THE UNIVERSITY OF HULL

Dynamic Networks for Robotic Control and Behaviour Selection in Interactive Environments

being a Thesis submitted for the Degree of
Doctor of Philosophy
in the University of Hull

by

Brian Peach MEng.

February 2019

Abstract

Traditional robotics have the capabilities of the robot hard coded and have the robot function in structured environments (structured environments are those that are predefined for a given task). This approach can limit the functionality of a robot and how they can interact in an environment. Behaviour networks are reactive systems that are able to function in unstructured dynamic environments by selecting behaviours to execute based on the current state of the environment. Behaviour networks are made up of nodes that represent behaviours and these store an activation value to represent the motivation for that behaviour. The nodes receive inputs from a variety of sources and pass proportions of that input to other nodes in the network.

Behaviour networks traditionally also have their capabilities predefined. The main aim of this thesis is to expand upon the concepts of traditional robotics by demonstrating the use of distributed behaviours in an environment. This thesis aims to show that distributing object specific data, such as; behaviours and goals, will assist in the task planning for a mobile robot.

This thesis explores and tests the traditional behaviour network with a variety of experiments. Each experiment showcases particular features of the behaviour network including flaws that have been identified. Proposed solutions to the found flaws are then presented and explored. The behaviour network is then tested in a simulated environment with distributed behaviours and the dynamic behaviour network is defined. The thesis demonstrates that distributed behaviours can expand the capabilities of a mobile robot using a dynamic behaviour network.

Acknowledgements

First and foremost I would like to thank everyone that has supported me throughout my time study for the PhD. I am especially thankful to Dr. Peter Robinson for his continuous support as my supervisor, for his patience, criticism and guidance. Without his help this thesis would not be what it is today.

I would like to extend my gratitude to Dr. Darryl Davis and Dr. Yongqiang Cheng who also co-supervised me during my PhD as the panel chair and technical expert. Their constructive criticism helped to keep me on the path during the panel meetings and ensured that I made great progress. I am also grateful to all the staff in computer science department who were able to support me throughout the PhD.

I am especially grateful to all of my colleagues and friends who I was able to share this experience with. A special thanks to Dr. John Stamford who I was able to work with on many side projects and share many discussions with. I am also grateful for the help from all of the administrative staff for their support and assistance during my PhD.

Finally, I would like to give my deepest thanks to my partner Louise Foster for her continuous support over the years. Thank you for your support and patience during my PhD and supporting my many late nights to see this through to completion. I would also like to thank all of my family and friends for their support too.

Declaration

The research conducted in this thesis represents the original work conducted by the author. One key area of the work covered in this thesis has been published in a conference and scientific publication:

1) Peach, B. and Robinson, P. (2016) 'The Use of Data Packets in a Behaviour Network to Improve the Action Selection Mechanism', PlanSIG 2016.

List of Abbreviations

This section will contain a list of all of the abbreviations used in this thesis.

AI	Artificial Intelligence
ANA	Agent Network Architecture
ASM	Action Selection Mechanism
ASMO	Attentive Self-MOdifying
BN	Behaviour Network
DBN	Dynamic Behaviour Network
GDA	Goal Driven Autonomy
GUI	Graphical User Interface
HTN	Hierarchical Task Network
MASM	Mae's Action Selection Mechanism
RRT	Rapidly-exploring Random Tree
STRIPS	Stanford Research Institute Problem Solver
UAV	Unmanned Aerial Vehicle
UI	User Interface
URL	Uniform Resource Locator
VMT	Vehicle Model Trajectory

Table of Contents

1. INTRODUCTION 11
1.1. General Statement 11
1.2. Research Hypothesis 13
1.2.1. A gap in the topic domain 14
1.2.2. Hypothesis 14
1.2.3. The importance of the research 15
1.3. Research Objectives 16
1.3.1. Aims 16
1.3.2. Limitations 17
1.3.3. Method 18
1.4. Thesis Structure 18
1.5. Contribution 20
1.6. Publications 21

2. BACKGROUND 22
2.1. Traditional AI 22
2.2. Behaviour-Based AI 26
2.2.1. Behaviour-Based Architecture 30
2.2.1.1 Basics of the Behaviour-Based Architecture 31
2.2.1.2 Competition Vs Cooperation 33
2.2.2 Subsumption Architecture 35
2.2.3 Action Selection 43
2.2.3.1 Symbolic Approach 44
2.2.3.2 Distributed Approach 45
2.2.3.3 Dynamic Planning Approach 48
2.2.4 Behaviour Networks 49
2.2.4.1 Introduction to Behaviour Networks 49
2.2.4.2 Components of a Behaviour Network 50
2.2.4.2.1 Behaviours 51
2.2.4.2.2 Components 52
2.2.4.2.3 Action Selection 55
2.2.4.2.4 Activation Spreading and The Division Rule 56
2.2.4.3 Factors of Favourability 61
2.2.4.4 Liabilities 62
2.2.4.5 Improvements to Behaviour Networks 64
2.2.4.6 Recent Work 65
2.3. Alternative Approaches to Robot Control 67
2.3.1. Hierarchical Mechanisms 67
2.3.2. Reinforcement Learning 71
2.3.3. Neural Networks 73

3. PROPOSED APPROACH AND ARCHITECTURE 78
3.1. Overview 78
3.2. Wireless Tag Manager 81
3.3. Environment Manager 83
3.4. Behaviour Network Manager 84
3.5. User Interaction 85
3.6. Benefits of proposed architecture 87
3.7. Limitations of proposed architecture 87

4. BEHAVIOUR NETWORK 89
4.1. A Solution to the Division Rule 89
4.2. Results of the Division Rule 93
4.2.1. Benefits of the new division rule 96
4.2.2. Limitations of the new division rule. 96
4.3. Energy Spreading in the Behaviour Network 97

4.3.1. Energy Packets 105
4.3.3.1. Experiment 1 107
4.3.3.2. Experiment 2 108
4.3.3.3. Experiment 3 114
4.3.3.4. Experiment 4 124
4.3.3.5. Overview of the results 135

5. DYNAMIC BEHAVIOUR NETWORK 137
5.1. TEST CASES FOR THE DYNAMIC BEHAVIOUR NETWORK 139
5.1.1. Distributed Behaviours 140
5.1.2. Distributed behaviours with multiple goals 150
5.1.3. User Interactions 157

6. DISCUSSION 162
6.1. OVERVIEW OF BEHAVIOUR NETWORKS 162
6.2. PROPOSED ARCHITECTURE 163
6.2.1. Purpose of the proposed architecture 164
6.2.2. Benefits of proposed architecture 164
6.2.3. Disadvantages of proposed architecture 165
6.2.4. Overview of the results 166
6.3. BEHAVIOUR NETWORKS 167
6.3.1. Discussion of the division rule 167
6.3.2. Discussion on the data packet approach 169
6.3.3. Overview of the results 172
6.4. DYNAMIC BEHAVIOUR NETWORK 173
6.5. LIMITATIONS AND FUTURE WORK 175
6.5.1. Connection mechanism of internal behaviours 176
6.5.2. Binary nature of behaviours 177
6.5.3. Parameter choosing for behaviour networks 178
6.5.4. Additional metadata on wireless tags 179

7. CONCLUSION 181
7.1. SUMMARY OF THESIS 181
7.2. CONTRIBUTIONS TO RESEARCH 181
7.3. FUTURE WORK 184

REFERENCES 186

List of Figures

Figure 2-1 - Hierarchical decomposition of a mobile robot control system. 22
Figure 2-2 - Shakey the robot. 24
Figure 2-3 - Deliberative Vs Reactive 27
Figure 2-4 - Vertical decomposition of a behaviour-based control system 28
Figure 2-5 - The components of a hybrid architecture 29
Figure 2-6 - The Structure of the Behaviour-Based Control Architecture 31
Figure 2-7 - Different methods for behaviour selection a) competitive b) cooperative .. 34
Figure 2-8 - A representation of a subsumption architecture 35
Figure 2-9 - Subsumption Architecture - Example 1 37
Figure 2-10 - Subsumption Architecture - Example 2 38
Figure 2-11 - Subsumption Architecture - Example 3 39
Figure 2-12 - Subsumption Architecture - Example 4 40
Figure 2-13 - Structure of a behaviour network 50
Figure 2-14 - Components of a behaviour 51
Figure 2-15. First step of the activation spreading function. 58
Figure 2-16 - Demonstrates the unbalanced competition between the nodes due to the division of the number of outputs. 59
Figure 2-17 - Demonstrates the unbalanced competition due to the division of the number of inputs 60
Figure 2-18 - Lack of cooperation in action selection 68
Figure 2-19 - Lack of cooperation in subsumption architecture 68
Figure 2-20 - Cooperation in the Rosenblatt and Payton 69
Figure 2-21 - The neuron of a neural network. 73
Figure 2-22 - The components of a Neural Network 74
Figure 3-1 - Overall structure of the proposed architecture for the dynamic behaviour network. 79
Figure 3-2 - Internal components of the deliberative control layer 80
Figure 3-3 - Data structure of a wireless tag for use in a behaviour network 82
Figure 3-4 - Structure of Environment Node. 83
Figure 3-5 - Initial behaviour network that the robot will start with. 84
Figure 3-6 - Proposed Web GUI to allow users to interact with the behaviour network 86
Figure 4-1 - Scenario where a node can have multiple inputs from different goals. 90
Figure 4-2 - The data that can be stored in each link. 90
Figure 4-3 - An example of a behaviour network displayed automatically from metadata. 90
Figure 4-4 - Pseudocode for embedding goal information to the links of a behaviour network 91
Figure 4-5 - A behaviour network to test the division rule. 93
Figure 4-6 - Results from testing the original division rule and the new division rule for spreading energy in a behaviour network. 94
Figure 4-7 - Results from using an arbitrary order of behaviour nodes. The order was: G1, G2, B1, B2, B3, B4, B8, B5, B6 100
Figure 4-8 - An example behaviour network consisting of two goals and eight behaviours, each connected with either goal or predecessor links. 101
Figure 4-9 - Results from using an arbitrary order of behaviour nodes. The order was: G1, G2, B1, B2, B3, B4, B5, B6, B8 102
Figure 4-10 - Results from using a random order of behaviour nodes. The order was: G1, G2, B1, B2, B3, B4, B6, B8, B5 103
Figure 4-11 - Results from using the second approach. The order was: G1, G2, B1, B2, B3, B4, B5, B8, B6 104
Figure 4-12 - The contents of a behaviour and a packet 106

