle to be calculated. Detected signals are classed as being in one of the following cardinal or ordinal directions: W, NW, N, NE, and E. Thus signals are detected from all directions above the horizontal plane. The calculated angle interfaces with the remaining subnetworks at pool AE. Pool AE is the output of an OR gate (Table 1, motif 8), and its inputs are activated by either the WDnet or the Cnet. AE also forms the inhibitory input of a SWITCH (Table 1, motif 7), where the presence of chemical in MA and absence in AE activates pool MB of the MOnet In the actual organism, receptors are set around the cell perimeter and direct movement appropriately. In this simulation, for simplicity, a count of the number of time-steps "*n*", that MA contains the token unit is processed to gain the new heading "*h*" relative to the agents' current heading "*c*" using Eq. (3):

$$h \equiv (n - 90) + c \tag{3}$$

Statement 1

```
IF (h > 360)THEN h = h – 360
IF (h < 0)  THEN h = h + 360
```

Thus, if the number time-steps is 120 and the agent is facing north, then the current heading would equal 0 and the new heading would equal 30.

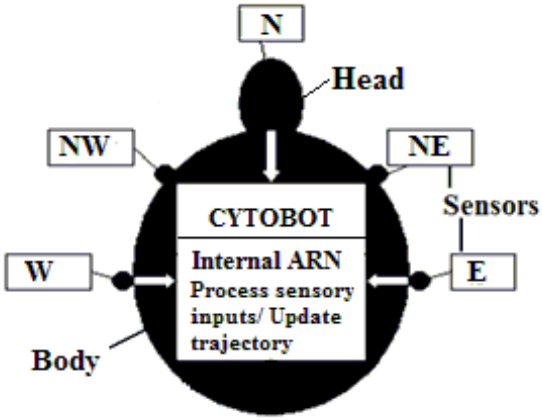


Fig. 3. Location of the Cytobot sensors around its perimeter.

4.2.5 The Chaotic Network

The Chaotic Network (Cnet), shown in Fig. 2, is responsible for generating the pseudo-random angles which agents use to perform each Foraging mode tumble. It is a networked implementation of a Logistic Map, given by Eq. (4), where X_n is a state variable of value $0 < X_n < 1$ at time-step *n* and λ is a system parameter of value $1 \leq \lambda \leq 4$:

$$X_{n+1} = \lambda X_n (1 - X_n) \tag{4}$$

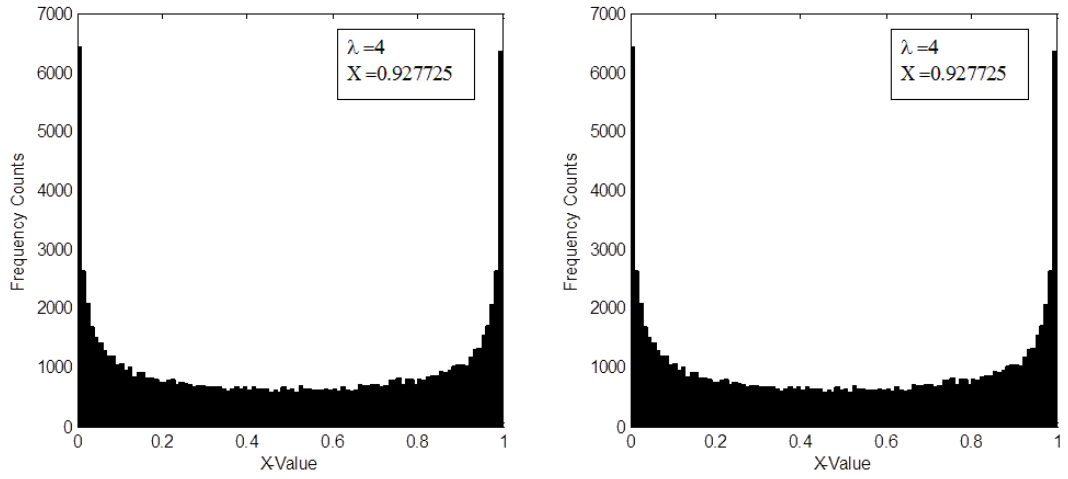
Without prior knowledge of the initial conditions, the output of the Logistic Map is not predictable; whereas, with prior knowledge it is deterministic. Therefore, the resulting series cannot be described as truly random, but as pseudo-random and its output has long been proposed as a pseudo-random number generator. Ulam and von Neumann (1947) were the first to examine this, and it has been successfully used in that capacity by several researchers (Patidar et al. 2009; Phatak and Rao 1995). The probability-density distribution of the Logistic Map, as given by Eq. (5) (where P(X) is the probability of X occurring at any time-step), is non-uniform (Patidar et al. 2009):

$$P(X) = \frac{1}{\pi\sqrt{X(1-X)}} \tag{5}$$

When λ=4, the distribution is "U" shaped with a higher probability of values closer to the minima and maxima of X and a symmetric distribution around the midpoint.

To implement the Logistic Map, a number of motifs are combined, including multiple branch motifs (Table 1, motif 6- KB activates KD and KE), PFLs (Table 1, motif 3- a multi component PFL exists where KA leads to activation of KE, which results in the activation of KA) and NFLs (Table 1, motif 4- KA activates KD which in turn inhibits KA). At the start of the simulation, pools KA and KB are initialized to the same random value (a unique number for

each Cytobot), between 0 and 1 (to 5 decimal places). This value represents the initial value of X of Eq. (4). All the other pools are initialized to 0, with the exception of the static pools KI and RK, whose initial values are 360 and 1 respectively. Reaction K2 is responsible for generating each new value of X and has a forward and reverse rate of 4 (the Logistic Map exhibits chaotic behaviour when λ is 4). The connection between KA and K2 has a weight of 1 and that between K2 and KB has a weight of 2. The remaining series of reactions function to copy the value of X 3 times; where 2 copies serve as the new initial values of KA and KB, and one participates in the final output of the network at KH. KI has a fixed value of 360 which allows the network to convert the pseudo-random number at KH to an angle value between 0 and 360 at reaction K0. However, reaction K0 cannot proceed until all 11 pools that inhibit it are empty.



a) b)

Fig. 4 The Frequency distribution for each value of X when X is initialised to 0.9277725 and λ=4 resulting from: a) the chaotic network b) Recursive relation given by Eq. (2) run using Matlab.

These inhibitory connections (Table 1, motif 2) ensure that random angles are not output while the agent is in starvation mode, and that pool AE is empty before adding more chemical.

The ARN implementation of the Logistic Map was verified against the recursive relation shown in Eq. (4) using Matlab, where λ=4, initial X = 0.927725, and iterated for $1x10^5$ steps. The complete range of state-variables between 0 and 1 were divided into 100 equal subintervals and the frequency of occurrence of each subinterval interval was plotted. Similarly, the Cnet was run for $1x10^5$ cycles, using the same parameters of X (initial value) and λ. These results were processed in the same way and are shown in Fig. 4. The frequency distribution gained from the ARN is identical to that obtained using Matlab and by other researchers using the same parameters (Patidar et al. 2009). The same comparison was repeated 100 times at different values of X, and the ARN consistently produced the same values as Eq. (2).

5 Experiments and Results

In the following sections the methodology and results for the following experiments are presented: 1) A Cytobot based simulation of the Vegetative and Aggregative life-cycle phases of Dd including the transition; between them and 2) application of the same Cytobots in a simulated oil-spill clean-up operation.

5.1 *D. discoideum* Simulation

5.1.1 *D. discoideum* Simulation Methodology

The experiments are grouped into two sets: Aggregation (AG1-10 of Table 2) which models the Aggregation phase only and Foraging to Aggregation (AGF3 and AGF8 of Table 2) where both the Vegetative and Aggregative phases are simulated, including the transition between them. Each experiment is performed at varying population densities of Cytobots (p) and different distance ranges of detection of cAMP (r). The experiments AGF3 and AGF8 are performed at the same p and r as experiments AG3 and AG8 respectively to compare the effect of the Foraging phase on the number of mounds formed and length of time to complete the Aggregation phase. The emergent patterns, numbers of mounds, and length of time to complete phases is examined and compared in both sets of experiments and with the literature. In each experiment the Cytobots move within a 2D simulated environment which represents an area of 5.06 mm^2- approximately half the maximum Aggregation territory reported in the literature (Dallon and Othmer 1997). A screen output shows the position of the Cytobots in real-time and is a grid of 500 × 500 pixels where each is represented by a square of side 4.5 μm. In nature, aggregating Dd cell densities are typically 250 to 1000 per mm^2 (Dallon and Othmer 1997). Due to the computational resources required to manage a population of Cytobots within the upper range, a cell density at the lower biological range of 250 agents per mm^2 (1250 Cytobots) and another at 150 per mm^2 (750 Cytobots) were chosen.

Table 2 *D. discoideum* Simulation Results

No.	Cytobots per mm^2 (p)	Range (r) in mm	Mean No. of mounds; (σ)	Aggregation Phase Mean time in Hours; (σ);
AG1	**150**	**5**	**1 (0)**	**8.98 (0.09)**
AG2	**150**	**2.5**	**4 (0.31)**	**9.63 (0.17)**
AGF3	**150**	**1**	**5.9 (1.16)**	**9.47 (0.65)**
AG3	**150**	**1**	**5.2 (0.82)**	**9.92 (0.34)**
AG4	**150**	**0.5**	**8.4 (1.19)**	**10.23 (0.59)**
AG5	**150**	**0.1**	**14.2 (2.36)**	**10.6 (1.82)**
AG6	**250**	**5**	**1 (0)**	**8.95 (0.11)**
AG7	**250**	**2.5**	**1 (0)**	**9.6 (0.20)**
AGF8	**250**	**1**	**6.8 (1.81)**	**9.71 (0.87)**
AG8	**250**	**1**	**4.3 (0.37)**	**10.05 (0.58)**
AG9	**250**	**0.5**	**6.7 (1.62)**	**12.65 (1.94)**
AG10	**250**	**0.1**	**-**	**-**

```
FOR each Cytobot
    Get current agents' facing direction C_F
    Assign a value to direction C_F using statement 1

        FOR each (index n) detected cAMP signal
            Get detected signal incoming direction C_A
            Assign a value to direction C_A using statement 1
            IF CA = C_F THEN k_n = 3
              ELSE IF C_A = C_F-1 OR C_A = C_F+1 THEN k_n=2
                  ELSE IF C_A = C_F-2 OR C_A = C_F +2 THEN k_n=1
              ELSE k_n=0
             END IF
            Calculate distance d_n
            Store each C_A with k_n and d_n
        END FOR

    Calculate W_A for current agent using Eq. (7)
END FOR

Statement 1: East = 1; North East = 2; North = 3; North West =4;
West = 5

Where:
W_A= total weight of direction A
N= total number of agents within range of detection
d_n= distance of current agent from agent n
C_A = direction of incoming signal detected by current agent
C_F = the current agents facing direction
k_n = value of cAMP signal from agent n
```

Fig. 5 The strength of signal for each cardinal or ordinal direction above the horizontal plane of a Cytobot is calculated using this pseudocode. The result for each direction is used to set the corresponding direction input pool of the ARN WDnet.

In both sets of experiments the Cytobots are initialized at random positions in Foraging mode within the simulated environment. In the AGF experiments, the environment is initialized with a radial outwardly-decreasing gradient of food (z), as described by Eq. (6), where x and y are Cartesian coordinates on the horizontal plane:

$$z = \sqrt{x^2 + y^2} \tag{6}$$

$$W_A = \sum_{n=1}^{N} \frac{k_n}{d_n} \tag{7}$$

The Cytobots begin the experiment in the previously described Foraging mode and remain in this mode until the food resource is depleted and Starvation mode is triggered. In a real environment, food is non-uniformly distributed, may be regenerated and can move (in the case of bacterial prey). Thus, this setup is highly simplified, but is comparable to other simulations (Becker 2010).

If a Cytobot does not detect food for a period of approximately 5 time steps (the exact number depends on the level of food detected in the recent past, because higher levels take longer to flow through the network) it will enter Starvation mode. Cytobots in Starvation mode emit a cAMP signal at equal strength in all directions around their circumference into the environment. Each Cytobot in Starvation mode detects the cAMP signal of all other starving agents within a radius r. The total value for each direction is calculated using the pseudocode given in Fig. 5 and these totals are used to set the weighted direction network input (receptor) pools. A range of r values were explored, including that of real Dd cells: 1, 0.5, and 0.1 mm (McCann et al. 2010). The cAMP signal degrades linearly with increasing distance from the emitting cell. Each cycle represents 1 minute of time. In this time an aggregating Cytobot moves 9μm- a distance which corresponds to that of actual aggregating Dd cells (Rifkin and Goldberg 2006). Therefore, after 1 hour of motion a Cytobot travels a distance of 540μm. In this simulation, just as in biology, there are always remaining cells that do not aggregate, and thus the simulation runs until 95% of agents are at a distance of less than 0.1mm from their nearest neighbour.

5.1.2 *D. discoideum* Simulation Results and Discussion

The results for all 12 experiments are given in Table 2. Each experiment was performed 100 times. In the AG experiments an increase in p by 100 per mm^2 resulted in a decrease in the number of mounds formed at each value of r, with the exception of experiment AG6. This is not surprising, as denser populations have more chance of interacting, and thus form fewer clusters, each having a higher number of agents. Similarly, decreasing r results in a general

increase in the number of mounds formed at both values of p. The likely reason for this is that as r decreases the Cytobots area of influence becomes increasingly smaller, and thus the number of isolated stable clusters with fewer agents increases. In the AGF experiments, agents generally focus on consuming food in each of the remaining areas of highest concentration (see Fig. 6K-L). Having consumed almost all the food, agents begin switching to starvation mode (Fig. 6M). In these experiments the number and location of resulting mounds differs from that of the AG experiments at the same values of r and d. For example experiment AG8 results in an average of 4.3 mounds within the test space (Fig. 6E) while AGF8 results in an average of 6.8 mounds and a general shift in mound formation further away from the centre of the environment (as shown in Fig. 6O). The likely explanation is that, at the time of switching to aggregation, the majority of cells were forced outward toward the next remaining highest concentration of food. Emergent behaviours and clustering patterns similar to the biological organism were also observed.

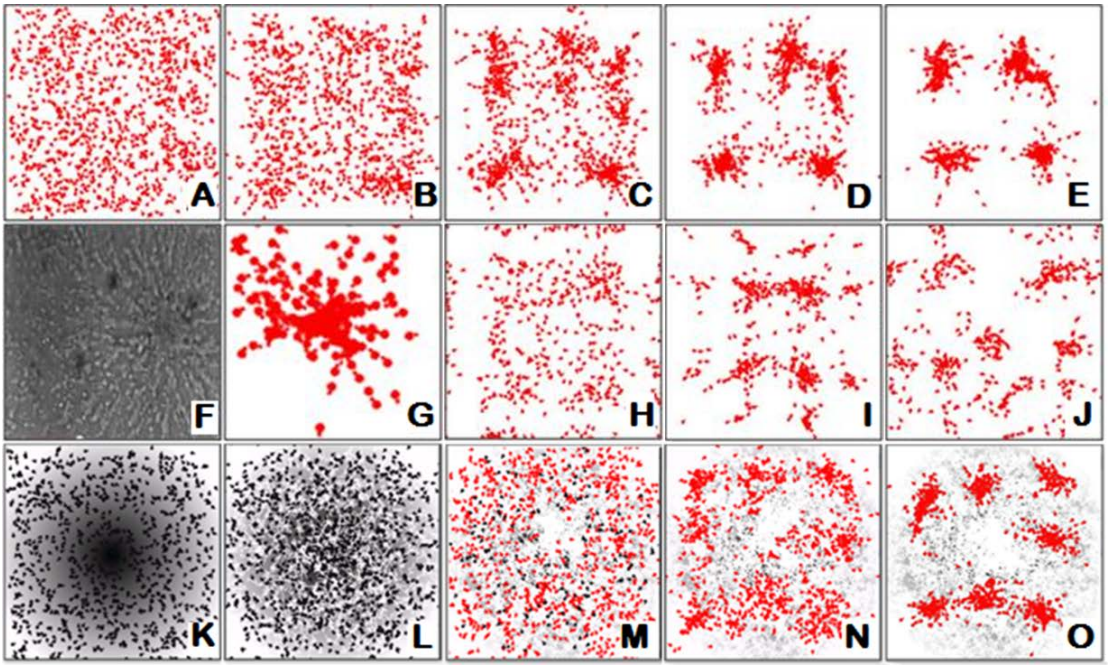


Fig. 6 Screenshots of the Dd simulation. Dots represent the Cytobots (black- vegetative and red- aggregative cells), and greyscale colour represents the food distribution. A-E: Cytobot aggregation experiment AG8 at A- 1hr, B- 2hr, C-5hr, D- 8hr, E- 10hr; Image F- real Dd cells aggregating; G- Lower right hand corner of image C demonstrating streaming behaviour; H-J Shows pattern formation; K-O Cytobot experiment AGF8 at K-0hr vegetation, L-4hr vegetation, M-transition to aggregation 0hr aggregation, N-5hrs aggregation, O-10hr aggregation.

Diagram F courtesy of T, Gregor, Laboratory for the Physics of Life, Princeton University, 2013 Used with permission.

In experiments AG8-10 and AGF8 the value of r and p are within the ranges for real Dd cells. These experiments are used to compare the behaviors and aggregation time with the values for real Dd in the literature. In experiments AG8-9 and AFG8 mound formation completes within

the range reported for the actual organism of 9-13 hours (Cotter et al. 1992; Becker et al. 2010). These results are comparable with other work. For instance, Becker et al. (2010) report an aggregation time of 11.6 hours for a simulated population of Dd with a cell density of 200mm^2. In experiment AG10, the population never satisfied the criteria for completion of mound formation where instead the agents appeared to move in a fashion reminiscent of Brownian motion. The likely explanation is firstly because the simulation does take into account glycoprotein's which allow aggregating cells to attach together on contact. Furthermore, because r is small, fewer agents are detected by each Cytobot. Thus momentarily larger clusters with higher attraction strength go undetected and quickly dissipate- an effect that would not occur if agents stayed together. As previously discussed, the Cytobots are polarized. Implementing the agents in this way allows us to observe whether or not the previously described streaming behavior occurs. A close-up of the right-hand corner of screenshot C is shown in Fig. 6G showing agents beginning to form a cluster. The protruding head of each agent can be seen clearly, and each lines up its head to the rear of another to form a stream. As can be seen in Fig. 6F, this is very similar to the streaming behavior in real cells of Dd. Other emergent patterns occurred during different experiments including spirals (Fig. 6J), symmetric patterns (Fig. 6I), and waves (Fig. 6H).

These results show that the Cytobots are able to simulate behaviour of individual unicellular organisms, and the emergent behaviours arising from their interaction. It highlights a potential use, as a means to simulate groups of interacting cells, for example a bacterial colony or tissue component within a multicellular organism. Applications include the modelling of the effects of disease (e.g. faulty gene expression) and pharmaceuticals on global behaviour. The results demonstrate the parallels between ARN agents and their biological counterpart; like amoebae, their internal network of spatially distributed dynamic chemical species allows them to autonomously coordinate and direct their movement, recognize and respond to patterns in the environment, and produce high-level behaviour.

5.2 Oil-spill Confrontation Simulation

5.2.1 Oil-spill Simulation Methodology

To illustrate a practical application of the Cytobot system within robotics they were used to tackle a simplified oil-spill clean-up simulation. The Cytobots move within a 2D environment containing an oil-spill on water. This oil is analogous to a distribution of food within a nutrient landscape. In the following 4 experiments the length of time it takes for a swarm of 3, 5, 8 and 15 Cytobots to clean up 95% of a simulated oil-spill is recorded. The agents move through the environment by switching between the two previously described behavioural modes- Foraging and Starvation. In these experiments, each Cytobot is controlled using the same ARN network

as used in the Dd simulation. To enable the Cytobots to behave differently, rather than modify the network, the interface between the Cytobots and the environment was altered. To achieve this, the concentration of oil surrounding the agents was used to represent both food and cAMP attractants. Thus, the amount of oil at each new position was fed into both the receptor pools of the WDnet and of the Fnet. At the start of each experiment, the Cytobots are distributed randomly within the environment, and the ARN network is initialized as previously described. The agents start the simulation in Foraging mode but during the simulation alternate between Foraging and Starvation modes. Starvation behaviour is triggered when the last positions (minimum of 2) contained zero food. In Starvation mode, instead of turning in a random direction, the new direction is weighted toward higher concentrations of food within its surrounding area. This behaviour forces exploration of unexplored search space because previously visited positions have a food level of 0. Consumption of environmental food therefore acts as a stigmergic signal, where agents are inclined to move up the nutrient gradient created by their foraging activities. On consuming a sufficient amount of food, the Cytobot switches back to Foraging mode, repeating this behaviour until 95% of the oil is consumed. Here, we model the spillage of 100 tonnes of Statfjord crude oil at 15^0C under a wind speed of $5ms^{-1}$ The oil is distributed over a 2D sea surface of 300m by 200m, thus an area of area $60000m^2$, where 2 pixels corresponds to 1m, as shown in Fig. 7A. This particular oil type and parameter set were chosen in order to compare directly with work by Kakalis and Ventikos (2008) who present a robotic swarm concept for oil-spill confrontation. For this reason, we account for an initial response time of 14 hours. Based on the mathematical models found in Kakalis and Ventikos which account for the main factors of short term changes in oil characterization, the volume of oil after 14 hours is reduced to $150m^3$. Beyond this starting state, the volume is only influenced by the Cytobots. The speed of each agent is $0.5ms^{-1}$ and is based on other robotic agents in oil cleaning scenarios (Kakalis and Ventikos 2008), thus the Cytobots move 1 pixel (0.5m) for every time step. The actual cleaning surface is 1m, thus the Cytobots clean a 2 pixel wide area in each time step.

Mathematical modelling of an oil-spill is non-trivial and at best can offer a crude approximation of its actual trajectory. Most oil-spills quickly form a comet shape with most of the oil within the head, and a trail of sheen (Wang and Stout 2007). To represent a simplified version of the comet shaped spread, the area is divided into 100 3m x 200m segments. The first segment contains 0.015 tonnes of oil, and each subsequent segment increases by 0.03 tonnes from right to left.

5.2.2 Oil-Spill Results and Discussion

In each experiment, a different number of Cytobots was deployed- 3, 5, 8 and 15 and the recovery rate achieved by each group were compared. The simulation time was measured from

deployment of the Cytobots at 0 hours (14 hours after oil was spilled) and stopped when the Cytobots had collectively removed 95% of the 150m^3 of oil. Each experiment was run 100 times, and the average volume of oil consumed at 6 minute intervals was calculated. Figure 8 presents the average volume of oil consumed by the group of Cytobots against time. Figure 9 provides the average length of time taken to clean 95% of the oil (Avg. time) and standard deviation (σ) for each experiment. By adding 2 additional agents to the group of 3 the length of time is reduced by 3.7 hours, thus 1.85 hour average difference per extra Cytobot. This difference decreases 1.12 hours per Cytobot for 8 agents, then to 0.76 per agent for 15. The variation can be accounted for by examining the agents' paths through the oil. Rates are much faster at the beginning of the experiments, where Cytobots move toward the oil-rich left side of the environment. This can be seen in the series of screenshots of a typical experiment shown in Fig. 7, where A shows the starting position at time 0, and B shows that after 2 hours the Cytobots have moved toward the left-hand side, focusing mainly on highly concentrated areas (consumed oil is shown in white). Initially, the rate of oil removal is high because Cytobots focus on the highest concentration areas and cannot go over their path, thus each new location results in consumption of oil. However, as time progresses, large patches become cleaned and a higher probability exists for the Cytobots to revisit previously cleaned areas. The consumption of oil in Fig. 7C-D at 4 and 9.6 hours respectively shows more clearly that Cytobots focus cleaning efforts on the area of highest concentration first, and are gradually forced to move toward the next highest concentration by the gradient created by their foraging activities.

Figure 7D shows the state of the oil at the end of the simulation, where only small patches remain mainly in areas of low oil concentration. These results can be compared to the simulation by Kakalis and Ventikos. Here, varying numbers of simulated EU-MOP robots are deployed to tackle 150m^3 of Strajford oil over 60000m^3 (as before). In this case, the robots have a slightly faster speed of 0.54m/s but have the same 1m skimming face. Each EU-MOP robot has a storage capacity of 2m^3 and a transit speed of 2.1ms^{-1}. The times taken for 3, 5, 8, and 15 EU-MOPS are 54, 32, 20 and 10 hours respectively. For comparison, the results of our simulation can be adjusted to include unloading of the oil at a servicing vessel. Using the same storage capacity and transit speed and assuming the distance to the ship and back is 2 times 300m and that each Cytobot fills the same amount simultaneously, then the new times are 17.2, 12.7, 10.3 and 6.5 for 3, 5, 8 and 15 Cytobots respectively. The Kakalis and Ventikos simulation has several differences to the one reported here, particularly in the distribution of the oil. Also, some key parameters are missing from their paper, for example, the distance to the boat. Despite these differences, our results are very similar. For example, the reported simulation time for 15 EU-MOPS is 10 hours and in our simulation 5 and 8 Cytobots took 12.7 and 10.3 hours respectively. Given the differences in the simulation and differences in operation of the robots, the resulting clean-up times are comparable, showing that the Cytobots

have potential application as distributed robotic agents in real-world environments. This application demands an internal control system which can function without reference to other agents within the environment which are operating in parallel. By modifying the environment, (which in this case was consumption of food), the agents can stigmergically communicate and facilitate emergent behaviour. The Cytobots offer a unique range of abilities. Like cells, their internal network of spatially distributed dynamic chemical species allows them to autonomously coordinate and direct their movement, recognize and respond to patterns in the environment, and produce high-level behaviour.

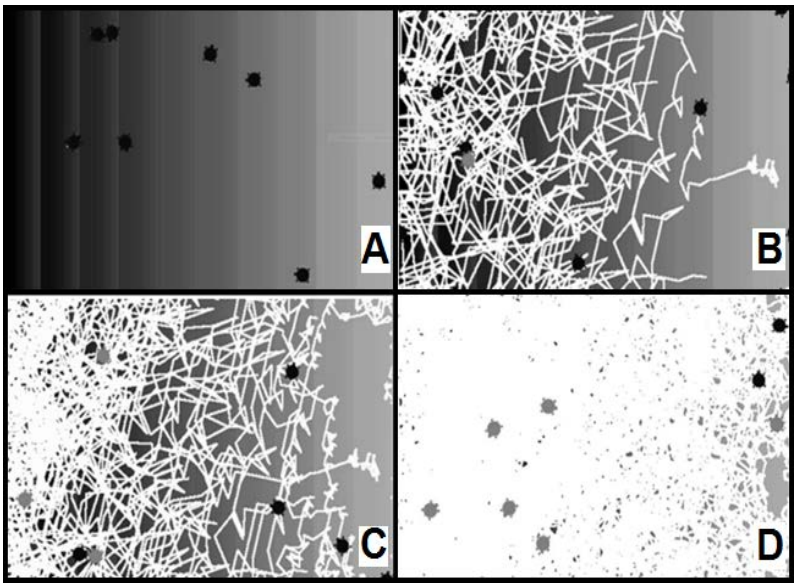


Fig. 7 Oil simulation using 8 Cytobots at A- 0 hours, B- 2 hours, C- 4 hours and D- 9.6 hours

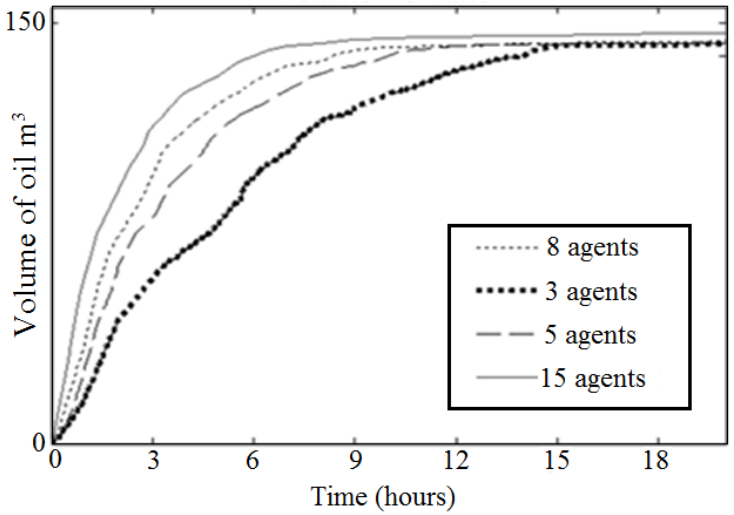


Fig. 8 Average volume of oil cleaned against time for each group of Cytobots

No. of Cytobots	Avg. finish times	Standard deviation (σ)
3	15.2	3.4
5	11.5	2.7
8	9.6	2.8
15	6.1	3.1

Fig. 9 Average length of time taken to clean 95% of the oil-spill for each group of Cytobots.

6 Conclusions

The experiments outlined in this paper show the advantages of considering cell-signalling networks as a connectionist paradigm. This approach allows their structure to be easily visualised, manipulated and organised into hierarchical modular structures. These in turn facilitate the exploration of the limits of their processing capabilities and, from an artificial intelligence perspective, allows them to be compared directly with other forms of simulated and biological intelligence. It may also prove useful tool in biomedical research as it allows, for example, the effect of mutated proteins to be examined simply in isolation or in interconnected groups - this is of particular importance in cancer research.

The results presented above illustrate how common simple motifs, present in all CSNs, can be integrated together to form structured networks with sophisticated processing capabilities. This indicates that these may form universal building blocks from which higher-level functions can be built. The ARN based agents, constructed from these, behave in a very similar way to the real organisms; displaying two of their most interesting behaviours (foraging and aggregation), and so we may conclude that the ability to evolve this level of behaviour is probably fairly universal among such single-celled microbes.

In the next stage of work, it is hoped to use an ARN to explore how learning might arise in protists. In particular the extent to which learning and memory is genetically programmed into invariant CSNs, and how much of it is extrinsic to the organism – as a stigmergic system. Or whether it has a variable intrinsic aspect – for example, the use of modulating elements within the CSNs, which might act as primitive “memories”. In the experiments outlined in this paper, the stigmergic aspect of such memory was illustrated. For example, like the cAMP trail-following behaviour described above, the environment acts as a shared information depository in which to both facilitate collective manipulation of data and communicate the current global system state.

There may also be a role for the ARN in other areas which have yet to be explored, particularly those which would benefit from its connectionist approach described in the paragraphs above. For example, it would appear ideal for modelling the complex but interconnected pathways present in environmental science and soil chemistries.

Supplementary information and code can be found at the following link:

https://drive.google.com/folderview?id=0B-xGVfJFH9UmZlJTV1pROFFRb00&usp=sharing

References

Andrew N, Insall R (2007) Chemotaxis in shallow gradients is mediated independently of PtdIns 3-kinase by biased choices between random protrusions. Nat. cell boil. 9(2):193-200

Arkin A, Ross J (1994) Computational functions in biochemical reaction networks. Biophys. J 67:560-578

Becker M (2010) Simulation model for the whole life cycle of slime mould Dictyostelium Discoideum. In: Proceedings of the European conference on modeling and simulation, pp 247-253

Bhalla U (2003) Understanding complex signaling networks through models and metaphors. Prog. Biophys. Mol. Bio. 81:41-65

Bray D (1995) Protein molecules as computational elements in living cells. Nature. 376(6538):307-12

Brock D, Douglas T, Queller D, Strassmann J (2011) Primitive agriculture in a social amoeba. Nature 469 (7330): 393-396

Cotter D, Sands T, Virdy K, North M, Klein G, Satre M (1992) Patterning of development in Dictyostelium discoideum: factors regulating growth, differentiation, spore dormancy and germination. Biochem. Cell Biol. 70(10-11):892-919

Dallon J, Othmer H (1997) A discrete cell model with adaptive signaling for aggregation of Dictyostelium discoideum. Philos. T. Roy. Soc. B 352(1351): 391–417

Devreotes P (1989) Dictyostelium discoideum: a model system for cell-cell interactions in development. Science 245(4922): 1054-1058

Ford B (2009) On intelligence in cells: The case for whole cell biology. Interdisipl Sci Rev 34(4):350-365

Gerrard C, McCall J, Coghill G, Macleod C (2011) Artificial Reaction Networks. In: Proceedings of the 11th UK Workshop on Computational Intelligence, Manchester: UK, pp 20-26

Gerrard C, McCall J, Coghill G, Macleod C (2012a) Temporal patterns in Artificial Reaction Networks. In: Proceedings of The 22nd International Conference on Artificial Neural Networks Lausanne, part 1, vol. 7552, pp 1-8

Gerrard C, McCall J, Coghill G, Macleod C (2012b) Adaptive Dynamic Control of Quadrupedal Robotic gaits with Artificial Reaction Networks. In: Proceedings of the 19th International Conference on Neural Information Processing Doha, vol. 7663, part 1, pp 280-287

Gerrard C, McCall J, Coghill G, Macleod C (2013) Exploring aspects of cell intelligence with artificial reaction networks. Soft Computing (in press)

Guo H, Meng Y, Jin, Y (2009). A cellular mechanism for multi-robot construction via evolutionary multi-objective optimization of a gene regulatory network. BioSystems 98(3):193-203

Joachimczak M, Kowaliw T, Doursat R, Wrobel B (2013) Controlling development and chemotaxis of soft-bodied multicellular animats with the same gene regulatory network In: Advances in Artificial Life, ECAL, vol. 12, pp. 454-461

Kakalis N, Ventikos, Y (2008) Robotic swarm concept for efficient oil spill confrontation. J. Hazard. Mater. 154(1-3):880-7.

Kholodenko B (2006) Cell Signaling dynamics in Time and Space. Nat Rev Mol Cell Biol 7(3):165-176

Manahan C, Iglesias P, Long Y, Devreotes P (2004) Chemoattractant signaling in Dictyostelium discoideum. Annu. Rev. Cell Dev. Biol. 20, 223–253

McCann C, Kriebel P, Parent C, Losert W (2010) Cell speed, persistence and information transmission during signal relay and collective migration. J. Cell Sci. 123:1724-1731

Patidar V, Sud K, Pareek N (2009) A pseudo random generator based on chaotic logistic map and its statistical testing. Informatica 33:441-452

Phatak S, Rao S (1995) Logistic map: A possible random-number generator. Phys. Rev. 51(4): 3670.

Rifkin J, Goldberg R (2006) Effects of chemoattractant pteridines upon speed of D. discoideum vegetative amoeba. Cell Motil. Cytoskeleton, 63(1): 1-5

Shen W, Will P, Galstyan A, Chuong C (2004) Hormone-inspired self-organization and distributed control of robotic swarms. Auton. Robot 17(1): 93-105

ten Eikelder T, Crigins S, Steijaert M, Liekens A, Hilbers P (2009) Computing with feedforward networks of artificial biochemical neurons. In: Proceedings of the 2nd International Workshop on Natural Computation, Japan: Springer: vol 1 pp 38-47

Tyson J, Novák B (2010) Functional motifs in biochemical reaction networks. Annu. Rev. Phys. Chem 61:219-240

West S, Diggle S, Buckling A, Gardner A, Griffin A (2007) The social lives of microbes Annu. Rev. Ecol. Evol. Syst.38: 53-77

Ziegler J, Banzhaf W (2000). Evolving a" nose" for a robot. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)

Paper 2: ‘Exploring aspects of Cell Intelligence with Artificial Reaction Networks’

Reference:

Gerrard, C., McCall, J., Coghill, G. and Macleod, C. (in press) ‘Exploring aspects of cell intelligence with Artificial Reaction Networks’, A Fusion of Foundations Methodologies and Applications, *Soft Computing*, Springer.

Exploring Aspects of Cell Intelligence with Artificial Reaction Networks

Claire E. Gerrard • John McCall • George M. Coghill • Christopher Macleod

Claire. E. Gerrard • John McCall
IDEAS Research Institute, Robert Gordon University, Aberdeen, AB10 7GJ, UK
email: c.e.gerrard@rgu.ac.uk

John McCall
email: j.mccall@rgu.ac.uk

Geroge M. Coghill
School of Computing Science, University of Aberdeen, Aberdeen, AB24 3FX, UK.
email: g.coghill@abdn.ac.uk

Christopher Macleod
School of Engineering, Robert Gordon University, Aberdeen, AB10 7GJ, UK.
email: chris.macleod@rgu.ac.uk

Keywords Artificial Biochemical Network (ABN), Artificial Chemistry , Artificial Neural Network (ANN), Artificial Reaction Network (ARN), Cell Signalling Network (CSN), Robotics

Abstract The Artificial Reaction Network (ARN) is a Cell Signalling Network inspired connectionist representation (CSN) belonging to the branch of A-Life known as Artificial Chemistry. Its purpose is to represent chemical circuitry and to explore computational properties responsible for generating emergent high-level behaviour associated with cells. In the paper, the computational mechanisms involved in pattern recognition and spatio-temporal pattern generation are examined in robotic control tasks. The results show that the ARN has application in limbed robotic control and computational functionality in common with Artificial Neural Networks. Like spiking neural models, the ARN can combine pattern recognition and complex temporal control functionality in a single network, however it offers increased flexibility. Furthermore, the results illustrate parallels between emergent neural and cell intelligence.

1 Introduction

In recent years, researchers have become increasingly interested in the complex behaviours displayed by individual cells. Such behaviours are exemplified by simple eukaryotic organisms called protists. These show an astonishingly varied repertoire of seemingly intelligent behaviours. For example, some have simple eye-spots to help avoid high light levels; others have locomotory appendages and stinging arrows to actively hunt and subdue their prey (Ford 2009). All this is accomplished without recourse to a neural network, the foundation-stone of intelligence in higher animals. The behaviour of such simple organisms may be labelled as "Cell Intelligence".

In order to generate this emergent high-level behavior, a cell must be able to store and process information. Data is represented internally by a set of spatially distributed molecular concentrations. Cell Signalling Networks (CSNs) process this information within elaborate hierarchical network control structures which connect species together in productive or inhibitory unions. In this way, cells are able to respond to changes within their environment, communicate with other cells, and perform internal self maintenance operations (Bray 1995). Several researchers have highlighted the processing capabilities of these networks (Bray 1995; Arkin and Ross 1994; Bhalla 2003) and similarities to Artificial Neural Networks (ANNs) (Bray 1995; Bhalla 2003). For example, Bray (1995) claims that individual network units can perform Boolean and fuzzy logic and act as a Turing machine. In other work, Stadtman and Chock (1997) demonstrated that such a network can act as a flexible computational unit. Similar results were documented by Arkin and Ross (1994), and recently a number of researchers have developed these ideas (Hild et al. 2010; Wang 2011).

In this paper the properties and applications of a connectionist model inspired by CSNs termed the Artificial Reaction Network (ARN) are discussed. This representation was introduced previously (Gerrard et al. 2010) where it was used to create a simulation of the chemotaxis pathway of *Escherichia coli*. In later work biochemical network motifs were investigated as a means to perform computational processing in a single ARN based system (Gerrard et al. 2011a; 2011b).

The ARN belongs to the branch of Artificial Life known as Artificial Chemistry Computing (ACC). This field of study utilizes principles of the Chemical Metaphor to construct novel software or hardware architectures *in silico* (Dittrich et al. 2001). In the Chemical Metaphor, data is stored in the form of molecular species and information processing occurs through interactions (reactions) between these molecules. The result of this computation appears as emergent global behaviour (Dittrich et al. 2001). ACC has been previously used in two main applications: simulating complex systems (biological, social or ecological) and in developing novel solutions to engineering or computational problems. Its approach can be broadly categorised into microscopic or macroscopic methods (Dittrich 2005). Microscopic methods

treat each molecule explicitly, while in macroscopic methods, all the molecules of one type are represented by a value signifying, for example, concentration. Microscopic ACCs tend to model dynamics as stochastic molecular collisions, while macroscopic models tend to use continuous differential or discrete difference equations. The ARN is a macroscopic ACC and has a networked representation similar to other ACC models (Zeigler and Banzaf 2000; Eikelder et al. 2009). For instance, in the Artificial Biochemical Neuron, concentrations of reactants form weighted links between reactions and their dynamics are modelled using Ordinary Differential Equations (ODEs) (Eikelder et al. 2009). Other related chemically inspired approaches can be found in the literature, for instance, the Gene Regulatory Network algorithm (Guo et al. 2009), the Digital Hormone System (Shen et al. 2004), the Artificial Homeostatic Hormone System (Hamann et al. 2010) and idiotypic Farmer based Artificial Immune Systems (Krautmacher and Dilger 2004). Like these models, the ARN represents molecular species as continuous concentrations where dynamics are modelled using ODEs. Its networked representation is specifically designed to represent "biological circuitry" and allows temporal and spatial dynamics to unfold real-time. As discussed later, it has properties in common with other models from both Artificial Intelligence and Systems Biology fields, including: Artificial Neural Networks, Random Boolean Networks, Petri Nets, and S-Systems.

The specific objectives of the results presented here are as follows. Firstly, to explore the mechanisms and computational properties that leads to emergent high-level behaviour in cells. Secondly, to further investigate applications of this technique- specifically the control of motion in limbed robots. In this paper the following novel work is presented: 1) A complete overview of the ARN including its development, computational properties, advantages and disadvantages; and 2) the production of a complete ARN based control system for a limbed robot which combines pattern recognition and generation of time varying waveforms in a single network.

The paper is structured as follows: section 2 discusses the ARNs development, representation, advantages and disadvantages. The ARN is then used to explore several computational aspects of Cellular Intelligence. The first of these is its pattern recognition capability (section 3.1). In these experiments ARN parameters are set using a Genetic Algorithm (GA) and input patterns representing external environmental chemicals are mapped to output patterns. In section 3.2, further processing capabilities of the ARN are investigated by determining its ability to regulate complex temporal dynamics. Its application in robotic control is then explored by using the resulting system to create waveforms which control the gaits of limbed robots. This network is then extended into a complete control system by combining it with the previous pattern recognition network (section 3.3) in a single ARN. The results show that the ARN can function in both sensory input and motor output tasks which usually only more complex models can fulfil. Moreover the ARN allows offers increased flexibility over existing methods in robotic

control tasks. The report concludes that the ARN is a versatile and powerful technique which has application in both simulation of chemical systems, and in robotic control, where it can offer a higher degree of flexibility and computational efficiency than benchmark alternatives. Furthermore, it provides a tool which may possibly throw further light on the origins and limitations of the primitive intelligence associated with cells and its parallels with neural intelligence.

2 The Artificial Reaction Network

The ARN was briefly introduced in our previous work (Gerrard et al. 2011). This section provides a complete overview of the ARN starting with its basic formulation, and followed by its networked representation and computational properties.

2.1 Basic Formulation

Rate equation models can be used to represent many different physical systems and so are very general and flexible in their applications. In the domain of chemistry, they can directly represent (or be slightly modified to represent) all the common reaction types. They form the basis of S-systems (Savageau and Voit 1987) and are well characterised in biochemical simulations. The basic rate equation is described by Eq. (1) and is described by two terms. The first half corresponds to the rate of generation of product j (P_j) and is equal to the forward reaction rate (k_f), multiplied by the product of the concentrations of the N reactants ([R_n]), each raised to the power of its reaction order α_n.

$$\frac{d[P_j]}{dt} = k_f \prod_{n=1}^{N} [R_n]^{\alpha_n} - k_r \prod_{i=1}^{M} [P_i]^{\beta_i} \tag{1}$$

The second term represents the rate of decomposition of product back into its original reactants. This depends on the reverse reaction coefficient (k_r) multiplied by the product of the concentrations of the M products [P_i], each raised to the power of its reaction order β_i. For example, consider the simple reaction between two reactants labelled A and B with reaction orders of q and s respectively. These produce a single product P. In this case, Eq. (1) is reduced to Eq. (2).

$$\frac{d[P]}{dt} = k_f [A]^q [B]^s - k_r [P] \tag{2}$$

When used in S-systems, a group of rate equations are normally set up - one for each reaction. The left hand of each equation is then set to zero and they are solved simultaneously to yield the steady-state response. If the dynamic responses are required, then numerical solution methods like Runge-Kutta are normally applied.

2.2 A Networked Representation

Clearly a large set of simultaneous ODEs (Ordinary Differential Equations), in their basic mathematical form, limit the conceptualisation, visualisation and communication of complex topologies. Furthermore, in this form, each ODE term is tightly coupled, and is difficult to isolate and manage. Therefore, in order to create a connectionist representation with distinct biological processing units, capable of constructing complex biological circuits, the method needs to be modified. This may be done by isolating each reaction in the network to form a discrete node which may then be modified independently of the other reactions. Such a node can be viewed as analogous to a neuron in an ANN and has been named an Artificial Reaction Node; by analogy networks of such nodes may be termed Artificial Reaction Networks (ARNs). Similarly to an ANN, each ARN node is a processing unit, transforming a number of inputs into an output. In an interconnected network of such units, global behaviour is determined by the connections, and unit parameters. Furthermore, by isolating each reaction like this, the individual pathways or units which make up the system can be changed, reconnected or evolved by (for instance) a genetic algorithm. This also allows an individual part of the network to be independently modified and its effects studied. Such a feature is useful in simulating disease pathways. Isolating the reactions in this way facilitates two other important practical advantages. Firstly, visual “drag and drop” interfaces can be developed. These allow researchers to quickly change network or reaction parameters in order to study their effect. This, in turn, allows simple visualisation of the system in a graphical form which makes its conceptualisation easier. Secondly, it makes the application of object-orientated programming techniques very simple, as each node can be coded as an instance of an object.

$$\Delta[P] = (k_f[A]^q[B]^s - k_r[P])\Delta t \tag{3}$$

$$\Delta[A] = \Delta[P]q \tag{4}$$

$$\Delta[A] = \Delta[P]\frac{q}{q+s} \tag{5}$$

In developing the system described, Euler's method was chosen in order to solve the rate equations. This offers some advantages, in that it is simple and computationally cheap, but more importantly, it allows the whole network to run quickly in simulated real time- so that its temporal dynamics can be seen to unfold during a run. This gives the option of changing parameters in real time so that a user can observe any dynamic resulting behaviour. Furthermore, the temporal output of the network could potentially be used as a control system for an "artificial cell" robot - a cytobot.

Using the simple two input system shown in Eq. (2), multiplying through by dt and changing to a discrete finite time-step Δt, the Euler approximation is described by Eq. (3). This reaction needs to be isolated from the others, so that it can form a discrete "unit". This can be done most easily by borrowing the concept of "pools" from Petri-nets (Murata 1989). Petri-nets pass tokens between such pools as part of their operation. In the system discussed here, the pools may hold the number of available molecules, the concentration of the reacting chemicals (for example in moles per litre) or the mass of reactants. As the reaction proceeds, the reacting species pass from the input pools (depleting them) to the output pools (enriching them). So, in the previous example, to generate one molecule of product requires q molecules of reactant A and s molecules of reactant B. In this case, the pool containing A would get depleted by an amount ΔA as described by Eq. (4). Where ΔP is the amount of product generated (which would be added to pool P). This equation works if the units used are number of molecules or moles per litre (which are not conserved quantities). However if mass or a similarly conserved quantity is used then ΔA is given by Eq. (5). The whole system using more general symbols is shown diagrammatically in Fig. 1 (for a conserved quantity).

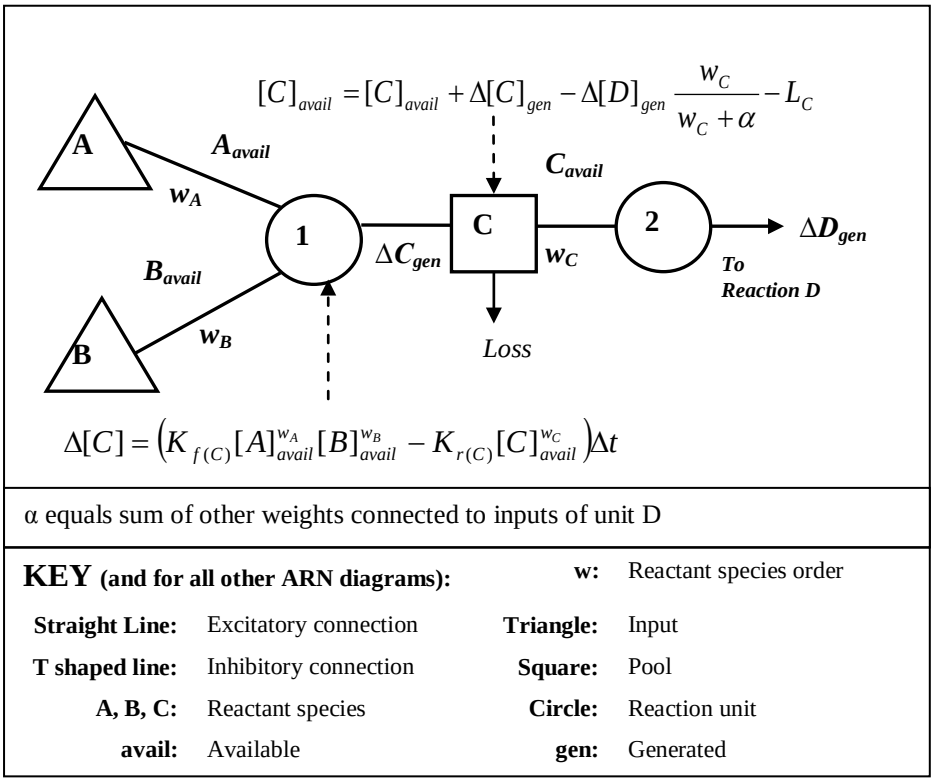


Fig. 1 Schematic diagram of a simplified Artificial Reaction Network (ARN). Reactant chemicals A and B react at unit 1. The rate of the reaction at unit 1 at time t is given by Eq. (3). The current concentration in pool C is updated using Eq. (5).

It comprises a set of connected reaction nodes (circles), pools (squares), and inputs (triangles). Each pool represents the current available protein species concentration (*avail*) in a compartment and each circle corresponds to a reaction unit, representing an interaction (reaction) between a numbers of chemicals.

The use of pools allows current concentration of species and their dynamics to be simply viewed. As a biological modelling tool, chains of pools could be used to represent gradients and translocation of species across membranes. Similarly, a loss component can also be added to the pools to represent the destruction of reactants or products by specific or general proteases or other degradation routes as shown in Fig. 1.

Connections symbolize the flow of species into and out of reaction units and their weight (*w*) corresponds to reaction order. The connections can be either excitatory, or inhibitory. A reaction with both excitatory and inhibitory connections will proceed if all connected inhibitory pools are empty and its excitatory connected pools have the required concentrations. Thus the input pools serve as pre-conditions which must be met before the reaction can proceed. The inhibitory connections act as discrete on/off switches to either the forward or reverse reaction.

The ARN has been extensively verified against standard methods of representing chemical systems (Gerrard, 2011), where it was shown to provide the same degree of accuracy as other ODE models.

2.3 Computational Properties

The overall structure may be compared to a perceptron, where the pools correspond to inputs, the reaction units to the weighted sum function, and these are joined together by weighted connections.

It is fairly easy to see that the computational properties of the ARN are similar to those of the ANN. For example, consider the simple network shown in Fig. 2.

$$[\dot{P}] = k_{Af}[A] + k_{Bf}[B] \tag{6}$$

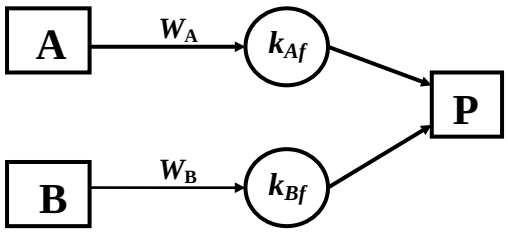


Fig. 2 A simple ARN network

If we assume that the orders W_A and W_B are unity and the reverse reaction rates are zero, then the rate of change of the product pool P is given by Eq. (6). Which is the same expression as for the activity of a perceptron if A and B are the inputs and the *k* terms the weights. So a network of such nodes has at least the same computational capabilities as MLPs (Rumelhart and McClelland 1986). In fact the addition of non-unity orders means that effectively the node can produce non-linear separators in a similar way to polynomial neurons (Woo and Khor 2004) and are rather similar to so-called “sigma-pi” units (Gurney 1992) - although with the added dimension of dynamic behaviour which will be discussed in section 3.2.

2.4 Disadvantages

There are some potential disadvantages associated with the ARN. Firstly, the Euler approximation has an associated cumulative error. This is because it is an iterative linear approximation to a complex function. It may be thought of as the first-order term of a Taylor expansion of the function. For example, if we say that the rate equation is a function of at set of

reactants and products **R**, we could write an abbreviated version of Eq. (1) as given by Eq. (7). The full Taylor series for Euler approximation to the second order would then be described by Eq. (8).

$$\frac{d[P]}{dt} = f(\mathbf{R}) \tag{7}$$

$$R_0 + f(\mathbf{R})\Delta t + \frac{\Delta t^2}{2!}\frac{d^2 f(\mathbf{R})}{dt^2} + \ldots \text{ and so on} \tag{8}$$

Because the series is truncated to the linear term the error of the approximation is the sum of the missing terms. In reality the error contribution from successive terms is usually negligible providing that the step-size is small. The error may be of consequence if the user is trying to simulate a complex biochemical system very accurately. However, as previously discussed, this is not the main purpose of the ARN.

Other difficulties can arise using hybrid models and detailed discussions are provided in the literature (Kowalewski 2002). Two such issues can occur in this representation where both produce an unnatural result due to the problem of trying to represent a discrete system (of individual molecules) by a continuous mathematical expression. In a real biological system there are a finite number of molecules and the chemistry acts the same way on all of these until they are exhausted. This however is not always the case when applying the governing equation (Eq. 3). For example, consider the case where the order of the reactants is above 1. When the current reactant concentration is above one this is fine, however if it is less than one then the resultant activity decreases unnaturally. In practice this is easily sorted by restricting the range of the concentrations or using different units.

A similar issue occurs where a pool, for example S, inhibits a reaction unit by an inhibitory connection. This reaction will always be inhibited while there remains any amount of chemical in S. Meanwhile S is involved in another reaction where the resultant flux is depleting S at each time step. As the concentration of S decreases so too does the flux. This leads to an infinite sequence of decreasing concentrations of S which asymptotically approaches zero. Therefore, S will always contain a smaller but positive value and as a result the inhibited reaction can never occur. In reality this would not occur since individual molecules would react in an individual manner. This problem is solved by simply setting a threshold- if a pool concentration is less than the threshold its concentration is set to 0.

3 Experiments and Results

In the following sections the methodology and results for the following experiments are presented: 1) An ARN based pattern recognition system; 2) the use of an ARN based system to regulate time varying waveforms and its application in control of limbed robotic gaits; and 3) an ARN which combines the previous networks into a complete quadrupedal robotic control system capable of recognising input patterns and generating the required gait response.

3.1 Pattern Recognition

A key mechanism of cell intelligence is the ability of a cell to recognise and respond to specific patterns of chemical signals within its environment. Receptors recognise and bind to particular environmental chemicals. These are transduced and cell response is determined by a chain of signalling events.

The ARNs pattern recognition capability was tested in both the context of a general pattern recognition device and in an abstract biological setting. In each case, 4 separate patterns composed of 4 input and 4 associated output mass values were applied to the ARN. Each pattern comprised values of either 0.1, representing low concentration, or 1 corresponding to high concentration. The ARN was set up as shown in Fig. 3 and consists of 7 pools, 4 inputs and 7 reaction units organised into 2 layers.

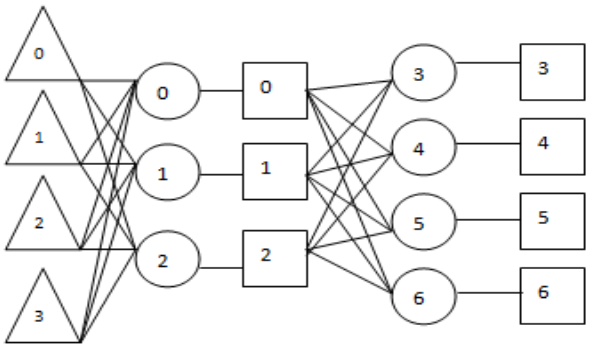


Fig. 3 The structure of the ARN used for pattern recognition experiments. The network consists of 4 inputs (triangles), 7 reaction units (circles) and 7 pools (squares). Each index of the input pattern array is fed into the corresponding input number. Output patterns are output at pools (squares) 3-6.

In biological CSNs, network parameters are determined by genetic factors which are subject to evolution. To achieve a related effect within this artificial setting a genetic algorithm (GA) was adopted to train the network to produce the correct output. The initial value of all pools was 0.01. Each input value of a pattern was fed into its corresponding input unit. For example, the first, second and third input value of pattern 1 is 0.1 and the fourth is 1 (see Table 1) - thus input unit 0-3 (see Fig. 3) were initialised to 0.1 and input unit 3 to 1. The output values were generated by the final layer of pools (3-6). The target output values for each pattern are given under the heading “Output” of Table 1, and the actual values associated after training are given

under “Actual Output”. A population of 100 solutions were randomly initialised. Each solution comprised a complete set of network parameters including the forward and reverse rates for each unit and the weights for each connection between pools (or inputs) and units. Due to its temporal properties the network was run for 100 cycles (a cycle ends when the complete set of pools in the network are updated once using Eq. (3) where $\Delta t = 1$) in order to obtain steady-state output values. The solution fitness was then calculated where fitness was the inverse of the error on output and the target error was 0.01. The least fit half of the population was discarded and the remaining solutions were subject to mutation and crossover in order to create the new population. To minimise the number of generations, the mutation and crossover rates were adjusted to final settings of 0.4 single point crossover and 10% uniform mutation. The average number of generations required to reach the target error was 387. The parameters of one solution are given in Table 2. The results from this general pattern recognition experiment are shown in Table 1. As can be seen the ARN was able to recognize all 4 patterns correctly.

Table 1 Patterns and results for both general and abstract biological setting experiments.

General Pattern Setting				**Abstract Biological Setting**			
Pattern	Input	Output	Actual Output	Pattern	Input	Output	Actual Output
1	0.1 0.1 0.1 1	0.1 0.1 0.1 0.1	0.1 0.1 0.1 0.1	1	1 (WR) 1 (SR) 0.1 (SA) 0.1 (WA)	1 (IS) 0.1(F) 1 (O) 0.1(DS)	1 0.1 1 0.1
2	1 0.1 1 0.1	1 1 1 0.1	1 1 1 0.1	2	0.1 (WR) 0.1 (SR) 0.1 (SA) 1 (WA)	0.1 (IS) 1 (F) 0.1 (O) 0.1(DS)	0.1 1 0.1 0.1
3	1 1 1 1	1 0.1 1 0.1	1 0.1 1 0.1	3	0.1 (WR) 1 (SR) 1 (SA) 0.1 (WA)	1 (IS) 0.1 (F) 1 (O) 0.1(DS)	1 0.1 1 0.1
4	1 0.1 1 1	1 1 1 0.1	1 1 1 0.1	4	1 (WR) 0.1 (SR) 0.1 (SA) 1 (WA)	0.1 (IS) 0.1 (F) 1 (O) 0.1(DS)	0.1 0.1 1 0.1
Key Inputs:	WR : weak repel		SR : strong repel		SA : strong attract		WA: weak attract
Key Outputs:	IS : increase speed		F : reorientation (up chemical gradient)		O : reorientation (down gradient)		DS : decrease speed

Table 2 Resulting network parameters for one solution after training using the genetic algorithm.

General Pattern Setting Parameters					
Pool	**Initial Concentration**	**Weight of Connection**	**Reaction unit**	**Forward Rate**	**Reverse Rate**
0	1st Pattern value (e.g. if pattern is no.1 input is 0.1)	2.999	0	0.723	2.816
1	2nd pattern value	-2.915	1	5.411	0.837
2	3rd pattern value	0.424	2	0.969	0.643
3	4th pattern value	-0.278	3	0.120	4.310
4	0.01	-1.714	4	1.003	1.455
5	0.01	0.750	5	0.093	0.006
6	0.01	-0.435	6	1.081	0.580
7	0.01	1.319	Note that in this case to simplify the program the hidden layer pool concentrations were updated using the unweighted flux of the product.		
8	0.01	-0.104			
9	0.01	0.501			
10	0.01	1.492			

Multilayer Perceptron ANNs (MLPs) (Rumelhart and McClelland 1986) have similar properties. For instance, each neuron can be approximated as either active or inactive and is comparable to the ARN whose concentration is either high or low. However, MLPs lack an explicit time dimension whereas the ARN processes inputs over a time period. In this case the ARN was subject to a continuous flux of inputs over 100 cycles causing the pool concentrations to enter a transient phase and stabilise at steady-state. The implications are that, unlike the MLP where processing is discrete-time, stored patterns are recalled only if inputs are applied for a length of time greater than that required to reach steady-state. Thus, this experiment demonstrates that the ARN is an appropriate pattern recognition technique when the requirement is to establish if a set of conditions have held true over a time period. This functionality is not so easily generated in other neural models. Discrete-time neural models provide a direct mapping from input to output and in their basic form they are unsuited for temporal pattern recognition. Continuous time models can provide this functionality but are generally more computationally complex. One such model is the Artificial Biochemical Network (ABN) (Macleod and Capanni 2010). It is a connectionist representation which, like the ARN, can be used to recognise continuous data streams. It has a weighted sum activation function combined with leaky integrator and generates a pulse width modulated output. In a similar experiment an ABN was setup using 11 ABN units. The network was trained using a GA to map identical sized patterns to those used here. Like the ARN, the ABN recognised all patterns, but the training time was longer (average of 496 generations) (Macleod and Capanni 2010) and the ABN network used 4 additional ABN units (1 ABN unit is approximately as complex as a single ARN unit).

In a further experiment, using the previously described network structure and set-up, the ARN was trained to recognize an additional 4 patterns, where the inputs were chosen to correspond to chemical signatures (for example, attractants or repellents) that trigger specific movement

responses. These patterns are given in Table 1. Here, the ARN network represents a highly abstracted CSN that controls chemotactic motion of a generalised single celled organism. This artificial amoeba is assumed to have a default slow swim behaviour and in the presence of chemoeffectors the behaviour is updated accordingly. Each input signifies an environmental chemical, where input: 0 is a weak repellent (WR), 1 a strong repellent (SR), 2 a strong attractant (SA) and 3 a weak attractant (WA). Specific combinations of environmental chemicals generate specific output response, where repellents have precedence over attractants. The presence of chemical concentration to a value approximate to 1, in an output pool, corresponds to a particular behavioural response, where output pool: 0 increases speed (IS), 1 reorientation to face up chemical gradient (F), 2 reorientation down chemical gradient (O) and output 3 decreases speed (DS). Therefore, as an example, on detecting both a strong repellent and strong attractant the cell re-orientates to face down the chemical gradient and increases speed. As can be seen in Table 1, the network generated the correct response for all the abstract biological patterns.

One property of a CSN is robustness, where correct response maybe generated in the presence of noise or loss of connections. In order to test this property within the ARN, random noise was introduced to the trained general pattern recognition network. Each pattern was subjected to 10% increments of uniformly distributed random noise to a total level of 60% of the input range. At each noise level outputs were obtained for all four patterns. It can be seen in the graph in Fig. 4 that the performance of the network gently degrades as noise is added. Error levels within 5% are reported for both the ABN and MLP models (Macleod and Capanni 2010) at levels of up to 50% noise in pattern recognition tasks of the same complexity.

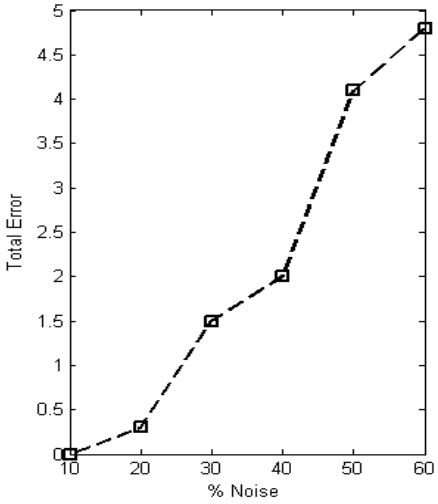


Fig. 4 Total error (y-axis) for all four patterns after introduction of random noise (x-axis) to patterns at 10% level increments

Similarly to an ANN, the ARN pattern recognition system, is a robust connectionist network and thus provides an intuitive bridge between biology and AI. Furthermore, this experiment illustrates that that such pattern recognition mechanisms are plausible in single celled organisms.

3.2 Regulation of Temporal Dynamics and Control of Limbed Robots

A common motif of CSNs is periodic oscillatory patterns of protein concentrations. Such patterns relate to particular cellular behaviours (Bray 1995; Ankers 2008; Kholodenko 2006). Many illustrations of these oscillatory patterns can be found within the literature (Ankers et al. 2008; Ferrel 2004).Such temporal dynamics are explored within the ARN in order to validate its ability to represent such patterns, to explore potential application, and to gain deeper understanding of the regulatory mechanisms involved within CSNs and their role in cell intelligence.

One method of exploring the ARNs ability to reproduce such temporal patterns, while investigating its potential AI applications, is by creating an ARN based controller to reproduce the patterns associated with robotic gaits. Terrestrial locomotion of limbed animals is achieved by multiple phase locked patterns of limb movements known as gaits. For example, depending on speed of locomotion and terrain, quadrupeds commonly walk, trot and gallop (Dagg 1973). The gait phase is a value that ranges from 0 to 1 as the gait cycle proceeds. Therefore, the motion of each limb can be described relative to the gait phase. The ideal quadrupedal gaits are described by Dagg (1973) and others (Hildebrand 1997) and are used as a standard for comparison here and similarly in other studies (Collins and Richmond 1994). In the walk gait the legs move a quarter cycle out of phase; in the trot gait each pair of diagonal limbs move half a cycle out of phase.

In these experiments, an ARN controller was implemented to generate gaits of a Lynxsmotion dual-servo quadruped 2 (Q2) robot. Each robotic leg is controlled by two servo motors, one for each degree of freedom (DOF). One motor raises the leg and the other turns it. The structure of the robotic legs is shown in Fig. 5, further details of which are given by Toth and Parker (2003).

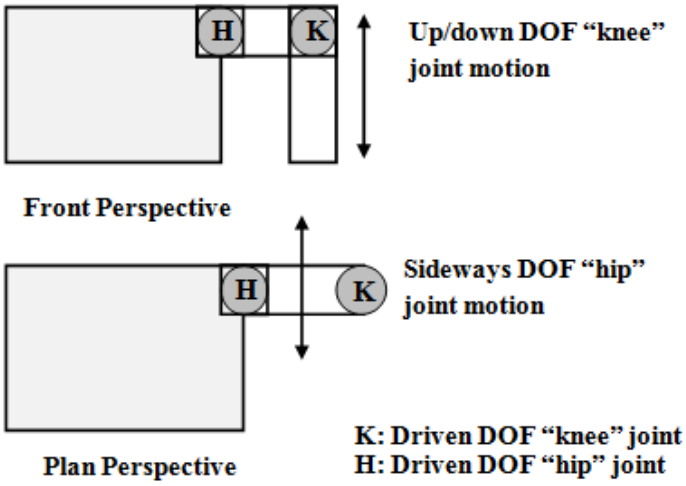


Fig. 5 The structure of a Lynxsmotion quadrupedal robot leg. Each leg has two DOFs and each DOF is controlled by a separate motor.

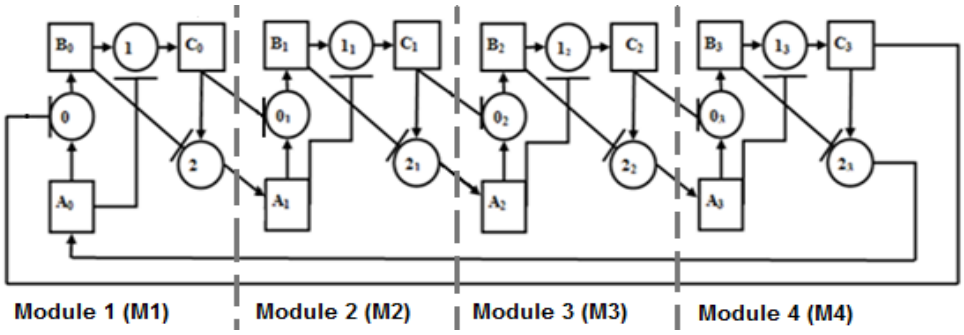


Fig. 6 The ARN based controller. Each module (shown separated by a dashed line) is mapped to a single leg and is responsible controlling the 2 motors which generate its motion. Pool A of each module controls the up/down motor, pool B the back/forward motor and pool C the stop period for each of these motors.

Signals are sent by the ARN to each motor and control the angle of the rotor for each DOF, using a simple position to pulse width modulator interface circuit to control the servo. The structure of the ARN based controller is shown in Fig. 6 and was designed to include abstractions of regulatory mechanisms found in CSNs including inhibitory/excitatory reactions, cyclic loops, and feedback structures. The controller comprises a network of four repeating structural units or modules, where a module is separated by a dashed line. Each module controls the two motors of a separate leg and comprises 3 reaction units and 3 pools: A, B and C. Pool A controls the up/down (U/D) motor, Pool B the back/forward (B/F) motor and Pool C controls the off period for both motors. Pool activity is regulated by a series of excitatory and inhibitory connections between reaction units. The type of connection represents the inhibitory and excitatory properties of specialized regulatory proteins common to CSNs such as enzymes. The overall network structure is organized as a closed loop allowing protein species to be recycled to the first module and thus generate a temporal oscillatory pattern. The structure of

the ARN controller is capable of producing all the common gaits. The type of gait is easily modified by a simple adjustment of the initial pool values. For example, by initializing a C pool a walk gait will be generated, where the C pool chosen will determine the starting leg, and the value determines the angle to which the leg is raised. Similarly, a trot gait is achieved by initializing two C pools within alternate modules. In this particular design, the value to which the C pool(s) are initialized determines the DOF angle and were set specifically for the physicality of the particular robot although it can be freely varied.

The network architecture remains fixed throughout these experiments and the network parameters are manually set. This method was employed so that the outputs could be directly compared with other published work on similar Central Pattern Generators (CPGs) (Billard and Ijspeert 2000; Collins and Richmond 1994; Liu et al. 2009). However, there is no reason why connection weights cannot be set using an Evolutionary Algorithms or a similar pseudorandom search technique, and the current authors have employed this in other examples (for instance in the pattern recognition experiments). The use of gradient decent algorithms, however, would be difficult in this application because of the recurrent nature of the network topology. The ARN controller was considered to generate a specific gait if the relative phases of the respective oscillatory signals were within 2% of the standard gait cycle described previously.

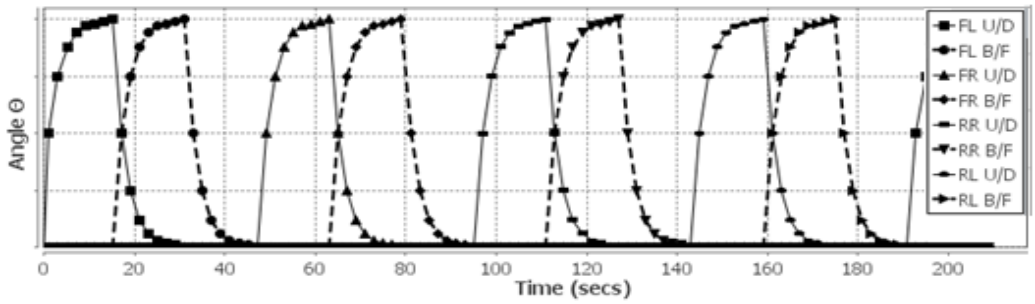


Fig. 7 Output generated for the walk gait. Legs are front left (FL), front right (FR), rear right (RR) and rear left (RL). The up/down (U/D) motor is displayed as a solid line and the back/forward (B/F) motor is displayed as a dashed line.

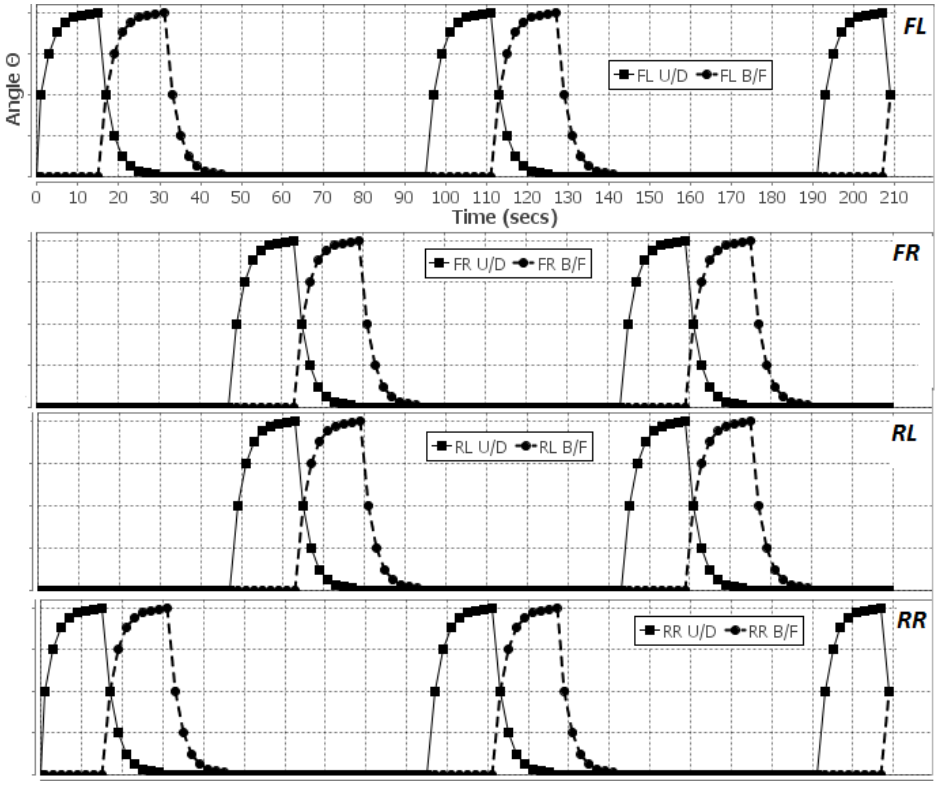


Fig. 8 Output generated for the trot gait. Legs and motors are labelled as before- see Fig. 7.

Higher values of 10% were used in other studies (Collins and Richmond 1994), and this was considered reasonable due to the variation found in real animal gaits (Afelt 1983). In each case, the controller first generates the U/D motor oscillation and on reaching the maximum value the B/F motor is initiated.

As can be seen in Fig. 7, the walk gait results show that the legs are a quarter cycle out of turn, with phases of 0.0, 0.25, 0.5, 0.75 between limbs in clockwise order from FL (front left) leg. Similarly, the trot gait results in Fig. 8 show that the opposite legs are half a cycle out of turn with phases respectively of 0.0, 0.5, 0.0, 0.5. The frequency of oscillations and therefore the gait speed is easily adjusted by applying uniform increase or decrease to k_f of each unit.

Both phase locked limb patterns produced by the ARN match the standard, and compare well with other connectionist models. For example, Billard and Ijspeert present a CPG (Central Pattern Generator) based neural controller for a quadrupedal AIBO robot, similarly with 2 DOFs for each leg (Billard and Ijspeert 2000). The limb phases generated by this network correspond to the standard and to those produced by the ARN. Here, the network is composed of 8 coupled non-linear oscillators and each oscillator consists of 6 leaky integrator neurons (a total of 96 neurons). Each neuron implements an activation function which is approximately as complex as the reaction unit function of the ARN, and therefore the complexity of the network is equivalent to approximately 96 ARN reaction units. Similar correspondence is found in other sources. For instance, Collins explores a CPG based neural controller for a quadrupedal robot

with 1 DOF per limb, and compares 3 types of activation function models: Stein, Van der Pol, and FitzHugh-Nagumo. The controller is composed of a network of 4 coupled non-linear oscillators (Collins and Richmond 1994), where each oscillator controls a separate limb. The Stein model consist of 3 first order differential equations, the Van der Paul model consists of a second order differential equation and the FitzHugh-Nagumo model consists of two first order differential equations. All these models have approximately twice the complexity as the output produced by the ARN unit. In this case all 3 models require a pulsing signal to drive the network. Generally speaking the structure of these models is less flexible then either the Billard and Ijspeert (2000) model or the ARN due to their rigidly fixed internal parameters. All these models produced the gait patterns within 10% of the standard, whereas the ARN matched the standard for both trot and walk.

Overall the ARN has a very similar capacity to generate both walk and trot gaits as the compared controllers. However, in general, it affords a higher degree of flexibility and is less computationally complex. Although robotic gaits might seem unconnected with cellular intelligence, the ARNs ability to produce them illustrates how cellular networks can generate the complex temporal patterns necessary in emergent behaviour.

3.3 Complete Robotic Control System

It was demonstrated in section 3.1 that an ARN can recognize patterns. Furthermore it was demonstrated in section 3.2 that such a system can generate temporal output patterns which can be used in control tasks. Of course in the natural world these two behaviors are linked together.

In the following experiment it is illustrated that both pattern recognition and control function can be combined within a single ARN based system. Here, a more complex ARN was created to recognize specific patterns and in response automatically generate the associated temporal gait. The ARN in this experiment reuses the pattern recognition and gait network previously described in sections 3.1 and 3.2 respectively. The complete ARN system is shown in Fig. 9, and is functionally divided into 3 smaller ARN components: pattern recognition, control, and a connecting network.