Figure 4-13 - Results from using the packet approach in experiment 1. 107
Figure 4-14 - An example real world situation shown as a behaviour network consisting of three goals and four behaviours, each connected with different link types. 109
Figure 4-15 - Behaviour Network - Experiment 3 115
Figure 4-16 - Experiment 3 - Results 123
Figure 4-17 - The simulated environment for experiment 4 124
Figure 4-18 - Internal memory map for the agent in the simulated environment 125
Figure 4-19 - Behaviour Network - Experiment 4 130
Figure 4-20 - Experiment 4 - Results 131
Figure 5-1 - Simulated environment with distributed behaviours 140
Figure 5-2 - Behaviour Network – distributed behaviours scenario 143
Figure 5-3 - Results for distributed behaviours – Bake Cake Scenario 144
Figure 5-4 - Distributed behaviours added to behaviour network over time. 146
Figure 5-5 - The robot is trapped between selecting two different paths. 148
Figure 5-6 - Results for distributed behaviours 153
Figure 5-7 - Evolution of the dynamic behaviour network 155
Figure 5-8 - Initial state of the Web GUI 158
Figure 5-9 - Evolution of the Web GUI as the robot finds new goals and when goals are completed. 159
Figure 5-10 - Web GUI - Updating the motivation for the 'Bake Cake' goal. 159
Figure 5-11 - Results from changing the motivation of a goal at run-time. 161

List of Tables

Table 2-1 - The effects of the different variations of the division rule (Tyrrell 1994). .. 60
Table 3-1 - Types of errors that can be exhibited by the architecture. 88
Table 4-1 - Results from using a random order of behaviour nodes. 99
Table 4-2 - Results from using an arbitrary order of behaviour nodes. The order was: G1, G2, B1, B2, B3, B4, B5, B6, B8 102
Table 4-3 - Results from using a random order of behaviour nodes. The order was: G1, G2, B1, B2, B3, B4, B6, B8, B5 103
Table 4-4 - Results from using the second approach. The order was: G1, G2, B1, B2, B3, B4, B5, B8, B6 104
Table 4-5 - Results from using the packet approach in experiment 1. 107
Table 4-6 - Results from using the packet approach with different types of links. 110
Table 4-7 - Results from using the packet approach with different types of links. 111
Table 4-8 - Results from using the packet approach with merging using different types of links. 112
Table 4-9 - Results from using the packet approach with merging using different types of links. 112
Table 4-10 - Precondition list for Experiment 3 117
Table 4-11 - Add list for Experiment 3 117
Table 4-12 - Delete list for Experiment 3 117
Table 4-13 - Experiment 3, Scenario 1 119
Table 4-14 - Experiment 3, Scenario 1 - Results 119
Table 4-15 - Experiment 3, Scenario 2 119
Table 4-16 - Experiment 3, Scenario 2 - Results 119
Table 4-17 - Experiment 3, Scenario 3 120
Table 4-18 - Experiment 3, Scenario 3 - Results 120
Table 4-19 - Experiment 3, Scenario 4 121
Table 4-20 - Experiment 3, Scenario 4 - Results 121
Table 4-21 - Experiment 3, Scenario 5 121
Table 4-22 - Experiment 3, Scenario 5 - Results 121
Table 4-23 - Experiment 3, Scenario 6 122
Table 4-24 - Experiment 3, Scenario 6 - Results 122
Table 4-25 - Value representation of the simulated environment. 124
Table 4-26 - Precondition list for Experiment 4 126
Table 4-27 - Add list for Experiment 4 127
Table 4-28 - Delete List for Experiment 4 128
Table 4-29 - Initial world state for Experiment 4 129
Table 4-30 - World state when the robot is in the proximity of the kitchen (Experiment 4) 133
Table 4-31 - World state when the robot leaves the proximity of the kitchen. 133
Table 5-1 - Value representation of the simulated environment with associated distributed behaviours and goals. 141
Table 5-2 - Precondition List for distributed behaviours scenario 142
Table 5-3 - Add List for distributed behaviours scenario 142
Table 5-4 - Delete List for distributed behaviours scenario 142
Table 5-5 - Initial world state for the distributed behaviours scenario. 142
Table 5-6 - World state when the robot is in the proximity of the kitchen (Distributed Behaviour Scenario). 147
Table 5-7 - World state when the robot leaves the proximity of the kitchen (Distributed Behaviours Scenario). 148
Table 5-8 - Value representation of the simulated environment with associated distributed behaviours and goals. 152

1. Introduction

1.1. General Statement

Robots are growing in number. A great deal of research has been spent on making them more intelligent and capable of completing the everyday tasks a human would normally have to do (Weiser, 1993, 1995; Gates, 2007; Richards and Smart, 2013; IFR, 2016).

Typical approaches for robots to navigate and interact with an environment, involve embedding many different sensors in the robot and the environment, then having the robot process the data from those sensors to interpret the environment. Examples include (iRobot, 2007; Fitzgerald, 2013). It is common for the capabilities of the robot to be hard coded and to have the robot function in structured environments (structured environments are those that are predefined for a given task). The traditional approach (knowledge representation) is widely used and discussed in further detail in section 2.1. This approach can be computationally expensive, due to the vast amounts of information that need to be processed. It becomes even more computationally expensive for a robot to function in an unstructured dynamic environment (an environment that is not predefined and can change unexpectedly) and so alternate solutions are required.

The overall aim of the research reported here is to investigate how embedded wireless tags can simplify the everyday tasks of a robot. One approach to solving this problem is outlined by (Diekmann, Melski and Schumann, 2007; Pais and Symonds, 2011) who discuss a concept that shows that data relating to the location of information can be stored on a wireless device. It shows that data such as a web URL can be stored and point to data relating to either the object the tag is attached to or some information that can be downloaded. This is useful as one of the limitations of wireless tags is the

amount of data that can be stored locally. The limitation of this concept is that the agent will need access to the internet to get the information it needs. This study is based on the concept of storing data on a wireless tag and further expands upon it by finding a solution to store the information directly in a wireless tag. It focuses on what external information a robot would need, to assist its decision-making process, to be able to complete tasks, and how to embed that information into wireless tags in an environment.

In the early 1990s Maes (Maes, 1991c) developed a reactive decision making process in response to the limited processing capabilities that were available to robots during that time period. This decision making process, the behaviour network (Maes, 1989, 1991b, 1991d, 1993), used a selection mechanism to determine the most appropriate action at any given time given the external environment. Tyrrell (Tyrrell, 1994) further extended this work by providing improvements to the action selection mechanism of the behaviour network. Lee and Cho (Lee and Cho, 2014) extended this again by adding an additional planning layer, called a deliberative control layer, to the behaviour network. The behaviour network has not seen a great deal of development since then. This is due to many factors such as: other approaches in AI becoming more established; the previous technical limitations no longer being a factor; and the amount of development time needed to create a behaviour network for each predetermined structured task. This study uses and extends upon all of the previously mentioned concepts and exploits the idea of distributing data into the environment, with the intention of proving that control and task data such as behaviours can be discovered dynamically in an environment and added to an existing behaviour network. This concept offers a possible solution to allow a robot to perform relevant actions in a dynamic environment without the need to hard code every possible behaviour in

advance. Compared to the more established alternatives, this approach would have the benefit of enabling an agent to learn new tasks / goals without the need for any extra processing / computational resources.

1.2. Research Hypothesis

Robotic systems using a variety of sensors and actuators have developed over time to use large complex architectures in order to handle the complexity generated from the combination of environment, sensor data and task or behavioural definition. This is to allow robots to function in dynamic environments under uncertainty and in real time. Knepper et al. (Knepper, Srinivasa and Mason, 2010) suggests that there is no single best architecture to manage the complexity caused by dynamic environments and diverse components. Knepper et al. (Knepper, Srinivasa and Mason, 2010) also state that there is no reliable methodology for determining the best architecture for any given application.

An ideal robot architecture would allow a robot to achieve tasks and to be reactive to a variety of situations. It would grant the robot a diverse set of capabilities including: functioning and reacting in real time; control of actuators and sensors; dealing with uncertainty; and planning and actions (or sequences) to achieve goals. Such an architecture would be decomposable into smaller modules. Each module in the architecture would communicate and send data to other modules in the architecture.

There are different styles of architecture to choose from, including; subsumption architectures, layered robot control, behavioural control and hierarchical control. These different styles of robot architecture are discussed in further detail in Chapter 3.

1.2.1. A gap in the topic domain

This study focuses on the use of behaviour networks in dynamic, unstructured environments and reveals a potential gap in the topic domain. The few extensive articles that have been published relating to behaviour networks all demonstrate the same limitation. That is, the assumption that the behaviours of an agent are predefined and are all that the agent needs to know, either for the environment it is in or for the study in which it is implemented. Each implementation of a behaviour network shows that the agent is typically hand-coded to work in a structured environment. The literature on behaviour networks, reviewed in section 2.2.4, has concentrated principally on finding the best action selection mechanism for these predefined networks (Maes, 1991b, 1991c; Tyrrell, 1994), though it has been expanded to include planning (Lee and Cho, 2014). The concept of allowing behaviour networks to change or to work in unstructured environments has not been given much attention and will form a fundamental part of this study. There are also many different options for action selection in behaviour networks and this study will examine these mechanisms and expand upon some of the better choices.