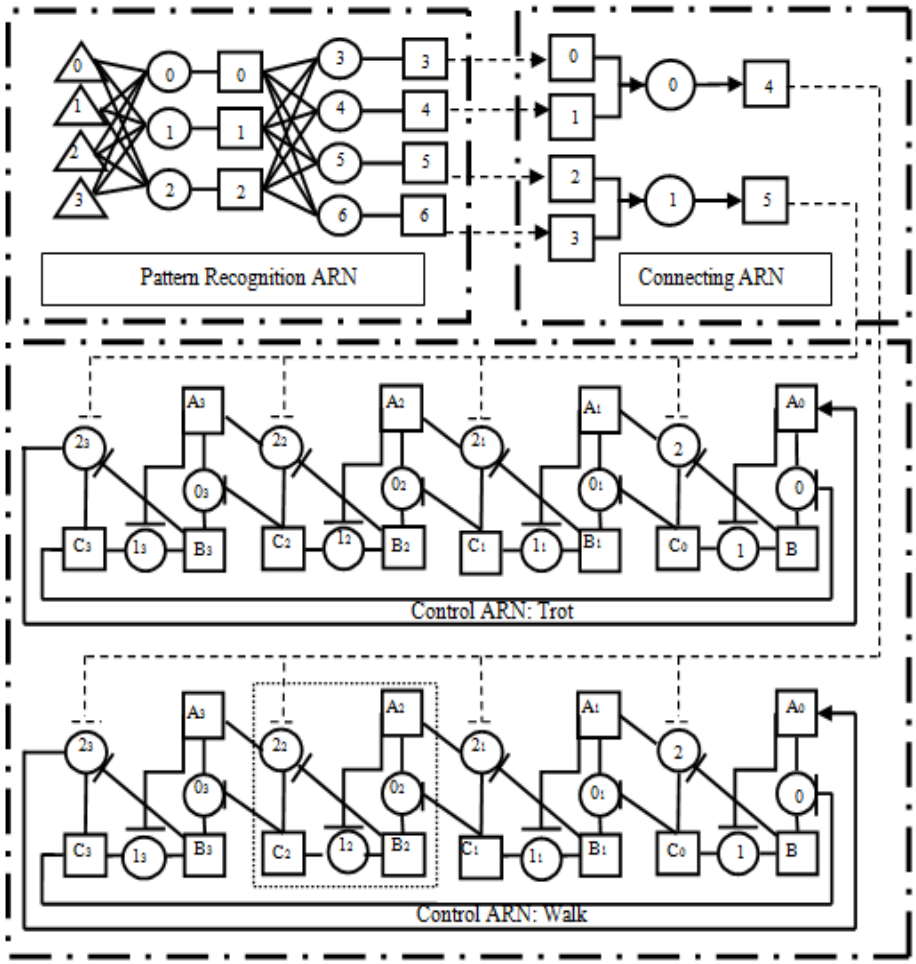


Fig. 9 A complete control system for a quadrupedal robot. On recognition of particular patterns the pattern recognition ARN generates the associated output pattern. The connecting network implements two parallel Boolean AND gates which act as switches turning the walk or trot components of the control ARN off/on. The control ARN generates the required waveform which controls the robotic gait.

The structure of the pattern recognition network, its implementation, and training methods are identical to those described in section 3.1. In this case the network was trained to recognize 3 separate patterns (as shown in Table 3) composed of 4 input and 4 associated output mass values. The output pools of the pattern recognition network are equal to the input pools 0, 1, 2, and 3 of the connecting network. The connecting module comprises 6 pools (4 inputs and 2 outputs) and 2 reaction units. Essentially this component operates like two parallel Boolean AND gates, where a value of 1 at pools 0 and pool 1 will output a value of 1 at pool 4, as will a value of 1 at pools 2 and 3 output a 1 at pool 5. Two negative feedback connections between the interface network and both ARN control system subunits (shown as dashed line connections) are responsible for switching between the gaits. Therefore, if a value of 1 is output at the interface network pool 4, it will inhibit all the reaction 2's of the ARN walk subunit, thus stopping the walk gait pattern from being generated. Conversely, if a value of 0 is output at pool 4 the walk will be generated. In the same way, pool 5 of the interface controls the

switching on/off of the trot control subunit. Table 3 shows the range of required behaviors in response to particular outputs generated by the connecting network.

The control system comprises two separate ARN subunits, both identical in structure and implementation to the ARN described in section 3.2. Each of these subunits is responsible for generating a specific temporal gait pattern: one generates walk the other trot. The two ARN subunits provide distinct gait patterns due to the differences in initialization of the concentration values of C pools.

Table 3 Patterns applied to the complete control system and expected gait generated.

Pattern	PR Network Input Pool No.	PR Network Input Value	CN network Input Pool No.	CN Input Value (& output of the PR network)	CN Output Pool No.	CN Output Value	Gait
1	0	1	0	1	4	1	Inhibit Walk
	1	0.1	1	1			
	2	1	2	0	5	0	Trot
	3	0.1	3	0			
2	0	0.1	0	0	4	0	Walk
	1	1	1	0			
	2	0.1	2	1	5	1	Inhibit Trot
	3	1	3	1			
3	0	1	0	1	4	1	Inhibit Walk
	1	0.1	1	1			
	2	0.1	2	1	5	1	Inhibit Trot
	3	1	3	1			
KEY:	PR	Pattern recognition		CN	Connecting network		

Table 4 Pattern applied to the network and expected durations of gaits.

Pattern	Walk ARN Network	Trot ARN Network	Start Time	End Time	Duration
2	On	Off	0	210	210
1	Off	On	210	440	230
2	On	Off	440	560	120
1	Off	On	560	700	140
3	Off	Off	700	800	100

The complete system was tested to confirm its ability to both generate the correct behavior and automatically transition between the behaviors in response to firing input patterns 0-3. The time periods in which patterns were applied, and the expected output states are shown in Table 4.

The results for this experiment are displayed in Fig. 10. The phases produced for each gait are exactly as described previously in section 3.2. The on/off periods of both trot and walk gaits are in agreement with the expected durations displayed in Table 4 with a slight transitional delay. The ARN controller, and gait phases produced have previously been compared with CPG models in section 3.2. The transitions between gaits generated by these models may now be compared with those of the ARN. The results given for the Billard and Ijspeert model, show transitions from walk to gallop in approximately 4 leg cycles, whereas the ARN transitions from walk to trot within 2 leg cycles. In both cases the transitions are very smooth. There are 3

models described by Collins, and although gait graphs are provided for all these, gait transitions are only given for the Stein model. Here gaits transition quickly within approximately 2 leg cycles. However the leg movements during transition are very irregular- in contrast to the ARN and the Billard and Ijspeert model.

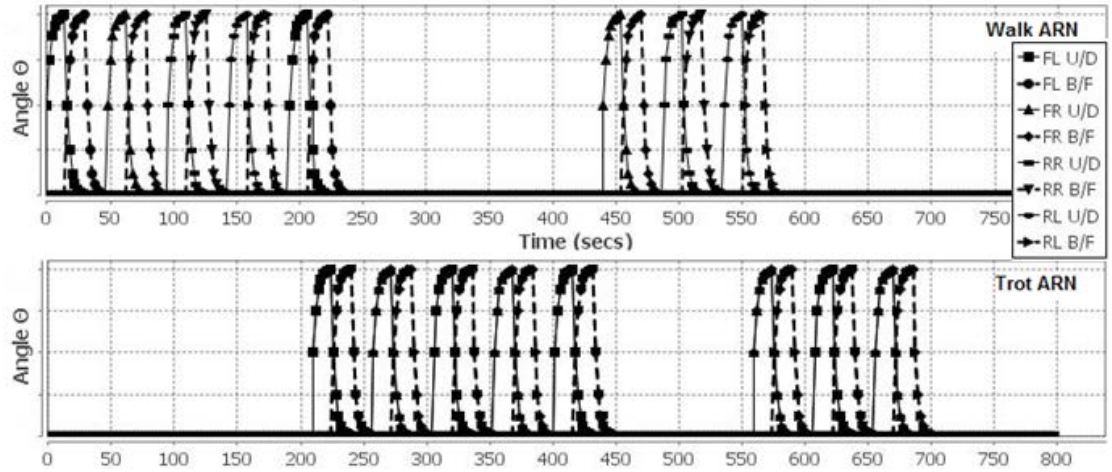


Fig. 10 The output of the complete ARN control system over 800 secs. Legs and motors are labelled as before- see Fig 9.

This complete control system demonstrates that the ARN, like a CSN, is capable of both recognizing patterns and controlling overall behavior in a single network. With the exception of spiking models, few ANNs can achieve this functionality. However, spiking models are often less flexible. For example, in the Integrate and Fire model information is rate coded and all the spikes generated are uniform (Maass 1997). Thus, unlike the ARN this model lacks the flexibility to produce pulse-width and pulse-amplitude coded information. The gait phases and transitions compared well with CPG neural controllers and showed that the ARN has application in similar robotic control tasks and can offer lower computationally complexity. These experiments illustrate how a CSN might perform the complex processing associated with the high-level behaviors displayed by single celled organisms. Furthermore, it shows that abstractions of both neural networks and CSNs operate in similar ways, and have comparable functionality. This illustrates a close relationship between neural and cell intelligence.

6 Conclusions

The ARN is a new connectionist model, based on the dynamics of CSNs. It is accurate enough to represent actual chemical concentrations in the cytoplasm of a cell, but simple enough to construct biological circuitry with applications in AI. Perhaps most importantly it is a useful tool for investigating the surprising emergent behaviours of single cells. It may help to elucidate the mechanisms involved in these, and their similarities and differences from neural

based intelligence (and intelligence in its widest philosophical sense). Although other researchers have used techniques such as S-systems and Petri Nets to do related work, the ARN is unique in that it was conceived as a much more connectionist, unit-based representation, designed specifically to investigate cell intelligence.

The results presented above show that the ARN (and by extension, cellular networks) are capable of performing pattern recognition in a similar way to artificial neural models and also producing complex temporal dynamics reminiscent of spiking neural models. Additionally, it was shown that the ARN can model biological reactions and simulate real CSN pathways with an accuracy matching those of standard simulation methods (Gerrard et al. 2010). This combination of attributes makes it a unique and useful tool. The ARN systems presented above show clearly that biochemical networks are quite capable of producing many of the behaviours normally ascribed to neural networks. This helps to illuminate the many interesting results now emerging from the behavioural biology of single cells. Of course the neuron is itself a biochemical network, and one future application of the ARN may be to help unravel its more complex internal dynamics.

The simplicity of the ARN makes it a potentially useful model in more practical AI and engineering systems. As demonstrated in the case of robotics, its ability to function in both input (sensory) and output (efferent or motor) networks and in the interconnection between these, gives it applications which usually only much more complex models can fulfil. This is particularly useful in the field of robotics, where such flexibility has particular application in evolutionary control networks.

The authors intend to extend the work reported here by producing more complex cell based robots (cytobots). These will allow us to explore more aspects of cellular intelligence (for example the role of learning in these systems) as well as some practical applications such as vehicles to clear oil spills- pollution by moving along chemical gradient, rather like that in which chemotaxis operates. The ARN may also have useful applications in other areas of science – for example in modelling the complex interconnected chemical networks present in environmental and soil chemistry.

10 References

Afelt Z, Blaszczyk J, Dobrzecka C (1983) Speed control in animal locomotion: transitions between symmetrical and nonsymmetrical gaits in the dog. Acta Neurobiol Exp, 43:235-250

Ankers J, Spiller D, White M, Harper C (2008) Spatio-temporal protein dynamics in single living cells. Curr Opin Biotechnol 19:1-6

Arkin A, Ross J (1994) Computational functions in biochemical reaction networks. Biophys J 67:560-578

Bhalla U (2003). Understanding complex signaling networks through models and metaphors. Prog Biophys Mol Bio 81:41-65

Billard A, Ijspeert A (2000). Biologically inspired neural controllers for motor control in quadruped robot. In: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, Italy: IEEE, pp 637-641

Bray D (1995). Protein molecules as computational elements in living cells. Nature. 376(6538):307-12

Collins J, Richmond S (1994) Hard-wired central pattern generators for quadrupedal robots. Biol Cybern 71:375-385

Dagg A (1973) Gaits in mammals. Mammal Rev 3:135-154

Dittrich P (2005) Chemical computing. Unconventional programming paradigms. Springer Berlin Heidelberg, pp. 19-32.

Dittrich P, Ziegler J, Banzhaf W (2001) Artificial chemistries-a review. Artif. Life 7(3):225-275

ten Eikelder T, Crigins S, Steijaert M, Liekens A. Hilbers P (2009). Computing with feedforward networks of artificial biochemical neurons. In: Proceedings of the 2nd International Workshop on Natural Computation, Japan: Springer: vol 1 pp 38-47.

Eungdamrong N, Iyengar R. (2004) Modeling cell signaling networks. Biol Cell 96:355–362

Ferrell J (2004) Self perpetuating states in signal transduction: positive feedback, double negative feedback and bistability. Cur Opin Cell Biol 14(2):142-148

Ford B (2009) On intelligence in cells: The case for whole cell biology. Interdisipl Sci Rev 34(4):350-365

Gerrard C, McCall J, Coghill G, Macleod C (2011) Artificial Reaction Networks. In: Proceedings of the 11th UK Workshop on Computational Intelligence, Manchester: UK, pp 20-26

Gerrard C, McCall J, Coghill G, Macleod C (2012a) Temporal patterns in Artificial Reaction Networks. In: Proceedings of The 22nd International Conference on Artificial Neural Networks Lausanne, part 1, vol. 7552, pp 1-8

Gerrard C, McCall J, Coghill G, Macleod C (2012b) Adaptive Dynamic Control of Quadrupedal Robotic gaits with Artificial Reaction Networks. In: Proceedings of the 19th International Conference on Neural Information Processing Doha, vol. 7663, part 1, pp 280-287

Guo H, Meng Y, Jin, Y (2009). A cellular mechanism for multi-robot construction via evolutionary multi-objective optimization of a gene regulatory network. BioSystems, 98(3):193-203.

Gurney K (1992). Training nets of hardware realizable sigma-pi units. Neural Networks 5:289-303

Hamann H, Stradner J, Schmickl T, Crailsheim K. (2010). A hormone-based controller for evolutionary multi modular robotics: From single modules to gait learning. In: Evolutionary Computation (CEC) IEEE, pp 1-8

Helikar T, Konvalina J, Heidel J, Rogers J (2008). Emergent decision making in biological signal transduction. P Natl Acad Sci USA 104:1913–1918

Hellingwerf K (2005). Bacterial observations: a rudimentary form of intelligence. Trends Microbiol 13:152-158

Hild W, Pollinger K, Caporale A, Cabrele C, Keller M, Pluym N, Buschauer A, Rachel R, Tessmar J, Breunig M, et al. (2010) G protein-coupled receptors function as logic gates for nanoparticle binding and cell uptake. P Natl Acad Sci USA 107:10667–10672

Hildebrand M (1997). Analysis of asymmetrical gaits. J Mammel 58: 131-156

Kholodenko B (2006). Cell Signaling dynamics in Time and Space. Nat Rev Mol Cell Biol 7(3):165-176

Kowalewski S (2002) Modeling, Analysis and Design of Hybrid Systems. Lect Notes Contr Inf 279:153-171

Krautmacher M, Dilger W (2004) AIS based robot navigation in a rescue scenario. In: Springer Lecture Notes Computer Science 3239, pp 106-118

Liu C, Chen Y, Zhang J, Chen Q (2009) CPG driven locomotion control of quadruped robot. In: IEEE International conference on Systems, Man and Cybernetics, San Antonio, pp 2368-2373

Maass W. Networks of spiking neurons: the third generation of neural network models. Neural networks 10, no. 9 (1997): 1659-1671.

MacLeod C, Capanni N (2010). Artificial biochemical networks: a different connectionist paradigm. Artificial intelligence review 33(1-2):123-134.

Saigusa T, Tero A, Nakagaki T, Kuramoto Y (2008) Amoebae Anticipate Periodic Events. Phys Rev 100(1):1-4

Savageau M, Voit E (1987) Recasting Nonlinear Differential Equations as S-Systems: A Canonical Nonlinear Form. Math Biosci 87:83–115

Savageau M. (1988) Introduction to S-systems and the underlying power-law formalism. Math Comput Model 11(1):546-551

Shen W, Will P, Galstyan A., Chuong C (2004). Hormone-inspired self-organization and distributed control of robotic swarms. Autonomous Robots 17(1): 93-105.

Stadtman E, Chock P (1997) Superiority of interconvertible enzyme cascades in metabolic regulation: analysis of multicyclic systems. P Nat Acad Sci USA, 74:2766–2770

Toth D, Parker G (2003) Evolving Gaits for the Lynx motion Hexapod II Robot. In: Proceedings of the 7th World Multiconference on Systems, Cybernetics, and Informatics, Orlando, USA pp 229-234

Wang B, Kitney R, Joly N, Buck M (2011) Engineering Modular and Orthogonal genetic logic gates for robust digital-like synthetic biology. Nat Commun 2:508

Woo W, Khor L (2004) Blind restoration of nonlinearly mixed signals using multilayer polynomial neural network. IEEE Proc Vis Image Signal Proc 151(1):51–61

Ziegler J, Banzhaf W (2000). Evolving a" nose" for a robot. In: Proceedings of the Genetic and Evolutionary Computation Conference (GECCO-2000).

Paper 3: ‘Artificial Reaction Network Agents’

Reference:

Gerrard, C, McCall, J., Coghill, G., Macleod, C. (2013) ‘Artificial Reaction Network Agents’, *The 12th European Conference on the Synthesis and Simulation of Living Systems (ECAL)*, Advances in Artificial Life, MIT press, Taormina, pp. 957-964.

Artificial Reaction Network Agents

Claire E. Gerrard[1], John McCall[1], Christopher Macleod[1], and George M. Coghill[2]

[1]IDEAS Research Institute,
Robert Gordon University,
Aberdeen, Scotland, UK
[2] Department of Computing Science,
University of Aberdeen,
Aberdeen, UK, Scotland
c.e.gerrard@rgu.ac.uk, j.mccall@rgu.ac.uk, g.coghill@abdn.ac.uk, chris.macleod@rgu.ac.uk

Abstract

The Artificial Reaction Network (ARN) is an Artificial Chemistry representation inspired by cell signaling networks. The ARN has previously been applied to the simulation of biological signaling pathways and to the control of limbed robots. In this paper we create multiple cell-like autonomous agents using ARN networks. It is shown that these agents can simulate some aspects of the behavior of biological amoebae. To demonstrate practical applications of such agents they are then reconfigured as a swarm of robots in a simulated oil spill clean-up operation. We demonstrate that ARN agents, like amoebae, can autonomously recognize environmental patterns and produce emergent behavior. The results show that such agents may be useful in biological simulation and furthermore may have practical applications in swarm robotics.

Introduction

Unicellular organisms have evolved an astonishing array of complex behaviors. Some can avoid light with photo-sensitive spots; some actively hunt prey; while others can build protective shelters (Ford, 2009). It has been shown that single cells achieve such primitive intelligence by storing and processing information through the complex dynamics of interacting chemicals (Bray, 1995; Arkin and Ross, 1994). Within a cell, data is represented by a set of spatially distributed concentrations of chemical species; the instantaneous set of which corresponds to the cell's current state. Intricate networks of chemical reactions termed cell signaling networks (CSNs), process this information by transforming input species into output species. In this way, cells are able to respond to changes within their environment, communicate with other cells, and perform internal self maintenance operations. Several researchers highlight the processing capabilities of CSNs and their similarities to Artificial Neural Networks (ANNs) (Bray, 1995; Bhalla, 2003). For example, it has been demonstrated that a network of such reactions can perform Boolean and Fuzzy Logic functions and are equivalent to a Turing machine (Bray, 1995; Arkin and Ross, 1994). Furthermore, CSNs contain features such as feedback loops and interconnectivity, thus forming highly complex systems (Bray, 1995; Bhalla, 2003). It is possible to exploit computational features of such chemical processing to create an Artificial Chemistry (AC). In its broadest sense, an AC describes a man-made system which is similar to a real chemical system (Dittrich, et al., 2001). The Artificial Reaction Network is an example of an AC and is based on properties and mechanisms found in CSNs. In our previous work, it was applied to simulate the chemotaxis signaling pathway of *Escherichia coli*, and later investigated as a means to produce complex temporal waveforms to control limbed robots (Gerrard, et al., 2011; 2012a; b).

In this paper, a single ARN network is instantiated and used as the internal control system for multiple instances of cell-like autonomous distributed agents. Our first objective is to show that ARN agents have application in the simulation of biological cells, their interactions, and resulting emergent behaviors. This is addressed by using the agents to simulate aggregating cells of the slime mould *Dictyostelium discoideum* and comparing the emergent behaviors with the literature. Our second objective is to show that by reconfiguring the inputs to each agent's ARN, the same agents can produce other distinct behaviors. Our final objective is to show that ARNs have application as the control systems for distributed robotic agents within real world environments. Here, we apply the agents to the task of cleaning up a simulated oil spill within a simplified search environment.

The paper is structured as follows: the first section provides an overview of the ARN representation. This is followed by an overview of the operation of the ARN agents. The experimental details and results are presented next; these are followed by the conclusions.

The Artificial Reaction Network

In this section we provide a brief summary of the ARN representation. A full account can be found in our previous work (Gerrard, et al., 2011; 2012a; b).

The ARN comprises a set of networked reaction nodes (circles), pools (squares), and inputs (triangles) as shown in figure 1. Each pool stores the current available chemical

species concentration (avail); this concentration represents data within the system. Thus, the complete set of pool concentrations at time t, corresponds to the system's current state. Inputs are a special type of pool, the only difference being that they are not updated by flux at each time step, and are used to represent continuous concentrations, for example, environmental inputs or enzymes. Each circle corresponds to a reaction unit, representing a reaction between a number of chemicals. Data is processed by reaction nodes transforming incoming pool values to connected outgoing pool values. Connections symbolize the flow of chemical into and out of reaction units and their weight (w) corresponds to reaction order. Connections provide the facility to create complex control structures using combinations of inhibitory and excitatory connections.

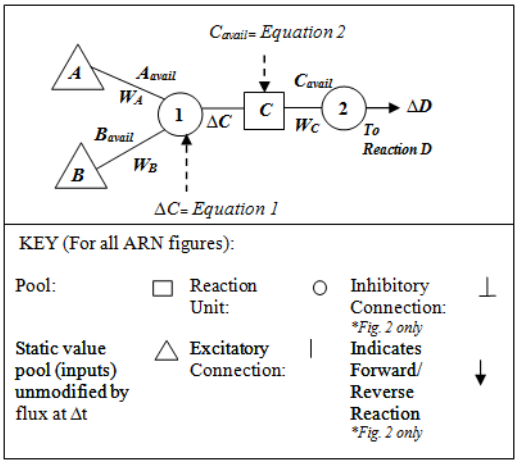


Figure 1: The Artificial Reaction Network representation.

$$\Delta C = \left(K_{f(C)} \left(A_{avail}^{W_A} B_{avail}^{W_B} \right) - K_{r(C)} \left(C_{avail}^{W_C} \right) \right) \Delta t \quad (1)$$

$$C_{avail} = C_{avail} + \left(\Delta C - \frac{W_C}{W_C + \alpha} \Delta D \right) \quad (2)$$

Where:
A, B, C, D = Species Concentrations
W = Reaction order (weight)
avail = Available species concentration
K_f = Forward rate constant
ΔC = Change in species concentration C
K_r = Reverse rate constant
α=sum of other incoming weights

Figure 1 shows the reaction between species A and B to produce species C. At time interval Δt, each reaction unit's temporal flux value is calculated by applying Euler's approximation to the differential rate equation given in (1). This value is then used to update the current concentration of each reaction's connecting pools as shown in (2). Pools may asymptotically approach 0, and thus below a particular threshold a pool is considered empty and its value set to zero. A reaction step may proceed if it meets its preconditions. Preconditions are met if incoming inhibitory pools are inactive, and incoming excitatory pools are active. Similarly a reaction step will fulfill a number of post conditions: participating reactants are consumed and products generated- the amount of which will depend on the parameters of the reaction step.

ARN Agents

This section describes the behavioral modes of each agent and the structure and operation of the ARN network controlling them. In the experiments outlined in this paper, a number of autonomous ARN controlled software agents termed "Cytobots" ("cyto" from Greek for cell, and "bot" from robot) are initialized and move around asynchronously within a 2D simulated environment containing a distribution of artificial chemicals. The artificial chemicals represent attractants of either food or cAMP (cyclic adenosine monophosphate). When an agent moves to a new position, the surrounding level of chemical is used to set the inputs to its ARN. Consequently this changes the internal state of the ARN and updates the agent's trajectory. During this process, the agent modifies the state of the environment by, for example, consuming food or releasing cAMP. Similar to the way in which a CSN acts as the control system to a cell, the behavior of each cytobot is controlled by its own instance of an ARN network. The ARN network architecture is based on a combination of functional structural motifs found in actual biochemical networks (Tyson and Novak, 2010). Each ARN instance is updated asynchronously with all other instances. In this way, each instance directs an agents' movement asynchronously to other agents, enables it to react to situated environmental patterns, and allows it to stigmergically communicate with other cytobots to contribute to higher level function.

The cytobot ARN network was designed to produce two simple behavioral modes: foraging and starvation, both are based on the movement patterns of single celled organisms as described in the following sections. The cytobot ARN is composed of 6 subnetworks as shown in figure 2. Each subnetwork contributes a functional aspect to either or both starvation and foraging behaviors. The subnetworks are discussed in the following sections.

Cytobot Foraging Behavior

Cytobots forage by performing a biased random walk. This pattern of movement is exemplified by the bacteria *E. coli*, where foraging cells alternate periods of runs (forward motion) and tumbles (random redirections). By comparing concentrations of attractants and repellants in a temporal fashion, the organism is able to reduce the frequency of tumbles up concentration gradients of attractants, and down gradients of repellants, resulting in overall travel to more favorable conditions (Vladimirov and Sourjik, 2009). In the foraging mode a cytobot performs a similar random biased walk movement pattern. At each new position X, an agent redirects to a new random angle between 0 and 360 degrees (tumble). The agent then moves forward in a straight line for a number of time steps based on the level of detected food at position X (run). The cytobot consumes food (if present) at each passing location.

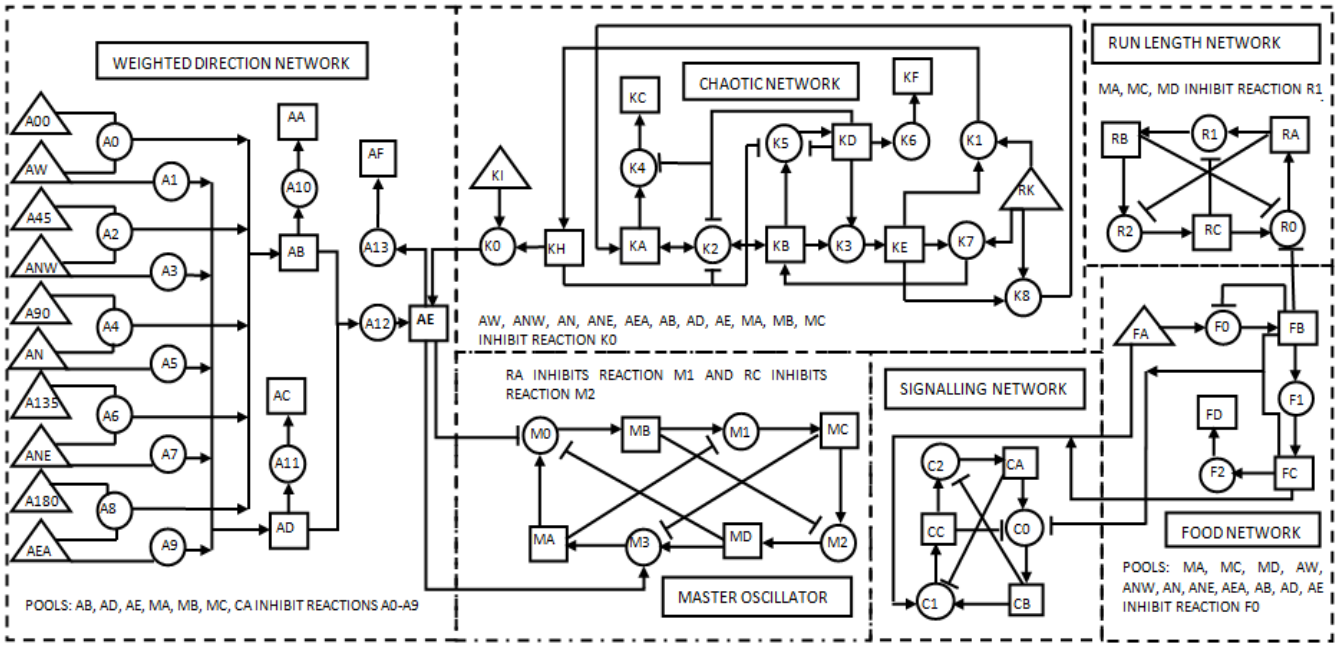


Figure 2: The cytobot ARN network comprising 6 subnetworks. Each cytobot is controlled by an instance of this network.

Cytobot Starvation Behavior

The starvation behavior is based on the pattern of motion displayed by starving cells of the cellular slime mould *D. discoideum*. During the organisms' vegetative stage, cells move up gradients of folic acid secreted by its bacterial prey. When the food resource has been depleted, the amoebae begin to starve and enter the aggregation phase of their life cycle. During aggregation, starving cells secrete cAMP which serves as a signal to attract surrounding amoebae towards a central location (McCann, et al., 2010). During the aggregation phase, *D. discoideum* cells are polarized, thus one side becomes the leading edge which always faces in the direction of travel (McCann, et al., 2010). Depending on parameters such as environmental conditions, and the cell population density, migrating cells often form transient emergent patterns such as streams, waves and spirals (McCann, et al., 2010; Dallon and Othmer, 1997). Streaming describes a pattern of motion where cells line up in close order files, with the head of one following the rear of another (McCann, et al., 2010).

In these experiments the agents enter starvation mode if food has not been consumed within a time period. Here, instead of turning in a random direction, the new direction is weighted toward the highest concentration of cAMP within its surrounding area. As discussed later, by representing the external chemicals in different ways within the simulated environment, different high level behaviors can be produced by the agents.

The Master Oscillator

The master oscillator network (see figure 2) functions to synchronize all the outputs from all the other subnetworks together and is what each agent references at each time step to ascertain its current behavior. It is a simple closed loop, with a token unit of chemical cycling around it. It consists of 4 reaction units: M0, M1, M2, and M3 (all with reaction rate of 1) and 4 pools MA, MB, MC and MD. Each pool activates one of three behaviors, and for every time step that a particular pool contains the token unit, its corresponding behavior is performed. Pool MA activates turn, MC activates run and pools MB and MD activate stop. If these pools were switches to motor actuators on a simple wheeled robot, pool MC would switch on all wheel motors, while pool MA would switch on wheel motors on the left side only, thus turning the robot. The remaining pools would act as off switches. The other subnetworks inhibit or excite the reaction units of the master oscillator to allow or prevent chemical flow. The number of time steps that a chemical is present in a particular pool indicates the length of time that a particular behavior is performed. Thus if pool MC contains a chemical for 10 time steps, then the agent will move forward for 10 time steps; similarly if this were pool MA, the agent would turn for 10 time steps.

The Food Network and the Run Length Network

The food network senses the level of food within the environment and connects to the run length network to modify the number of steps forward according to the level of food sensed. The value of food at a cytobot's current position is stored at input pool FA. The forward rate of reaction node F0 is 1, thus all food is transferred to pool FB in a single time step. The presence of chemical in pool FB inhibits the run network reaction R0 for a number of time steps according to the level of food (by setting forward rate of unit F1 to 1 and weight to 0 this can be an exact correlation). This in turn stops pool RC in the run length network from emptying. Pool RC inhibits reaction M2 of the master oscillator thus preventing pool MC from emptying for the same number of time steps.

As discussed previously, the number of time steps which pool MC contains the token unit represents the number of time steps to move forward.

The Signaling Network

The signaling network functions as a switch between starvation and foraging mode. Low food levels trigger the starvation response and allow the weighted direction network to control each new angle. Sufficient food will switch off the weighted direction network and allow the chaotic network to control each new angle. It is a simple closed loop with a token unit of chemical flowing around it. Pool CA acts as a switch between foraging and starvation behavior, where the presence of chemical in CA inhibits the weighted direction network-while its absence switches on the weighted direction network; this in turn inhibits the chaotic network, as shown in figure 2. In this component, all reaction units have a forward flux of 0.5; which ensures a minimum number of time steps for each behavior.

The Weighted Direction Network

The weighted direction network senses cAMP within the agent's immediate environment and calculates a tumble angle which is weighted toward higher cAMP levels. This network interfaces with the environment via a number of receptor pools (AW, ANW, AN, ANE, AEA) which sense the level of cAMP around the cytobot. These pools represent receptors positioned at fixed points around the front of its perimeter. Limiting the signal detection to one side facilitates representation of polarization in *D. discoideum*, where that side becomes the leading edge. For each receptor input pool, there is a static pool containing a fixed level of chemical which represents the angle of the receptor relative to the cytobot. Directions start from AW (west) with a corresponding numeric value of 0 (A00) and progress in 45 degree steps through each direction to east (thus maximum value is 180). Detected signals are classed as being in one of the following cardinal/ordinal directions: W, NW, N, NE, and E. Thus signals are detected from all directions above its horizontal plane. All connections have a weight of 1 with the exception of the connection between pool AD and reaction A12 which has a weight of -1. This negative connection raises the sum of chemical detected in pool AD to -1, which multiplied by AB, allows the average angle to be calculated. The calculated angle interfaces with the remaining subnetworks at pool AE. In an actual organism, receptors are set around the cell perimeter and direct movement appropriately.

$$h \equiv ((n-90)+c) \bmod 360 \tag{3}$$

Where:
h= new heading (relative to external frame)
n = count of time steps pool MA contained chemical
c = current heading (relative to the external frame)

In this simulation, for simplicity, a count of the number of time steps that MA contains the token unit is processed to gain the turn angle using (3). Thus if the number time steps is 120 and the agent is facing north, then the current heading would equal 0 (relative to the external frame) and the new heading would equal 30.

The Chaotic Network

The chaotic network, as shown in figure 2, is responsible for generating pseudo random angles which agents use to perform the foraging tumble behavior. It is a networked implementation of a Logistic Map, see (4). Ulam and von Neumann (1947) were the first to examine a Logistic Map as a pseudo random number generator and it has been successfully used in this capacity by several researchers (Patidar, et al., 2009). The probability density distribution of the Logistic Map is non-uniform and is given in (5).

$$X_{n+1} = \lambda X_n (1 - X_n) \tag{4}$$

Where:
X_n= state variable of value $0 \leq X_n \leq 1$
λ= system parameter of value $1 \leq \lambda \leq 4$

$$P(X) = \frac{1}{\pi\sqrt{X(1-X)}} \tag{5}$$

Where:
P(X) = probability of X occurring

At the start of the simulation, pools KA and KB of each cytobot's chaotic network are initialized to the same random value between 0 and 1 (to 5 decimal places). This value represents the first value of X (where X represents the state variable of (4)). All the other pools are initialized to 0 with the exception of the static pools KI and RK whose initial values are 360 and 1 respectively. Reaction K2 is responsible for generating each new value of X and has a forward and reverse rate of 4 (the logistic map exhibits chaotic behavior when λ is 4). The connection between KA and K2 has a weight of 1 and the connection between K2 and KB has a weight of 2. The remaining series of reactions function to copy the value of X 3 times, where 2 copies serve as the new initial values of KA and KB and the remaining copy participates in the final output of the network at KH. Static pool KI has a fixed value of 360 which in reaction K0, allows the network to convert the pseudo random number at KH to an angle value between 0 and 360. However, reaction K0 cannot proceed until all 11 pools that inhibit it are empty. These inhibitory connections ensure that random angles are not output while the agent is in starvation mode, and that pool AE is empty before adding more chemical.

Slime Mould Aggregation Simulation

In the following experiments cytobots are used to model the behavior of aggregating *D. discoideum* cells, where each cytobot represents a cell. In each experiment the emergent patterns, numbers of mounds, and length of time to mound formation is examined. A total of 10 experiments are performed at varying population densities of cytobots (p) and

different ranges of detection of cAMP (r), as shown in table 1. The environment contains no food, thus each agent immediately enters and stays in the previously described starvation mode. The agents' behavior is initially explored at biologically realistic p and r values and compared with the behavior of the actual organism and other simulations. These parameters are then extended into ranges outwith the biological range in order to examine the emergent properties of the system.

```
FOR each cytobot
   Get current agents' facing direction C_F
   Assign a value to direction C_F using statement 1

      FOR each (index n) detected cAMP signal
         Get detected signal incoming direction C_A
         Assign a value to direction C_A using statement 1
         IF CA = C_F THEN k_n = 3
         ELSE IF C_A = C_F-1 OR C_A = C_F+1 THEN k_n=2
         ELSE IF C_A = C_F-2 OR C_A = C_F +2 THEN k_n=1
         ELSE k_n=0
         END IF
         Calculate distance d_n
         Store each C_A with k_n and d_n
      END FOR

    Calculate W_A for current agent using Equation 6
END FOR

Statement 1: East = 1; North East = 2; North = 3; North West
=4; West = 5

Where:
W_A= total weight of direction A
N= total number of agents within range of detection
d_n= distance of current agent from agent n
C_A = direction of incoming signal detected by current agent
C_F = the current agents facing direction
k_n = value of cAMP signal from agent n
```

Figure 3: Pseudocode to calculate the strength of detected cAMP at each direction relative to the cell.

$$W_A = \sum_{n=1}^{N} \frac{k_n}{d_n} \tag{6}$$

Cytobots move within a simulated 2D environment of area 5.06 mm^2- approximately half the maximum recorded aggregation territory reported in the literature (Dallon and Othmer, 1997). Each pixel represents 4.5 μm and the grid is 500 × 500 pixels, giving a total area of 5.06 mm^2. In nature, aggregating *D. discoideum* cell densities are typically 250 per mm^2 to $1x10^4$ per mm^2 (Dallon and Othemer, 1997). Due to the computational resources required to manage a population of cytobots within the upper range, two cell densities of 250 agents per mm^2 (1250 agents) and 150 per mm^2 (750 agents) were chosen. The agents are initialized at random positions within the simulated environment. Each starving agent emits a cAMP signal at equal strength around its circumference into the environment. This signal is detected by other agents within or equal to r. In these experiments a range of r values are explored, including that of real cells of 1, 0.5, and 0.1 mm (McCann, et al., 2010). The actual cAMP signal degrades linearly with increasing distance (d) from the emitting cell. Each agent detects the cAMP signal of all starving cells within or equal to r, and a total value for each direction (A) is calculated using the pseudocode given in figure 3. Each cycle represents 1 minute of time. In this time the agent moves 9μm- a distance which corresponds to that reported in the literature (Rifkin and Goldberg, 2006). Therefore, after 1 hour motion the agent travels a distance of 540μm. In reality there are always remaining cells that do not aggregate, and thus the simulation runs until 95% of agents are at a distance of less than 0.1mm from their nearest neighbor.

Results

The results for all 10 experiments are given in table 1. Each experiment was performed 100 times. In experiments 8, 9, and 10 the value of r and d are within the ranges reported for real *D. discoideum* cells. These experiments are used to compare the behaviors and length of time taken to aggregate with the literature. In experiments 8, and 9 aggregation completes after an average formation of 4.3 mounds in 10.05 hours, and 6.7 mounds in 12.65 hours respectively. In nature the organism takes between 9-13 hours to aggregate (Cotter, et al., 1992; Becker, et al., 2010), thus the results of these experiments have an aggregation time within the reported range. This is also comparable to other simulations. For example, Becker et al. (2010) reports an aggregation time of 11.6 hours for a simulated population of *D. discoideum* with a cell density of $200mm^2$. In experiment 10, the population never satisfied the criteria for completion of aggregation, where instead the agents appeared to move in a fashion reminiscent of Brownian motion. The likely explanation for this is twofold. Firstly, the simulation does not consider the effect of glycoproteins where aggregating cells making contact with each other attach together. Secondly, because the attraction range is so small, agents are only able to detect other agents within their immediate neighborhood, thus momentarily larger clusters with higher attraction strength go undetected and quickly dissipate- an effect that would not occur if agents stayed together. The complete set of results shows that by increasing p by $100mm^2$ the number of mounds formed at each r decrease with the exception of experiment 6. This is not surprising, as denser populations should have more chance of interacting, and thus form fewer clusters, but with higher numbers of agents. Similarly, decreasing r results in a general increase to the number of mounds formed at both values of p. The likely reason for this is that as r decreases the agent becomes unable to influence increasing quantities of area, thus larger numbers of stable clusters can form but with fewer numbers of agents.

Emergent behaviors and clustering patterns similar to the biological organism were also observed. As previously discussed, the cytobots are polarized.

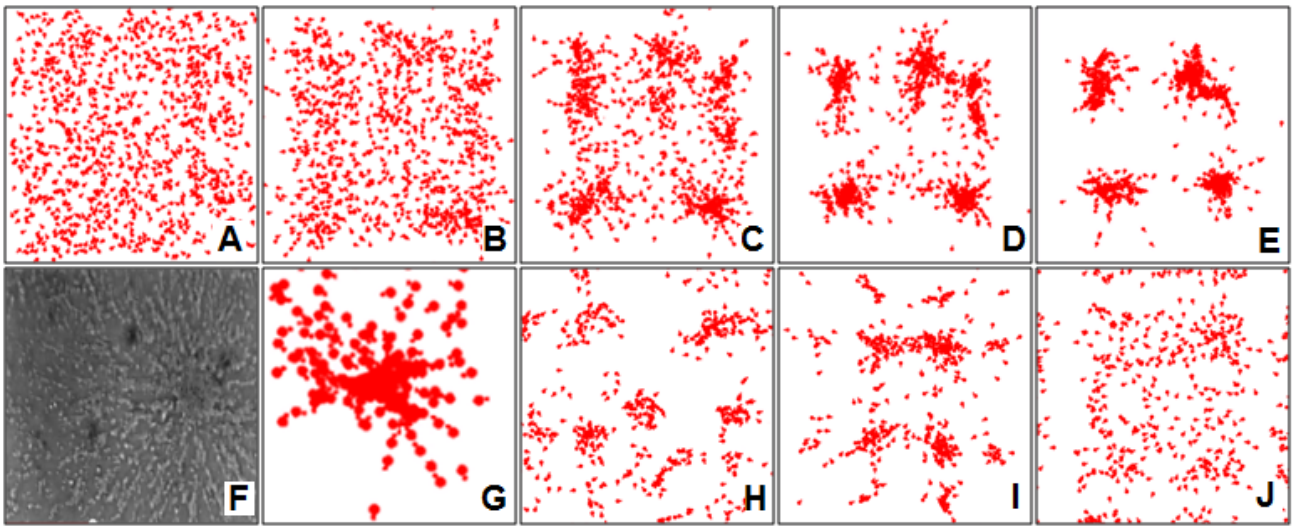


Figure 4: A-E Cytobot aggregation for experiment 8 at: A- 1hr, B- 2hr, C-5hrs, D- 8hrs, E- 12hrs; Image F- *D. discoideum* cells aggregating; G- Is the lower right hand corner of image C demonstrating streaming behavior; H- Spiral patterns in experiment 4 after 8 hours; I- Symmetrical patterns for experiment 2 at 7 hours; J- Wave pattern for experiment 2 at 2 hours.

Diagram F is courtesy of T, Gregor, Laboratory for the Physics of Life, Princeton University, 2013 Used with permission.

Table 1: Aggregation experiment simulation results

No	Density (p) per mm^2	Range (r) in mm	Mean No. of mounds; (σ= Sta. Dev.)	Mean time (hours); (σ); *Literature Range 9-13 hours
1	150	5	1 (0)	8.98 (0.09)
2	150	2.5	4 (0.31)	9.63 (0.17)
3	150	1	5.2 (0.82)	9.92 (0.34)
4	150	0.5	8.4 (1.19)	10.23 (0.59)
5	150	0.1	14.2 (2.36)	10.6 (1.82)
6	250	5	1 (0)	8.95 (0.11)
7	250	2.5	1 (0)	9.6 (0.20)
8	250	1	4.3 (0.37)	10.05 (0.58)
9	250	0.5	6.7 (1.62)	12.65 (1.94)
10	250	0.1	-	-

Implementing the agents in this way allowed us to observe whether or not the previously described streaming behavior occurs. A close up of the right hand corner of screenshot C is shown in figure 4G showing agents beginning to form a cluster. The protruding head of each agent can be seen clearly, where each lines up its head to the rear of another agent and forms a stream. As can be seen in figure 4F this is very similar to the streaming behavior in real cells of *D. discoideum*. Other emergent patterns occurred during different experiments including spirals (figure 4H), symmetric patterns (figure 4I), and waves (figure 4J).

Oil Spill Clean-up Simulation

To illustrate a practical application, the cytobots are used to tackle a simplified oil-spill clean-up simulation. In these experiments, the same ARN used previously produces different behaviors by altering its interface with the environment. In the following 4 experiments the length of time it takes for 3, 5, 8 and 15 cytobots to clean up 95% of the oil is recorded. These results are compared with similar work.

The cytobots move within a 2D simulated environment containing an oil spill. This oil is analogous to a distribution of food within a nutrient landscape. The task of the cytobots is to clean up the spill as quickly as possible by consuming oil at each location. The agents move through the environment by switching between the two previously described behavioral modes- foraging and starvation. In the aggregation experiments, no food was present, thus the foraging behavior remained inactive. In this case, the concentration of oil surrounding the agents was fed into both the receptor pools of the weighted direction network and the food network. Thus in this case oil represents both food and cAMP. At the start of each experiment, the cytobots are distributed randomly within the environment, and the ARN network is initialized as previously described. The agents start the simulation in foraging mode but during the simulation alternate between foraging and starvation modes. Starvation behavior is triggered after the most recent positions (minimum of 2) contained zero food. In starvation mode, instead of turning in a random direction, the new direction is weighted toward higher concentrations of food within its surrounding area. This behavior forces exploration of unexplored search space because previously visited positions have a food level of 0.

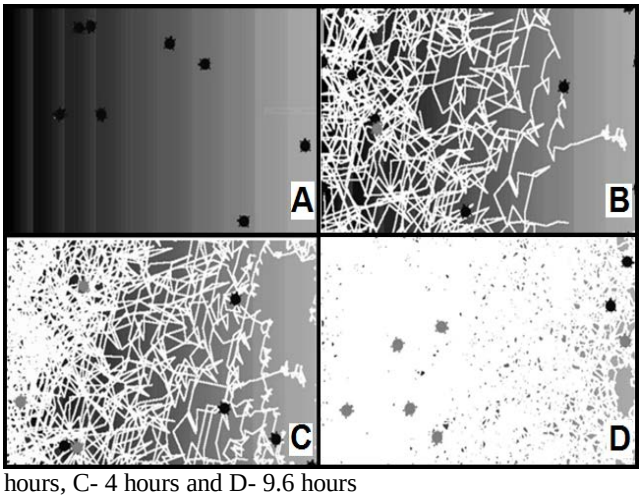
Consumption of environmental food therefore acts as a stigmergic signal, where agents are inclined to move up the nutrient gradient created by their foraging activities. Here, we model the spillage of 100 tonnes of Statfjord crude oil at 15^0C under a wind speed of $5ms^{-1}$ The oil is distributed over a 2D sea surface of 300m by 200m, thus an area of area $60000m^2$, where 2 pixels corresponds to 1m, as shown in figure 4A. This particular oil type and parameter set were chosen in order to compare directly with work by Kakalis and Ventikos (2008) who present a robotic swarm concept for oil spill confrontation. For this reason, we account for an initial response time of 14 hours. Based on the complex mathematical models found in Kakalis and Ventikos which account for the main factors of short term changes in oil characterization, the volume of oil after 14 hours is reduced to $150m^3$. Beyond the starting state, the volume is only influenced by the cytobots. The speed of each agent is $0.5ms^{-1}$ and is based on other robotic agents in oil cleaning scenarios (Kakalis and Ventikos, 2008), thus the cytobots move 1 pixel (0.5m) for every time step. The actual cleaning surface is 1m, thus the cytobots clean a 2 pixel wide area in each time step.

Mathematical modeling of an oil spill is non-trivial and at best can offer a crude approximation of its actual trajectory. Most oil spills quickly form a comet shape with most of the oil within the head and a trail of sheen (Wang and Stout, 2007). To represent a simplified version of the comet shaped spread, the area is divided into 100 3m x 200m segments. The first segment contains 0.015 tonnes of oil, and each subsequent segment increases by 0.03 tonnes from right to left.

Results

In each experiment, a different number of cytobots was deployed- 3, 5, 8 and 15 and the recovery rate achieved by each group was compared. The simulation time was measured from deployment of the cytobots at 0 hours (14 hours after oil was spilled) and stopped when the cytobots had collectively removed 95% of the $150m^3$ of oil. Each experiment was run 100 times, and the average volume of oil consumed at 6 minute intervals was calculated. Figure 6 presents the volume of oil consumed by each group of cytobots against time. The finishing times in hours are 15.2, 11.5, 9.6, and 6.1 for 3, 5. 8, and 15 cytobots respectively. By adding 2 additional agents to the group of 3 the length of time is reduced by 3.5 hours, thus 1.75 hour difference per extra cytobot. This difference decreases 1.12 hours per cytobot for 8 agents, then to 0.76 per agent for 15. This variation can be accounted for by examining the agents' paths through the oil. Rates are much faster at the beginning of the experiments, where cytobots move toward the oil rich left side of the environment. This can be seen in the series of screenshots in figure 5 where A shows the starting position at time 0, and B shows that after 2 hours the cytobots have moved toward the left-hand side, focusing mainly on highly concentrated areas (consumed oil is shown in white). Initially, the rate of oil removal is high because cytobots focus on the volume rich areas and cannot go over their path, thus each new location results in consumption of oil. However, as time progresses, large patches become cleaned and a higher probability exists for the cytobots to revisit previously cleaned areas. The consumption of oil in figure 5 C and D at 4 and 9.6 hours respectively shows more clearly that cytobots focus cleaning efforts on the richest volume area first, and are gradually forced to move toward the next highest concentration by the gradient created by their foraging activities.

Figure 5: Oil simulation using 8 cytobots at A- 0 hours, B- 2 hours, C- 4 hours and D- 9.6 hours



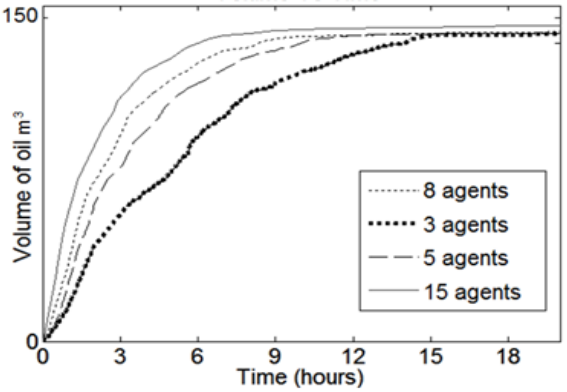


Figure 6: Volume of oil cleaned against time for each group of cytobots

Figure 5 D shows the state of the oil at the end of the simulation, where only small patches remain mainly in areas of low oil volumes. These results can be compared to the simulation by Kakalis and Ventikos. Here, varying numbers of simulated EU-MOP robots are deployed to tackle $150m^3$ of Strajford oil over $60000m^3$ (as before). In this case, the robots have a slightly faster speed of 0.54m/s but have the same 1m skimming face. Each EU-MOP robot has a storage capacity of $2m^3$ and a transit speed of $2.1ms^{-1}$. The time taken for 3, 5, 8, and 15 EU-MOPS are 54, 32, 20 and 10 hours respectively. For comparison, the results of our simulation can be adjusted to include unloading of the oil at a servicing vessel. Using the same storage capacity and transit speed and assuming the distance to the ship and back is 2 times 300m and that each cytobot fills the same amount simultaneously, then the new times are 17.2, 12.7, 10.3 and 6.5 for 3, 5, 8 and 15 cytobots respectively. The Kakalis and Ventikos simulation has several differences to the one reported here, particularly in the distribution of the oil. Also, some key parameters are missing from their paper, for example, distance to boat. Despite these

differences, our results are very similar. For example the reported simulation time for 15 EU-MOPS is 10 hours and in our simulation 5 and 8 cytobots took 12.7 and 10.3 hours respectively. Given the differences in the simulation and differences in operation of the robots, the resulting clean up times are comparable showing that the cytobots have potential application as distributed robotic agents in real-world environments.

Conclusions

The aggregation experiment results presented above show that the agents are able to simulate behavior of individual unicellular organisms, and model emergent behavior arising from interactions among such groups. These results demonstrate the parallels between ARN agents and the biological counterpart from which they were inspired. It also highlights a potential use as a means to simulate groups of interacting cells such as a bacterial colony or tissue component within a multicellular organism.

The results for the oil spill simulation demonstrate potential application for the ARN agents as autonomous agents within real world environments. This application demands an internal control system which can function without reference to other agents within the environment which are operating in parallel. By modifying the environment, (which in this case was consumption of food), the agents can stigmergically communicate and facilitate emergent behavior. The cytobots offer a unique range of abilities. Like cells, their internal network of spatially distributed dynamic chemical species allows them to autonomously coordinate and direct their movement, recognize and respond to patterns in the environment, and produce high-level behavior.

In future work, it is intended to further explore the AI applications of the cytobot agents, and later, to create swarms of cytobot robots with applications in real world environments.

References

Arkin, A. and Ross, J. (1994) Computational functions in biochemical reaction networks, *Biophysical Journal*, 67:560-578.

Becker, M. (2010). Simulation model for the whole life cycle of slime mould *Dictyostelium Discoideum*. In proceedings of the European conference on modeling and simulation, pages 247-253.

Bhalla, U. S. (2003). Understanding complex signaling networks through models and metaphors. *Progress in Biophysics and Molecular Biology*, 81(1):45-46.

Bray, D. (1995). Protein molecules as computational elements in living cells. *Nature*, 376(6538):307-12.

Cotter, D. A., Sands, T. W., Virdy, K. J., North, M. J., Klein, G., Satre, M. (1992). Patterning of development in Dictyostelium discoideum: factors regulating growth, differentiation, spore dormancy and germination. *Biochemistry and Cell Biology*, 70(10-11):892-919.

Dallon, J. C. and Othmer, H. G. (1997). A discrete cell model with adaptive signaling for aggregation of Dictyostelium discoideum. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 352(1351): 391–417.

Dittrich, P., Zeigler, J., Banzhaf, W. (2001). Artificial chemistries- a review. *Artificial Life*, 7(3):225-275.

Ford, B. J. (2009). On intelligence in cells: The case for whole cell biology. *Interdisciplinary Science Reviews*, 34(4):350-365.

Gerrard, C. E., McCall, J., Coghill, G. M. and Macleod, C. (2011). Artificial Reaction Networks, In Proceedings of the 11th UK Workshop on Computational Intelligence, pages 20-26.

Gerrard, C. E., McCall, J., Coghill, G. M. and Macleod, C. (2012a). Adaptive dynamic control of quadrupedal robotic gaits with Artificial Reaction Networks. In Proceedings of the 19th international conference on neural information processing, pages 280-287.

Gerrard, C. E., McCall, J., Coghill, G. M. and Macleod, C. (2012b). Temporal patterns in artificial reaction networks, In Proceedings of the 22nd international conference on Artificial Neural Networks, pages 1-8.

Kakalis, N. M. P., Ventikos, Y. (2008). Robotic swarm concept for efficient oil spill confrontation. *Journal of Hazardous Materials*, 154(1-3):880-7.

McCann, C. P., Kriebel, P. W., Parent, C. A., Losert, W. (2010). Cell speed, persistence and information transmission during signal relay and collective migration. *Journal of Cell Science*, 123:1724-1731.

Patidar, V., Sud, K. K., and Pareek, N. K. (2009). A pseudo random generator based on chaotic logistic map and its statistical testing. *Informatica*, 33:441-452.

Rifkin, J. L. and Goldberg, R. R. (2006). Effects of chemoattractant pteridines upon speed of D. discoideum vegetative amoeba. *Cell Motility and the Cytoskeleton*, 63(1): 1-5.

Tyson, J. J. and Novák, B. (2010). Functional motifs in biochemical reaction networks. *Annual Review of Physical Chemistry*, 61:219-240.

Ulam, S. M. and von Neumann, J. (1947). On combinations of stochastic and deterministic processes. *Bulletin of the American Mathematical Society*, 53:1120.

Vladimirov, N. and Sourjik, V. (2009). Chemotaxis: how bacteria use memory. *The Journal of Biological Chemistry*, 390(11):1097-1104.

Wang, Z. and Stout, S. A. (2007). Oil spill environmental forensics: fingerprinting and source identification. Academic Press, Burlington MA.

Paper 4: ‘Combining Biochemical Network Motifs within an ARN-Agent Control System’

Reference:

Gerrard, C., McCall, J., Coghill, G., Macleod, C. (2013) ‘Combining Biochemical Network Motifs within an ARN-Agent Control System’, *Proceedings of the 13th Annual Workshop on Computational Intelligence (UKCI)*, IEEE, Surrey, pp. 8-15.

Combining Biochemical Network Motifs within an ARN-Agent Control System

Claire E. Gerrard, John McCall, Christopher Macleod,
IDEAS Research Institute,
Robert Gordon University,
Aberdeen, Scotland.
c.e.gerrard@rgu.ac.uk, j.mccall@rgu.ac.uk,
chris.macleod@rgu.ac.uk

George M. Coghill,
Department of Computing Science
University of Aberdeen
Aberdeen, Scotland.
g.coghill@abdn.ac.uk

***Abstract*—The Artificial Reaction Network (ARN) is an Artificial Chemistry representation inspired by cell signaling networks. The ARN has previously been applied to the simulation of the chemotaxis pathway of *Escherichia coli* and to the control of limbed robots. In this paper we discuss the design of an ARN control system composed of a combination of network motifs found in actual biochemical networks. Using this control system we create multiple cell-like autonomous agents capable of coordinating all aspects of their behavior, recognizing environmental patterns and communicating with other agent's stigmergically. The agents are applied to simulate two phases of the life cycle of *Dictyostelium discoideum*: vegetative and aggregation phase including the transition. The results of the simulation show that the ARN is well suited for construction of biochemical regulatory networks. Furthermore, it is a powerful tool for modeling multi agent systems such as a population of amoebae or bacterial colony.**

Keywords— Artificial Reaction Networks; Artificial Chemistry; Swarm Agents

I. INTRODUCTION

Artificial Chemistry (A-Chem) is a subfield of A-Life and, in its broadest sense, it describes man-made systems which are similar to real chemical systems [1]. Chemical information processing has many desirable properties, it is: decentralized asynchronous, fault tolerant, evolvable, self-organizing and adaptive [1]. A-Chem focuses on harnessing these properties by either creating Molecular Computing devices where computation is achieved using either real chemicals or by utilizing the principles of the chemical metaphor to construct novel software or hardware architectures *in silico* [1]. The latter is termed Artificial Chemistry Computing (CCM) and is the focus of this paper. In the chemical metaphor, data is stored in the form of molecular species and information processing occurs through interactions (reactions) between these molecules. The result of this computation emerges from the numerous low-level interactions and appears as a global behavior [1]. ACC is used in two main applications: simulating complex systems (biological, social or ecological) and in developing novel solutions to engineering or computational problems. The Artificial Reaction Network (ARN) is an ACC representation inspired by the properties and mechanisms of information processing found in biological Cell Signaling Networks (CSNs). In our previous work, it was applied to simulate the chemotaxis signaling pathway of *E. coli* [2], and later investigated as a means to produce complex temporal waveforms to control limbed robots [3,4].

Within a cell, data is represented by a set of spatially distributed molecular concentrations; CSNs process this information within elaborate hierarchical network control structures which connect species together in productive or inhibitory unions. In this way, cells are able to respond to changes within their environment, communicate with other cells, and perform internal self maintenance operations [5]. The ability of chemical networks to perform computational processing is well documented. For example it has been shown both theoretically and in wet lab experiments that such networks can perform Boolean and Fuzzy logic functions [1,5]. A number of researchers have identified structural motifs in such biochemical networks which can form basic computational processing units [5,6].

The aim of this paper is to show that the ARN is a powerful modeling tool and can produce realistic approximations of the complex network circuitry that exists within and between cells. This ARN is firstly used to construct real biochemical network regulatory motifs. These motifs are then combined to create a control system for an autonomous ARN-agent. The control system implements a set of cell-like behaviors which allow the agent to recognize and respond to environmental patterns by modifying its trajectory. A swarm of ARN-agents are then instantiated within an artificial environment and used to simulate the collective behavior of a population of *D. discoideum* (Dd) cells throughout two phases of the organism's life cycle: vegetative and aggregative.

The paper is structured as follows: Section II provides a summary of the ARN representation; this is followed by a discussion about network motifs and the ARN-agent control system in Section III and IV. The experimental details are discussed in Section V followed by results in Section VI. Finally, Section VII presents the conclusions.

II. Artificial Reaction Networks

A summary of the ARN is provided here. Verification and further discussion of the ARN representation can be found in our previous work [2-4]. The ARN comprises a set of networked reaction nodes (circles), pools (squares), and inputs (triangles) and is depicted as a directed weighted graph as shown in Fig. 1 Each pool stores a current available chemical species concentration (*avail*); thus, the complete set of pool concentrations at time t, corresponds to the current state of the system.

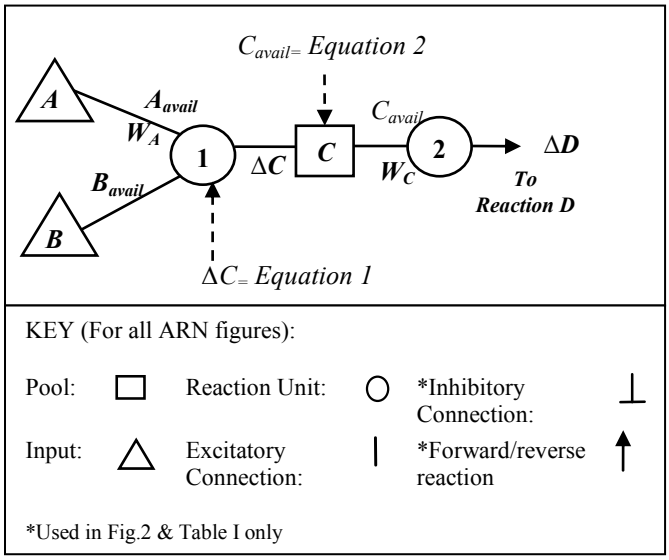


Fig. 1. The Artificial Reaction Network representation.

$$\Delta C = \left(K_{f(C)}\left(A_{avail}^{W_A} B_{avail}^{W_B}\right) - K_{r(C)}\left(C_{avail}^{W_C}\right)\right)\Delta t \quad (1)$$

$$C_{avail} = C_{avail} + \left(\Delta C - \frac{W_C}{W_C + \alpha}\Delta D\right) \quad (2)$$

$$C_{avail} = C_{avail} + \left(\Delta C - \Delta D W_C\right) \quad (3)$$

Where:
A, B, C, D = Species Concentrations
W = Reaction order (weight)
avail = Available species concentration
K_f = Forward rate constant
ΔC = Change in species concentration C
K_r = Reverse rate constant
α=sum of other incoming weights to D

In this paper and the following example we use conserved mass quantities, however, the ARN may also be used to model non-conserved quantities such as the number of available molecules. The choice affects the way in which each pool is updated and is discussed later. While many ACs assume a well-stirred reactor, the use of pools within the ARN provides a discrete spatial structure. Inside a biological cell concentrations of chemical species are spatially distributed into localized compartments. This localization restricts which molecules may react together, thus affects the overall dynamics of the system. Representing the spatial distribution of chemicals allows fine grained control over the system dynamics and thus is highly beneficial when modeling biochemical circuitry. For example it allows the representation of flow structures such as membrane channels, transport processes; network motifs, and provides a means to explore disease pathways [7]. Thus, in the ARN, each pool is represented as a well-stirred reactor and approximates a spatial compartment. Inputs are a special type of pool; the only difference is that they are of fixed value and thus can be used to represent the continuous flow of environmental inputs or enzymes. Each circle corresponds to a reaction unit, representing a reaction between a number of chemicals. Data is processed by reaction nodes transforming incoming pool values to connected outgoing pool values. Connections symbolize the flow of chemical into and out of reaction units and their weight (*W*) corresponds to reaction order. Connections provide the facility to create complex control structures using combinations of inhibitory and excitatory connections. Fig. 1 shows the reaction between species *A* and *B* to produce species *C*. At time interval Δt, each reaction unit's temporal flux value is calculated by applying Euler's approximation to the differential rate equation as shown in (1). This value is then used to update the current concentration of each reaction's connecting pools as shown in (2). A reaction step may proceed if it meets its preconditions. Preconditions are met if incoming inhibitory pools are inactive, and incoming excitatory pools are active. In a similar way the completion of a reaction step will fulfill a number of post conditions, which depend on the parameters and connections of the reaction step. Pools may asymptotically approach 0, and thus below a particular threshold a pool is considered empty and its value set to zero.

As previously mentioned the ARN is not restricted to representing conserved quantities such as mass, one may choose to model the exchange of molecules between pools. Thus using the example in Fig. 1, to generate one molecule of product *C* requires W_A molecules of reactant *A* and W_B molecules of reactant *B*. In this case pool *C* is updated by applying (3).

III. Background Theory

The ARN presented in Section IV is composed of regulatory motifs found in real biochemical networks. These motifs are combined in such as way to provide the control system for an ARN-agent. In this Section a summary of two behavioral modes implemented within the ARN-agent control system are described. This is followed by the structure and function of common network motifs, including biological examples.

A. Foraging and Starvation Modes of ARN-agents

Dd is a cellular slime mould which exists as a collection of amoeba and transitions to a multicellular slug during the aggregation phase of its life cycle. During its vegetative stage, amoebae move up gradients of folic acid (FA) secreted by its bacterial prey. The behavior of these unpolarised cells is characterized as a random biased walk where cells extend random pseudopods in a biased manner toward the source of FA resulting in overall movement up the gradient of FA [8]. Dd amoebae begin to starve when the food resource has been depleted, and begin the aggregation phase. During

aggregation, starving cells secrete cAMP (cyclic adenosine monophosphate) which serves as a signal to attract surrounding amoebae towards a central location [9]. Aggregating Dd cells are polarized, thus one side becomes the leading edge which always faces in the direction of travel [9, 10]. Depending on parameters such as environmental conditions, and Dd population density, migrating cells often form transient emergent patterns such as streams, waves and spirals [9, 10]. Streaming describes a pattern of motion where cells line up in close order files, with the head of one following the rear of another [10].

The agent performs two behavior modes- foraging and starvation based respectively on vegetative and aggregative behaviors of Dd amoebae. During the foraging phase, agent's alternate periods of forward motion termed "runs" and random redirections called "tumbles". The bias is provided by reducing the period of tumbles when moving up the food gradient. At each passing location food is consumed. The agents enter starvation mode if food has not been consumed within a time period. Each starving agent emits a continuous signal of cAMP into its surrounding environment. Instead of turning in a random direction, agents turn in the direction weighted toward the highest concentration of cAMP within its surrounding area.

B. *Functional Motifs In Biochemical Networks*

Cell membranes are studded with receptors which are sensitive to external parameters such as chemicals, pH, temperature and light. These receptors are responsible for detecting and transducing environmental signals. These signals trigger information processing events within the CSN which update cell activity such as changes in gene expression.

Components of CSNs are linked through productive unions (union of reactants triggers production/activation of other components) and inhibitory unions (union of two reactants inhibits production/activation). In the same way the network is separated through productive and inhibitory isolations (union does not occur or does not produce any effect). Such links arrange chemicals into elaborate circuitry which functions as the information processing machine of the cell.

A fundamental challenge in Molecular Biology is to understand such signal processing and thus enable the prediction of the effects of disease and intervention of pharmaceuticals. To this end a number of researchers have identified functional structural motifs within these networks [5, 6]. A summary of the structure (in ARN format), function and biological examples of a number of the most common motifs is provided in Table 1. A more detailed discussion is provided by Tyson [6]. Note that these motifs are shown for simplicity as 2 or 3 component but there are larger versions with the same function. For example, an additional component may be added to motif 9 to create a 4 component oscillator.

IV. ARN-AGENT CONTROL SYSTEM

In this Section we discuss an ARN control system composed of the structural motifs in Table 1. This system is designed to control an autonomous ARN-agent termed a "Cytobot" ("cyto" from Greek for cell, and "bot" from robot). The cytobot has two behavior modes: foraging and starvation, based on the previously described behavior of Dd amoeba during its respective vegetative and aggregative phases. In the experiments described in Section V a number of these cytobots are instantiated and interact to produce global emergent behavior. The relationship between the ARN control system and the cytobot is similar to that of a CSN to an amoeba. Thus the control system allows a cytobot to recognize environmental patterns, updating its trajectory within an artificial 2D environment and to communicate stigmergically with other cytobots. The environment contains a distribution of artificial chemicals. These chemicals represent attractants of either food or cAMP. When a cytobot moves to a new position, the surrounding level of chemical is used to set the inputs to its ARN. Consequently this changes the internal state of the ARN and updates the agent's trajectory. During this process, the agent modifies the state of the environment by consuming food or releasing cAMP.

The cytobot ARN is composed of 6 subnetworks as shown in Fig. 2. Each subnetwork contributes a functional aspect to either or both starvation and foraging behaviors. The design of the subnetworks is discussed in the following Sections.

A. *The Master Oscillator*

The Master Oscillator functions to synchronize the outputs from all the other subnetworks together and is what each cytobot references at each time step to ascertain its current behavior. It is a 4 component oscillator (Table 1 motif 9) with a token unit of chemical cycling around it. It consists of 4 reaction units: M0, M1, M2, and M3 (all with reaction rate of 1) and 4 pools MA, MB, MC and MD and generates a pulsed width modulated waveform. Each pool is associated with 1 of 3 behaviors. Every time step that a particular pool contains the token unit, its corresponding behavior is performed. Pool MA activates turn, MC activates run and MB and MD activate stop. Thus, if pool MC contains a chemical for 10 time steps, the agent will move forward for 10 time steps. Note that this motif could control motor actuators on a simple wheeled robot: MC would switch on all wheel motors, while MA would switch on left wheel motors only, thus turning the robot. The remaining pools would act as off switches. The other subnetworks inhibit (motif 2) or excite (motif 1) the reaction units of the master oscillator to allow or prevent chemical flow.

B. *The Food and Run Length Network*

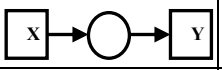
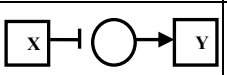
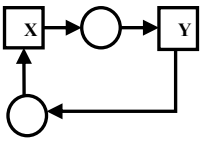
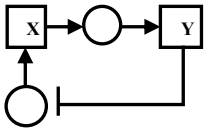
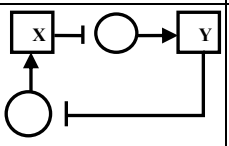
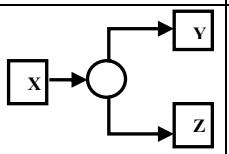
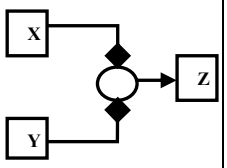
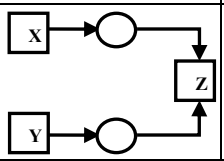
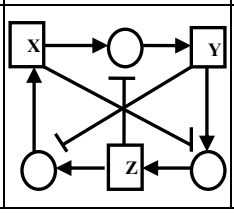
The food network interfaces with the environment at pool FA using an excitatory connection (motif 1) and inhibits the runlength network in accordance with the level of detected food. The forward rate of reaction node F0 is 1, thus the content of FA is transferred to pool FB in a single time step. The presence of chemical FB inhibits (motif 2) R0 for a number of time steps according to the level of food (by setting forward rate of unit F1 to 1 and weight to 0 this can be an exact correlation). The run length network is a 3 component oscillator (motif 9). While reaction R0 is inhibited it prevents pool RC from emptying. RC inhibits reaction M2 (motif 2) of the master oscillator thus preventing pool MC from emptying for the same number of time steps. As discussed previously,

the number of time steps which pool MC contains the token unit represents the number of time steps to move forward.

C. The Signalling Network

The signaling network functions as a switch between starvation and foraging mode. Low food levels trigger the starvation response and allow the weighted direction network to control each new angle. Sufficient food will switch off the weighted direction network and allow the chaotic network to control each new angle. It is a 3 component oscillator (motif 9) with a token unit of chemical flowing around it. Pool CA acts as a switch between foraging and starvation behavior.

TABLE I EXAMPLES OF FUNCTIONAL MOTIFS FOUND IN BIOCHEMICAL NETWORKS

Motif No., Name and Description	Structure (in ARN format)	Biological Example
1. Excitatory (E) The presence of X activates Y		Elementary motif common throughout pathways. E.g. Ras is a membrane associated protein that is normally activated in response to the binding of extracellular signals such as growth factors [6].
2. Inhibitory (Y) The presense of X inhibits Y. Acts as a NOT gate.		Elementary motif common throughout pathways. E.g. E-cadherin (a calcium-dependent cell–cell adhesion molecule) suppresses cellular transformation by inhibiting β-catenin [6].
3. Positive Feedback Loop (PFL) The presence of X activates Y and in turn the presence of Y activates X		The pathway of caspase activation is essential for apoptosis induction. A PFL exists between caspase-3 and caspase-9 [6].
4. Negative Feedback Loop (NFL) The presence of X activates Y and in turn the presence of Y inhibits X		The proteins Mdm2 and p53 (p53 is a tumor suppressor protein) are involved in a NFL which functions to keep the level of p53 low in the absence of p53-stabilizing signals [6].
5. Double-negative Feedback (DNF) The presence of X inhibits Y and the presence of Y inhibits X		BAX is protein which promotes apoptosis by competing with BCL. A DNF is formed between the proteins BAX and BCL [6].
6. Branch (B) The presence of X activates Y and Z		Transcription factors such as E2F or P53 frequently modulate the expression of more than one gene. Enzymes often modify more than one substrate e.g. CycB-dependant kinase [6].
7. Logic Gate (LG1) **AND gate**: 2 excitatory connections from X and Y when both X and Y are present they activate Z **NOR gate**: two inhibitory connections from X and Y. Both X and Y must be absent for Z to be activated **SWITCH:** Excitatory connection from X and inhibitory connection from Y. The presence of X but not Y activates Z		AND: The protein gCam 2 kinase becomes active when both calcium ions (Ca^{2+}) and Calmodulin (CaM) are present [5]. NOR: The activity of transcription factor E2F is a NOR function of RB and CycB where E2F is active when both RB and CycB are inactive [6]. SWITCH: The enzyme aspartate transcarbamylase is composed of multiple catalytic sites. It is activated by binding of its substrates (aspartate and carbamoyl phosphate) and inactivated by cytidine triphosphate causing its substrates to dissociate [5].
8. Logic Gate (LG2) **OR Gate:** : 2 excitatory connections from X and Y when either X or Y are present they activate		Ras is a membrane associated protein that is activated by a number of different signals. E.g. in response to the binding of extracellular signals such as a number of growth factors [6].
9. Oscillator (OSC) The presence of X activates Y. In turn the presence of Y activates Z but inhibits X. The presence of Z inhibits Y and activates X.		There are many examples e.g. in the cyanobacteria clock protein KaiC has a well defined closed cycle of phosphorylation and dephosphorylation states (composed of KaiA, KaiB and KaiC). In the motif shown here, all 3 components oscillate and each inhibits the reaction clockwise left. Oscillators may have less inhibitory connections, the number of which is dependent on the mobility of the reaction species. However, the presence of all inhibitors increases stability in the presence of fluctuating environmental parameters e.g. temperature. Note that this oscillator can also be thought of as a PFL (motif 3) combined with a system of DNFs (motif 5).
Key:  Either inhibitory or excitatory. X/Y/Z: Chemical species * Note that these motifs may combine arbitrary numbers of components.		

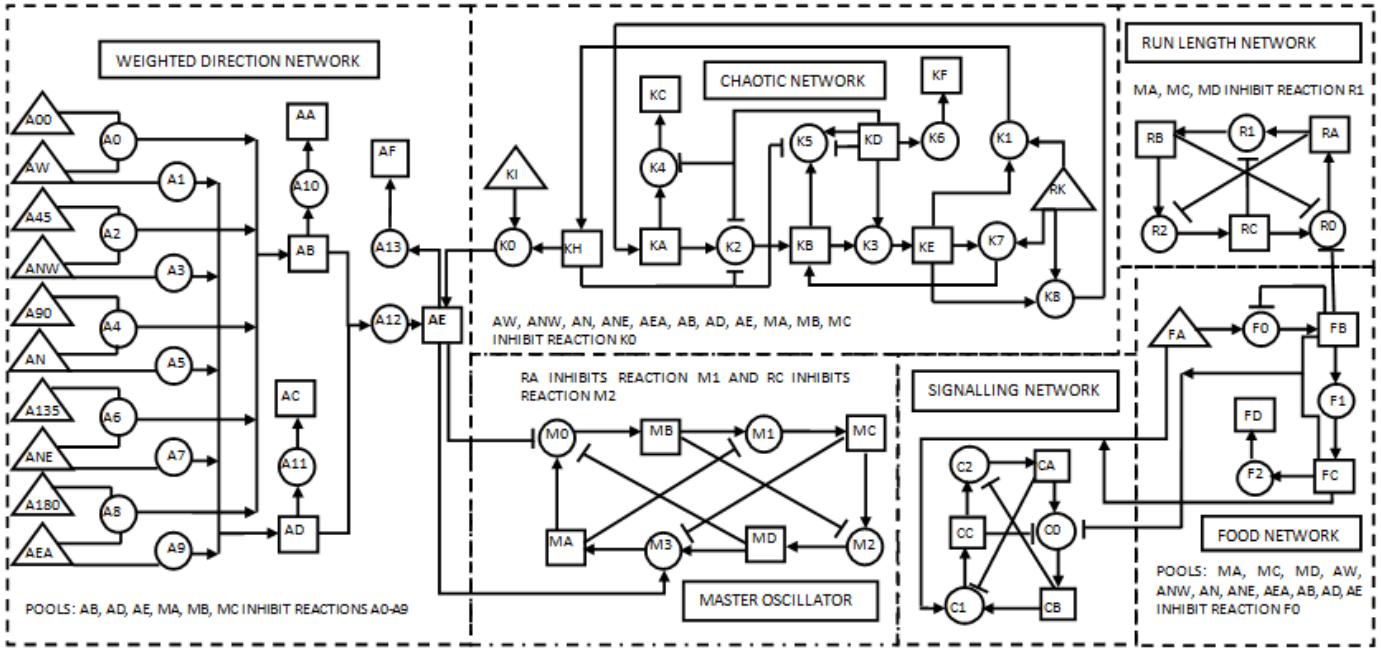


Fig. 2. The cytobot ARN network. Each cytobot is controlled by an instance of this network. The network is composed of 6 subnetworks

Here the presence of chemical in CA inhibits the weighted direction network (motif 2) while its absence switches on the weighted direction network; this in turn inhibits the chaotic network, as shown in Fig. 2. In this oscillatory network, all reaction units have a forward flux of 0.5; which produces a continuous oscillating waveform and ensures a minimum number of time steps for each behavior. A NOR gate (motif 7) activates pool CB in the absence of food chemical in both pools FB and FC of the food network, thus allowing pool CB to empty. While an AND gate (motif 7) will lead to pool CA to eventually refill by activating pool CC only when food is present in input FA and pool FC of the food network.

D. The Weighted Direction Network

The weighted direction network senses food within the agents' immediate environment and calculates a tumble angle which is weighted toward higher food levels. This network interfaces with the environment via a number of receptor pools (AW, ANW, AN, ANE, AEA) which sense the level of food around the cytobot. These pools represent receptors and are positioned at points around the front of its perimeter (as shown in Fig. 3), allowing the agent to travel in a similar way to that of a polarized Dd cell. For example, during the aggregation phase of their life cycle, Dd cells are polarized, and one side becomes the leading edge which always faces in the direction of travel [10]. Each receptor input pool forms one input of an AND gate (motif 7), the other input is a static pool containing a fixed level of chemical in correspondence to its direction. Directions start from AW (west) with a corresponding numeric value of 0 (A00) and progress in 45 degree steps through each direction to east. As the receptor positions around the agent are fixed, directions are always relative to that in which the agent is facing. All connections have a weight of 1 with the exception of the connection between pool AD and reaction A12 which has a weight of -1. This negative connection raises the sum of food detected in AD to -1, which multiplied by AB, allows an average angle to be calculated.