1.2.2. Hypothesis

The hypothesis is that embedding wireless tags on to objects in the environment and including object specific data, such as; behaviours and goals, will enhance the task planning of a mobile robot, allowing the robot to successfully complete goals in dynamic and both unstructured and structured environments. This study will test this hypothesis.

1.2.3. The importance of the research

Behaviour networks have been used to some success. Jiang et al. (Jiang *et al.*, 2012; Yang and Cho, 2013) both developed simulations involving a behaviour network and Paikan et al. (Paikan, Metta and Natale, 2013; Lee and Cho, 2014) both created real world examples. Expanding upon this field adds to the value of this research and may spark more interest and further research. If this research can demonstrate the benefits of using embedded devices to assist robots, then that could lead to further research and applying this technique in more examples and environments.

1.3. Research Objectives

1.3.1. Aims

The objective of this research is to explore the concept of embedding data into the environment and show that, by doing so, the capabilities of a robot will be improved. To achieve this outcome this study will focus on embedding information about behaviours into the environment to provide a robot with the ability to enhance and evolve their action selection mechanism.

Task planning in robotics is a computationally expensive task every robot will need a solution for; this is due to the agent needing to have a workable knowledge representation of the environment it is in. Typical approaches involve hard coding the tasks the robot needs to complete in order to reduce the computational complexity of the task (Bolmsjo, Neveryd and Eftring, 1995). While this is accepted in some fields it does not help in unstructured environments where new objects can be added and new behaviours required. Behaviour networks are designed to work in these unstructured environments by continuously selecting the best action to perform at a given time rather than rigidly following a predefined plan. Previous work in this field again consists of hard coding the behaviours that an agent can perform in a given environment (Paikan, Metta and Natale, 2013; Lee and Cho, 2014).

This study begins by developing the technical components and infrastructure for a behaviour network, based on the work of (Maes, 1991b, 1991c; Tyrrell, 1994). This work is expanded to receive new behaviours dynamically (for example, from wireless tags in an environment) and add those behaviours to the agent's existing behaviour network. This is achieved by adding data such as, goals and behaviours to wireless devices and embedding those into objects in the environment. When the robot is in the

environment it will be able to scan for nearby wireless tags, read the data from those tags and use that data in its decision-making process.

The technical objectives that this project aims to achieve are as follows:

- Implement and test behaviour networks.
- Develop and test improved mechanisms for spreading activation energy in a behaviour network.
- Design an architecture to support dynamic behaviour networks.
- Implement a dynamic behaviour network in a simulated environment.
- Test distributed behaviours / goals in a simulated environment.

1.3.2. Limitations

It is the purpose of this study to show how embedded devices in an environment can expand the capabilities of a robot. This study will cover existing techniques and how those techniques can be used with embedded wireless tags. Wireless tags have been shown to have limited storage (Pais and Symonds, 2011) which restricts the amount of data that can be used. The communication range of wireless tags is also limited and can vary in distance. The behaviour networks that will be used in this study also have documented limitations (Tyrrell, 1994). Part of this study will be to research these limitations and develop methods to overcome them. These limitations include:

- Predefined nature of behaviours in a behaviour network
- Problems documented (Tyrrell, 1994) regarding the passing of activation energy in the network

1.3.3. Method

The main topic area of this study uses and extends the concepts found in (Maes, 1991b, 1991c; Tyrrell, 1994). The study uses a behaviour network for the action selection mechanism, which allows behaviours to be embedded into the environment; this is covered in detail in section 2.2.4. Many techniques and methods have been implemented to develop a workable system, which can be used to test the hypothesis. This includes; developing a system architecture to support distributed behaviours and goals, improving the base behaviour network with alternative energy spreading mechanisms and testing the behaviour network with a variety of scenarios in a simulated environment.

1.4. Thesis Structure

The content and structure of the remaining chapters of the thesis are as follows:

2. Background Information

   The background chapter provides a review of the relevant literature to give context to the rest of the thesis. It is divided into four sections. First, a discussion of traditional AI methods leading into behaviour-based AI. Second, action selection mechanisms and behaviour networks are explained and justification for their use given. Third, other mechanisms for action selection are examined. Finally, alternative approaches to AI are discussed.

3. Proposed Architecture

   The proposed software architecture is discussed in this chapter. It details how the architecture can work with the behaviour networks, how the architecture enables the processing of external information and how all this information is

combined to allow the system to make decisions. The benefits and the limitations of the system are then examined.

4. Behaviour Network

   This chapter extends what has been discussed in the background chapter. The flaws that have been found during the implementation of the behaviour network are discussed and the solutions presented.

5. Dynamic Behaviour Network

   This chapter covers the novelty of this research: it begins with an in-depth explanation of what is trying to be achieved with behaviour networks and how the embedded wireless tags will improve the behaviour network. The results of a dynamic behaviour network using embedded wireless tags are then presented together with the results from a series of tests.

6. Discussion

   This chapter begins with a discussion of the problems found with the standard behaviour network, the proposed solutions to those problems and concludes with the dynamic behaviour network and its solutions. The successes and failures of the previous experiments are then critically analysed and compared with current methodologies.

7. Conclusion

   This chapter summarises the points that have been raised in this thesis and includes suggestions for future work or potential extensions to the work submitted.

1.5. Contribution

The work presented in this thesis extends the field of behaviour networks. This thesis addresses problems found with the traditional behaviour network, proposing solutions that allow for more appropriate action selection. The work shows that for the first time, a behaviour network can evolve in a dynamic environment and argues that behaviour networks are not an outdated concept. This is demonstrated with a series of tests demonstrating that a dynamic behaviour network can function in an unstructured environment and that the use of wireless tags can enhance this process.

1.6. Publications

The work presented in this thesis has also been published (in a condensed form) in a conference. Peach and Robinson (Peach and Robinson, 2016) describes how the use of data packets can be used as a replacement for the existing activation spreading process, in a behaviour network, to allow for more accurate selection of behaviours.

2. Background

This chapter begins with an overview of the subject of Behaviour-Based Architectures. It starts with a discussion on the history of traditional AI which then leads to the emergence of behaviour-based architectures. The discussion on behaviour-based architectures is followed by an overview of more modern approaches such as hybrid architectures. This chapter concludes with a thorough analysis of behaviour networks (Maes, 1991b, 1991c; Tyrrell, 1994), detailing all of the components and their benefits and flaws.

2.1. Traditional AI

Autonomous robotics often use a variety of Artificial Intelligence (techniques used to allow computer systems to perform tasks that require human intelligence) techniques. Artificial Intelligence approaches are typically classified as 'Traditional AI' and adopts the top-down (symbolic) approach. This approach subdivides the problem into a series of sub-problems that are easier to solve. This approach is also knowledge-based, as it requires a symbolic representation of the world that it can use to verify the sensor input and to generate actions to perform.

The architecture of Traditional AI is often broken down into a sequence of components (Brooks, 1986) and used in robot control systems, as shown in Figure 2-1.

Figure 2-1 - Hierarchical decomposition of a mobile robot control system. (Brooks 1986; Carreras 2004) detailed the components of a mobile robot, breaking it down into functional modules.

The components shown in Figure 2-1 are broken down into five functional modules;

1. Perception

   This module takes in the sensor data as input and is tasked with reducing the noise and any conflicts caused by other sensors (e.g, two cameras could create conflicting data). Computer vision techniques may be applied to find objects and other useful information (such as obstacles) in the environment.

2. Modelling

   This module typically uses data from the perception module to build a symbolic representation of the world internal to the robot (Brooks, 1986, 1991). This world model can include all of the objects the robot has seen, including their positions, as well as where the robot believes it is in the world.

3. Planning

   The planning model will usually use the symbolic representation of the world created in the previous module to produce a sequence of tasks that can be followed to complete a given goal. Many different planning techniques can be followed in this module, such as searching the state space (STRIPS) (Fikes and Nilsson, 1971), to find the best sequence of tasks to solve the goal and Hierarchical Task Network (HTN) (Erol, Hendler and Nau, 1994) which uses abstract operators to start a plan and an task decomposition to convert the plan into primitive operators.

4. Task Execution

   This module is in charge of representing each of the tasks (given from the planning module) into smaller operations that can be used to control the actuators. This module typically oversees the whole of the task execution.

5. Motor Control

This module is in charge of controlling the actuators of the robot allowing it to interact with the physical environment.

One of the first applications of the traditional AI architecture into a robot was the deliberative control architecture, an example of a robot using this was Shakey the robot (Fikes and Nilsson, 1971). Here the robot takes in all of its sensor information and its internal knowledge to create a plan. Shakey (Figure 2-2) was able to reason about its own actions by breaking down commands into more basic commands that it could execute. Some of the more notable results from the development of Shakey included the A* search algorithm (Hart, Nilsson and Raphael, 1968) and the Hough Transform (Hough, 1962; Ballard, 1981). The experiments for Shakey involved it navigating in specially prepared rooms in order to complete given goals. The main flaw of this experiment was that the rooms had to be structured and designed to meet Shakey's perceptual constraints otherwise it would struggle with the planning process and accurately representing the world. This was due to a combination of the low processing power and poor sensor hardware which was available at that time.

Figure 2-2 - Shakey the robot.
Shakey the robot (Fikes & Nilsson 1971) was one of the first autonomous robots that used a deliberative control architecture.

There have been many implementations using the traditional approach. (Swere and Mulvaney (Swere and Mulvaney, 2003) used decision trees (a model of decisions and their possible consequences) as a form of navigation for a mobile robot and more

recently Leiva and Barragán (Leiva and Barragán, 2011) used decision trees to control the behaviours of simulated bots. Collins and Ruina (Collins and Ruina, 2005) used a finite-state machine for the control architecture of a bipedal walking robot.

Due to the limitations defined previously, alternative approaches were developed, notably the subsumption architecture in the 1980s (Brooks, 1986), hierarchical networks (Rosenblatt and Payton, 1989) and action selection mechanisms in the 1990s (Maes, 1991b, 1991c; Tyrrell, 1994).

The main problem with 'traditional AI' is that it has difficulties working in complex and dynamic environments. Depending on the mobile robot, this can be because of the limitations in the processing power and that the planning techniques are not robust enough for dynamic environments. The previous examples of traditional AI have shown some positive results (the robots achieving their respective goals), however they are all tested in structured, predictable environments. With the traditional AI approach; the mobile robots need to create a symbolic model of the world, which can later be broken down into a set of rules that the robot can use to solve goals. This can lead to a combinatorial explosion of the number of rules depending on the complexity of the environment. This traditionally meant that the mobile robot would require a large amount of storage and computational power, which may be limited on a mobile robot.

Another problem that is found in traditional AI approaches is the difficulty in processing and executing tasks in real time because the robot will need to process its sensor data, create a symbolic world model, break the model into a sequence of tasks and finally execute those tasks quickly. The robot can find that by the time it has made a decision on what to do, the environment has changed and it will need to process the new data instead. Creating a symbolic representation of the world is also a challenge for traditional AI because the robot will need to be able to maintain a model of a

continuously-changing world. The Frame Problem (Hayes, 1971) is a situation where an agent can have difficulty describing the environment and the effects of its actions. Another related problem, called the 'symbol-grounding problem' occurs when the robot has difficulty linking symbols (such as 'door' or 'cup') to real-world perceptions (Harnad, 1990). Vinyals et al. (Vinyals *et al.*, 2015) attempted to address this problem by using a convolutional neural network, making it possible to accurately describe and caption an image.

2.2. Behaviour-Based AI

Because of the problems with traditional AI, in the 1980s a number of researchers began to look for a solution that was more reactive to environment changes. The idea was to break away from the concept of creating a symbolic model of the world and to instead rely on input from sensors, to be interpreted as the world model for each moment in time. One of the more influential researchers was (Brooks, 1986), who developed the subsumption architecture. This then led to the new fields of behaviour-based architectures (Maes 1991a) and behaviour-based robotics (Arkin, 1998). Figure 2-3 shows a comparison of deliberative control architecture (traditional AI) with a more reactive control architecture (behaviour-based architecture).

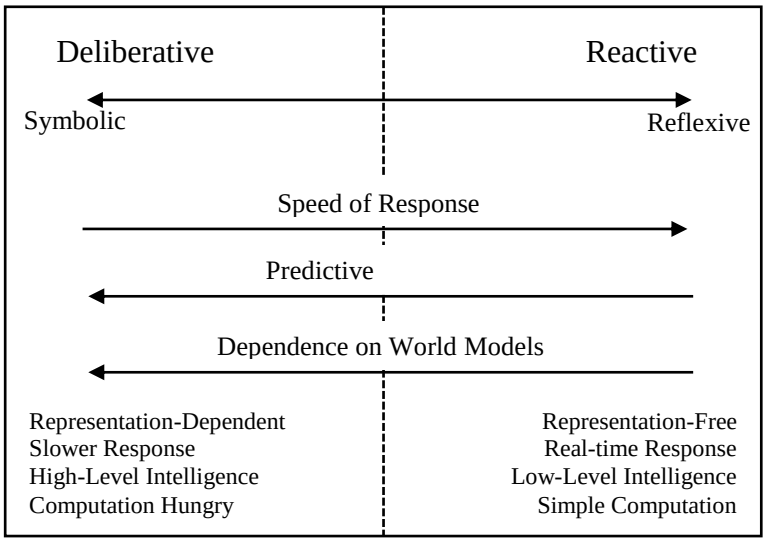

Figure 2-3 - Deliberative Vs Reactive

Cifuentes Costa (Cifuentes Costa 2013) detailed a comparison between deliberative and reactive systems.

The new behaviour-based architectures used a bottom-up approach (Maes, 1991b, 1991c; Tyrrell, 1994) as opposed to the Traditional AI using a top-down approach. The bottom-up approach defines the low-level elements of the system in detail. These elements are then linked together to form a larger subsystem. Subsystems are then linked, forming layers, until a complete top-level system is created. Using a set of rules, a reactive system (behaviour-based architecture) is able to provide real-time responses to changes in the environment. For example, if an obstacle were to present itself to a robot, a reactive architecture would be able to move around the obstacle without creating a new model of the world to plan around the obstacle. This approach does not use any form of deliberation or internal interpretations for planning and it is typically unable to learn any new behaviours as the behaviours are typically hand-coded (Maes, 1991b; Tyrrell, 1994; Cifuentes Costa, 2013). A behaviour-based architecture is able to store high-level interpretations of tasks (behaviours), e.g, 'explore' or 'avoid obstacles', and this can give the appearance of high-level deliberation as an emergent property.

The behaviours in a behaviour-based architecture work in parallel, reacting to changes in the perceived environment and providing the robot with an appropriate

response, which also helps to achieve some set goals. Figure 2-4 shows the structure of the behaviour-based architecture. The previous problems with traditional AI (accurate real-time world modelling and real-time processing for real-time task planning) are no longer an issue in behaviour-based architecture, as it is reactive rather than deliberative, meaning the agent only needs to decide on a single behaviour to execute instead of a detailed plan. However, a new challenge of how to select the most appropriate behaviour at any particular moment to achieve a set goal will now need to be addressed.

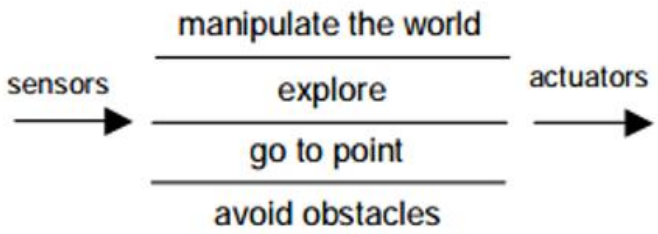

Figure 2-4 - Vertical decomposition of a behaviour-based control system (Brooks 1986; Carreras 2004).

This relatively new field (at the time) has since been widely researched and has seen a variety of different implementations. Seraji and Howard (Seraji and Howard, 2002) used a behaviour-based architecture combined with fuzzy logic to measure terrain traversability and use this information in real-time to update a robot's navigation strategy. Carreras (Carreras, 2004) used a behaviour-based architecture combined with reinforcement learning to allow an autonomous underwater robot to complete simple tasks.

As both traditional AI and behaviour-based architectures have merit, some researchers have combined these approaches to create hybrid architectures (Arkin, 1987; Nicolescu and Matarić, 2002; Lee and Cho, 2014). These architectures aim to find the compromise between the classical top-down approach and the reactive bottom-up approach. Figure 2-5 shows the components of a typical hybrid architecture.

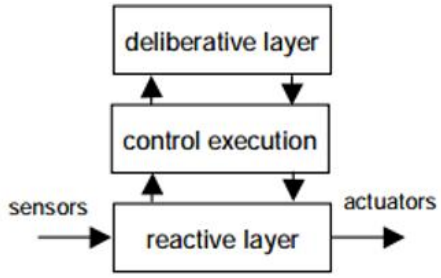

Figure 2-5 - The components of a hybrid architecture

(Carreras Pérez 2003) defines a three layered hybrid architecture that combines a deliberative layer architecture with a reactive architecture.

Typical approaches to the hybrid architecture (Figure 2-5) split the architecture into three layers:

1. Deliberative layer

   This is the planning layer and relates closely to traditional AI. Here the goals are broken down into a sequence of behaviours, which forms a plan to solve a given goal. This plan can be interrupted by the lower layers of the architecture and runs in parallel to the other modules.

2. Control layer

   This is the 'middle man' layer. It is used to supervise the completion of actions / tasks and to communicate between the reactive lower layer and the deliberative higher layer (Cifuentes Costa, 2013).

3. Reactive layer

   This layer deals with the real-time problems that are presented by the environment. If there are no problems, then the robot is able to execute the high-level plan of the deliberative layer and, should a problem arise (such as an obstacle), the reactive layer will take over and react to the given problem.

The hybrid architecture is able to take advantage of both the hierarchical planning and the reactive real-time aspects of the behaviour-based architectures. Hybrid architectures are discussed in more detail in section 2.2.4.6.

2.2.1. Behaviour-Based Architecture

A behaviour is typically characterized as an entity of an agent that encapsulates a process. A process can be anything, ranging from moving to a specific position to completing a given task. A behaviour can also store something that can be used to execute the process that it represents. Maes (Maes, 1991b; Tyrrell, 1994) uses executable code, for this component of a behaviour. This component can range from executable code to a hierarchy of behaviour networks (Nicolescu and Matarić, 2002; Lee and Cho, 2014).

Behaviour-based architecture was developed from traditional AI concepts in order to create autonomous robots that can achieve goals in dynamic environments. All behaviours in the architecture take inputs from the robot's sensors, and send their output to the coordinator for evaluation. Carreras (Carreras, 2004) explains the purpose of the coordinator in a behaviour-based control architecture, which is that it is used to send only one action to each actuator at a given time (Figure 2-6). Section 2.2.1.2 details the two approaches that the coordinator can use to select which action to send to the actuators; the coordination method or competitive method.