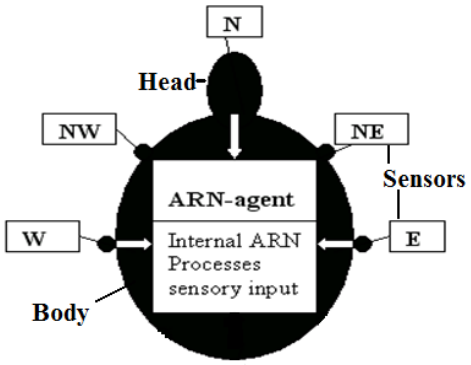


Fig. 3. Location of the ARN-agent (cytobot) sensors around its perimeter.

The calculated angle interfaces with the remaining subnetworks at pool AE. Pool AE is the output of an OR gate (motif 8) where inputs from either the weighted direction network or the chaotic angle network activate AE. AE also forms the inhibitory input of a SWITCH (motif 7) where the presence of chemical in MA and absence in AE activates pool MB of the master oscillator. In an actual organism receptors are set around the cell perimeter and direct movement appropriately. In this simulation, for simplicity, a count of the number of time steps "n" that MA contains the token unit is processed to gain the new heading "h" relative to the agents' current heading "c" using (4). Thus if the number time steps is

120 and the agent is facing north, then the current heading would equal 0 and the new heading would equal 30.

$$h = ((n - 90) + c) \bmod 360 \quad (4)$$

E. The Chaotic Network

The chaotic network, as shown in Fig. 2, is responsible for generating pseudo random angles which agents use to perform the foraging mode tumble behavior. It is a networked implementation of a Logistic Map, see (5). Without prior knowledge of the initial conditions the output of the logistic map is unpredictable, while it is deterministic with prior knowledge. Therefore, the series cannot be described as truly random but as pseudo random. Its output has long been proposed as a pseudo-random number generator [11] and it has been successfully used in this capacity by several researchers [12]. The probability density distribution of the Logistic Map is non-uniform and is described in [12]. When λ=4 the distribution is "U" shaped, with higher probability of values closer to the minima and maxima of X and a symmetric distribution at the midpoint. The general shape of the distribution is invariant for the complete range of state variables from 0 to 1.

$$X_{n+1} = \lambda X_n (1 - X_n) \quad (5)$$

Where:
X_n= state variable of value $0 \leq X_n \leq 1$
λ= system parameter of value $1 \leq \lambda \leq 4$

To implement the logistic map a number of motifs are combined including multiple branch motifs (motif 6- KB activates KD and KE), PFLs (motif 3- a multi component PFL exists where KA leads to activation of KE, which results in the activation of KA) and NFLs (motif 4- KA activates KD which in turn inhibits KA). At the start of the simulation, pools KA and KB are initialized to the same random value (a unique number for each cytobot) between 0 and 1 (to 5 decimal places). This represents the first value of X of (5). All the other pools are initialized to 0 with the exception of the static pools KI and RK whose initial values are 360 and 1 respectively. Reaction K2 is responsible for generating each new value of X and has a forward and reverse rate of 4 (the logistic map exhibits chaotic behavior when λ is 4). The connection between KA and K2 has a weight of 1 and that between K2 and KB has a weight of 2. The remaining series of reactions function to copy the value of X 3 times, where 2 copies serve as the new initial values of KA and KB and one participates in the final output of the network at KH. KI has a fixed value of 360 which allows the network to convert the pseudo random number at KH to an angle value between 0 and 360 at reaction K0. However, reaction K0 cannot proceed until all 11 pools that inhibit it are empty. These inhibitory connections (motif 2) ensure that random angles are not output while the agent is in starvation mode, and that pool AE is empty before adding more chemical. The ARN implementation of the Logistic Map was tested against the recursive relation (5), the details of which are given in our previous work [13]. The frequency distribution gained from the ARN is identical to that of (5).

V. METHODOLOGY

In the following experiments, multiple cytobots are instantiated and used to model aggregating and vegetative Dd cells, where each cytobot represents a cell. Two sets of experiments are performed: aggregation (AG) and foraging to aggregation (AGF). In the AG experiments (AG1-10 of Table 2) only the aggregative phase is modeled, where each experiment is performed at varying population densities of cytobots (p) and different ranges of detection of cAMP (r). In experiments AGF3 and AGF8 the vegetative and aggregative phases are simulated (and the transition between these phases) using the same population density and range as experiments AG3 and AG8 respectively. The emergent patterns, numbers of mounds, and length of time to complete phases is examined and compared in both sets of experiments.

In the AGF experiments, the environment is initialized with a radially outward decreasing gradient of food as described by (6), where x and y are on the horizontal plane. Here the cytobots remain in foraging mode until the food resource is depleted and starvation mode is triggered. In a real environment food is non-uniformly distributed, may be regenerated and can move (in the case of bacterial prey). Thus, this setup is highly simplified, but is comparable to other simulations [14].

$$z = \sqrt{x^2 + y^2} \quad (6)$$

The results of the AGF experiments are compared with those of the AG experiments, where, the environment never contains food, thus agents immediately enter and stay in the starvation mode. The agents' behavior is initially explored at biologically realistic p and r values and compared with the behavior of the actual organism and other simulations. These parameters are then extended outwith the biological range in order to examine the emergent properties of the system. The Cytobots move within a simulated 2D environment of area 5.06 mm^2- approximately half the maximum recorded aggregation territory [9]. Each pixel represents 4.5 μm and the grid is 500 × 500 pixels, giving a total area of 5.06 mm^2. In nature, aggregating Dd cell densities are typically 250 per mm^2 to $1x10^4$ per mm^2 [9]. Due to the computational resources required to manage a population of cytobots within the upper range, two cell densities of 250 agents per mm^2 (1250 agents) and 150 per mm^2 (750 agents) were chosen. In all experiments, the agents are initialized at random positions within the simulated environment. Foraging cytobots consume food at each passing location, while those in starving mode emit a cAMP signal at equal strength around their circumference into the environment. This cAMP signal is detected by other agents within or equal to r. If an agent in foraging mode detects cAMP it will switch to starvation mode behavior.

In these experiments a range of r values are explored (see Table 2), including that of real cells of 1, 0.5, and 0.1 mm [8].

The actual cAMP signal degrades linearly with increasing distance (d) from the emitting cell. Each agent detects the cAMP signal of all starving cells within or equal to r, and a total value for each direction is calculated. Each cycle represents 1 minute of time. In this time the agent moves 9μm- a distance which corresponds to that reported in the literature [15]. Therefore, after 1 hour motion the agent travels a distance of 540μm. In reality there are always remaining cells that do not aggregate, and thus the simulation runs until 95% of agents are at a distance of less than 0.1mm from their nearest neighbor.

VI. RESULTS

The results for all 12 experiments are given in Table 2. Each experiment was performed 100 times. In experiments AG8-10 and AGF8 the value of r and d are within the ranges for real Dd cells. These experiments are used to compare the behaviors and aggregation time with the values for real Dd in the literature. In experiments AG8-9 and AFG8 mound formation completes within the range reported for the actual organism of 9-13 hours [14, 16]. These results are comparable with other work. For instance, in [14] the aggregation time reported was 11.6 hours for a cell density of 200mm^2. In experiment AG10, the population never satisfied the criteria for completion of mound formation. The likely explanation is firstly because the simulation does take into account glycoprotein's which allow aggregating cells to attach together on contact. Furthermore, because r is small, fewer agents are detected by each cytobot. Thus higher numbers of momentarily larger clusters with higher attraction strength go undetected and quickly dissipate. In the AG experiments increasing p by 100mm^2 the number of mounds formed at each r decrease with the exception of experiment AG6. This is not surprising, as denser populations should have more chance of interacting, and thus form fewer clusters, but with higher numbers of agents. Similarly, decreasing r results in a general increase to the number of mounds formed at both values of p. In the AGF experiments, agents generally focus on consuming food in each remaining highest concentration area (see Fig. 4K-L). Having consumed almost all the food, agents begin switching to starvation mode (Fig. 4M). In these experiments the number and location of resulting mounds differs from that of the AG experiments at the same values of r and d. For example experiment AG8 results in 4.3 mounds while AGF8 results in 6.8 mounds with a general shift in mound formation further away from the centre of the environment (as shown in Fig. 4).

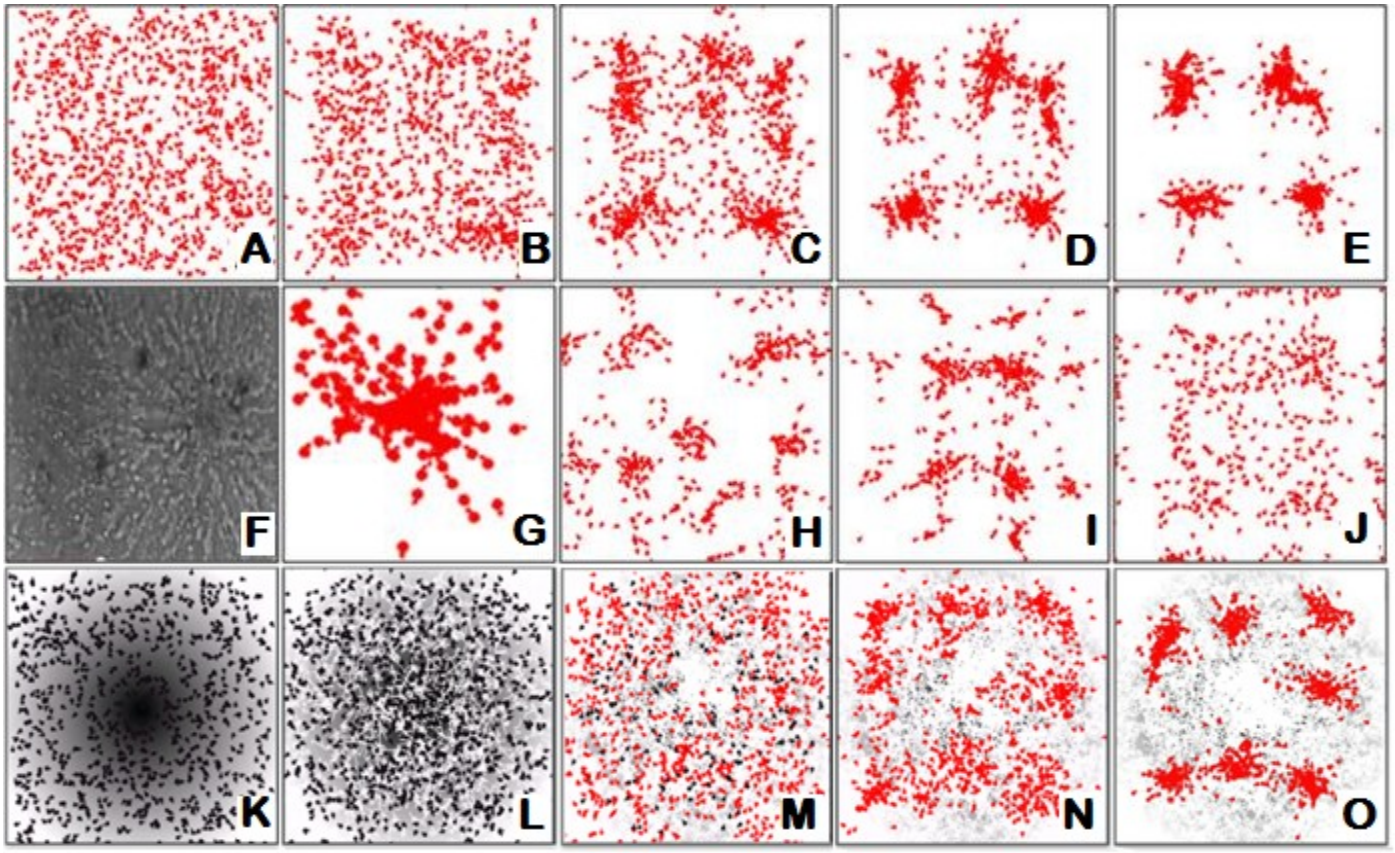


Fig. 4. Screenshots of the Dd simulation. Dots represent the cytobots (black- vegetative and red- aggregative cells), and greyscale color represents the food distribution. A-E: Cytobot aggregation experiment AG8 at A- 1hr, B- 2hr, C-5hr, D- 8hr, E- 10hr; Image F- real Dd cells aggregating; G- Lower right hand corner of image C demonstrating streaming behavior; H-J Shows pattern formation; K-O Cytobot experiment AGF8 at K-0hr vegetation, L-4hr vegetation, M-transition to aggregation 0hr aggregation, N-5hrs aggregation, O-10hr aggregation.

Diagram F courtesy of T, Gregor, Laboratory for the Physics of Life, Princeton University, 2013 Used with permission.

The likely explanation for this is that, at the time of switching to aggregation, the majority of cells had been forced outward, toward the next remaining highest concentration of food. Emergent behaviors and clustering patterns similar to the biological organism were also observed. As previously discussed, the cytobots are polarized. Implementing the agents in this way allowed us to observe whether or not the previously described streaming behavior occurs. A close-up of the right-hand corner of screenshot C is shown in Fig. 4G showing agents beginning to form a cluster. The protruding head of each agent can be seen clearly, where each lines up its head to the rear of another agent and forms a stream. As can be seen in Fig. 4F, this is very similar to the streaming behavior in real cells of Dd. Other emergent patterns occurred during different experiments including spirals (Fig. 4H), symmetric patterns (Fig. 4I), and waves (Fig. 4J).

TABLE II CYTOBOT SIMULATION RESULTS

No.	Density (p) per mm^2	Range (r) in mm	Mean No. of mounds; (σ)	Aggregation Phase Mean time in Hours; (σ); *Literature 9-13 hours
AG1	150	5	1 (0)	8.98 (0.09)
AG2	150	2.5	4 (0.31)	9.63 (0.17)
AGF3	150	1	5.9 (1.16)	9.47 (0.65)
AG3	150	1	5.2 (0.82)	9.92 (0.34)
AG4	150	0.5	8.4 (1.19)	10.23 (0.59)
AG5	150	0.1	14.2 (2.36)	10.6 (1.82)
AG6	250	5	1 (0)	8.95 (0.11)
AG7	250	2.5	1 (0)	9.6 (0.20)
AGF8	250	1	6.8 (1.81)	9.71 (0.87)
AG8	250	1	4.3 (0.37)	10.05 (0.58)
AG9	250	0.5	6.7 (1.62)	12.65 (1.94)
AG10	250	0.1	-	-

VII. CONCLUSIONS

The results of the Dd experiments presented above show that ARN-agents are able to simulate individual behaviors, stigmergic interactions and emergent behaviors of unicellular organisms by combining structural motifs found in real biochemical networks. This highlights a potential use as a means to simulate groups of interacting cells - such as a bacterial colony or tissue component within a multicellular organism, including the effects of disease (e.g. faulty gene expression) and pharmaceuticals on global behavior. The results demonstrate the parallels between ARN agents and the biological counterpart from which they were inspired. Like amoebae, their internal network of spatially distributed dynamic chemical species allows them to autonomously coordinate and direct their movement, recognize and respond to patterns in the environment, and produce high-level behavior. This application demands an internal control system which can function without reference to other agents within the environment which are operating in parallel.

In future work, it is intended to further investigate ARN-agents in biological simulations. Importantly a study into ways in which the pathways of ARN-agents can be evolved and how such agents can learn and adapt to the environment autonomously.

REFERENCES

[1] P. Dittrich, J. Zeigler, and W. Banzhaf, "Artificial Chemistries- a reivew". Artifi. Life vol. 7, no. 3, pp. 225-275, 2001.

[2] C. E. Gerrard, J. McCall, G. M, Coghill, and C. Macleod, "Artificial Reaction Networks," Proceedings of the 11th UK Workshop on Computational Intelligence, Manchester, UK, pp. 20-26, September 2011.

[3] C. E. Gerrard, J. McCall, G. M, Coghill, and C. Macleod, "Temporal patterns in Artificial Reaction Networks," Proceedings of The 22nd International Conference on Artificial Neural Networks Lausanne, part 1, vol. 7552, pp. 1-8, September 2012.

[4] C. E. Gerrard, J. McCall, G. M. Coghill, and C. Macleod, "Adaptive Dynamic Control of Quadrupedal Robotic gaits with Artificial Reaction Networks," Proceedings of The 19th International Conference on Neural Information Processing Doha, vol. 7663, part 1, pp. 280-287, November 2012.

[5] D. Bray. "Protein molecules as computational elements in living cells," Nature, vol. 376, no. 6538, pp. 307-12, 1995.

[6] J. J. Tyson, and B. Novák. "Functional motifs in biochemical reaction networks." Annu. Rev. Phys. Chem., vol. 61, pp. 219-240, 2010.

[7] P. Kreyssig, P Dittrich, "Reaction flow artificial chemistries," Advances in Artificial Life, ECAL 2011: Proceedings of the Eleventh European Conference on the Synthesis and Simulation of Living Systems, pp. 431-437. MIT Press, 2011.

[8] C. P McCann, P. W. Kriebel, C. A. Parent, W. Losert, "Cell speed, persistence and information transmission during signal relay and collective migration," J. Cell. Sci., vol. 123, pp. 1724-1731, 2010.

[9] J. C. Dallon, and H. G. Othmer, "A discrete cell model with adaptive signaling for aggregation of Dictyostelium discoideum," Phil. Trans. R. Soc. B, vol. 352, no. 1351, pp. 391–417, 1997.

[10] R. H. Kessin, "Making Streams,"Nature, vol. 422, pp. 481-482, 2003.

[11] S. M. Ulam and J. von Neumann, "On combinations of stochastic and deterministic processes," Bull. Amer. Math. Soc, vol. 53, pp. 1120, 1947.

[12] V. Patidar, K. K. Sud, and N. K Pareek, "A Pseudo Random Bit Generator Based on Chaotic Logistic Map and its Statistical Testing," Informatica, vol. 33, pp. 441-452, 2009.

[13] C. E. Gerrard, J. McCall, G. Coghill and C. Macleod. "Artificial Chemistry Approach to Solving Search Spaces using Artificial Reaction Network Agents", Presented at the IEEE Congress in Evolutionary Computing, Cancún, Mexico, 2013, in press.

[14] M. Becker. "Simulation model for the whole life cycle of slime mould *Dictyostelium Discoideum,"* Proceedings of the European conference on modeling and simulation, pp. 247-253, 2010.

[15] J. L. Rifkin. and R. R. Goldberg. "Effects of chemoattractant pteridines upon speed of D. discoideum vegetative amoeba. Cell Motility and the Cytoskeleton," vol. 63 no.1, pp 1-5, 2006.

[16] D. A. Cotter, T. W. Sands, K. J. Virdy, M. J. North, G. Klein, M. Satre, "Patterning of development in Dictyostelium discoideum: factors regulating growth, differentiation, spore dormancy and germination," Biochem. Cell Biol., vol. 70, no. 10-11, pp. 892-919, 1992.

Paper 5: ‘Artificial Chemistry Approach to Exploring Search Spaces using Artificial Reaction Network Agents’.

Reference:

Gerrard, C., McCall, J., Coghill, G., and Macleod, C. (2013) ‘Artificial chemistry Approach to Exploring Search Spaces using Artificial Reaction Network Agents’, Congress on Evolutionary Computation (CEC)’, IEEE, Cancún, pp.1201-12.

Artificial Chemistry Approach to Exploring Search Spaces Using Artificial Reaction Network Agents

Claire E. Gerrard, John McCall, Christopher Macleod,
IDEAS Research Institute,
Robert Gordon University,
Aberdeen, Scotland.
c.e.gerrard@rgu.ac.uk, j.mccall@rgu.ac.uk,
chris.macleod@rgu.ac.uk

George M. Coghill,
Department of Computing Science,
University of Aberdeen,
Aberdeen, Scotland.
g.coghill@abdn.ac.uk

***Abstract*— The Artificial Reaction Network (ARN) is a cell signaling network inspired representation belonging to the branch of A-Life known as Artificial Chemistry. It has properties in common with both AI and Systems Biology techniques including Artificial Neural Networks, Petri Nets, Random Boolean Networks and S-Systems. The ARN has been previously applied to control of limbed robots and simulation of biological signaling pathways. In this paper, multiple instances of independent distributed ARN controlled agents function to find the global minima within a set of simulated environments characterized by benchmark problems. The search behavior results from the internal ARN network, but is enhanced by collective activities and stigmergic interaction of the agents. The results show that the agents are able to find best fitness solutions in all problems, and compare well with results of cell inspired optimization algorithms. Such a system may have practical application in distributed or swarm robotics.**

Keywords*— *Artificial Reaction Networks; Artificial Chemistry; Swarm Robotics

I. INTRODUCTION

Single celled organisms display an astonishing array of complex behaviors. Some can avoid light with photo-sensitive spots; some actively hunt prey; while others can build protective shelters [1]. Such behaviors improved these organisms' chances of survival through the process of natural selection. In recent years a growing body of research has illuminated the remarkable capabilities of single cells to store and process information [2, 3]. The mechanisms involved are quite different from those of a digital computer. Within a cell, the current state is represented as a set of spatially distributed concentrations of chemical species. This data is processed by vast networks of chemical reactions termed cell signaling networks (CSNs). In this way, cells are able to respond to current environmental conditions, communicate with other cells, and perform internal self maintenance operations.

Several researchers have highlighted the processing capabilities of these networks [2, 4, 5] and similarities to Artificial Neural Networks (ANNs) [2, 5]. For example, it has been demonstrated that such networks can perform Boolean and fuzzy logic functions and are equivalent to a Turing machine [2, 4]. Furthermore, CSNs contain topological features such as feedback loops and interconnectivity, thus forming highly complex systems [2, 5, 6].

It is possible to abstract the computational properties of such chemical processing to create a type of model called an Artificial Chemistry. Artificial Chemistry is a subfield of A-Life, and in its broadest sense, it describes man-made systems which are similar to real chemical systems [7]. In previous work, a new Artificial Chemistry representation of CSNs- the Artificial Reaction Network (ARN) was introduced and investigated as a means to control limbed robots [8, 9, 10].

Our first aim is to show that an ARN network can be instantiated and used as the internal control system for multiple instances of cell-like autonomous distributed agents. Like biological cells these agents react to their environment, and stigmergically communicate to facilitate collective emergent behavior. Our second aim is not to present a new optimization algorithm; rather it is to show that these agents can perform a range of useful search behaviors in a variety of situations, and that their search strategy can compare to that of established optimization algorithms using similar cell inspired strategies. The agents are placed within a simulated environment with the task of finding the global minima of a set of well-known benchmark problems. The search spaces chosen are not high dimensional, but chosen to reflect problems which situated robotic agents could perform in real world environments.

The paper is structured as follows: section 2 provides an overview of the ARN representation; this is followed by an overview of the ARN agents in section 3. The experimental details are discussed in section 4 followed by results in section 5. Finally section 6 presents the conclusions.

II. ARTIFICIAL REACTION NETWORKS

A full account and verification of the ARN representation can be found in our work [8, 9, 10]; thus to preserve space only a brief summary is provided here.

The ARN comprises a set of networked reaction nodes (circles), pools (squares), and inputs (triangles) as shown in Fig. 1. Each pool stores the current available chemical species concentration (avail); this concentration represents data within

the system. Thus, the complete set of pool concentrations at time t, corresponds to the current state of the system. Inputs are a special type of pool, the only difference being that they are not updated by flux at each time step, and are used to represent continuous concentrations, for example, environmental inputs or enzymes. Each circle corresponds to a reaction unit, representing a reaction between a number of chemicals. Data is processed by reaction nodes transforming incoming pool values to connected outgoing pool values. Connections symbolize the flow of chemical into and out of reaction units and their weight (w) corresponds to reaction order. Connections provide the facility to create complex control structures using combinations of inhibitory and excitatory connections. Fig. 1 shows the reaction between species A and B to produce species C. At time interval Δt, each reaction unit's temporal flux value is calculated by applying Euler's approximation to the differential rate equation as shown in (1).

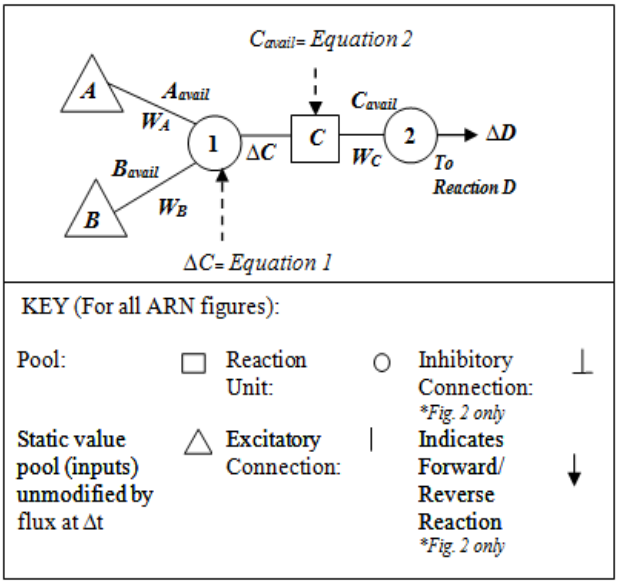


Fig. 1. The Artificial Reaction Network representation.

$$\Delta C = \left(K_{f(C)} \left(A_{avail}^{W_A} B_{avail}^{W_B} \right) - K_{r(C)} \left(C_{avail}^{W_C} \right) \right) \Delta t \quad (1)$$

$$C_{avail} = C_{avail} + \left(\Delta C - \frac{W_C}{W_C + \alpha} \Delta D \right) \quad (2)$$

Where:
A, B, C, D = Species Concentrations
W = Reaction order (weight)
avail = Available species concentration
K_f = Forward rate constant
ΔC = Change in species concentration C
K_r = Reverse rate constant
α=sum of other incoming weights

This value is then used to update the current concentration of each reaction's connecting pools as shown in (2). Pools may asymptotically approach 0, and thus below a particular threshold a pool is considered empty and its value set to zero. A reaction step may proceed if it meets its preconditions. Preconditions are met if incoming inhibitory pools are inactive, and incoming excitatory pools are active. In a similar way the completion of a reaction step will fulfill a number of post conditions, which depend on the parameters and connections of the reaction step.

III. ARN Controlled Agents

In the following experiments a number of autonomous ARN controlled software agents termed "cytobots" ("cyto" from Greek for cell, and "bot" from robot) are created and initialized within an artificial environment containing a nutrient landscape. The cytobots task is to find the maximum food level by moving around within this simulated environment in as few evaluations (reading the value of food at the current position) as possible. Similar to the way in which a CSN acts as the control system to a cell, the behavior of each cytobot is controlled by its own instance of an ARN network. In this way, the ARN directs the agent's movement, enables the agent to react to situated environmental patterns, and allows it to stigmergically communicate with other cytobots to contribute to higher level function. The cytobot ARN network was designed to produce two simple behavioral modes: foraging and starvation, both are based on the movement patterns of unicellular organisms. Cytobots forage by performing a biased random walk behavior while consuming food at each passing location. This pattern of movement is exemplified by the bacteria *Escherichia coli* (*E. coli*), where foraging cells alternate periods of runs (forward motion) and random redirections known as tumbles. By comparing concentrations of attractants and repellants in a temporal fashion, the organism is able to reduce the frequency of tumbles up concentration gradients of attractants, and down gradients of repellants, thus providing the bias. This behavior implements a type of optimization where biased periods of movement in the direction of attractants lead to overall travel toward more favorable conditions [11].

The starvation behavior is based on the pattern of motion displayed by starving cells of the cellular slime mould *Dictyostelium discoideum* (*D. discoideum*). During the vegetative stage of *D. discoideum*, cells move up gradients of folic acid secreted by its bacterial prey. When the food resource has been depleted, the amoebae begin to starve and enter the aggregation phase of their life cycle. During aggregation, starving cells secrete cAMP (cyclic adenosine monophosphate) which serves as a signal to attract surrounding amoebae towards a central location [12]. In this simulation, when the cytobots enter starvation mode, the level of food surrounding the agent represents corresponding levels of attractant cAMP. When a cytobot travels over areas of low or zero food it enters the starvation mode. Instead of turning in a random direction, the new direction is weighted toward higher concentrations of food within its surrounding area. This behavior forces exploration of unexplored search space because previously visited positions have a food level of 0.

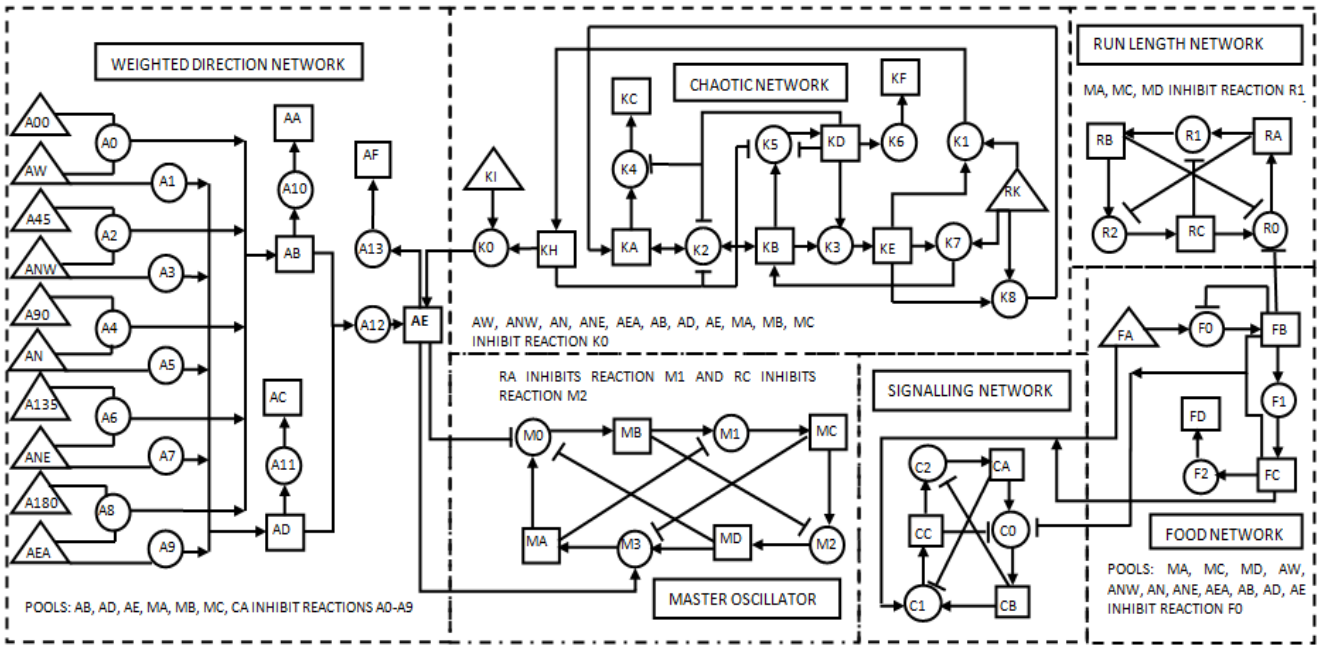


Fig. 2. The cytobot ARN network. Each cytobot is controlled by an instance of this network. The network is composed of 6 subnetworks

Consumption of environmental food therefore serves as a stigmergic signal, where agents are inclined to move up the nutrient gradient created by their foraging activities. The cytobot ARN was designed to perform the starvation and foraging behaviors described above and is composed of 6 subnetworks as shown in Fig. 2. Each subnetwork contributes a functional aspect to either or both starvation and foraging behaviors. The subnetworks are discussed below.

A. The Master Oscillator

The master oscillator functions to synchronize all the outputs from all the other subnetworks together and is what each agent references at each time step to ascertain its current behavior. It is a simple closed loop, with a token unit of chemical cycling around it. It consists of 4 reaction units: M0, M1, M2, and M3 (all with reaction rate of 1) and 4 pools MA, MB, MC and MD. Each pool activates one of three behaviors, and for every time step that a particular pool contains the token unit, its corresponding behavior is performed. Pool MA activates turn, MC activates run and pools MB and MD activate stop. If these pools were switches to motor actuators on a simple wheeled robot, pool MC would switch on all wheel motors, while pool MA would switch on wheel motors on the left side only, thus turning the robot. The remaining pools would act as off switches. The other subnetworks inhibit or excite the reaction units of the master oscillator to allow or prevent chemical flow. The number of time steps that a chemical is present in a particular pool indicates the length of time that a particular behavior is performed. Thus if pool MC contains a chemical for 10 time steps, then the agent will move forward for 10 time steps; similarly if this were pool MA, the agent would turn for 10 time steps.

B. The Food Network and The Run Length Network

The food network senses the level of food within the environment and connects to the run length network to modify the number of steps forward according to the level sensed. The value of food at the cytobots' current position is stored at input pool FA. The forward rate of reaction node F0 is 1, thus the content of FA is transferred to pool FB in a single time step. The presence of chemical in pool FB inhibits the run network reaction R0 for a number of time steps according to the level of food (by setting forward rate of unit F1 to 1 and weight to 0 this can be an exact correlation). This in turn stops pool RC in the run length network from emptying. Pool RC inhibits reaction M2 of the master oscillator thus preventing pool MC from emptying for the same number of time steps. As discussed previously, the number of time steps which pool MC contains the token unit represents the number of time steps to move forward.

C. The Signaling Network

The signaling network functions as a switch between starvation and foraging mode. Low food levels trigger the starvation response and allow the weighted direction network to control each new angle. Sufficient food will switch off the weighted direction network and allow the chaotic network to control each new angle. It is a simple closed loop with a token unit of chemical flowing around it. Pool CA acts as a switch between foraging and starvation behavior, where the presence of chemical in CA inhibits the weighted direction network-while its absence switches on the weighted direction network; this in turn inhibits the chaotic network, as shown in Fig. 2. In this component, all reaction units have a forward flux of 0.5; which ensures a minimum number of time steps for each behavior.

D. The Weighted Direction Network

The weighted direction network senses food within the agents' immediate environment and calculates a tumble angle which is weighted toward higher food levels. This network interfaces with the environment via a number of receptor pools (AW, ANW, AN, ANE, AEA) which sense the level of food around the cytobot. These pools represent receptors positioned at points around the front of its perimeter, allowing the agent to travel in a similar way to that of a polarized biological cell. For example, during the aggregation phase of their life cycle, *D. discoideum* cells are polarized, and one side becomes the leading edge which always faces in the direction of travel [12]. For each receptor input pool, there is a static pool containing a fixed level of chemical in correspondence to its direction. Directions start from AW (west) with a corresponding numeric value of 0 (A00) and progress in 45 degree steps through each direction to east. As the receptor positions around the agent are fixed, directions are always relative to that in which the agent is facing. All connections have a weight of 1 with the exception of the connection between pool AD and reaction A12 which has a weight of -1. This negative connection raises the sum of food detected in pool AD to -1, which multiplied by AB, allows an average angle to be calculated.

The calculated angle interfaces with the remaining subnetworks at pool AE. In an actual organism receptors are set around the cell perimeter and direct movement appropriately. In this simulation, for simplicity, a count of the number of time steps that MA contains the token unit is processed to gain the turn angle relative to the agents' current heading using (3). Thus if the number time steps is 120 and the agent is facing north, then the current heading would equal 0 and the new heading would equal 30.

$$h \equiv ((n-90)+c) \bmod 360 \tag{3}$$

Where:
h= new heading
n = count of time steps pool MA contained chemical
c = current heading

E. The Chaotic Network

The chaotic network, as shown in Fig. 2, is responsible for generating pseudo random angles which agents use to perform the foraging tumble behavior. It is a networked implementation of a Logistic Map, see (4). Without prior knowledge of the initial conditions the output of the logistic map is unpredictable, whereas with prior knowledge it is deterministic- therefore the series cannot be described as truly random but as pseudo random. Its output has long been proposed as a pseudo-random number generator. Ulam and von Neumann [13] were the first to examine this and it has been successfully used in this capacity by several researchers [14]. The probability density distribution of the Logistic Map is non-uniform and is described by (5) [14]. When λ=4 the distribution is "U" shaped with higher probability of values closer to the minima and maxima of X and symmetric distribution at the midpoint. The general shape of the distribution is invariant for the complete range of state variables from 0 to 1.

$$X_{n+1} = \lambda X_n (1 - X_n) \tag{4}$$

Where:
X_n= state variable of value $0 \le X_n \le 1$
λ= system parameter of value $1 \le \lambda \le 4$

$$P(X) = \frac{1}{\pi\sqrt{X(1-X)}} \tag{5}$$

Where:
P(X) = probability of X occurring

The chaotic network component, as shown in Fig. 2, operates in the following manner. At the start of the simulation, the pools KA and KB of each cytobots' chaotic network are initialized to the same random value between 0 and 1 (to 5 decimal places). This represents the first value of X where X is the state variable of (4). All the other pools are initialized to 0 with the exception of the static pools KI and RK whose initial values are 360 and 1 respectively. Reaction K2 is responsible for generating each new value of X and has a forward and reverse rate of 4 (the logistic map exhibits chaotic behavior when λ is 4). The connection between KA and K2 has a weight of 1 and the connection between K2 and KB has a weight of 2. The remaining series of reactions function to copy the value of X 3 times, where 2 copies serve as the new initial values of KA and KB and the remaining copy participates in the final output of the network at KH. Static pool KI has a fixed value of 360 which in reaction K0, allows the network to convert the pseudo random number at KH to an angle value between 0 and 360. However, reaction K0 cannot proceed until all 11 pools that inhibit it are empty. These inhibitory connections ensure that random angles are not output while the agent is in starvation mode, and that pool AE is empty before adding more chemical.

The ARN implementation of the Logistic Map was tested against the recursive relation shown in (4). The results generated for (4), were obtained using Matlab, where λ=4, initial X = 0.927725, and iterated 100000 steps. The complete range of state variables between 0 and 1 were divided into 100 equal subintervals and the frequency of occurrence of each subinterval interval was plotted. Similarly, the chaotic network component of the ARN was run for 100000 cycles, using the same parameters of X and λ. These results were processed in the same way and are shown in Fig. 3. The frequency distribution gained from the ARN is identical to that obtained using matlab and by other researchers using the same parameters [14]. The same comparison was repeated 100 times at different values of X, and the ARN consistently produced the same values as (4).