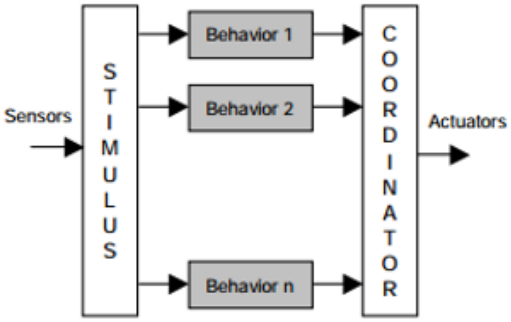

Figure 2-6 - The Structure of the Behaviour-Based Control Architecture

(Carreras Pérez 2003) details the benefits of a behaviour-based control architecture

Brooks (Brooks, 1986) suggests that each behaviour in a behaviour-based architecture must be independent the others as typically only one behaviour can be selected to execute at any given time, however the internal structure of a behaviour can be open to interpretation (e.g, the internal structure of a behaviour can include any combination of behaviours (different levels of abstraction), sensors, internal data and a coordinator). The resulting structure is a network of interconnected behaviours. One of the key benefits of the behaviour-based architecture is that it allows for real-time responses with low computational cost (Carreras Pérez 2003).

2.2.1.1 Basics of the Behaviour-Based Architecture

Behaviour-Based Architectures typically follow a few basic design principles (Brooks, 1986; Carreras, 2004) that are key to creating successful implementations:

1. Parallelism

   It is important that a behaviour-based architecture is able to quickly process the data about the environment and execute the most appropriate behaviour. This is achieved via parallelism and often is implemented by running each behaviour on a separate thread. This concept allows the behaviours to either work together and coordinate or compete with one another for selection (Maes, 1991b; Tyrrell, 1994).

2. Modularity

In every implementation of autonomous robots, one of the key tests is robustness, the ability for the system to cope with errors and still be able to function to a high degree of accuracy (Singh and Kelly, 1996; Simmons *et al.*, 1997). A behaviour-based architecture is able to achieve a high level of robustness from the concept of modularity. Each behaviour in the behaviour-based architecture is separate from the other behaviours and the behaviours are executed individually. If a behaviour fails in the system (due to hardware or software issues) then other behaviours can still be executed and the robot can retain some form of functionality. Modularity also allows the network of behaviours to grow dynamically without affecting or altering any of the existing behaviours. This concept is one of the key motivations in this thesis.

3. Situatedness / Embeddedness

The term 'situatedness' refers to the robot being physically situated in the world. This means that the robot uses real sensor data to inform its decision on what to do next and to perform the associated action (Maes and Brooks, 1990). The term 'embeddedness' refers to, again, the idea of the robot being in the real-world but noting that the robot is subjected to any of the influences of the environment, such as potential damage to the robot. Asimov (Asimov, 1963; Clarke, 1993, 1994) refer to this concept as the third law of robotics, which is used to keep the robot safe from harm. This was later researched (Murphy and Woods, 2009) and alternative laws discussed, including robot safety.

4. Emergence

   A behaviour-based architecture appears to exhibit high level deliberative planning as an emergent property of the interaction between behaviours. This is demonstrated in (Maes, 1991b; Tyrrell, 1994) who gave an agent simple behaviours in a situated environment and it is shown that the sequence of selected behaviours can appear like a plan. The behaviour-based architecture does not have any planning modules (unless a hybrid architecture is used); however, the sequence of executed behaviours can give the appearance of a carefully-formulated plan. Meeden et al. (Meeden, McGraw and Blank, 1993) had a similar experience with their autonomous robot that exhibited plan-like behaviours.

2.2.1.2 Competition Vs Cooperation

While behaviours are modular and can be independent, is it also possible for behaviours to work together. Some implementations of behaviours have running processes, such as (Brooks, 1986) where the output (commands) of each behaviour is subsumed by higher level behaviours in the network. Some implementations of behaviours hold a set of instructions or even executable code that is run when the behaviour is selected for execution (Maes, 1991b; Tyrrell, 1994). For example, a robot with a robotic arm could be issued with a task 'Pick up Object', with the object being far away requiring the robot to navigate to it. The behaviours 'Pick up' and 'Move to point' can act individually, competing with one another for selection, or they can coordinate together to both solve the goal. It is important to remember the role of the 'Coordinator' in the behaviour-based architecture. Figure 2-7 shows a representation of the different methodologies (Competition and Cooperation) and how the coordinator is able to select an action to be executed.

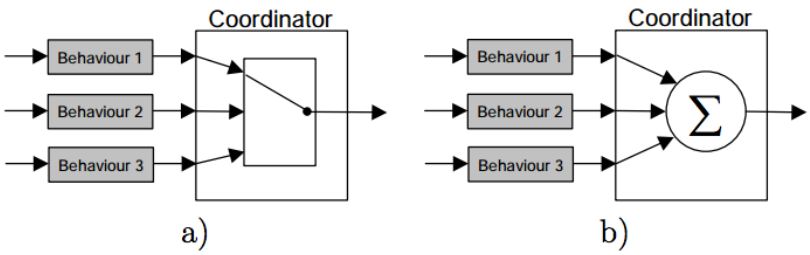

Figure 2-7 - Different methods for behaviour selection a) competitive b) cooperative

(Carreras Pérez 2003) details two different methods for selecting behaviours. Behaviours either compete with one another for selection or cooperate with one other to find a middle ground.

1. Cooperative

   The coordinator takes input from a range of behaviours and a combination function is applied. Vector summation or context-dependent behaviour blending are often used (Saffiotti, Konolige and Ruspini, 1995). Behaviour blending uses behaviour schemas to describe the behaviours of a robot (expressed as trajectories of control actions in an environment). This allows for the combination of behaviours and goals. The level of abstraction of the behaviour can control which combination function to use; if some behaviours such as 'move to point a' and 'move to point b' are selected then a vector summation can be applied. Here an agent would travel to a midway point until it was more certain in which action to execute. However, if the behaviours are 'go home' and 'explore' then behaviour blending will need to be applied allowing the agent to perform an action that achieves the outcome of both behaviours.

2. Competitive

   The coordinator of the competitive methods always outputs a single behaviour for activation. There are many different methods that can be applied for the coordinator to determine which behaviour should be selected for activation. Some popular examples of these methods include subsumption architecture

(Brooks, 1986), action selection mechanisms (Maes, 1991b, 1991c; Tyrrell, 1994) and voting-based coordination (Rosenblatt and Payton, 1989; Tyrrell, 1993).

2.2.2 Subsumption Architecture

Subsumption architecture is another behaviour-based architecture and was introduced in the 1980s and further expanded upon in the 1990s (Brooks, 1986). The subsumption architecture splits behaviours into a hierarchy of sub-behaviours with each level able to perform actions to a set competence. The lower layers of the subsumption architecture can relate to atomic actions such as 'move forward' or 'turn left'. The higher layers of the subsumption architecture can relate to more abstract behaviours such as; 'explore' or 'move to location' with the higher levels able to utilize the lower levels when the sensor data requires it. Any layer of the subsumption architecture can send commands to the actuators of a system, but may be prevented by another layer. Figure 2-8 shows a representation of a subsumption architecture.

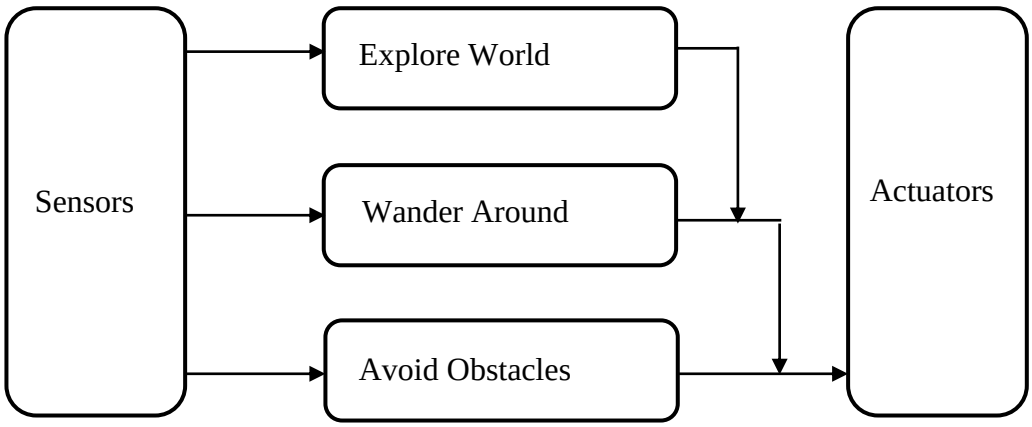

Figure 2-8 - A representation of a subsumption architecture

The diagram shows the input of sensors to a subsumption architecture, with high level behaviours being subsumed by the lower level behaviours.

The aim of this architecture was to address a problem with AI during the 1980s, which was that it was difficult to model the world and to be able to react to the environment in real time. This architecture achieves this when the behaviours receive input from the sensors and send output to the actuators. This method is performed

asynchronously resulting in many input signals being discarded, this happens when a new input is sent and there are still old input signals to process. By discarding the old input signals, it allows the system to work in real-time using the most recent information. Brooks (Brooks, 1986; Maes and Brooks, 1990) demonstrated this architecture on a robot named Allen; this robot used three layers of control with the lowest layer being able to avoid obstacles and the highest layer to explore the environment. Allen was the first robot to implement the subsumption-based architecture; it used a sonar for distance, odometry to calculate its position and a Lisp machine to simulate the subsumption architecture.

There are many benefits to the subsumption-based architecture, some of these benefits include;

- Abstraction

  The subsumption-based architecture offers different layers of abstraction of its behaviours where each layer of the architecture allows the robot to operate at different levels of competence, which also require different levels of resource. The higher levels of competence are applied only when they are needed allowing the agent to run more efficiently.

- Simplicity

  Compared to many other behaviour-based architectures the subsumption-based architecture has the simplest architecture.

- Extendibility

  The subsumption-based architecture can be extended with additional layers of abstract behaviours.

- Time

  As the subsumption-based architecture is the simplest of the behaviour-based architectures, it is also the quickest to implement. This is ideal for researchers that need to test out their theories.

In an example scenario where a robot (implementing the subsumption architecture) is given a goal to explore and map an environment, the subsumption architecture could look similar to that in Figure 2-9. As the goal is to explore the environment, we do not want the robot to be navigating in already seen terrain. At the lowest layer of the subsumption, architecture is an atomic behaviour 'Move Forward'. At this bottom layer the robot will simply move forward, until it is subsumed by a higher level behaviour. At the next level is the 'Avoid Obstacle' behaviour and at the top of the architecture is an 'Explore' behaviour, which will point the robot in the direction of an unexplored area.

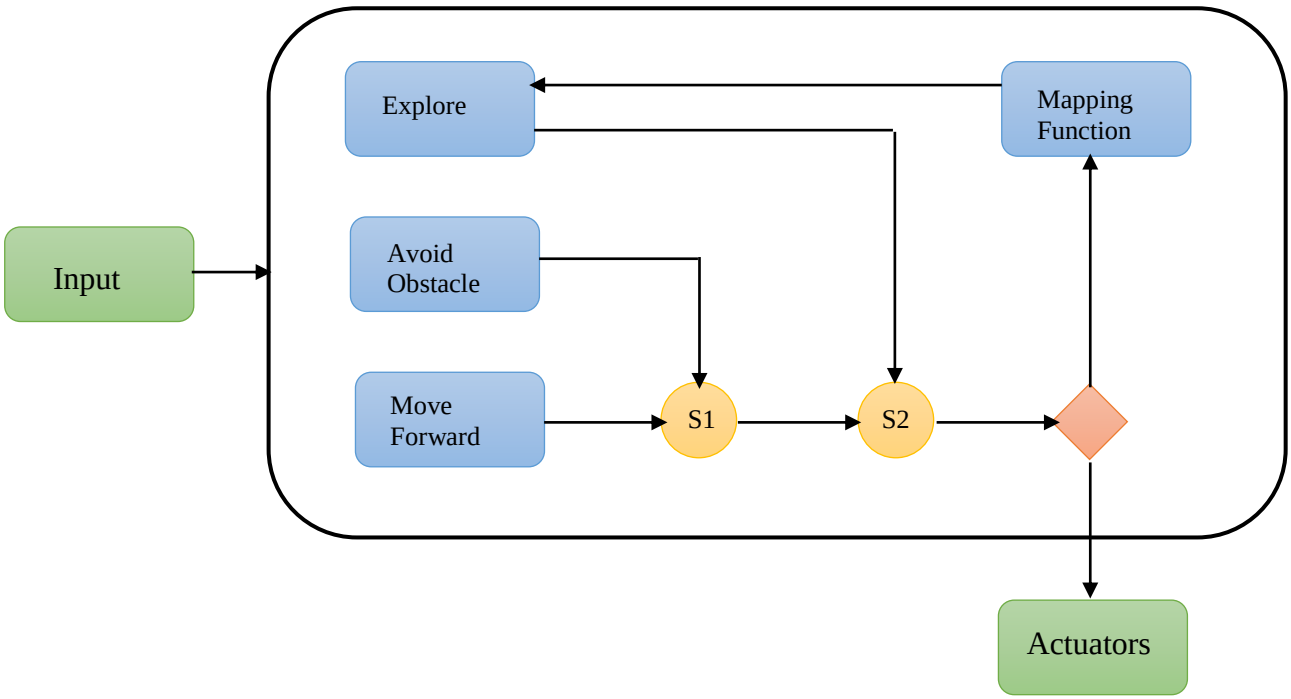

Figure 2-9 - Subsumption Architecture - Example 1

This simple example shows a subsumption architecture to assist with a robot's navigation. Here the move forward behaviour gets subsumed by the avoid obstacle behaviour and both of these are subsumed by an explore behaviour.

In this model of the subsumption architecture there is input to the system, which is sent to each of the three behaviours. These behaviours are isolated so that the other behaviours cannot interfere with the inner workings of another behaviour. Each process (behaviour, suppressor or splitter) is connected via links, which pass data from one process to the next. Figure 2-10 shows an example of the data flow when the 'Move Forward' behaviour is activated.

The 'Move forward' behaviour sends its commands to a suppressor process (S1), Figure 2-10. This is then sent to the other suppressor process (S2) and from there the data is sent to a splitter (sends the data to multiple processes). The actuators receive the data and are able to perform a 'Move Forward' action. The data is also sent to a mapping function which, determines where the robot is located and whether or not it is in an already explored area.

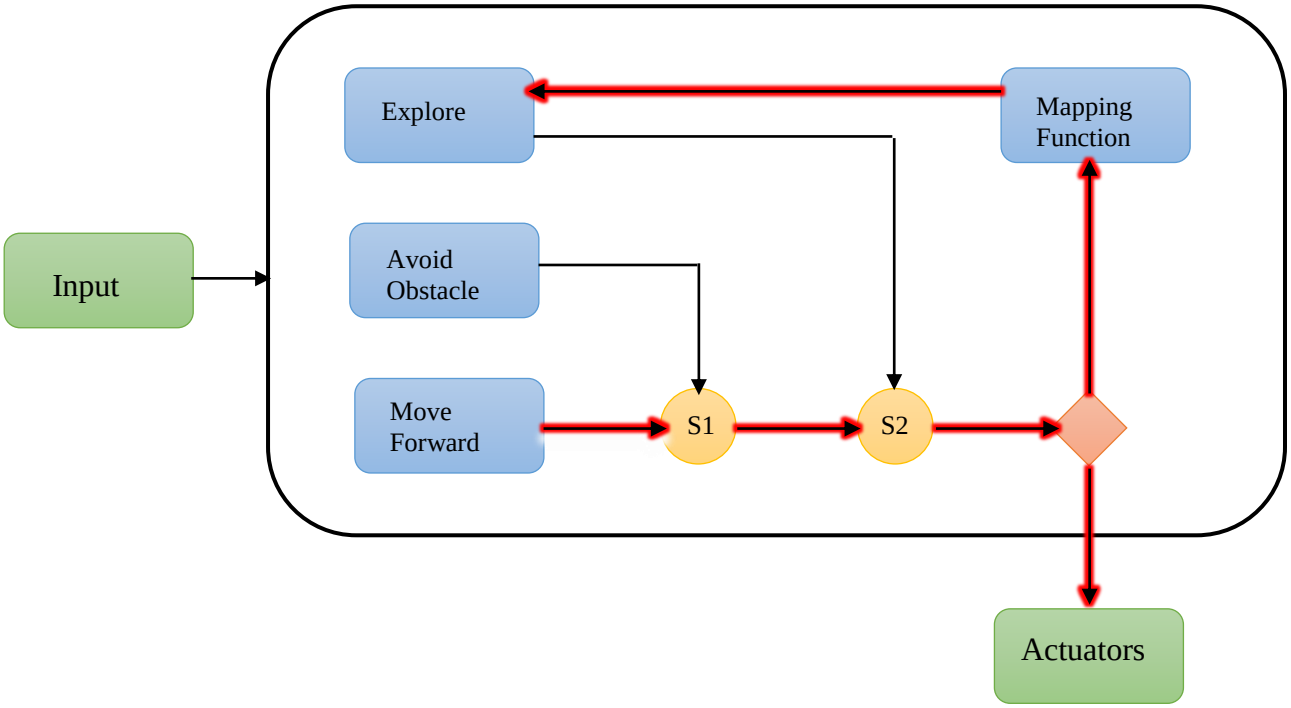

Figure 2-10 - Subsumption Architecture - Example 2

This example of the subsumption architecture shows the data flow when the 'Move Forward' behaviour is selected.

Figure 2-11 shows the path the data flows to the actuators when the robot detects an object and the 'Avoid Obstacle' behaviour is activated. The 'Move Forward' behaviour has not been stopped, all of the behaviours in the subsumption architecture run concurrently. When an object is detected in front of the robot, both the 'Avoid Obstacle' behaviour and the suppressor (S1) are activated. The suppressor suppresses the flow of data from an input when it is activated. The 'Avoid Obstacle' behaviour will send signals to the actuators to make the robot pivot and face a new direction, then allowing the 'Move Forward' behaviour to continue. The combination of these two behaviours from the different levels of the subsumption architecture as well as the concept of suppression, allow the robot to dynamically avoid obstacles in real-time.

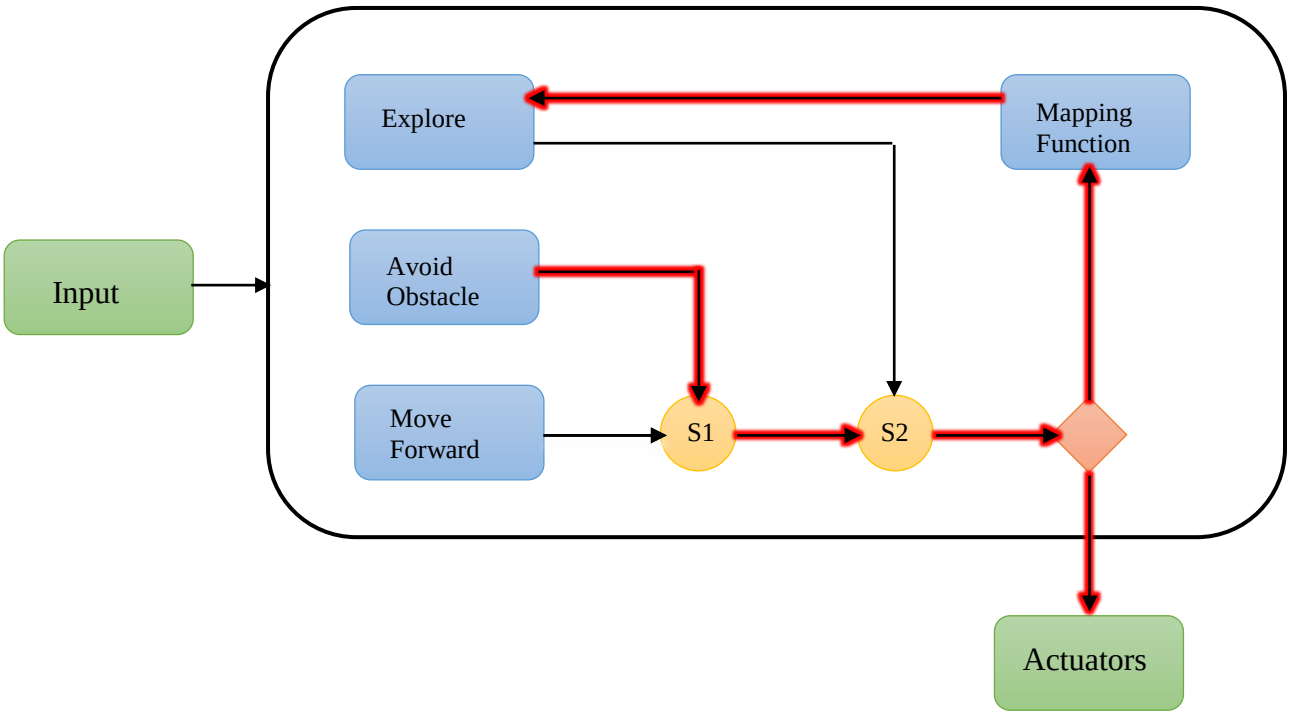

Figure 2-11 - Subsumption Architecture - Example 3

This example of the subsumption architecture shows the data flow when the 'Move Forward' behaviour is subsumed by the 'Avoid Obstacle' behaviour.