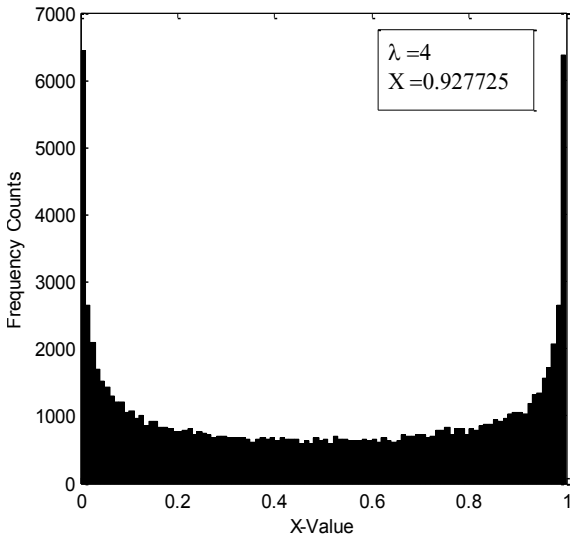


Fig. 3. Frequency distribution for each value of X resulting from the chaotic network when the first value of X is 0.927725 and λ=4

IV. METHODOLOGY

In the following experiments, cytobot agents are applied to the task of finding the minima in a number of benchmark 2D optimization problems. These are the following functions: Rosenbrock, Peaks, Inverted sinc, and Bowl (see Table 1 for formulae, domains, and minima). Three experiments were performed for each function, where each uses either 1, 3 or 6 cytobots and is performed 100 times. The task of the agents is to find the minima of the functions within as few evaluations (reading value of food at current x, y coordinate) as possible. The range of output values for each of the functions represents the concentration of food (also the fitness of an agent at that point) within a simulated environment. Values approaching the minima represent higher food levels, and values approaching the maxima represent lower food levels. The simulated environment consists of a 2D area of 400 x 400 pixels. A scaling factor is used to map the domain to the actual dimensions of the simulation, e.g. Rosenbrock domain of [-2, 2] mapped to a simulation space of [-200, 200] by a scaling factor of 100. For display purposes, a corresponding grayscale color is used to show the distribution of food within the environment as displayed in the screenshot of the simulation in Fig. 4. Each agent consists of a token to mark its current position and an instance of an ARN network, as discussed in section 3. At the start of each experimental run, each agent's ARN network is initialized as described in section 3, and each is positioned at random x,y coordinates within the search space. The agents undergo alternating phases of "searching" and "repositioning," for a number of cycles until one reaches a position of within 0.04 of the global minima of the function. This value was chosen as it is within 1% of the global minima for all the functions used. The high level pseudocode describing the searching and repositioning phases is provided in Fig. 5. Searching is characterized by the 2 ARN controlled behaviors- foraging and starvation, as described in section 3. In each search phase each agent performs a total of 3 moves (3 evaluations of the environment). The length of a run corresponds to the number of pixels a cytobot moves forward and is subject to the output from the run length network. After each tumble, and before moving forward, the food level at the current position is input into the ARN network as described previously. The agents travel at a speed of 1 pixel per time step, thus the number of time steps produced by the run length network corresponds directly to the number of pixels the agent moves forward. As a cytobot travels, the food at each passing position is consumed and its path within the simulation is represented in black (as shown in Fig. 4). During the search phase, a central control unit, external to all cytobot agents, keeps track of each cytobots best fitness and the coordinates of that value.

Let $A = \{a_0, \cdots, a_N\}$ equal the set of all agents

$$f_{tot} = \sum_{a=0}^{N} f_a \tag{6}$$

Then $\forall a \in A \setminus \{a^*\}$

$$f_{ra} = \frac{f_a}{f_{tot}} \tag{7}$$

$$d_{ax} = x_a - x_{a^*} \tag{8}$$

$$d_{ay} = y_a - y_{a^*} \tag{9}$$

$$p_x = x_{a^*} + \left(\sum_{a=0}^{N} d_{ax} \times f_{ra} \right) \tag{10}$$

$$p_y = y_{a^*} + \left(\sum_{a=0}^{N} d_{ay} \times f_{ra} \right) \tag{11}$$

$$x_a = rand(r) + p_x \tag{12}$$

$$x_a = rand(r) + p_y \tag{13}$$

Where:

a*= agent with highest fitness
f_a= fitness of agent a_n
f_{tot}= total fitness of all agents
f_{ra}= ratio of agent a_n fitness to f_{tot}
x_{a^*}= the x coordinate of a*
y_{a^*}= the y coordinate of a*
x_a= agent a_n x coordinate
y_a = agent a_n y coordinate
d_{ax}=difference between x_a and x_{a^*}
d_{ay}= difference between y_a and y_{a^*}
p_x=total of all d_{ax}
p_y= total of all d_{ay}
rand(r)= a random value within a defined radius r

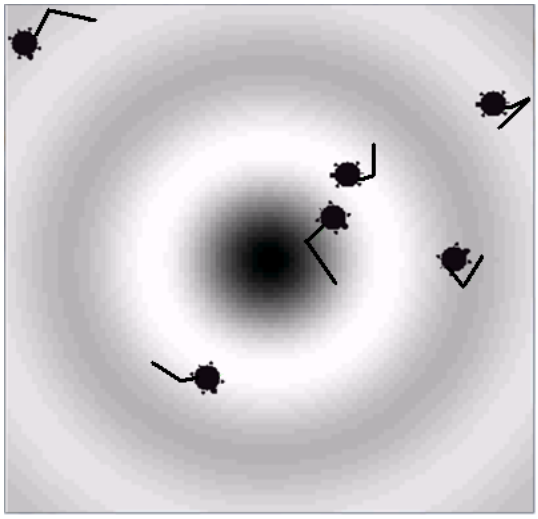


Fig. 4. A screen shot of the simulation showing 6 cytobots in the inverted sinc search space. The greyscale color represents the food distribution.

```
INITIALISE cytobots
WHILE (best fitness outwith 0.04 of global minima)

  START search phase
    WHILE (more searching phase moves)
      FOR each agent start searching phase
        Turn agent
        Set receptor pools of food network
        Set receptor pools of weighted direction network
        Move agent forward
        IF (new food level > previous food level)
            Record fitness
            Record current position
        END IF
      END FOR
    ENDWHILE
  END search phase

  START reposition phase
  CALCULATE new central point P to reposition
  INITIALISE agents at new position
 END reposition phase
END WHILE
```

Fig. 5. High level pseudocode for each experiment

After completing the searching phase, agents switch to the repositioning phase. This phase is used to focus searching toward areas containing higher food levels and is inspired by stages of the life cycle of *D. discoideum*. Having depleted the level of nutrients within the immediate environment, *D. discoideum* cells begin to starve, and aggregate to form a slug. The slug travels in the direction of more favorable conditions by moving toward attractants such as light, warmth, and humidity. On finding a suitable location, it eventually forms a fruiting body which disperses spores within its immediate surroundings. The spores mature into cells, and begin foraging within the new environment [12]. When the cytobots enter the repositioning phase, the central control unit processes each agent's best fitness position to compute a new central point P, weighted in favor of higher fitness, as described by equations (6-13). Agents are then repositioned randomly within an area of radius r from point P to begin the next search phase. For the purposes of this simulation travelling to the new position was not modeled, as this does not affect overall behavior and would only occur if the cytobots were applied to real world environments.

V. RESULTS

The experimental results are displayed in Table 1. For each experiment, the average best, f_a, and best solution, f_b, for 100 independent runs are presented. The average number of evaluations and the standard deviation for all agents is displayed as "Avg Eval for all agents" and "Std Dev" respectively. The average number of relocations for each agent is presented in the final column as "Avg Reloc per agent".

In all experiments the cytobots were able to find the global minima. Cytobots performed best in Bowl and Rosenbrock functions, where, using 6 cytobots, the average number of total evaluations and relocations per agent respectively for Bowl was 56.4 and 2.1 and for Rosenbrock was 79.8 and 3.4. The cytobots performed least well in the Inverted Sinc search space, where the lowest number of total evaluations was 94.8 using 6 cytobots. In all the experiments, a slight increase in the number of cytobots generally results in a significant reduction in the total number of evaluations performed. This is most significant for Peaks where using 3 and 6 cytobots results in approximately 30% and 60% respective reductions in the total number of evaluations when compared to the results for 1 cytobot. The Mann Whitney U test was used to determine any significant (95% confidence) statistical difference in the total number of evaluations between experiments using 1 and 3 and 3 and 6 cytobots. In all experiments there was a significant difference between 1 and 3 agents, with the exception of the Inverted Sinc function. In Peaks there was a significant difference in all experiments, while in the Inverted Sinc there was no significant difference found. Thus increasing the number of cytobots from 1 to 3 both reduces the time to find the global minima and the number of evaluations, but this effect can be quickly reversed if too many cytobots are added.

The paths of agents through the search space indicate reasons for variation in results. In simple landscapes such as Bowl, agents descend steadily toward the minima, as shown in Fig. 6. Similarly in Rosenbrock, agents quickly descend to the narrow valley and are forced to steadily move along it by moving up the nutrient gradient created by the consumption of food, until finding the global minima. In Peaks, agents move from their initial positions and search many parts of the domain. Fig. 7 shows the agents' trajectories, one can see that peaks are avoided and troughs are pursued. However, if fewer agents are used they may quickly become trapped in local minima causing a significant rise in the number of evaluations. Increasing the number of agents by a small amount expands the amount of search space explored per cycle, and increases the chance of finding better solutions and/or leaving local minima. Another possibility is to increase the number of moves for each searching phase, thus allowing an agent to travel a sufficient distance to escape local minima. Similar solutions could be adopted in the Inverted Sinc function.

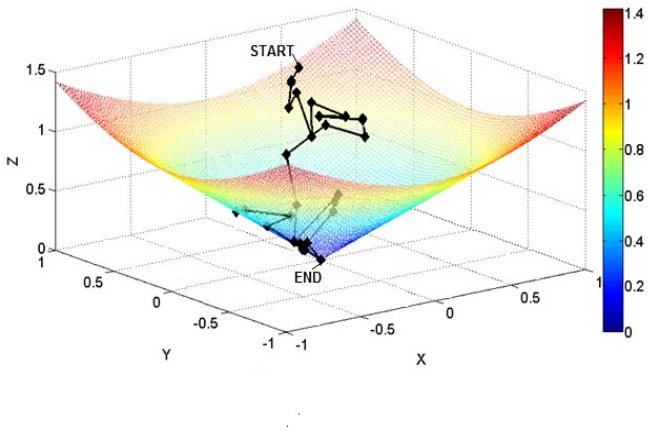


Fig. 6. Typical path of 1 cytobot in Bowl search space

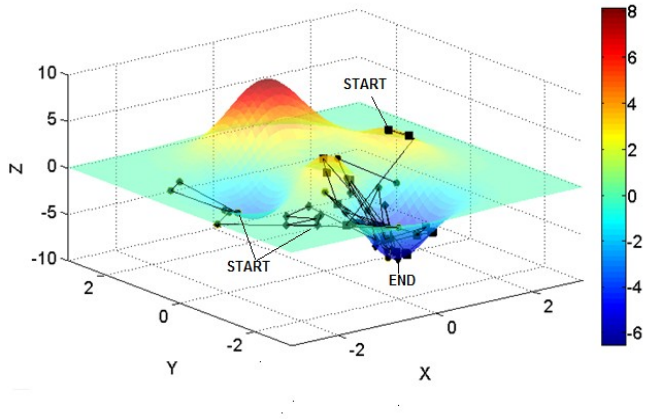


Fig. 7. Typical path of 3 cytobots in Peaks search space

TABLE I. CYTOBOT AGENTS 2D SEARCH SPACE RESULTS

Functions	**No. of Agents**	**Formulae**	**Avg (fa) Best and Best (fb) Results**	**Domain**	**f(x*) = f***	**Avg Eval for all agents (Std Dev)**	**Avg Reloc per agent**
Rosenbrock's Function	1	$f(x) = 100 \cdot (x_2 - x_1^2)^2 + (1 - x_1)^2$	f_a = 0.04 $f_b = 0$	$x_1, x_2 \in [-2,2]$	$f(1,1) = 0$	98.5 (16.3)	32
Rosenbrock's Function	3	As above	f_a = 0.03 $f_b = 0$	As above	As above	84.6 (9.5)	8.4
Rosenbrock's Function	6	As above	f_a = 0.01 $f_b = 0$	As above	As above	79.8 (4.8)	3.4
Bowl	1	$f(x) = \sqrt{x_1^2 + x_2^2}$	f_a = 0.03 $f_b = 0$	$x_1, x_2 \in [-1,1]$	$f(0,0) = 0$	81.9 (14.8)	26.3
Bowl	3	As above	f_a = 0.02 $f_b = 0$	As above	As above	64.2 (7.6)	6.1
Bowl	6	As above	$f_a = 0.02$ $f_b = 0$	As above	As above	56.4 (5.8)	2.1
Peaks	1	$f(x) = 3 * (1 - x_1)^2 \cdot \exp(-(x_1^2) - (x_2 + 1)^2)$ $- 10 \cdot (x_1/5 - x_1^3 - x_2^5) \cdot \exp(-x_1^2 - x_2^2)$ $- 1/3 \cdot \exp(-(x_1 + 1)^2 - x_2^2)$	f_a = -6.51 f_b = -6.55	$x_1, x_2 \in [-3,3]$	$f(0.23,-1.63)$ = -6.55	151.7 (59.2)	49.6
Peaks	3	As above	f_a = -6.51 f_b = -6.55	As above	As above	108.9 (17.4)	11.1
Peaks	6	As above	f_a = -6.52 f_b = -6.55	As above	As above	64.8 (5.7)	2.6
Inverted sinc function	1	$f(x) = -1 \cdot \left(\frac{\sin\left(\sqrt{x_1^2 + x_2^2}\right)}{\sqrt{x_1^2 + x_2^2}} \right)$	f_a = -1.04 f_b = -1	$x_1, x_2 \in [-10,$	$f(0,0) = -1$	163.5 (63.4)	53.5
Inverted sinc function	3	As above	f_a = -1.03 f_b = -1	As above	As above	109.5 (30.1)	11.2
Inverted sinc function	6	As above	f_a = -1.03 f_b = -1	As above	As above	94.8 (10.1)	4.2

These results are compared with other optimization algorithms inspired by behaviors of single celled organisms. For example, Passino developed the Bacterial Foraging Optimization Algorithm (BFOA), inspired by foraging behaviors, reproduction and dispersal events in the life cycle of *E. coli* [15]. Like the foraging behavior of the cytobots, movement is modeled as a biased random walk, where, after each random redirection, the cell moves forward a length according to current food levels. In a nutrient hill-climbing experiment (without swarming effects), 50 cells are initialized at random starting positions within a 2D search space. This search space is similar to Peaks but with 5 troughs and a domain of [30, 30]. Similarly to cytobots the cells tend toward valleys and avoid peaks. After 4 generations (4 reproductive steps), and moving 100 chemotactic steps (moves) between generations, the cells find the global minima.

Similarly in other work, Chen et al applied BFOA using 6 cells to the 2D Bowl function with domain [-5,5], and the global minima was found within 50 chemotactic steps [16]. In our experiments, 6 cytobots find the global minima after an average of 9.4 evaluations, which is the equivalent to 9.4 moves (or 9.4 chemotactic steps in the terminology of Chen et al). After adjusting for the difference in domain size, the numbers of moves are highly consistent for cytobots and the cells in BFOA. In other related work, Monismith et al created the slime mould optimization algorithm inspired by the life cycle of *D. discoideum* [17]. The state space is represented as a sparse mesh which cells populate and make modifications to, for example, deposit attractant. Using a combination of behavioral states inspired by the life cycle of *D. discoideum,* artificial cells perform local searches, and move to positions in favor of their personal best and the best fitness of their neighborhood. The slime mould optimization algorithm, like the cytobots, finds the global minima of the 2D Rosenbrock function.

VI. CONCLUSIONS

The results presented above show that the agents are able to find best fitness solutions in all problems, and match the performance of cell inspired optimization algorithms in similar search spaces. Increasing the number of agents by small increments (2 or 3), can half the number of function evaluations required to find the global minima. These experiments serve as a preliminary to implementing ARN systems to control real world distributed autonomous robotic agents. Such agents could be applied to similar search problems in real world environments, for example oil spill cleanup operations, where the objective is to travel to higher concentrations of oil, while consuming it at each passing location. The cytobots obviously do not compare directly with conventional optimization techniques like Genetic Algorithms, since they have a complex internal structure. However this is not their purpose and they may be much more effectively utilized as the control systems in autonomous agents. This application demands an internal control system which can function without reference to other agents within the environment which are operating in parallel. By modifying the environment, (in this case by consumption of food), the agents can stigmergically communicate and enhance and/or facilitate emergent behavior. The cytobots offer a unique range of abilities. Like cells, their internal network of spatially distributed dynamic chemical species allows them to autonomously coordinate and direct their movement, recognize and respond to patterns in the environment, and produce high-level behavior.

In future work, it is intended to further explore the AI applications of the cytobot agents, and later, to create swarms of cytobot robots with applications in real world environments.

REFERENCES

[1] B. J. Ford, "Are cells Ingenious?," The Microscope. vol. 52, no. 3-4, pp. 135-144, 2004.

[2] D. Bray, "Protein molecules as computational elements in living cells," Nature, vol. 376, no. 6538, pp. 307-12, July 1995.

[3] T. Nakagaki, H. Yamada, and A. Toth, "Maze-solving by an amoeboid organism,". Nature, vol. 407, no. 6803, pp. 470-470, September 2000.

[4] A. Arkin, J. Ross, "Computational functions in biochemical reaction networks," Biophys. J., vol. 67, pp. 560-578, August 1994.

[5] U. S. Bhalla, "Understanding complex signaling networks through models and metaphors," Prog. Biophys . Mol. Biol. vol. 81, no. 1, pp. 45-65, January 2003.

[6] B. Kholodenko, "Cell signaling dynamics in time and space," Nat. Rev. Mol. Cell Biol., vol. 7, no. 3, pp. 165-176, March 2006.

[7] P. Dittrich, J. Zeigler, and W. Banzhaf, W. Artificial Chemistries- a reivew. Artifi. Life vol. 7, no. 3, pp. 225-275, 2001.

[8] C. E. Gerrard, J. McCall, G. M, Coghill, and C. Macleod, "Artificial Reaction Networks," Proceedings of the 11th UK Workshop on Computational Intelligence, Manchester, UK, pp. 20-26, September 2011.

[9] C. E. Gerrard, J. McCall, G. M, Coghill, and C. Macleod, "Temporal patterns in Artificial Reaction Networks," Proceedings of The 22nd International Conference on Artificial Neural Networks Lausanne, part 1, vol. 7552, pp. 1-8, September 2012.

[10] C. E. Gerrard, J. McCall, G. M. Coghill, and C. Macleod. "Adaptive Dynamic Control of Quadrupedal Robotic gaits with Artificial Reaction Networks," Proceedings of The 19th International Conference on Neural Information Processing Doha, vol. 7663, part 1, pp. 280-287, November 2012.

[11] N. Vladimirov, and V. Sourjik, "Chemotaxis: how bacteria use memory," J. Biol. Chem., vol. 390, no. 11, pp. 1097-1104, November 2009.

[12] R. H. Kessin, "Making Streams,"Nature, vol. 422, pp. 481-482, April 2003.

[13] S. M. Ulam and J. von Neumann, "On combinations of stochastic and deterministic processes," Bull. Amer. Math. Soc, vol. 53, pp. 1120, 1947.

[14] V. Patidar, K. K. Sud, and N. K Pareek, "A Pseudo Random Bit Generator Based on Chaotic Logistic Map and its Statistical Testing," Informatica, vol. 33, pp. 441-452, 2009.

[15] K. M. Passino, "Biomimicry of bacterial foraging for distributed optimization and control", IEEE Control Systems, vol. 22, no. 3, pp.52–67, June, 2002.

[16] H. Chen, Y. Zhu, and K. Hu, "Cooperative bacterial foraging optimization," Discrete Dyn. Nat. Soc., vol. 2, no. 1, pp. 501-517, August 2009.

[17] D. R. Monismith, and B. E. Mayfield, "Slime mold as a model for numerical optimization," IEEE Swarm Intelligence Symposium, St Louis, USA, pp. 21-23, 2008.

Paper 6: ‘Adaptive Dynamic Control of Quadrupedal Robotic gaits with Artificial Reaction Networks’

Reference:

Gerrard, C., McCall, J., Coghill, G. and Macleod, C. (2012) ‘Adaptive Dynamic Control of Quadrupedal Robotic gaits with Artificial Reaction Networks’, *Proceedings of the 19th International Conference on Neural Information Processing (ICONIP)*, Lecture Notes in Computer Science, vol. 7663, part 1, Springer, Doha, pp 280-287.

Adaptive Dynamic control of Quadrupedal Robotic Gaits with Artificial Reaction Networks

Claire Gerrard[1] , John McCall[1], George M. Coghill[2] and Christopher Macleod[3].

[1] Ideas Institute, The Robert Gordon University, Aberdeen, AB25 1HG, Scotland, UK
[2] The Department of Computing, University of Aberdeen, AB24 3FX, Scotland, UK
[3] School of Engineering, The Robert Gordon University, Aberdeen, Scotland, UK

Abstract. The Artificial Reaction Network (ARN) is a bio-inspired connectionist paradigm based on the emerging field of Cellular Intelligence. It has properties in common with both AI and Systems Biology techniques including Artificial Neural Networks, Petri Nets, and S-Systems. In this paper, properties of temporal dynamics and pattern recognition are combined within a single ARN control system for a quadrupedal robot. The results show that the ARN has similar applicability to Artificial Neural Network models in robotic control tasks. In comparison to neural Central Pattern Generator models, the ARN can control gaits and offer reduced complexity. Furthermore, the results show that like spiky neural models, the ARN can combine pattern recognition and control functionality in a single network.

Keywords: Artificial Neural Networks, Artificial Reaction Networks, Cellular Intelligence, Biochemical Networks

1 Introduction

Researchers have become increasingly interested in the array of complex behaviors displayed by the simple, commonly unicellular organisms called protists. Some can avoid light with photo-sensitive spots; some actively hunt prey; while others can build protective shelters [1]. Such complex behaviors have led researchers to investigate how such traits of primitive intelligence might arise. Well known examples of such work are that by Nakagaki and Yamada, who demonstrated that the slime-mould *Physarum polycephalum* was able to solve a simple maze [2]. Similar research by Saigusa et al showed that this same organism was able to learn and change its behavior in anticipation of the next environmental stimuli [3]. These high level behaviors are mediated by Cell Signaling Networks (CSNs) [4]. Such networks are composed of interacting proteins within the cell's cytoplasm. Several researchers have highlighted the processing capabilities of these networks and similarities between Artificial Neural Networks (ANNs) [4-8]. For example, it has been demonstrated that such networks can perform Boolean and fuzzy logic and are equivalent to a Turing machine. Furthermore CSNs contain topological features such as feedback loops and interconnectivity, thus forming highly complex systems [9].

The overall aim of our research is twofold. Firstly, to continue exploration of our previously developed connectionist representation of CSNs- the Artificial Reaction Network (ARN) [10], in terms of its possible application in AI. Secondly, to investigate and elucidate mechanisms that contribute to high level behavior or "cell intelligence", which may help in the understanding of intelligence in its widest sense.

This paper investigates the ability, of the ARN like a CSN, to combine pattern recognition and control within a single networked system. A complete control system for a quadrupedal robot is explored, where the ARN responds dynamically to input patterns by generating the associated temporal pattern or "gait". The results are compared with those of similar Artificial Neural Network (ANN) models.

The paper is structured as follows: the first section provides an overview of the ARN representation; this is followed by experimental details and results, and finally conclusions.

1.1 The Artificial Reaction Network Representation

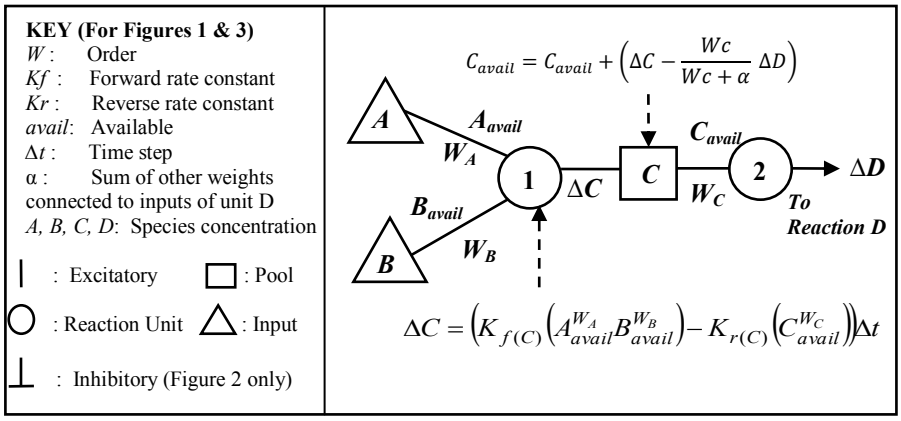


Fig. 1. The Artificial Reaction Network (ARN)

A brief summary of the ARN model is given below. A full account is provided in our previous paper [10]. The ARN, as shown in Figure 1, is a connectionist representation of a CSN, and is structured in a similar way to an ANN. It comprises a set of connected reaction nodes (circles), pools (squares), and inputs (triangles). The inputs are external and constant, each pool represents the current available protein species concentration (avail) and each circle corresponds to a reaction unit, representing an interaction (reaction) between a number of proteins. Figure 1 shows the reaction between species A and B to produce species C. Connections symbolize the flow of species into and out of reaction units and their weight (w) corresponds to reaction order. Flux (ΔA/ΔB/ΔC) at Δt is given by Equation (1). This is derived from the standard Rate Law equation [11], and is equal to the aggregate of connected incoming pools and connected outgoing pools raised to n powers of weighted connections and multiplied by rate constants. At time interval Δt, each reaction unit's temporal flux value is calculated using Euler's approximation as shown in Equation 1. This value is then used

to update the current concentration of each reaction's connecting pools. Thus, the complete set of pool concentrations at time t, corresponds to the current state of the system.

$$\Delta C = \left(K_{f(C)}\left(A_{avail}^{W_A} B_{avail}^{W_B}\right) - K_{r(C)}\left(C_{avail}^{W_C}\right)\right)\Delta t \quad \textbf{(1)}$$

Where:

A, B, C = Species Concentrations	*W* = reaction order (weight)
avail = available species concentration	K_f = Forward rate constant
ΔC = Change in species concentration C	K_r = Reverse rate constant

2 A Complete ARN System for Robotic Control

By means of their CSNs, cells are able to dynamically recognize and respond to environmental patterns [4]. The response is to update the spacio-temporal activations of intracellular species, which in turn encode the high level behavior of the cell [4, 8]. In the following experiments the computational properties and AI applications of such behaviors are explored using a quadrupedal robot.

A single ARN system was created, as shown in Figure 2 and is functionally divided into 3 components: pattern recognition, control, and a connecting network. This section first discusses the setup, function, and results of each component separately before providing the results for the overall system.

2.1 Control Component

The control component is responsible for generating particular temporal patterns, which correspond to robotic gaits. Terrestrial locomotion of limbed animals is achieved by multiple phase locked patterns of limb movements known as gaits. For example, quadrupeds commonly walk, trot and gallop [12]. The gait phase is a value that ranges from 0 to 1 as the cycle proceeds, and thus each limb can be described relative to the cycle. The ideal quadrupedal gaits are described by Dagg [12] and others [13], and are used as a standard for comparison here and similarly in other studies [14]. The walk gait is characterized where each leg is a quarter cycle out of phase with each other. In the trot gait each pair of diagonal limbs move half a cycle out of phase with one another. Here, the ARN control component was implemented, to generate the trot and walk gaits of a Lynxsmotion dual-servo quadruped 2 (Q2) robot. Each robotic leg is controlled by two servo motors, one for each degree of freedom (DOF), where one raises the leg, the other moves it. Further details of the robot legs are given by Toth and Parker [15]. Signals are sent by the ARN to each motor and control the angle of the rotor for each DOF, using a simple position to pulse width modulator interface circuit to control the servo. The ARN control component is shown in Figure 2 and consists of two copies of the same network- one for walk, the other for trot (each labeled). It comprises four identical modules (one module is

shown enclosed in a dotted line), where each controls the two motors (one for each DOF) of a separate leg.

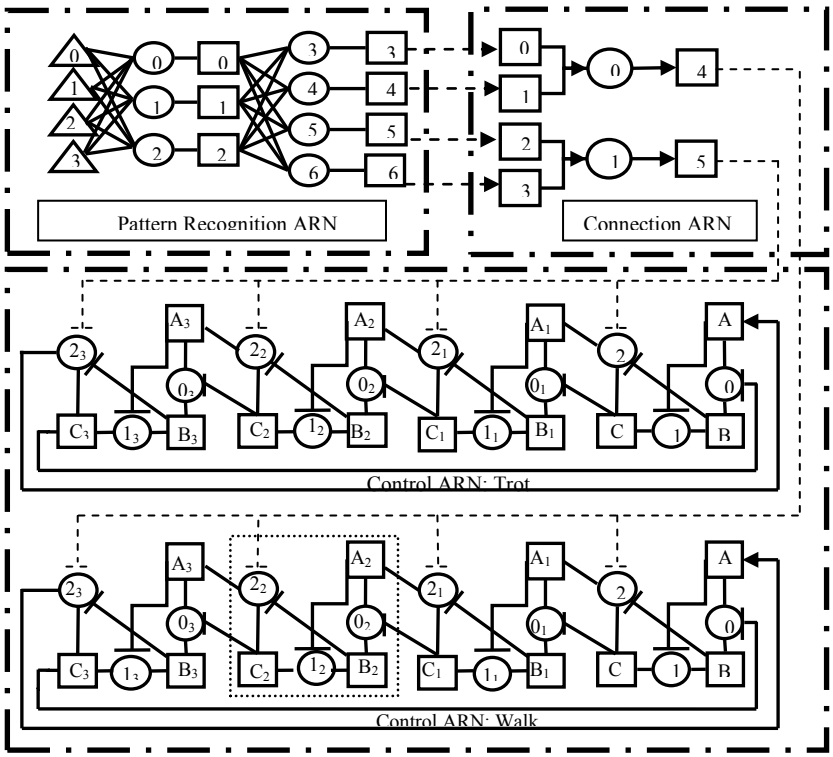


Fig. 2. The complete ARN control system comprising 3 smaller network components: Pattern recognition, Connection and Control.

A module comprises 3 reaction units, and 3 pools: A, B and C. Pool A controls the up/down (U/D) motor, Pool B the back/forward (B/F) motor and Pool C controls the off period for both motors. Pool activity is regulated by a series of excitatory and inhibitory connections between reaction units and represents properties of specialized regulatory proteins common to CSNs such as enzymes. The entire structure is organized as a closed loop, thus chemical species are recycled to the first module, and generate a temporal oscillatory pattern. The network structure and parameters were hardcoded so that the outputs could be directly compared with other published work on similar Central Pattern Generators (CPGs). However, there is no reason why connection weights cannot be set using an Evolutionary Algorithms as will be shown later. The gait produced by this network is modified by adjustment of the initial pool values. For example, initializing one C pool generates a walk gait, where the C pool chosen will determine the starting leg, and the value determines the angle to which the leg is raised (the DOF angle). Similarly, a trot gait is achieved by initializing 2 C pools within alternate modules. The output for the walk subunit is displayed in Figure 3, and shows legs are a quarter cycle out of turn, with phases of 0.0, 0.25, 0.5, 0.75 between limbs in clockwise order from front left (FL) leg. Similarly, the trot gait re-

sults were half a cycle out of turn with phases respectively of 0.0, 0.5, 0.0, 0.5. Both phase locked limb patterns match the standard, and compare well with other connectionist models. For example, Billard and Ijspeert present a CPG (central pattern generator) based neural controller for a quadrupedal AIBO robot with 2 DOFs for each leg [16]. The network is composed of 8 coupled non-linear oscillators and each oscillator consists of 6 leaky integrator neurons (total of 96 neurons). Each neuron implements an activation approximately as complex as the ARN reaction unit function. Thus the complexity of this network is equivalent to approximately 96 ARN reaction units. Similar correspondence is found in other sources. For instance, Collins explores a CPG based neural controller for a quadrupedal robot with 1 DOF per limb, and compares 3 types of activation function models. The controller is composed of a network of 4 coupled non-linear oscillators [14], where each oscillator controls a separate limb. These models produce gaits within 10% of the standard, whereas the ARN matches the standard for both gaits. Each model has approximately twice the complexity as the ARN reaction unit, and all require a pulsing signal to drive the network.

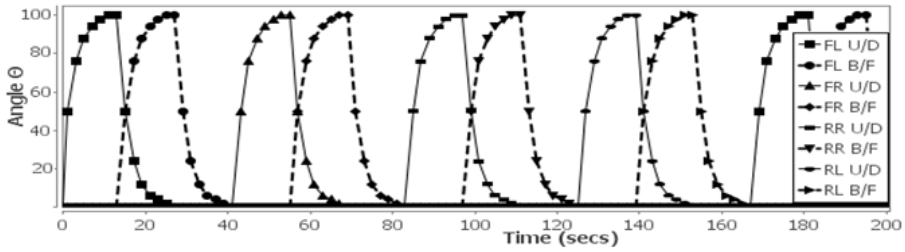


Fig. 3. Output generated by ARN controller for walk gait. Solid lines are legs up/down motor, dashed lines are back/forward motor. Legs move independently in order: FL, FR, RR, RL.

2.2 Pattern Recognition Component

The pattern recognition component serves as the interface between the environment and the ARN system. Here external concentrations are processed, where particular patterns switch off or on robotic gaits through the connecting network. The network was trained to recognize 3 patterns, each comprising 4 inputs (triangles 0-3) and these were associated with 4 output values. Each pattern comprised values of either 0.1, representing low concentration or 1 corresponding to high concentration. This component (shown in Figure 2) consists of 4 inputs, 7 pools, and 7 reaction units organized into 2 layers. The associated output generated corresponds to the steady state values of the final layer of pools (squares 3-6). The input and associated output patterns are given in Table 1. A genetic algorithm (GA) was used to train the network to associate the required outputs before being connected to the other components. In this GA a population of 100 solutions was randomly initialized, where each comprised a complete set of network parameters including the forward and reverse rates for each unit and the weights for each connection. Due to its temporal properties, the network was run for 100 cycles (a cycle ends when the complete set of pools are updated once) in order to obtain steady state output values. The solution fitness was then calculated, where fitness was the error on output. The least fit half of the population was discard-

ed, and the remainder was subject to rates of 0.4 single point crossover and 10% uniform mutation and trained to the target error value of 0.01. On completion of training, the network was able to associate all 3 patterns within the target error. Although there is not room for a full comparison, multilayer perceptron ANNs (MLPs) [17] produce comparable results. However, MLPs lack an explicit time dimension, whereas the ARN processes continuous inputs over a time period.

Table 1. Patterns applied to the pattern recognition network and their outputs (output is the input to connection component). Connection component output and expected gait generated.

Pattern	Pattern Recognition Network Input Pool No.	Pattern Recognition Network Input Value	Connection Network Input Pool No.	Connection Network Input Value (also output of the pattern ecognition network)	Connection Network Output Pool No.	Connection Network Output Value	Gait
1	0	1	0	1	4	1	Inhibit Walk
	1	0.1	1	1			
	2	1	2	0	5	0	Trot
	3	0.1	3	0			
2	0	0.1	0	0	4	0	Walk
	1	1	1	0			
	2	0.1	2	1	5	1	Inhibit Trot
	3	1	3	1			
3	0	1	0	1	4	1	Inhibit Walk
	1	0.1	1	1			
	2	0.1	2	1	5	1	Inhibit Trot
	3	1	3	1			

2.3 Connection Component and Results for the Complete System

The connecting module functions to process the output from the pattern recognition network, and produce a signal which switches off/on the required gait. This module comprises 6 pools and 2 reaction units, as shown in Figure 2. Each input (pools 0-3), is linked directly to a corresponding output pool of the pattern recognition network (pools 3-6). Essentially the network operates as two parallel Boolean AND gaits, where a value of 1 at pools 0 and pool 1 outputs a value of 1 at pool 4, as will a value of 1 at pools 2 and 3 output a 1 at pool 5.

Table 2. Pattern applied to the network and expected durations of gaits.

Pattern	Walk ARN Network	Trot ARN Network	Start Time	End Time	Duration
2	On	Off	0	210	210
1	Off	On	210	440	230
2	On	Off	440	560	120
1	Off	On	560	700	140
3	Off	Off	700	800	100

Two negative feedback connections between the connecting network and both ARN control system sub units (shown as dashed line connections) are responsible for switching between the gaits. Therefore if a value of 1 is output at pool 4, it will inhibit all the reaction 2's of the ARN trot subunit, thus stopping the trot gait from being generated. Conversely if a value of 0 is output at pool 4 the trot will be generated. In the same way pool 5 controls the switching on/off of the walk control subunit. Table 1

shows the input, and associated output of this component and the range of behaviors that should be generated in response to particular outputs. The complete system was tested to confirm its ability to both generate the correct behavior and automatically transition between the behaviors in response to firing input patterns 0-3. The time periods in which patterns were applied, and the expected output states are shown in Table 2. As shown in Figure 4 the on/off periods of both trot and walk gaits are in agreement with the expected durations displayed in Table 2 with a slight transitional delay, in order: walk, trot, walk, trot, off. The gait transitions are now compared with the same models used to compare the ARN controller, and gait phases in section 2.1. The results given for the Billard and Ijspeert model [16], show smooth transitions from walk to gallop in approximately 4 leg cycles. The ARN similarly transitions from walk to trot smoothly within 1 leg cycle. In the Collins paper [14], gaits transition quickly within approximately 2 leg cycles, whereas the transitions are very irregular in contrast to the ARN and the Billard and Ijspeert model.

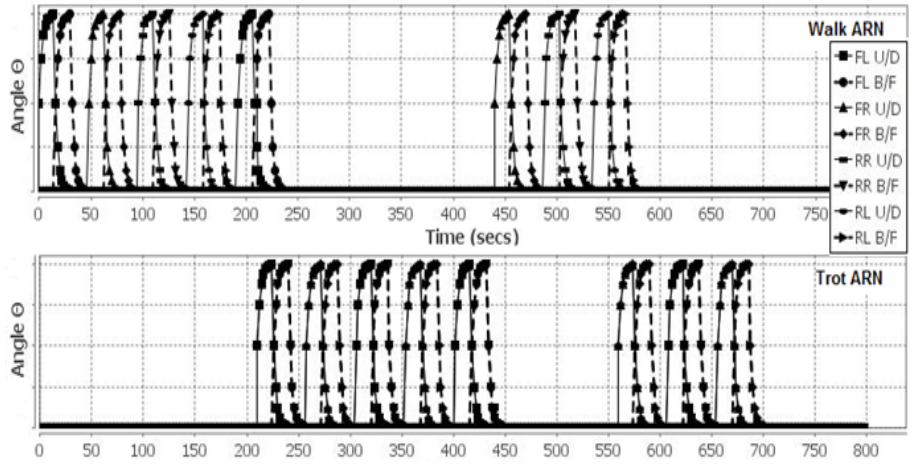


Fig. 4. The output of the complete ARN control system over 800 seconds.

3 Conclusions

The ARN is a bio-inspired connectionist representation based on properties and mechanisms found in CSNs that together result in emergent behavior or "cell intelligence". A complete ARN based control system was constructed to dynamically respond to external patterns, where each pattern triggers a specific gait of a quadrupedal robot. This system was designed to exploit topological features found in CSNs including negative feedback, and cycles. It was demonstrated that the ARN, like a CSN, is capable of both recognizing patterns and controlling overall behavior in a single network. With the exception of spiky models few ANNs can easily achieve this functionality, and thus the ARN provides an alternative in similar applications. The gait phases and transitions compared well with CPG neural controllers and showed that the ARN has application in similar robotic control tasks where it can offer lower

computationally complexity. These experiments illustrate how a CSN might perform the complex processing associated with the high level behaviors displayed by single celled organisms. Furthermore it shows that abstractions of both neural networks and CSNs operate in similar ways, and have comparable functionality. Thus this work illustrates a close relationship between emergent neural intelligence and emergent cell intelligence.

In future work, it is intended to further explore the AI applications of the ARN, including more complex networks that can recognize patterns and control simultaneous behaviors.