The mapping function is used to calculate the robot's relative position by creating an in-memory map and feeding this data to the 'Explore' behaviour. When the robot is navigating through an already explored area the 'Explore' behaviour will use the data from the mapping function to pivot the robot in a new direction, towards an unexplored area. When the 'Explore' behaviour is activated, the suppressor (S2) is also activated, suppressing the 'Move Forward' behaviour and the 'Avoid Obstacle' behaviour. This is demonstrated in Figure 2-12.

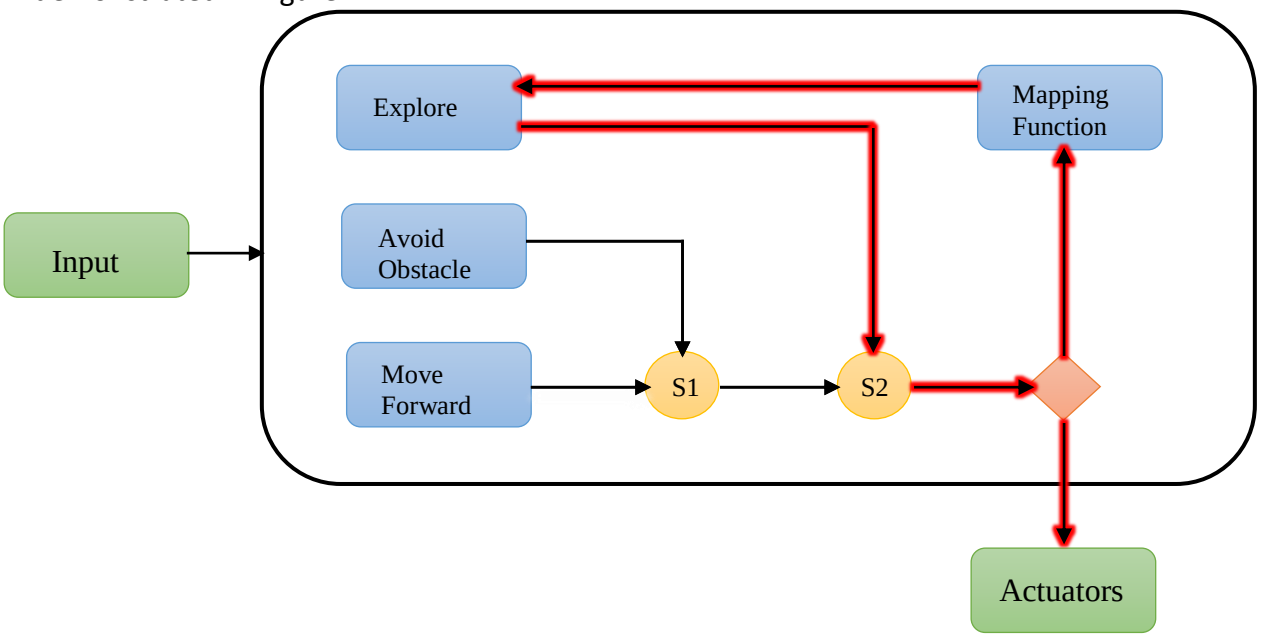

Figure 2-12 - Subsumption Architecture - Example 4

This example of the subsumption architecture shows the data flow when the 'Explore' behaviour is subsumes the other behaviour.

There are many benefits to the subsumption architecture, the first being that it is a modular architecture allowing new levels to be added to the system to give the robot greater functionality. The lower levels of the system would remain intact, providing the same capabilities only these can now be subsumed by the new higher levels. Another benefit is that it eliminates one of the problems found in traditional AI, that is the bottleneck caused when sensor data is received but there is still sensor data to be processed. Instead, in this architecture the behaviours only receive the sensor data that is relevant to their needs. As each layer in the architecture only needs its relevant

information, then there is no need for a representation model of the world to be created and this allows the robot to respond in real-time to changes in the environment.

Brooks (Brooks, 1986) explained that one of the requirements for a robot control system is that it should be capable of achieving multiple goals. In the subsumption architecture this can rarely be achieved as only one behaviour can be executed at one time and there is no ability to compromise. This is one of the limitations of the subsumption-based architecture: each behaviour is fighting for control of the robot's actuators and there can be some behaviours that are in direct conflict with one another. The subsumption architecture does have a mechanism for conflict resolution (the concept of suppression), in which the conflicting behaviours lower in the architecture are subsumed by the higher-level conflicting behaviour. Another limitation of the subsumption architecture is the concept of modularity, as the internal workings of each behaviour are kept separate from other behaviours in different layers it can become difficult to add additional layers to the system. For example, it is much more difficult to add layers between the different levels of the subsumption architecture, as you may not want some behaviours to subsume the new behaviours. It is also difficult to predict the needs of future layers and this can result in changes needing to be made to the lower level behaviours in order to accommodate and provide for the new behaviours. Finally, one of the subsumption architecture's strengths is also one if its greatest limitations and that is the concept of subsumption. For example; one layer in the system could have a behaviour 'Avoid Obstacle' to prevent the robot from colliding with obstacles however, if that layer is being subsumed by a higher layer then it is possible for the robot to collide with obstacles.

While the concept of the subsumption architecture has been around for many years, it is still a topic of research. Recent work involving subsumption architectures includes (Oland, Andersen and Kristiansen, 2016) who used the concept of the subsumption architecture to break down the task of flying a UAV (unmanned aerial vehicle) . The task was broken down into a hierarchy of behaviours; ground avoidance, obstacle avoidance and waypoint tracking with the more abstract levels subsuming the more detailed lower levels, showing that the subsumption architecture is not limited to a single field. Nagata et al. (Nagata, Otsuka and Watanabe, 2012) proposed a network-based subsumption architecture for multiple mobile robots. This is implemented in a server supervisory control system, which is shown to enable swarm intelligence in the mobile robots. Turner et al. (Turner, Givigi and Beaulieu, 2013) implemented a subsumption-based architecture using model-driven development and implemented the behaviours of the robot in a finite state machine. This shows that the subsumption-based architecture is still relevant and used in the field of robotics. It also shows the benefits of a reactive architecture for dynamic environments.

The subumption architecture has been widely used and heavily researched since it was originally developed in the 1980s. Some recent examples include Halal and Zaremba (Halal and Zaremba, 2018) who used subsumption as part of their architecture for allowing a mobile robot to navigate an unstructured environment, (Kragic *et al.*, 2018) who also based part of their architecture on subsumption to allow a robot to interact with Lego objects and *Gudwin et al.* (Gudwin *et al.*, 2018) who used the subsumption architecture as part of a traffic control manager in a traffic simulator.

2.2.3 Action Selection

Behaviour-based architecture has demonstrated that it is capable of dealing with dynamic, changing environments (Weser, Off and Zhang, 2010). An agent needs to be able to decide which action it needs to perform from its selection of possible actions. Action Selection is the term given to the solution to this problem. The function of an action selection mechanism is to select, at each moment in time, an action for the agent, which achieves or helps to achieve its global goals. The key to action selection is using the correct level of abstraction for the actions. e.g. 'turn left 90º' is a low-level atomic action whereas, 'explore' is a high-level action. These actions in an action selection mechanism are often predefined and fixed.

The problem of selecting the most appropriate action in a behaviour-based architecture has attracted a great deal of research into action selection mechanisms. Some of the more researched approaches to action selection are listed below and explained later in this chapter;

1. Symbolic Approach
2. Distributed Approach
3. Dynamic Planning Approach

Action selection mechanisms follow some basic key principles;

1. Dynamic Environments – The robot will need to be able to select an action to perform in dynamic and unpredictable environments where it may include humans or other robots that can either assist or hinder the robot.
2. Real-Time – The robot is expected to operate in as close to real-time as possible. It should be able to select an appropriate action in a timely manner.

3. The robot will typically need to be able to perform a variety of actions. Each of these actions may compete for resource allocation and the robot will need to be able to select the correct actions to perform, to allow it to achieve its own goals.

2.2.3.1 Symbolic Approach

This follows from the same approach as traditional AI where the robot must first take in sensor data to create a symbolic representation of the world and then infer a plan. The plan is then executed in a sequence of actions. As previously discussed, depending on the agent's resources, this approach may be too slow for real-time execution and it could have difficulty in creating an accurate description for the world model.

A decision-making strategy that has been adapted to overcome some of the flaws of the traditional AI concept is the satisficing strategy. Rather than identifying the optimum solution, this approach attempts to meet criteria for adequacy. This is achieved by selecting the first plan that can achieve a goal (even if it is sub-optimal) and saving time by not searching for a better plan (Dolgov and Durfee, 2002).