4 References

1. Ford, B. J.: Are cells Ingenious? The Microscope. 52, 135-144 (2004)
2. Nakagaki, T., Yamada, H., Toth, A.: Maze-solving by an amoeboid organism. Nature. 407 (6803), 470-470 (2000)
3. Saigusa, T., Tero, A., Nakagaki, T., Kuramoto, Y. Amoebae Anticipate Periodic Events. Phys. Rev. 100 (1), 1-4 (2008)
4. Bray, D.: Protein molecules as computational elements in living cells. Nature. 376 (6538) 307-12 (1995)
5. Arkin, A., Ross, J.: Computational functions in biochemical reaction networks. Biophys. J., 67, 560-578 (1994)
6. Wang, B., Kitney, R.I., Joly, N., Buck, M.: Engineering Modular and Orthogonal genetic logic gates for robust digital-like synthetic biology. Nat Commun. 2, 508 (2011)
7. Hellingwerf, K. J.: Bacterial observations: a rudimentary form of intelligence. T. I. M., 13 (4), 152- 158 (2005)
8. Bhalla, U. S.: Understanding complex signaling networks through models and metaphors. Prog. Biophys . Mol. Biol. 81, 41-65 (2003)
9. Kholodenko, B.: Cell Signaling dynamics in Time and Space. Nature Rev. Mol. Cell Biol. 7 (3), 165-176 (2006)
10. Gerrard, C.E, McCall, J., Coghill, G.M., Macleod, C.: Artificial Reaction Networks. In: Proceedings of the 11th UK Workshop on Computational Intelligence, pp. 20-26. UK (2011)
11. Morris, J.G. A Biologist's Physical Chemistry. 2nd ed. Barrington EJW, Willis AJ, editors. Great Britain: Edward Arnold; (1974)
12. Dagg, A., I.: Gaits in mammals. Mammal Rev. 3:135-154, (1973)
13. Hildebrand, M. Analysis of asymmetrical gaits: J. Mammal. 58:131-156, (1977)
14. Collins, J.J., Richmond, S.A.: Hard-wired central pattern generators for quadrupedal robots. Biol. Cybern. 71, 375-385 (1994)
15. Toth, D, Parker, G. Evolving Gaits for the Lynx motion Hexapod II Robot. In: Proceedings of the 7th World Multiconference on Systems, Cybernetics, and Informatics, 3, pp. 229-234. Orlando, USA (2003)
16. Billard, A., Ijspeert, A.J.: Biologically inspired neural controllers for motor control in quadruped robot. In: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks, pp. 637-641. IEEE, Italy (2000)
17. Yegnanarayana, B. Artificial neural networks for pattern recognition. Sadhana, 19, 147–169 (1994)

Paper 7: ‘Temporal patterns in Artificial Reaction Networks’

Reference:

Gerrard, C., McCall, J., Coghill, G. and Macleod, C. (2012) ‘Temporal patterns in Artificial Reaction Networks’, *Proceedings of The 22nd International Conference on Artificial Neural Networks (ICANN)*, Lecture Notes in Computer Science, vol. 7552, part 1, Springer, Lausanne, pp 1-8.

Temporal Patterns in Artificial Reaction Networks

Claire Gerrard[1], John McCall[1], George M. Coghill[2] and Christopher Macleod[3].

[1] Ideas Institute, The Robert Gordon University, Aberdeen, AB25 1HG, Scotland, UK
[2] The Department of Computing, University of Aberdeen, AB24 3FX, Scotland, UK
[3] School of Engineering, The Robert Gordon University, Aberdeen, Scotland, UK

Abstract. The Artificial Reaction Network (ARN) is a bio-inspired connectionist paradigm based on the emerging field of Cellular Intelligence. It has properties in common with both AI and Systems Biology techniques including Artificial Neural Networks, Petri Nets, and S-Systems. This paper discusses the temporal aspects of the ARN model using robotic gaits as an example and compares it with properties of Artificial Neural Networks. The comparison shows that the ARN based network has similar functionality.

Keywords: Artificial Neural Networks, Artificial Reaction Networks, Cellular Intelligence, Biochemical Networks

1 Introduction

When Artificial Intelligence (AI) researchers want to develop connectionist models of intelligence, it is only natural that they should look to the brain for inspiration. The result, of course, is the Artificial Neural Network (ANN). However, as discussed in this paper, there is an alternative, biologically inspired, connectionist paradigm based on the emerging field of Cellular Intelligence – the Artificial Reaction Network (ARN) [1].

In recent years, researchers have become increasingly interested in the behaviors displayed by single celled organisms, in particular protists. These eukaryotes, display an astonishing array of complex behaviors. Some can avoid light with photo-sensitive spots; some actively hunt prey; while others can build protective shelters [2].

These complex behaviors have led researchers to investigate how such traits of primitive intelligence might arise. Well known examples of such work are that by Nakagaki and Yamada, who demonstrated that the slime-mould *Physarum polycephalum* was able to solve a simple maze [3]. Similar research by Saigusa et al showed that this same organism was able to learn and change its behavior in anticipation of the next environmental stimuli [4].

These high level behaviors are mediated by Cell Signaling Networks (CSNs) which, as this paper will discuss, are analogs to ANNs. Such networks are composed of interacting proteins within the cell's cytoplasm that function to regulate virtually all cellular activity.

The ARN is a new representation based on CSNs. This paper explores the ARNs ability to generate temporal oscillations in protein species – a common theme in

CSNs. It discusses its similarities and differences to ANNs by comparing them in similar applications - specifically in the generation of robotic gaits. The aim of this research is firstly, to explore the mechanisms of cell intelligence in order to broaden understanding of intelligence in its widest sense as well as have possible applications in biological modeling. Secondly, to investigate the resulting representation in terms of its possible application for use as an AI technique.

1.1 Mechanisms of Cellular Intelligence

CSNs consist of different protein species, the interactions of which are shown by connecting lines in a similar way to a neural network. Via a system of complex mechanisms, CSNs adjust their set of protein activation levels to fine tune cellular activity appropriate to current conditions. An instantaneous set of these protein concentrations serves like a memory, containing an imprint of the current environmental state [5]. Individual spatio-temporal activation patterns of protein concentrations emerge from a multitude of low level interactions and result in a range of cellular responses and behaviors [6-8]. The network therefore represents cascades of numerous protein coupled interactions with topological features such as feedback loops and interconnectivity, forming highly complex systems [5, 8].

Bray claims that the processing performed by individual CSN units is similar to Boolean and fuzzy logic and further speculates that these networked logical units can perform computational processing equivalent to a Turing machine [5]. Similar reports were documented by a number of other researchers [9-11].

Many researchers highlight the similarities between CSNs and ANNs [5-7, 12]. Bray, observes both networks are made up of highly connected parallel distributed units, where each unit simultaneously integrates and processes signals. Both are able to recognize patterns, and provide the correct response in the presence of noise and loss of units, and are therefore robust [5, 12]. One difference is that while simple traditional ANNs like the perceptron lack an explicit time dimension, CSN functionality incorporates this in a similar way to spiking neuron models. Bhalla notes that the high level cellular behavior is encoded by temporal spatial patterns of intracellular species generated in this way [12]. One such common motif is oscillating patterns, resulting from feedback structures and cyclic loops [8].

2 The Artificial Reaction Network

2.1 Techniques Used to Develop Model

The ARN representation was designed to incorporate the previously discussed mechanisms of cell intelligence. Our previous paper provides a complete description, and verification of the ARNs accuracy and biological plausibility [1]

There are many methods used to model biochemical reactions, some are very simple Boolean-based techniques, others complex quantum mechanical abstractions [13], here the two most relevant adopted techniques are described. The first is S-Systems; these have proven themselves accurate and provide a similar degree of system abstraction to an ANN. They comprise sets of ordinary differential equations (ODEs)

that exploit a power law representation to approximate chemical flux [13]. Similarly to traditional rate law [13], each ODE is equal to the difference between two conceptually distinct functions; the first function includes all terms contributing to system influx, the second to decay. S-systems provide simple but accurate representations of temporal dynamics, including both steady and transient state. However, in their general form, terms are highly coupled, and therefore are difficult to manipulate without interference.

Like an ANN, Petri Nets (PNs) offer a modular approach. PNs are a graphical and mathematical modeling tool used to study processes characterized as parallel, distributed, concurrent, and asynchronous [14]. They are used extensively in several types of information processing, including modeling CSNs. Each PN is a networked structure of separate self-maintaining units called "places", where movement between connections is defined by separate transitions, thus PNs exploit benefits of modularization.

2.2 The Artificial Reaction Network Model

The authors combined the continuous mathematical nature of S-systems, the modular properties of PNs, and weighted connections of ANNs. The ARN, as shown in Figure 1, is a modular and expandable S-System. It comprises a set of connected reaction nodes (circles), pools (squares), and inputs (triangles). Each pool represents the current available protein species concentration (avail) and each circle corresponds to a reaction unit, representing an interaction (reaction) between a numbers of proteins. For example, Figure 1 shows the reaction between species A and B to produce species C. Connections symbolize the flow of species into and out of reaction units and their weight (W) corresponds to reaction order. This structure can be compared to a perceptron, where the pools correspond to inputs, the reaction units to the weighted sum function, and these are joined together by weighted connections. Both are instances of highly connected parallel distributed networks, where units simultaneously integrate and process signals.

$$\Delta C = \left(K_{f(C)}\left(A_{avail}^{W_A} B_{avail}^{W_B}\right) - K_{r(C)}\left(C_{avail}^{W_C}\right)\right)\Delta t \tag{1}$$

Where:
- A, B, C = Species Concentrations
- avail = available species concentration
- W = reaction order
- ΔC = Change in species concentration C
- K_f = Forward rate constant
- K_r = Reverse rate constant
- Δt = time step

Each reaction unit calculates flux ($\Delta A/\Delta B/\Delta C$) at Δt as given by Equation (1), and is equal to an aggregate of connected contributing (incoming) pools and connected decay (outgoing) pools raised to n powers of weighted connections and multiplied by pseudo rate constants. This can be compared to the Sigma-pi ANN model, where the

output depends on a function of the product of the inputs. Unlike the feedforward perceptron, species can flow in either direction, depending on the sign of the flux calculated by Equation 1. Dissimilarly to a perceptron, the ARN incorporates a temporal dimension, where at time interval Δt, each reaction unit's temporal flux value is calculated, which then is used to update the current concentration values of each reaction's connecting pools. Thus the complete set of pool concentrations at time t corresponds to the current state of the system. Euler's approximation was adopted in favor of other evaluation methods because it supports modularization. Its disadvantage is that net error accumulates with every cycle; however by decreasing step size error is reduced. The intention however, is to characterize high-level system properties and thus requires only sufficient low level detail to represent its contributing mechanisms such as temporal dynamics and complex network topologies.

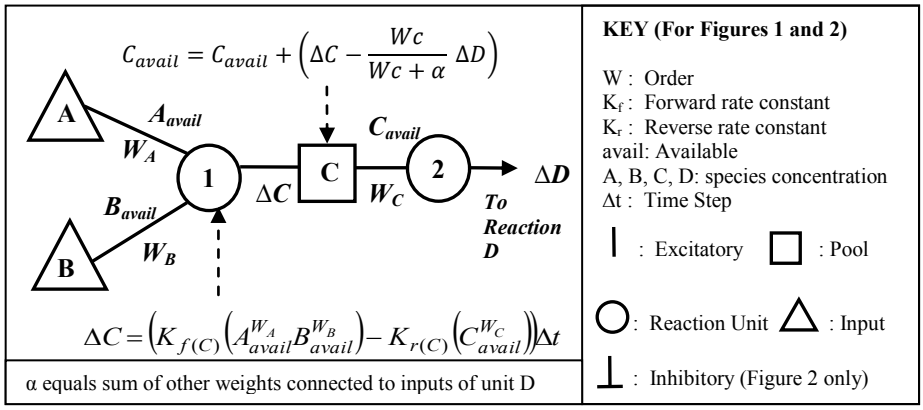


Fig. 1. The Artificial Reaction Network (ARN)

3 Experiments

As previously discussed, complex mechanisms found in CSNs lead to stable temporal patterns of species concentrations, where each relates to a high-level behavior. One way to investigate the ability of the ARN to produce such temporal oscillatory patterns is by applying it to generate those associated with robotic gaits. Furthermore, this allows comparison with similar results obtained using ANN models.

Terrestrial locomotion of limbed animals is achieved by multiple phase locked patterns of limb movements known as gaits. For example, depending on speed of locomotion and terrain, quadrupeds commonly walk, trot and gallop [15]. The gait phase is a value that ranges from 0 to 1 as the gait cycle proceeds. Therefore, the motion of each limb can be described relative to the gait phase. The ideal quadrupedal gaits are described by Dagg [15] and others [16], and are used as a standard for comparison here and similarly in other studies [17]. The walk gait is characterized where, each leg is a quarter cycle out of phase; in the trot gait each pair of diagonal limbs move half a cycle out of phase with one another. An ARN based robotic controller was implemented, to produce trot and walk gaits of a simulated Lynxsmotion dual-servo quadruped 2 (Q2) robot. The structure of the ARN controller was designed to include ab-

stractions of regulatory mechanisms found in CSNs including inhibitory/excitatory reactions, cyclic loops, and feedback structures.

3.1 The Robot and the ARN Controller

Each robotic leg is controlled by two servo motors, one for each degree of freedom (DOF), where one raises the leg, the other turns it. Signals are sent by the ARN to each motor and control the angle of the rotor for each DOF, using a simple position to pulse width modulator interface circuit to control the servo. The physical structure and control are described in detail in other papers [18].

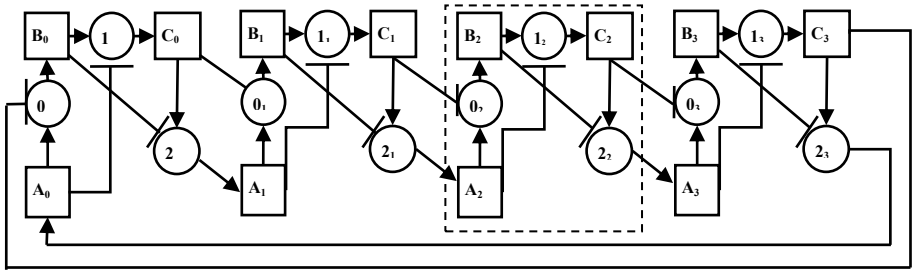


Fig. 2. The ARN based controller displayed contains 4 identically structured modules, a module is shown surrounded by a dashed line.

Figure 2 illustrates the structure of the ARN controller, it comprises four identical modules (a module is highlighted by a dashed line) each controlling the motors for a separate leg. Each module contains 3 reaction units, and 3 pools: A, B and C. Pool A controls the up/down (U/D) motor, Pool B the back/forward (B/F) motor and Pool C controls the off period for both motors. The activity of pools is regulated by a series of excitatory and inhibitory connections between reaction units. These connections represent properties of specialized regulatory proteins common to CSNs such as enzymes. The connection weights were hardcoded using the same method as used in the Billard and Ijspeert model [19]. The entire structure is organized as a closed loop, allowing chemical species to be recycled to the first module, and thus generate a stable repeating temporal pattern. The type of robot gait is easily modified by a simple adjustment of the initial pool values. For example, by initializing a C pool, a walk gait will be generated, where the C pool chosen will determine the starting leg. Similarly, a trot gait is achieved by initializing 2 C pools within alternate modules. In this particular design, the value to which the C pool(s) are initialized determines the DOF angle and were set specifically for the physicality of the particular robot, although it can be freely varied.

4 Results

The ARN controller was considered to generate a specific gait if the relative phases of the respective oscillatory signals were within 2% of the standard gait cycle described previously. Higher values of 10% were used in other studies [17], and this was con-

sidered reasonable due to the variation found in real animal gaits [20]. In each case, the controller first generates the U/D motor oscillation and on reaching the maximum value the B/F motor is initiated. As can be seen the walk gait results (Figure 3) show legs are a quarter cycle out of turn, with phases of 0.0, 0.25, 0.5, 0.75 between limbs in clockwise order from FL leg. Similarly the trot gait shows opposite legs are half a cycle out of turn with phases respectively of 0.0, 0.5, 0.0, 0.5. Both phase locked limb patterns match the standard, and compare well with other connectionist models. For example, Billard and Ijspeert present a CPG (central pattern generator) based neural controller for a quadrupedal AIBO robot, similarly with 2 DOFs for each leg [19]. Here, the network is composed of 8 coupled non-linear oscillators and each oscillator consists of 6 leaky integrator neurons (total of 96 neurons). Each neuron implements an activation approximately as complex as the ARN reaction unit function. Thus the complexity of this network is equivalent to approximately 96 ARN reaction units. The oscillatory signals produced by this network for both walk and trot gaits show that the limb phases correspond to the standard and to those produced by the ARN. Similar correspondence is found in numerous other sources. For instance, Collins explores a CPG based neural controller for a quadrupedal robot with 1 DOF per limb, and compares 3 types of activation function models. The controller is composed of a network of 4 coupled non-linear oscillators [17], where each oscillator controls a separate limb. The reported limb phases correspond to the standard, although those reported for the trot were within 10% of the ideal, whereas the ARN matches the standard for both gaits. Each model has approximately twice the complexity of the ARN reaction unit and, unlike the ARN, all require a pulsing signal to drive the network. Overall the ARN affords a higher degree of accuracy where fine tuning of parameters can provide finite levels of control. For instance, the frequency of oscillations and therefore the gait speed can be easily modified by uniform increase or decrease of K_f of each unit. Similarly, independent variation of speed for each type of DOF (B/F or U/D) or for a specific leg DOF motor. These results show the ARN has a very similar capacity in robotic control tasks as other connectionist robotic controllers, where it can offer reduced computational complexity. Furthermore the ARNs ability to produce gaits illustrates how cellular networks can generate the complex temporal patterns necessary in emergent behavior.

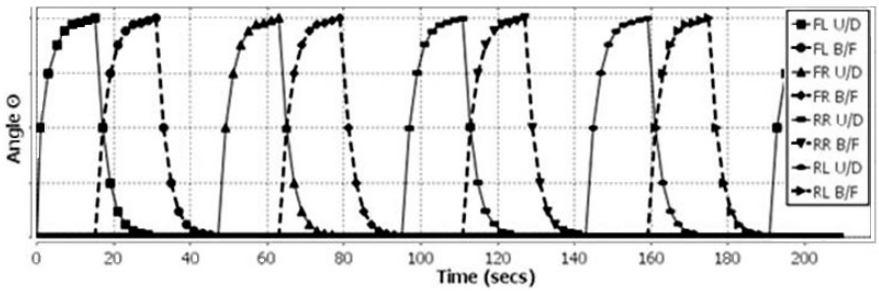


Fig. 3. Output generated by ARN controller for walk gait. Solid lines are legs up/down motor, dashed lines are back/forward motor. Legs move independently in order: FL (front left), FR (front right), RR (rear right), RL (rear left).

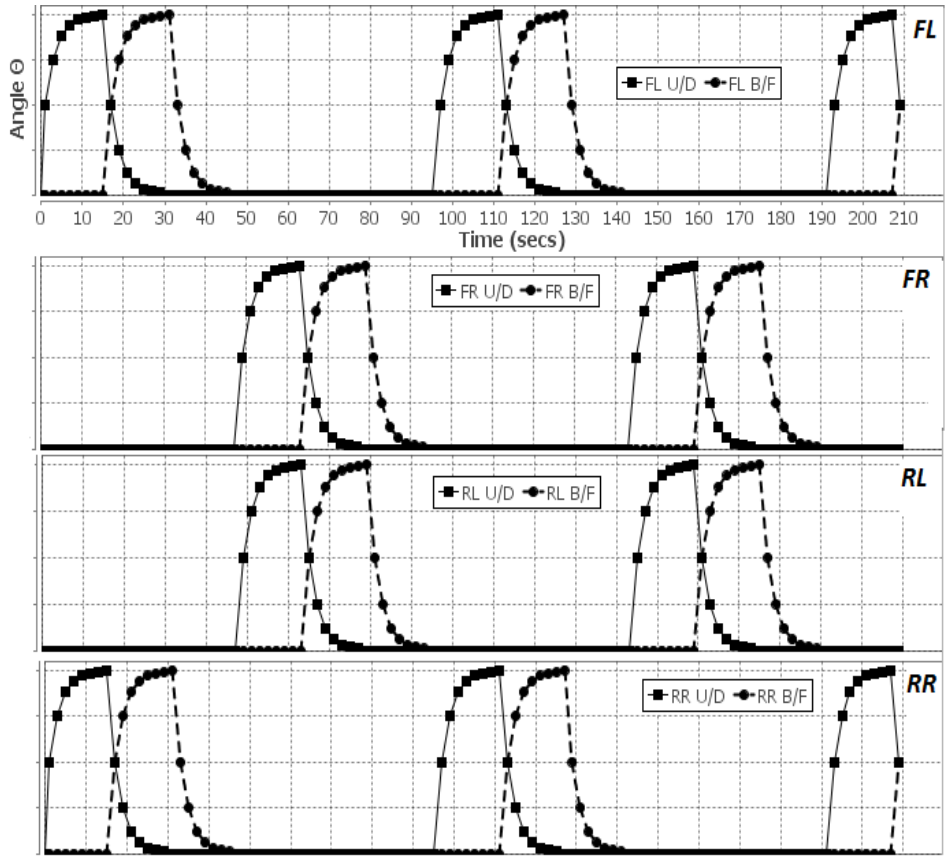


Fig. 4. ARN controller output for trot gait. Diagonal legs are in phase and operate in order FL and RR then FR and RL.

5 Conclusions

The ARN is a bio-inspired connectionist representation based on mechanisms found in CSNs that contribute to the emergence of cell intelligence. One feature of CSNs is the ability to generate high level behavior by regulating temporal activation patterns of its component proteins. The ARN was tested as a means to artificially produce similar pattern regulation, and its potential applicability was explored. Here an ARN based control system was designed to exploit topological features such as negative feedback, and cycles found in real CSNs. The controller was applied to produce the temporal oscillatory patterns associated with quadrupedal trot and walk gaits. The results confirmed the ability of the ARN to regulate temporal oscillating patterns with applicability in robotic control. These results are in good correspondence with ANN models, where both generate very similar spatial temporal patterns. A significant number of parallels between ARNs and ANNs were highlighted, suggesting the nature of cell intelligence may not be that different from neural intelligence. These simi-

larities highlight the potential of single celled organisms to produce complex behavior similar to that produced by a neural network. This will be explored further, in particular by generating more complex temporal patterns, regulating composite behavior and chaotic components.

6 References

1. Gerrard, C.E, McCall, J., Coghill, G.M., Macleod, C.: Artificial Reaction Networks. In: Proceedings of the 11th UK Workshop on Computational Intelligence, pp. 20-26. UK (2011)
2. Ford, B. J.: Are cells Ingenious? The Microscope. 52, 135-144, (2004)
3. Nakagaki, T., Yamada, H., Toth, A.: Maze-solving by an amoeboid organism. Nature. 407 (6803), 470-470, (2000)
4. Saigusa, T., Tero, A., Nakagaki, T., Kuramoto, Y. Amoebae Anticipate Periodic Events. Phys. Rev. 100 (1), 1-4, (2008)
5. Bray, D.: Protein molecules as computational elements in living cells. Nature. 376 (6538) 307-12, (1995)
6. Helikar, T., Konvalina, J., Heidel, J., Rogers, J.A.: Emergent decision making in biological signal transduction. In: Proc. Natl. Acad. Sci. USA. 105,1913–1918, (2008)
7. Hjelmfelt, A., Ross, J.: Mass Coupled Chemical Systems with Computational Properties. J. Phys. Chem. 97, 7988-7992, (1993)
8. Kholodenko, B.: Cell Signaling dynamics in Time and Space. Nature Rev. Mol. Cell Biol. 7 (3), 165-176, (2006)
9. Stadtman, E. R., Chock, P. B.: Superiority of interconvertible enzyme cascades in metabolic regulation: analysis of multicyclic systems. Proc. Natl. Acad. Sci. USA, 74, 2766-2770, (1977)
10. Arkin, A., Ross, J.: Computational functions in biochemical reaction networks. Biophys. J., 67, 560-578, (1994)
11. Wang, B., Kitney, R.I., Joly, N., Buck, M.: Engineering Modular and Orthogonal genetic logic gates for robust digital-like synthetic biology. Nat. Commun. 2, 508, (2011)
12. Bhalla, U. S.: Understanding complex signaling networks through models and metaphors. Prog. Biophys .Mol. Biol. 81, 41-65, (2003)
13. M. A., Savageau, E. O., Voit.: Recasting Nonlinear Differential Equations as S-Systems: A Canonical Nonlinear Form. Math. Biosci. 87 (1), 83–115, (1987)
14. Baldan, P., Cocco, N., Marin, A., Simeoni, M.: Petri Nets for Modeling Metabolic Pathways: A Survey. Natural computing . 9(4), 955-989, (2010)
15. Dagg, A. I.: Gaits in mammals. Mammal Rev. 3:135-154, (1973)
16. Hildebrand, M. Analysis of asymmetrical gaits.: J. Mammal. 58:131-156, (1977)
17. Collins, J.J., Richmond, S.A.: Hard-wired central pattern generators for quadrupedal robots. Biol. Cybern. 71, 375-385, (1994)
18. Macleod, C., Maxwell, G., Muthuraman, S.: Incremental Growth in Modular Neural Networks. Eng. Appl. Artif. Intel. 22 (4-5), 660-666, (2009)
19. Billard, A., Ijspeert, A.J.: Biologically inspired neural controllers for motor control in quadruped robot. In: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Neural Networks, pp. 637-641. IEEE, Italy (2000)
20. Afelt, Z., Blaszczyk, J., Dobrzecka, C.: Speed control in animal locomotion: transitions between symmetrical and nonsymmetrical gaits in the dog. Acta. Neurobiol. Exp. 43, 235-250, (1983)

Paper 8: ‘Artificial Reaction Networks’

Reference:

Gerrard, C., McCall, J., Coghill, G. and Macleod, C. (2011) ‘Artificial Reaction Networks’, *Proceedings of the 11th UK Workshop on Computational Intelligence (UKCI)*, Manchester UK, pp 20-26.

Artificial Reaction Networks

Claire E. Gerrard, John McCall, George M. Coghill and Christopher Macleod.

***Abstract*—In this paper we present a novel method of simulating cellular intelligence, the Artificial Reaction Network (ARN). The ARN can be described as a modular S-System, with some properties in common with other Systems Biology and AI techniques, including Random Boolean Networks, Petri Nets, Artificial Biochemical Networks and Artificial Neural Networks. We validate the ARN against standard biological data, and successfully apply it to simulate cellular intelligence associated with the well-characterized cell signaling network of *Escherichia coli* chemotaxis. Finally, we explore the adaptability of the ARN, as a means to develop novel AI techniques, by successfully applying the simulated *E. coli* chemotaxis to a general optimization problem.**

I. Introduction

Natural evolution has transformed the world into a resource rich in examples of elegant solutions to complex problems. However, these solutions are often hidden in layers of biochemical detail, and are consequently little understood. Cell Signaling Networks (CSNs) are an example of one such natural "solution". They refer to the network of biochemical reactions which allow communication, response and feedback within and between cells. Many scientists have reasoned that the characteristics of cellular intelligence such as recognition, classification, response, communication, learning and self-organization [1] are the result of these complex networks [2], [3].

Significant advances in biotechnology have resulted in a surge of biochemical data, allowing hidden aspects of cell signaling to be uncovered. As understanding of cell signaling becomes further developed, its significant role in cellular intelligence is emerging. Many parallels have been drawn between CSNs, computational processing and artificial intelligence techniques. For instance, their ability to perform processing analogous to Boolean logic, negative/positive feedback loops, integration, amplification, and temporal regulation [4]. However, the fact remains that no man-made system can yet compare to the degree of sophistication inherent in these networks.

Claire. E. Gerrard and John McCall are with the IDEAS Research Institute, Robert Gordon University, Aberdeen, AB25 1HG, UK (phone: +44(0)1224-26-2476; email: c.e.gerrard@rgu.ac.uk).

Geroge M. Coghill. is with the Department of Computing, University of Aberdeen, Aberdeen, AB24 3FX, UK.

Christopher Macleod is with the School of Engineering, Robert Gordon University, Aberdeen, AB25 1HG, UK.

Artificial intelligence has progressed enormously since the birth of bio-inspired approaches (for example: genetic algorithms (GAs), Particle Swarm Optimization (PSO), and Ant Colony Optimization (ACO) [5]), some such approaches are inspired by biochemical networks: Artificial biochemical networks [6] and Artificial Immune Systems (AIS) [5].

In this paper we focus on exploring the mechanisms of cellular intelligence to facilitate the development of novel CSN inspired AI techniques. For this purpose a new simple representation was developed: the "Artificial Reaction Network" ARN. Rather than focus on micro-molecular detail, the ARN aims to elucidate emergent behavior within a network of chemical reactions. Its biological basis is validated using real biochemical data, including simulation of the well characterized signaling network of *E.coli* chemotaxis. Furthermore, this network is examined as a source of inspiration for development of novel AI techniques.

II. Background

Nakagaki and Yamada demonstrated that the slime mould *Physarum polycephalum* was able to solve a simple maze [7]. A maze was built from plastic films set on agar gel with four possible routes of different length between two food sources. The organism eventually formed a thick plasmodial tube via the shortest pathway between the two food sources. This behavior increased its foraging capability, conserved its energy and thus increased its chances of survival. A further study by Saigusa et al showed that, when subjected to a distinct pattern of periodic environmental changes, this organism was able to learn and change its behavior in anticipation of the next stimulus [8]. The researchers argue that the behaviors illustrated in these experiments: problem solving, recalling, and anticipating events are the result of a "primitive intelligence" that emerges from the simple low-level cellular dynamics found in CSNs.

An account of how this primitive cellular intelligence arises is provided by Bray; he describes how interconnected protein units of CSNs result in a range of sophisticated processing capabilities analogous to computational components within a circuit [4]. CSNs continuously process changing environmental stimuli via this network to generate behavior suited to current conditions. Bray refers to an instantaneous set of protein concentrations as a random access memory containing an imprint of the current environmental state. The activity is determined by kinetic factors such as binding affinities or in reaction kinetic terminology: the reaction rate, reaction order and concentration of the reacting molecules. Where conditions are highly reactive, a processing unit acts like a molecular

switch giving a binary response. Such processing can be compared to that of Boolean logic. Or, in cases of lower reactivity, a unit may provide a more gradual response comparable to fuzzy logic. These processing units are linked together in cascades of protein coupled interactions with various network topological features such as feedback loops and interconnectivity and are thus capable of forming immensely complex networks. Bray claims that such a network of logical units can perform any kind of computational processing, equivalent to a finite state-machine with the same capability as a Turing machine. Evidence concerning the logical operation of protein units can be found in a number of independent studies. Stadtman et al demonstrated that the interconversion between phosphorylated and non-phosphorylated proteins can act as a flexible computational unit [9]. Similar results were documented by Arkin and Ross who examined the computational properties of enzymatic reactions [10].

Bray highlights the similarities between CSNs and ANNs. Both are examples of networked processors, simultaneously integrating and processing signals. Where weights in a neural network are set by a learning algorithm, the strength of connections within a CSN is set by natural evolution.

CSNs are the principle machinery of cellular intelligence. They may inspire new AI techniques, not only because they allow adaptive "intelligent" behavior, but also because of their intrinsic computational and processing abilities.

III. The E. coli Chemotaxis Pathway

The chemotaxis CSN of *Escherichia coli* is well characterized [11], and as such presents an ideal pathway to explore emergent properties of cell intelligence. *E. coli* have four types of transmembrane chemoreceptor proteins called methyl accepting proteins or MCPs responsible for sensing environmental chemoeffectors and a common set of cytoplasmic signaling proteins e.g. CheA, which transmit signals by reversible phosphorylation. Where no chemoeffectors are present, *E.coli* alternates between runs and tumbles, with runs lasting approximately 1 second and tumbles for 0.1 second [11]. In the presence of chemoeffectors, tumbling frequency is reduced up concentration gradients of attractants and down gradients of chemorepellents, resulting in a biased random walk. Thus, longer duration of swims in response to higher attractant gradients result in the emergence of a high level behavior characterized by net locomotion toward more favorable conditions.

To prevent the cell from being locked in either the swim or tumbling state, the cell also has a complex adaptation response. This response increases or decreases the sensitivity of the cell, depending on current ligand occupancy, by regulating the methylization of the MCP complex, so giving the cell a primitive memory.

In the two-state model [12] the MCP receptor complex is in equilibrium between two states: swim and tumble, where chemorepellents bind to the tumble form of receptor. As methylization of the MCP complex increases the receptors shift toward the tumble form of the receptor. In this form, the receptors phosphorylate CheA molecules which then transfer phosphoryl groups to aspartate residues on CheY and CheB. Phosphorylated CheY ($CheY_p$) interacts with the flagellar motor proteins triggering clockwise motor rotation (CW) resulting in a tumbling response. As $CheY_p$ concentration increases so does the tumbling frequency. CheZ is responsible for dephosphorylation of $CheY_p$. CheB and CheR are responsible for updating the methylation record and hence the adaptation response. The adaptation response drives the CSN toward its pre-stimulus equilibrium by demethylization of the MCP complex. A comprehensive description of this network is provided by Vladimirov and Sourjik [11].

IV. Related Techniques

The exploration of cellular intelligence requires a representation which focuses on high-level behaviors that emerge from CSN system dynamics, yet still capture the processing behaviors of individual reaction units. There are numerous methods of representing chemical reactions, ranging from the meticulously detailed quantum mechanical to the highly abstracted discrete Boolean models. Gilbert et al provides an excellent overview of current popular methods [13]. In this paper we shall consider only the most relevant, that is, those which capture their networked topology.

Random Boolean Networks, introduced by Kauffman, consist of a set of logical nodes, where each node corresponds to a real world object such as a gene or protein [14]. The nodes are connected to form a circuit, where the current state of each node is calculated by performing a Boolean function on its inputs. These, although focused on network dynamics, discard most unit behavior, preferring a binary switch response rather than continuous signals, and therefore cannot capture subtle system dynamics.

The Artificial Biochemical Network (AB-net) is a highly abstracted model of a CSN, intended for robotic control. It consists of a set of nodes representing protein activity, linked by weighted connections. The output of each node is a binary square-wave signal based on the input protein activities [6].

A more recent approach is the artificial biochemical neuron (AB-neuron); currently applied to phosphorylation cycles [15]. Similarly to the AB-net, it consists of a number of nodes with weighted connections. In this model the Michalis-Menton equation provides the unit output, representing the steady-state concentration of the product. Both the AB-neuron and the AB-net are simplified representations and neither capture realistic biological behavior.

Petri Nets are used extensively in several types of information processing, including modeling CSNs [16]. They work by passing tokens representing molecules between network units. In their simplest form they have similar functionality and limitations to RBNs. However, a

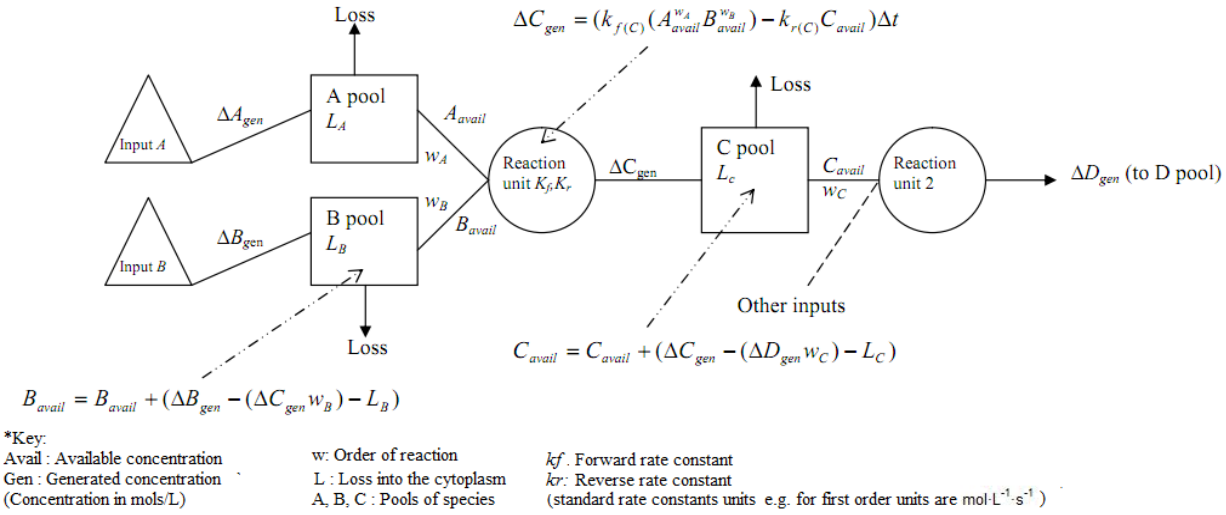


Fig. 1. The Artificial Reaction Network (ARN).

number of researchers have used them as a basis to produce more complex models.

Space precludes a complete discussion of all related models; however, it should be noted that there are several other network representations, less relevant to the problem at hand. For example, artificial immune network algorithms, and protein-protein interaction networks.

V. The Artificial Reaction Network

As explained in the previous sections, our focus is to capture the emergent cellular behavior that results from intracellular CSN processes. To achieve this, a model capable of representing sizeable networks and complex topologies, yet still maintaining biological plausibility was required. For this purpose, current methodologies were unsuitable, being either too simple or too complex, thus the authors created the ARN based on the following methods.

Developed by Savageau, S-systems are a popular representation used to model biological systems since the late 1960s [17]. They are composed of sets of ordinary differential equations (ODEs) that exploit a canonical power law representation to approximate chemical flux. Each ODE is composed of species concentration variables, raised to a power and multiplied by pseudo rate constants, as shown in Equation (1). Similarly to a traditional rate law, each ODE is equal to the difference between two conceptually distinct functions, the first term contributing to system influx, the second to decay.

To meet the previously discussed requirements, the authors combined the S-system approach with features found in RBNs and Petri Nets. By exploiting the simplified modular properties of RBNs with molecular transitions characteristic of Petri Nets, the ARN, as shown in Figure 1, represents a new, innovative, modular and expandable S-System. The ARN comprises a set of connected reaction nodes (circles), pools (squares), and inputs (triangles). Each pool represents a current species concentration (avail) measured in mols/L. Each circle represents a reaction, and calculates current flux at each time step (Δt), using Euler's approximation to the rate equation shown in Equation (1).

$$\dot{V} = k_F [A]^f [B]^g - k_R [S]^i [T]^j \quad (1)$$

Where:
$[S]^n$ = S is a species concentration, n its reaction order.
$\dot{V}$ = Current reaction rate
k_F = Forward rate constant
k_R = Reverse rate constant

Connections symbolize the flow of species into and out of reaction units and their weight (w) corresponds to reaction order. Flux (ΔA/ΔB/ΔC) as in Equation (1) and similar to S-systems, is equal to an aggregate of connected contributing (incoming) pools and connected decay (outgoing) pools raised to n powers of weighted connections and multiplied by pseudo rate constants. The pools are further subject to an optional degradation term (L), representing the natural cytoplasmic decay of species over time. This method provides each reaction with a temporal flux value, which is then used to update the current concentration values of each reaction's corresponding incoming and outgoing pools. Thus the complete set of pool concentrations at t, corresponds to the current state of the system.

The pool concept originates in Petri Nets and allows the system to account for accumulated molecular concentrations within the cytoplasm. By chaining several pools together chemical gradients and translocation through membranes can be represented; this facility is not available in standard S-systems.

Where S-systems are highly coupled sets of ODEs, the ARN is a modular approach offering finer degree of control, flexibility and adaptation. This not only supports simulation development by promoting object-orientation but is perceptually intuitive, mirroring the topology and modularization of its real-world counterpart. Thus the ARN representation is ideally suited to characterize emergent

behavior resulting from both subtle and high-level complex temporal system dynamics.

VI. RESULTS

Before the ARN could be applied to simulate cellular intelligence, its accuracy needed to be verified against known biological data and standard models. This was achieved by application of varied sets of real biochemical data to a single ARN unit. The resultant output was compared with those recorded in literature, manual calculation and by running the experiment on the Berkeley Madonna [18] programme. The outputs of these experiments confirmed its accuracy, with a minor error as expected from Euler's approximation. Figures 2 and 3 provide typical results from one such experiment. Here reaction kinetic data (rate constants, reaction order) were used to create a model of the reversible isomerisation reaction between cis and trans 1-ethyl-2-methyl cyclopropane on Berkeley Madonna and on a single ARN unit. Figure 2 shows the product output from Berkeley Madonna, and Figure 3 is that of the single ARN unit. After 2000 seconds, it can be seen that the product concentration produced by Berkeley Madonna and the single ARN are both 9.1×10^{-3} mol dm^{-3}. This result is the same as that recorded by the standard literature, thus confirming the biological plausibility of a single ARN unit.

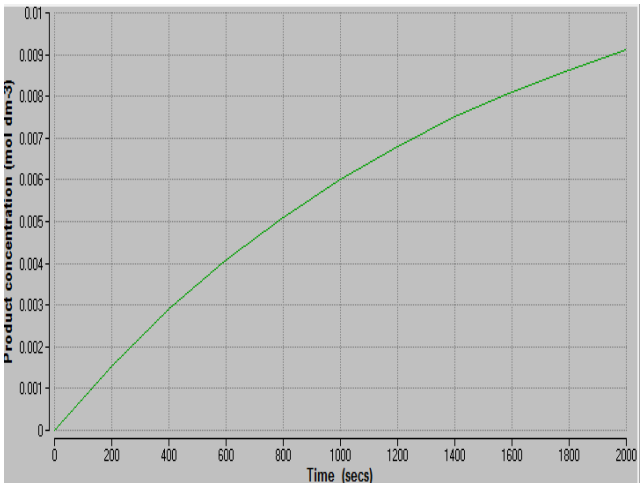


Fig. 2. The product concentration produced by Berkeley Madonna.

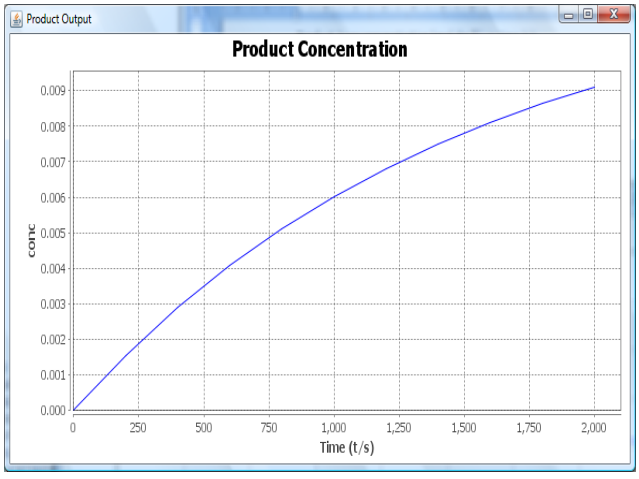


Fig. 3. The product concentration produced by the single ARN unit.

Having verified the biological plausibility of a single ARN unit, the ARN was tested as a means of capturing properties of cellular intelligence. A two state model, (refer to section3), was used as a basis to create a simulation of the chemotaxis CSN of *E. coli*. The structure of this simulation is shown in Figure 4 and is represented in the ARN format described in Figure 1 of the previous section. It is composed of a network of 10 reaction units numbered 0-9, 11 pools of intracellular signaling proteins, a single input representing the chemorepellent, and arrowed lines to show not only the connections but direction of signal flow through the network. The behavior of the simulated chemotaxis pathway in varying levels of environmental chemorepellent was setup using real biological data gathered from sources at the University of Cambridge [19], [20]. The output from this network is shown in Figures 5 and 6. Figure 5 shows the steady state concentration levels of $CheY_p$ in mols/L generated by the ARN simulation at four different continuous concentration levels of environmental chemorepellent. It can be seen from the graph that as the level of environmental chemorepellent increases so does the concentration of $CheY_p$ and therefore the tumbling frequency of the cell increases. The results are in clear agreement with published data from respected systems biology simulations [12].

To prevent the cell from being locked in either the swim or tumbling state the cell also has a complex adaptation response (refer to section 3). To ascertain the ability of the ARN to capture this behavior, the steady state concentration in mols/L of methylized MCP receptor complex obtained by the ARN simulation were examined at varying levels of continuous environmental chemorepellent.

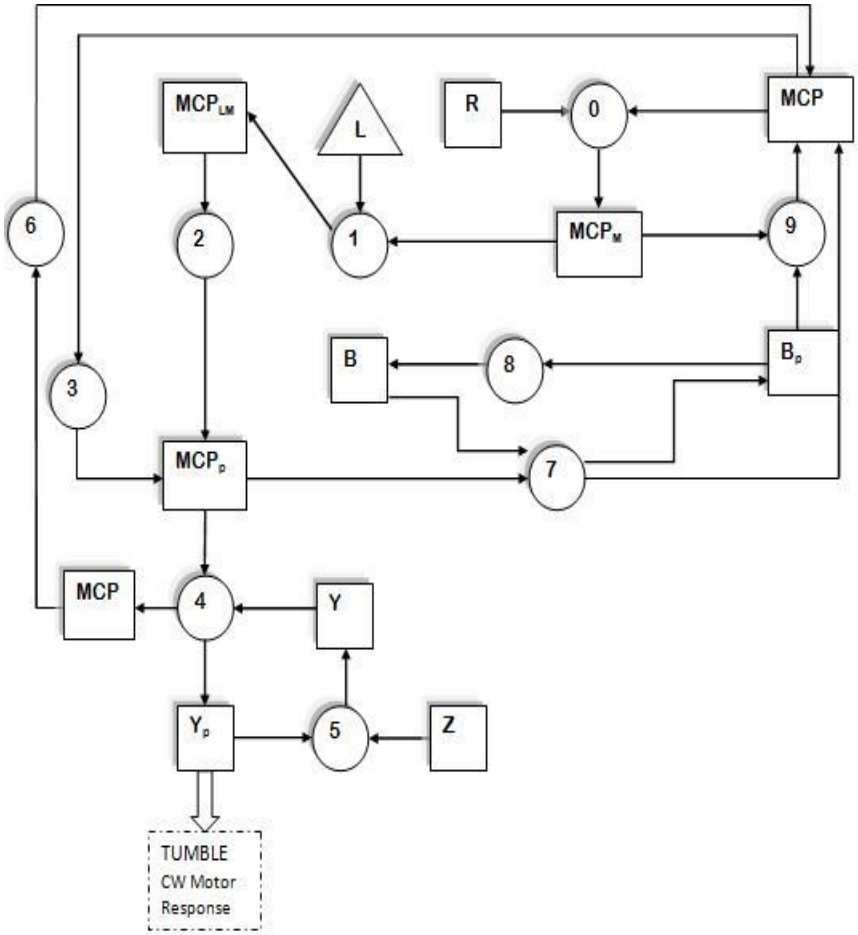
The output is displayed in Figure 6, where it can be seen that when chemorepellent concentration increases $CheY_p$ increases, and methylized MCP decreases thus driving the network back to the pre-stimulus equilibrium. Although a minor change to rate constant values were required, it can be seen that the adaptation response was attained and is in good agreement with previous work [21]

Finally to demonstrate the emergent behavior of the simulated CSN, it was decided to show the chemorepellent avoiding behavior in the context of an optimization problem. Here we observed the behavior of the simulated *E. coli* chemotaxis pathway to ascertain its ability to find a minimum chemorepellent level in an inverted bowl search space where x and y are on the horizontal plane:

$$z = \sqrt{x^2 + y^2} \tag{2}$$

Figure 7 displays the search space and an example run. The centre of the search space (solid black square) corresponds to an area of 0 chemorepellent concentration. With each progression outwards repellant concentration increases, and the outermost perimeter signifies a maximum concentration of 1×10^{-7} mols/L. The path of the simulated *E. coli* is displayed as a white line. Over 100 seconds the cell

remains in high concentration areas (above $1x10^{-9}$ mols/L) for 11 seconds and low (below $1x10^{-9}$ mols/L) for 89 seconds. These results were verified statistically over 100 run, and are in good correspondance with the reported behaviour of *E. coli* chemotaxis described in literature and using other simulation methods [21].



Key	
MCP	non-methylated methyl accepting chemotaxis proteins (consists CheA, CheW and receptor) TTWWAA
MCP_M:	methylated methyl accepting chemotaxis protein
MCP_{LM}	Methylated methyl accepting chemotaxis protein bound to ligand
Yp:	phosphorylated CheY
B:	CheB
Y:	CheY
A:	CheA
R	CheR
MCP_P:	phosphorylated CheA bound to MCP : TT_MWWAA_p, $TTWWAA_p$, TT_MWWAA_p
B_p:	phosphorylated CheB
Z:	CheZ
L	Ligand
o	(circles) reaction
□	(square) chemical pools
→	Direction of signal flow

Fig. 4. A two-state model of the chemotaxis CSN of E. coli is shown diagrammatically using the format specified in Figure 1.

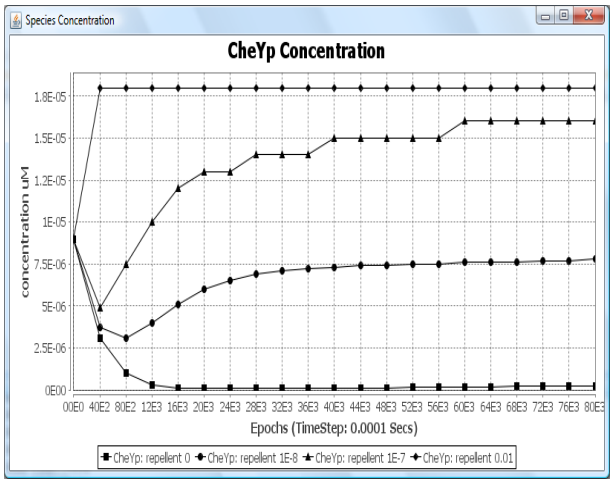


Fig. 5. The steady state concentration levels of $CheY_p$ in mols/L recorded by the ARN when subjected to varied levels of chemorepellent.

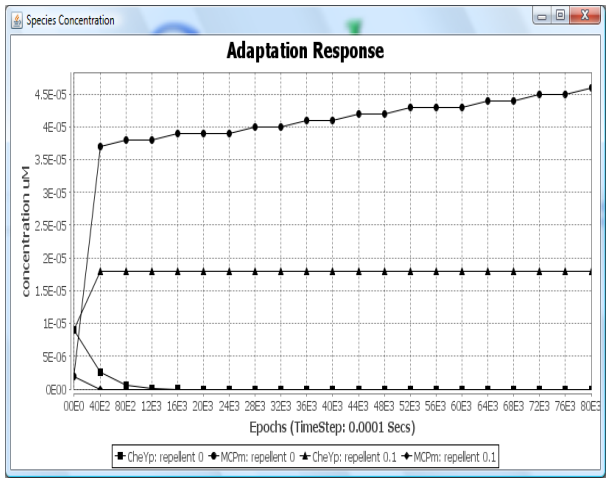


Fig. 6. The steady state concentration levels of $CheY_p$ and methylized MCP in mols/L recorded by the ARN when subjected to varied levels of chemorepellent.

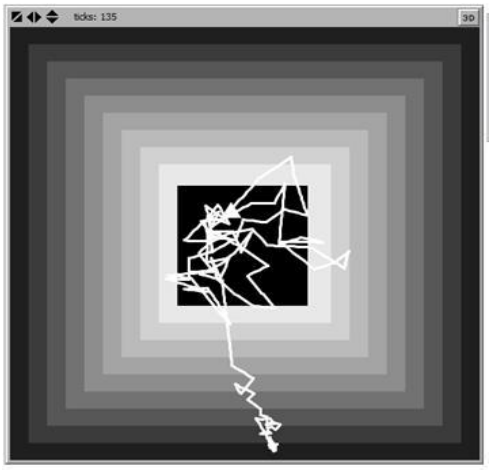


Fig. 7. Minimum seeking behavior in an inverted bowl search space.

VII. CONCLUSIONS

In this paper, the ARN representation was presented as a novel method of simulating cellular intelligence. Initially, its ability to successfully represent single node reaction dynamics was shown. Its efficacy and applicability was demonstrated by creating a working model of the CSN of *E.coli* chemotaxis. This confirmed its ability to effectively simulate both the tumbling frequency regulation and adaptation response behavior of the bacteria. Furthermore, the emergent random biased walk behavior generated by the ARN was demonstrated in a general optimization problem.

The ARN approach has several advantages over other similar techniques. Its network-like structure exploits the benefits of modularization found in RBNs. It uses the molecular accounting approach of Petri Nets; however, it also incorporates the complex temporal dynamics of individual reactions found in S-Systems. The addition of pools and loss mechanisms allows more flexibility to represent intracellular compartmentalization than other techniques. The authors therefore feel that its representation is ideally suited to the characterization of emergent behaviors resulting from both subtle and high-level temporal system dynamics. Furthermore, it offers a perceptually intuitive method, as it mirrors the topology and modularization of its real-world counterpart. Aside from biological systems, this approach may also have some advantages in the simulation of other chemical systems; in particular, in the complex networks of reactions present in soil and environmental chemistry.

The modularized form of the ARN makes it particularly suitable for the application of evolutionary algorithms. The success of simulating real biological systems is generally predicated on obtaining good experimental data, which is often missing or is unreliable. Thus, the ARNs evolvability may prove useful since it promotes the identification of network parameters.

The parallels between *E. coli* chemotaxis and robotic control should be obvious. The next stage of our work involves adapting the ARN into a cellular intelligence inspired AI technique. It is intended to explore its potential as a source for development of robotic control systems and optimization techniques.

VIII. REFERENCES

[1] B. J. Ford, "Are cells Ingenious?," *The Microscope*, vol. 52, no. 3-4, pp. 135-144, 2004.

[2] K. J. Hellingwerf, "Bacterial observations: a rudimentary form of intelligence," *Trends in Microbiology*, vol. 13, no. 4, pp. 152- 158, Apr. 2005.

[3] E. Ben-Jacob, "Bacterial self-organization: co-enhancement of complexification and adaptability in a dynamic environment," *Philosophical Transactions A*, vol. 361, no. 1807, pp. 1283-1312, Jun. 2003.

[4] D. Bray, "Protein molecules as computational elements in living cells," *Nature*, vol. 376, no. 6538, pp. 307-12, Jul. 1995.

[5] J. Y. Potvin, "A review of bio-inspired algorithms for vehicle routing," in *Bio-inspired Algorithms for the Vehicle Routing Problem*, F. B. Pereira, and J. Tavares, Ed. Berlin: Springer, 2009, pp. 1-34.

[6] C. MacLeod, and N. F. Capanni, "Artificial biochemical networks: a different connectionist paradigm," *Artificial intelligence review*, vol. 33, no. 1-2 , pp. 124-135, Feb. 2010.

[7] T. Nakagaki, H. Yamada, and A. Toth, "Maze-solving by an amoeboid organism," *Nature*, vol. 407, no. 6803, pp.470-470, Sep. 2000.

[8] T. Saigusa, A. Tero, T. Nakagaki and Y. Kuramoto, "Amoebae Anticipate Periodic Events," *Physical Review Letters*, vol. 100, no. 1, pp. 1-4, Jan. 2008.

[9] E. R. Stadtman, and P.B. Chock, "Superiority of interconvertible enzyme cascades in metabolic regulation: analysis of monocyclic systems," in *Proceedings of the National Academy of Sciences USA*, vol. 74, no. 7, pp. 2761-2765, Jul. 1977.

[10] A. Arkin, and J. Ross. "Computational functions in biochemical reaction networks," *Biophysical Journal*, vol. 67, no. 2, pp. 560-578, Aug. 1994.

[11] N. Vladimirov, and V. Sourjik, "Chemotaxis: how bacteria use memory," *Journal of Biological Chemistry*, vol. 390, no. 11, pp. 1097-1104, Nov. 2009.

[12] S. Asakura and H. Honda, "Two-state model for bacterial chemoreceptor proteins. The role of multiple methylation," *Journal* of Molecular Biology, vol. 176, no. 2, pp. 349-367, Jul. 1984.

[13] D. Gilbert, H. Fub, X. Gu, R. Orton, S. Robinson, V. Vyshemirsky, M. J. Kurth, C. S. Downes, and W. Dubitzky, "Computational methodologies for modeling, analysis and simulation of signaling networks," Briefings in Bioinformatics, vol.7, no.4, pp. 339-353, Jan. 2006.

[14] S. A. Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of Theoretical Biology*, vol. 22, no. 3, pp. 437-467, Mar. 1969.

[15] H. M. M. Ten Eikelder, S. P. M. Crijns, M. N. Steijart, A. M. L. Liekens, and P. A. J. Hilburs, "Computing with feedforward networks of artificial biochemical neurons," in *Natural computing Proceedings in Information and Communications Technology*, Nagoya, 2009, vol. 1, pp. 38-47.

[16] J. L. Peterson, Petri *Net Theory and the Modeling of' Systems*. Englewood Cliff, N.J: Prentice-Hall, 1981.

[17] M. A. Savageau, and E. O. Voit, "Recasting Nonlinear Differential Equations as S-Systems: A Canonical Nonlinear Form," *Mathematical Biosciences*, vol. 87, no. 1, pp. 83–115, Nov. 1987.

[18] R. Macey, G. Oster, and T. Zahnley (2011, Feb 12). Berkeley Madonna v.5, and User's Guide 2000. [Online]. Available: http://www.berkeleymadonna.com

[19] D. Bray (2011, May 20). Biochemical data (1998-2010) Bacterial chemotaxis in silico [Online] Available: http://www.pdn.cam.ac.uk/groups/comp-cell/Data.html

[20] C. J. Morton-Firth and R. B. Bourret (2011, May, 20). Experimental data in bacterial chemotaxis [Online] Available: www.pdn.cam.ac.uk/groups/comp-cell/Exp_data/exptdata.doc

[21] D. Bray, M. D. Levin, and K. Lipkow, "The chemotactic behavior of computer-based surrogate bacteria," *Current Biology*, vol. 17, no. 1, pp. 12-19, Jan. 2007.

Appendix 2

This appendix contains the data used to set the ARN network parameters to create a model of the chemotaxis CSN of the bacteria *E. coli* in *chapter 5 section 5.2-5.3*. The data represents a collection of wet lab results published by leading researchers in the field. These results were gathered by researchers at the University of Cambridge to create a central resource for those studying or modelling this pathway.

Network Parameters for the ARN Simulation of the *E. coli* Chemotaxis Pathway

Unit	Input pools	Chemical Species in	Initial conc. in μmol/L	Output pools	Chemical species out	Initial Conc. out μmol/L	Overall Order	Forward Rate (mol s^{-1})	Reverse rate (mol s^{-1})
Reaction 0: R + TTWWAA ➔ TT_M WWAA + R									
0	0	R	1	2	MCP_M	5	1	2.36×10^{-4}	------
	1	MCP	5						
Reaction 1: TTmWWAA + a ⇔ TT_{LM}WWAA									
1	2	MCP_M	5	4	MCP_{LM}	0	1	1×10^{9}	1×10^{3}
	3	ligand	0						
Reaction 2: TT_{LM}WWAA➔ TT_{LM}WWAAp									
2	4	MCP_{LM}	0	5	Ap	1	1	1.9×10^{0}	------
Reaction 3: TTWWAA ➔TTWWAAp									
3	1	MCP	5	5	Ap	1		4.7×10^{-2}	------
Reaction 4: Y + Ap➔ Yp + A (A_p part of MCP complex)									
4	5	Ap	2	7	Yp	10	2	3×10^{7}	------
	6	Y	10	8	MCP	5			
Reaction 5: Yp + Z➔ Y + Z									
5	8	Yp	10	6	Y	10	1	5×10^{5}	------
	9	Z	20						
Reaction 6: MCP (various forms)➔ MCP									
6	7	MCP	5	1	MCP	5	1		
Reaction 7: B + AAp➔ Bp + AA (Ap is part of MCP complex)									
7	5	Ap	1	10	Bp	1	2	6×10^{6}	------
	9	B	1	1	A	2			
Reaction 8: CheBp➔ CheB									
8	10	Bp	1	9	B	1	1	3.5×10^{-1}	------
Reaction 9: TTmWWAA + CheBp➔ TTWWAA + CheBp									
9	2	TTmWWAA	2	1	TTWWAA	2	1	1×10^{5}	------
	10	Bp	1						

KEY

TTWWAA	MCP complex	**MCP** (shorthand version of TTWWAA)	MCP complex	MCP_{LM}	Methylated MCP complex bound to ligand	A	cheA
TTmWWAA	Methylated MCP complex	MCP_m	Methylated MCP complex	MCP_L	MCP complex bound to ligand	A_p	Phosphorylated cheA
Y	CheY	CheB	CheB	R	CheR		
Y_p	Phosphorylated CheY	$CheB_p$	Phosphorylated CheB				

*Approximately equal amounts of MCP_M and MCP when no chemoreceptor is bound (initial state).

Data gathered from:

Morton-Firth, C. J, & Bourret, R. B. Experimental data in bacterial chemotaxis [Online]
Available: www.pdn.cam.ac.uk/groups/comp-cell/Exp_data/exptdata.doc, [Last Accessed 10 May 2011].

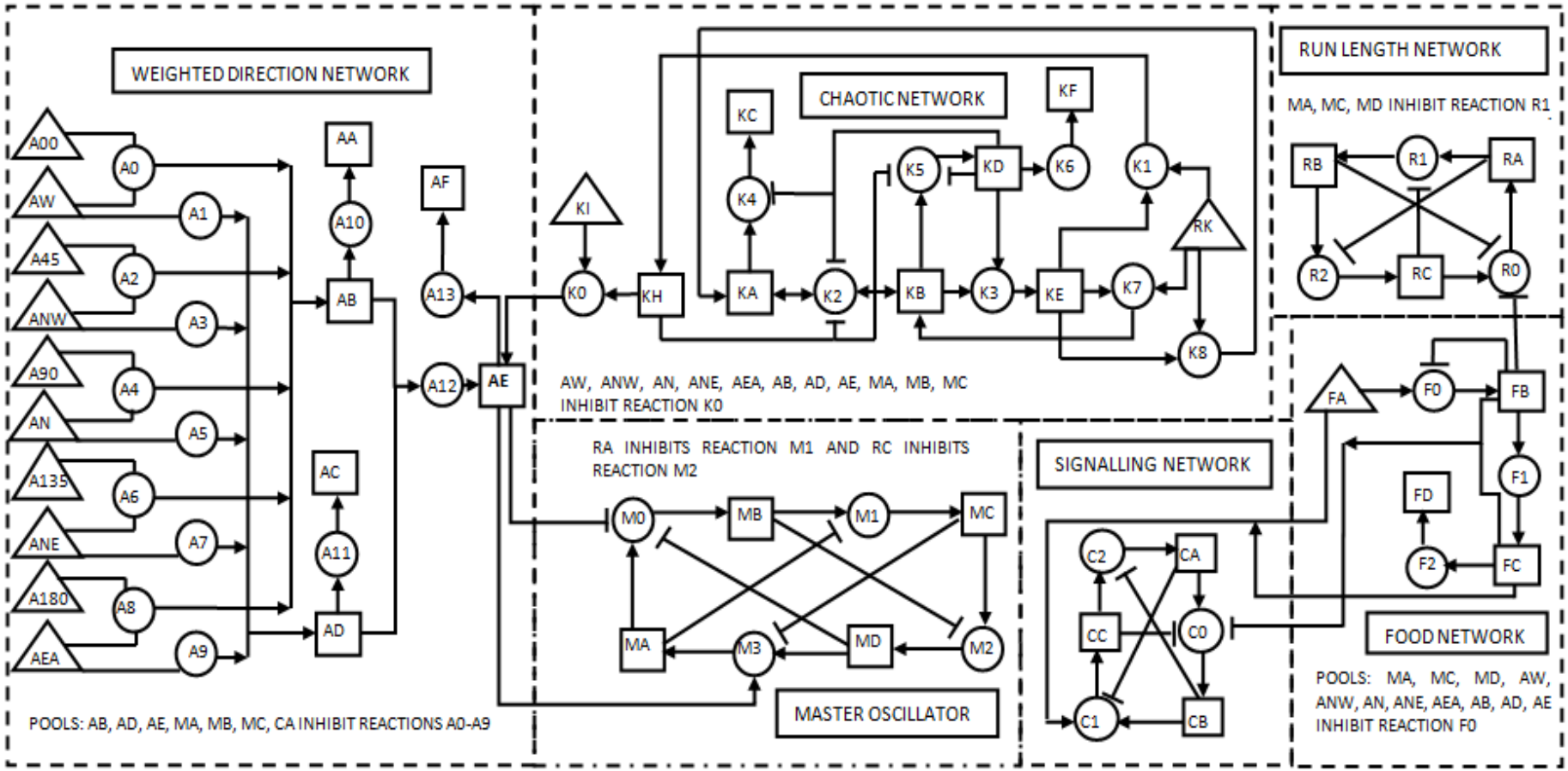
Appendix 3

This appendix provides, for clarity, a larger version of the diagram given in *chapter 7* which shows the Cytobot ARN structure. Each Cytobot is controlled by an instance of this network. It is composed of 6 subnetworks these are:

1. Master Oscillator
2. Run Length Network
3. Food Network
4. Signalling Network
5. Chaotic Network
6. Weighted Direction Network.

Each subnetwork contributes a functional aspect to either or both the Cytobot's Starvation and Foraging behavioural modes. The structure of each is based on functional motifs found in biochemical networks.

Note that Pools are considered empty when value of its component chemical is $\leq 1x10^{-3}$


WEIGHTED DIRECTION NETWORK
A00
AW
A45
ANW
A90
AN
A135
ANE
A180
AEA
A0
A1
A2
A3
A4
A5
A6
A7
A8
A9
AA
A10
AB
AC
A11
AD
AF
A13
A12
AE
POOLS: AB, AD, AE, MA, MB, MC, CA INHIBIT REACTIONS A0-A9
CHAOTIC NETWORK
KI
K0
KH
KC
K4
KA
K2
K5
KD
K6
KF
KB
K3
KE
K7
K1
RK
K8
AW, ANW, AN, ANE, AEA, AB, AD, AE, MA, MB, MC INHIBIT REACTION K0
RA INHIBITS REACTION M1 AND RC INHIBITS REACTION M2
M0
MB
M1
MC
MA
M3
MD
M2
MASTER OSCILLATOR
SIGNALLING NETWORK
C2
CA
CC
C0
C1
CB
RUN LENGTH NETWORK
MA, MC, MD INHIBIT REACTION R1
RB
R1
RA
R2
RC
R0
FA
F0
FB
F1
FD
F2
FC
FOOD NETWORK
POOLS: MA, MC, MD, AW, ANW, AN, ANE, AEA, AB, AD, AE INHIBIT REACTION F0

A107

Appendix 4

The complete listing of the Cytobot ARN parameters is provided in the table below.

This includes:

- Input and output pools of each reaction
- Weights of each connection
- Forward and reverse rates of each reaction
- Initial value of each pool.

The only parameters subject to change during the period of operation are the value of each pool. The initial values at time equals 0 are provided.

* Note that -5 means inhibitory connection and thus when active will switch off that reaction completely.

Run Length Network								
Reaction	Input pool	Weight	Output Pool	Weight	Forward Rate	Reverse Rate	Pool	Initial Value
R0	RB	-5	RA	1	0.9	0	RA	1
	RC	1					RB	0
	FB	-5					RC	0
R1	RA	1	RB	1	1	0		
	RC	-5						
	MA	-5						
	MC	-5						
	MD	-5						
R2	RA	-5	RC	1	1	0		
	RB	1						
Food Network								
Reaction	Input pool	Weight	Output Pool	Weight	Forward Rate	Reverse Rate	Pool	Initial Value
F0	FA	1	FB	1	1	0	FA	0
	FB	-5					FB	0
	MA	-5					FC	1
	MC	-5						
	MD	-5						
	AW	-5						
	ANW	-5						
	AN	-5						
	ANE	-5						
	AEA	-5						
	AB	-5						
	AD	-5						
	AE	-5						
F1	FB	0	FC	0	1	0		
F2	FC	1	FD	1	0.5	0		

Signalling Network								
Reaction	Input pool	Weight	Output Pool	Weight	Forward Rate	Reverse Rate	Pool	Initial Value
C0	CA	0	CB	1	0.5	0	CA	1
	CC	-5					CB	0
	FA	-5					CC	0
	FB	-5						
	FC	-5						
C1	CA	-5	CC	1	1	0		
	CB	1						
	FA	0						
	FC	0						
C2	CB	-5	CA	1	1	0		
	CC	1						
Master Oscillator								
Reaction	Input pool	Weight	Output Pool	Weight	Forward Rate	Reverse Rate	Pool	Initial Value
M0	MA	1	MB	1	1	0	MA	0
	MD	-5					MB	0
	AE	-5					MC	0
M1	MA	-5	MC	1	1	0	MD	1
	MB	1						
	RA	-5						
M2	MB	5	MD	1	1	0		
	MC	1						
	RC	-5						
M3	MC	-5	MA	1	1	0		
	MD	1						
	AE	0						

Weighted Direction Network								
Reaction	Input pool	Weight	Output Pool	Weight	Forward Rate	Reverse Rate	Pool	Initial Value
A0	A00	1	AB	1	1	0	A00	0
	AW	1					AW	0
	AB	-5					A45	45
	AD	-5					ANW	0
	AE	-5					A90	90
	MA	-5					AN	0
	MB	-5					A135	135
	MC	-5					ANE	0
A1	AW	1	AD	1	1	0	A180	180
	AB	-5					AEA	0
	AD	-5					AA	0
	AE	-5					AB	0
	MA	-5					AC	0
	MB	-5					AD	0
	MC	-5					AE	0
A2	A45	1	AB	1	1	0	AF	0
	ANW	1						
	AB	-5						
	AD	-5						
	AE	-5						
	MA	-5						
	MB	-5						
	MC	-5						
A3	ANW	1	AD	1	1	0		
	AB	-5						
	AD	-5						
	AE	-5						
	MA	-5						
	MB	-5						
	MC	-5						
A4	A90	1	AB	1	1	0		
	AN	1						
	AB	-5						
	AD	-5						
	AE	-5						
	MA	-5						
	MB	-5						
	MC	-5						

A5	AN	1	AD	1	1	0		
	AB	-5						
	AD	-5						
	AE	-5						
	MA	-5						
	MB	-5						
	MC	-5						
A6	A135	1	AB	1	1	0		
	ANE	1						
	AB	-5						
	AD	-5						
	AE	-5						
	MA	-5						
	MB	-5						
	MC	-5						
A7	ANE	1	AD	1	1	0		
	AB	-5						
	AD	-5						
	AE	-5						
	MA	-5						
	MB	-5						
	MC	-5						
A8	A180	1	AB	1	1	0		
	AEA	1						
	AB	-5						
	AD	-5						
	AE	-5						
	MA	-5						
	MB	-5						
	MC	-5						
A9	AE	1	AD	1	1	0		
	AB	-5						
	AD	-5						
	AE	-5						
	MA	-5						
	MB	-5						
	MC	-5						
A10	AB	1	AA	1	1	0		
A11	AD	1	AC	1	1	0		
A12	AB	1	AE	1	1	0		
	AD	-1						
A13	AE	0	AF	1	1	0		

Random Angle Network								
Reaction	Input pool	Weight	Output Pool	Weight	Forward Rate	Reverse Rate	Pool	Initial Value

K0	KI	1	AE	1	1	0	KA	0.67839
	KH	1					KB	0.67839
	AW	-5					KC	0
	ANW	-5					KD	0
	AN	-5					KE	0
	ANE	-5					KF	0
	AEA	-5					KH	0
	AB	-5					KI	360
	AD	-5					RK	1
	AE	-5						
	MA	-5						
	MB	-5						
	MC	-5						
K1	KG	0	KH	1	1	0		
	KE	1						
K2	KA	1	KB	2	4	4		
	KD	-5	KD	-5				
	KH	-5	KH	-5				
K3	KB	1	KE	1	1	0		
	KD	0						
K4	KA	1	KC	1	1	0		
	KD	0						
K5	KB	1	KD	1	1	0		
	KD	-5						
	KH	-5						
K6	KD	1	KF	1	1	0		
K7	KE	1	KB	1	1	0		
	KG	0						
K8	KE	1	KA	1	1	0		
	KG	0						

Appendix 5

In this appendix a working example of the Chaotic Network component of the Cytobot ARN control system presented in *chapter 7* is provided.

1. **At Δt_0** KA and KB are initialised to 0.38475.

2. **At Δt_1**, the flux generated at K2 will be 0.94687 (Applying Eq. (3) of *chapter 4* to calculate the flux: (0.38475)1-4(0.38475)2 = 0.94687). At the same time the values in pools KA and KB are being processed by reactions K4 and K5 respectively. In both these reactions the weights and forwards rates are 1, thus the entire content of pool KA and KB are transferred to pools KC and KD respectively in a single time step. KB has an additional connecting reaction, K3, but it cannot proceed until the connecting pool, KD, is active (not empty). Pools are updated at the end of each discrete time step, thus at Δt_1 pool KB will now have a value of 0.94687 (due to the positive flux generated at reaction unit K2) while pools KD and KC will each have a value of 0.38475.

3. **At Δt_2,** reaction K5 cannot proceed because pool KD is actively inhibiting, while reaction K3 may now proceed because pool KD and KB must both contain chemical in order for it to proceed. Reaction K3 has a forward rate and weight of 1, thus transferring its entire content of 0.94687 to pool KE in one time step. At the same time the entire content of pool KD is transferred to KF which acts as a waste pool (representing for example the environment) where chemical is eliminated from the system.

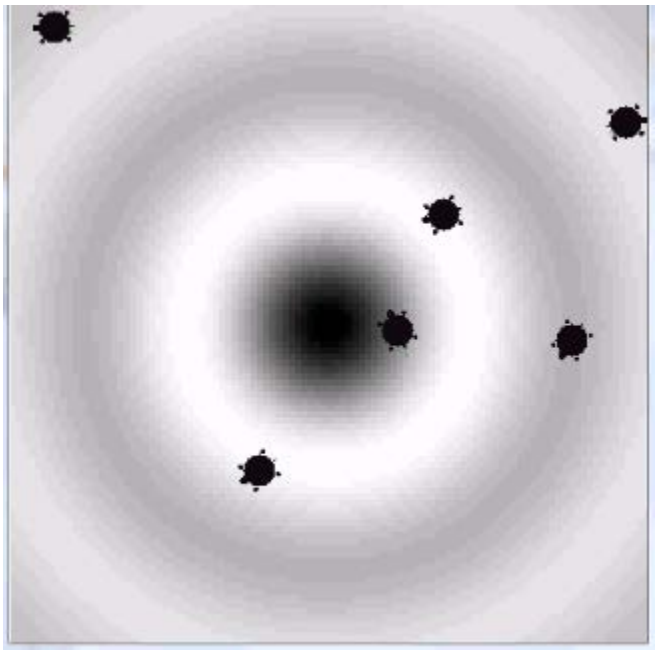
4. **At Δt_3** the content of pool KE is being simultaneously processed by reactions K1, K7 and K8. In all these reactions pools KE and the static pool RK react, where the forward rate is 1 the weight between the static pool and the reaction is 0. These reactions function to make copies of the current value, allowing one copy to be used as the

network output and the other two as the new initial values of pools KA and KB to restart the network cycle. In our example the flux generated for each reaction is 0.94687, and each copy of flows into pools KA, KB and KH.

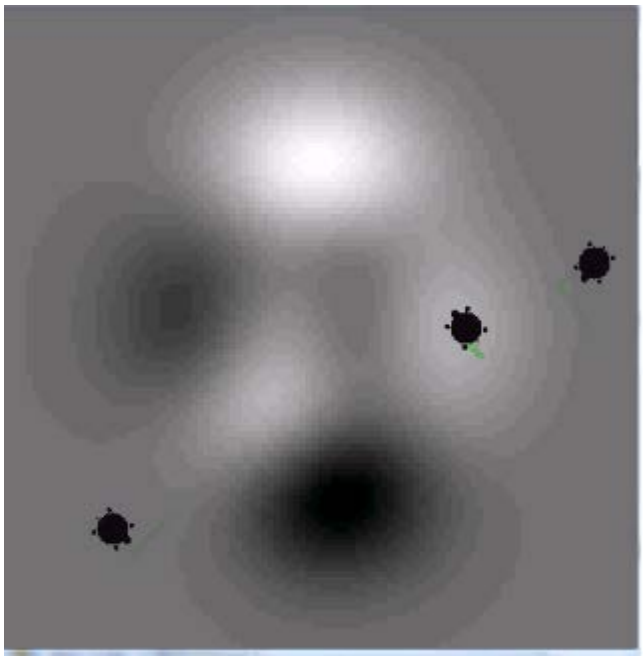
5. **At Δt_4** pool KH actively inhibits reaction K2 thus the next network cycle cannot commence until pool KH has emptied. Pool KH represents the output of the chaotic network and it interfaces with the rest of the network through reaction K0. Reaction K0 processes KH and static pool KI, in a single time step, where both forward rate and weight is 1. Static pool KI has a fixed value of 360 which allows the network to convert the pseudo random number to an angle value between 0 and 360. Thus in this example the value of 0.94687 becomes 340.8732. However, reaction K0 cannot proceed until all 11 pools that inhibit it are empty. These inhibitory connections ensure that random angles are not output while the agent is in starvation mode, and that pool AE is empty before adding more chemical.

Appendix 6

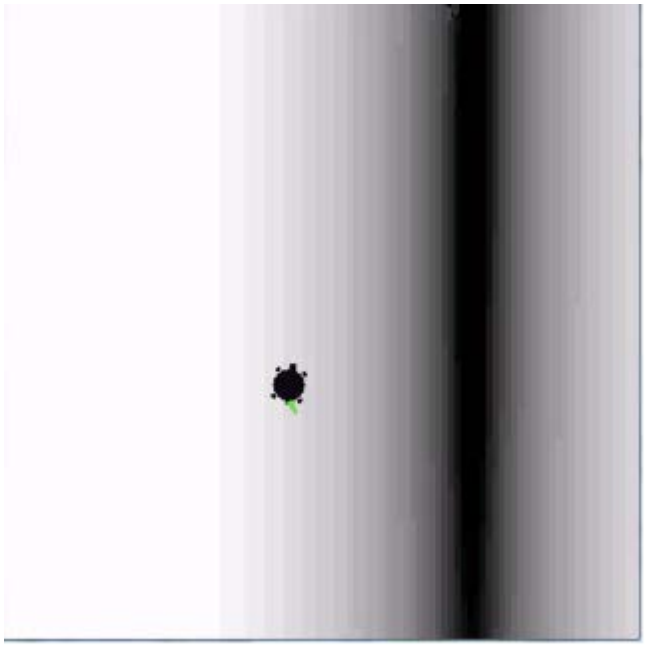
The following screenshots show the search spaces used in the Cytobot optimisation experiments outlined in *section 8.2*. This includes the following functions: 1. Inverted sinc; 2. Peaks; and 3. Rosenbrock.



1. Inverted sinc function with 6 Cytobots



2. Peaks with 3 Cytobots



3. Rosenbrock function with 1 Cytobot