The symbolic approach is most often implemented using a goal-driven architecture. In this symbolic approach each available behaviour is described by a set of goals and each of these goals can be achieved by a defined process. The robot must be able to choose which process to follow to accomplish its goals. This architecture tends to be a hybrid of the traditional AI and behaviour-based approaches.

Recent work in this field includes Klenk et al. (Klenk, Molineaux and Aha, 2013) who discussed a concept called Goal Driven Autonomy (GDA) which allowed their agent to reason and generate its own goals in response to changes in the environment. This is in response to the idea that goals need to be revised over time depending on the situation of the world (Roberts *et al.*, 2014).

2.2.3.2 Distributed Approach

The distributed approach differs greatly from the symbolic approach and follows the principles of the behaviour-based architecture. Here the distributed approach has many behaviours available and determines which behaviour to execute based on real-time local sensor information. There is also always some form of centralised decision system which determines which behaviour should be 'the most active'. Some examples of this approach include;

1. ASMO (Attentive Self-MOdifying)

   This is an attention-based architecture based on cognitive attention theory (William, 1983) and offers a mechanism for directing and creating behaviours, beliefs, anticipation, discovery, expectations and changes (Samsonovich, 2010). This architecture works similarly to the action selection mechanism (Maes 1991a;) in that behaviours are selected based on an attention value which the behaviours use to compete for selection.

2. Winner-Takes-All

   This is an architecture in which a behaviour that has the highest motivation (weighted sum of relevant sensory information and motivation / drives) is selected and takes control of the robot's motor system. Depending on the behaviour it could also take control of other aspects of the robot. The active behaviour has exclusive access to all of the robot's resources and does not share with any other behaviour. It was noted (Prescott, 2002) that this architecture can be slow when a behaviour releases access of the robot's actuators to another behaviour, which can give the appearance of the robot stalling.

3. Spreading Activation

   This is covered in detail in section 2.2.4. The spreading activation mechanism was pioneered by (Maes, 1991b, 1991c; Tyrrell, 1994) and is a model in which activation energy is passed amongst the behaviours in a non-hierarchical network. The activation energy is an arbitrary value that is used to identify the most appropriate behaviour in a behaviour network. This energy is first passed into the network from the goals and the environment nodes where it is then passed between the behaviours of the network. The behaviour with the largest amount of activation after this process is selected for execution.

4. Extended Rosenblatt and Payton

   In the late 1980s Rosenblatt and Payton (Rosenblatt and Payton, 1989) responded to the work of (Brooks, 1986), on the subsumption architecture, and created a connectionist (interconnected network), hierarchical, feed-forward network later known as the Rosenblatt and Payton. In this network, there is a hierarchy of nodes (behaviours) and these nodes receive information from; internal, indeterminate and external stimuli feeding their activation down the hierarchy until it reaches an action node. The action nodes are located at the bottom of the hierarchy named the behavioural final common path; it is at this level of the hierarchy that a winner-takes-all process is used to select which action node to execute. Unlike other action selection mechanisms, the Rosenblatt and Payton allows multiple nodes in the network to remain active at once. Each of these active nodes excites other nodes in the hierarchy by passing a degree of preference to the nodes beneath it (either a positive or a negative preference). The inputs to these nodes (degree of preference from other nodes)

are then combined following some predefined formulae, with the behaviour with the greatest preference being selected for activation.

Tyrrell (Tyrrell, 1993) reviewed the work of Rosenblatt and Payton (Rosenblatt and Payton, 1989) as well as many others in the field of behaviour networks and created his own variant the Extended Rosenblatt and Payton. Tyrrell (Tyrrell, 1993) discovered fundamental flaws in the original work: he found that some basic action nodes were being selected more than others because appetive sub-systems were similar in size to those of consummatory nodes (consummatory nodes are those that achieve goals and appetive nodes are those that enable consummatory nodes). Tyrrell (Tyrrell, 1993) introduced uncertainty and temporal penalties to ensure that consummatory nodes were preferable to appetive nodes (appetive nodes help other nodes become active); however, the penalties that were applied were too severe. More changes were made resulting in the penalties being reduced and the combination of inputs rule was changed from a simple summation to a more specific rule. This architecture is explained further in section 2.2.4.

5. Subsumption-Based Architecture

The subsumption-based architecture was created by Brooks (Brooks, 1986) in the 1980s in response to the lack of reactive systems available in robotics at the time. The subsumption-based architecture comprises of reactive processes that compete with each other for access for the robots control. A behaviour emerges from the fixed connections between the processes. Although the subsumption-based architecture is symbolic, it also follows a distributed approach as their behaviours are executed based on real-time sensor information and it uses a centralised decision system to select the most appropriate behaviours.

2.2.3.3 Dynamic Planning Approach

Distributed solutions (those discussed in section 2.2.3.2) can be very difficult to construct if the robot is to work in a dynamic and uncertain environment, because these systems require the behaviours of the agent to be predefined. Dynamic planning is an approach using explicit-hardcoded plans to determine the priorities of the system, which avoids combinatorial explosion in complexity. The dynamic planning approach only computes the next action for each time-step based on pre-scripted plans and the current context. An example of a dynamic planning approach applied to path finding in unstructured environments is given by (Ferguson and Stentz, 2007),who used a Rapidly-exploring Random Tree (RRT) algorithm combined with sampling techniques to allow an agent to successfully navigate a dynamic environment.

These are reactive hierarchical architectures and are commonly found in video game agents (Cutumisu and Szafron, 2009). A finite state machine works with a limited number of states with the agent being in one state at any given time. When conditions in the environment trigger changes, then the agent can transition to another state. The finite state machine consists of a list of its possible states and the conditions to transition between states (Chow, 1978). Due to the predefined nature of finite state machines, they are not commonly used at a top level, instead they are commonly used as elements in robot architectures and this is due to the dynamic nature of the environment.

2.2.4 Behaviour Networks

One architecture that uses action selection mechanisms is the behaviour network. This section begins with a basic introduction to behaviour networks before giving a thorough explanation and analysis of the behaviour network.

2.2.4.1 Introduction to Behaviour Networks

In the early 1990s Maes (Maes, 1991b, 1991c; Tyrrell, 1994) proposed the Agent Network Architecture (ANA) as a method of selecting behaviours. This incorporated a spreading activation process to an action selection mechanism, allowing an agent to become autonomous. ANA is based on the concept of a network of behaviours, which may be activated as required. The behaviour network consists of a variety of nodes, which are joined by a variety of links. The aim of the behaviour network is to pass activation energy between the nodes via the links until a behaviour reaches an activation energy greater than a set threshold, when it can be selected for execution. The activation spreading method is discussed in section 2.2.3.2. Figure 2-13 shows a basic example of the structure of a behaviour network and how the nodes might be connected to one another.

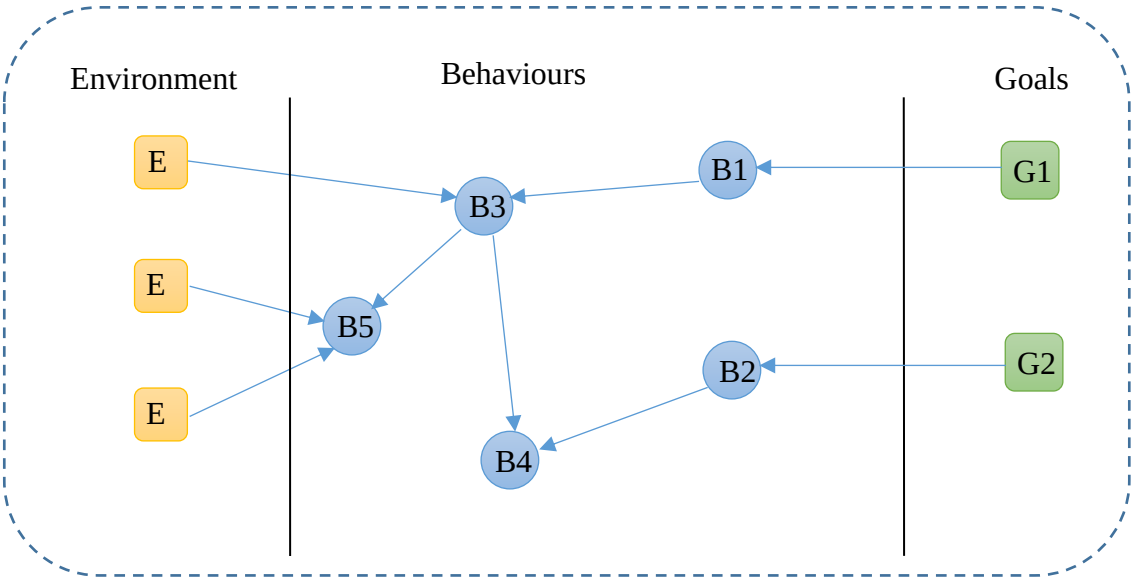

Figure 2-13 - Structure of a behaviour network

This diagram shows the basic decomposition of a behaviour network.

2.2.4.2 Components of a Behaviour Network

This section discusses in detail all the components of a behaviour network and how those components are able to work together to drive an autonomous agent. This section can be used as a tool to assist in the design and development of a behaviour network. Similar to many other AI techniques, the core of a behaviour network is the concept of a behaviour. The concept of behaviours owes greatly to the work of Brooks (Brooks, 1986, 1995), who helped to popularize the behaviour-based approach. This technique was extended by Bryson (Bryson, 2000) who described a combination of behaviour-based and object-oriented approaches leading to behaviour-oriented design.

2.2.4.2.1 Behaviours

A behaviour can be characterized as an entity of an agent that encapsulates a process. A process can be anything, ranging from moving to a specific position to completing a given task. The challenge has always been how to select the best behaviour for a given situation (Brooks, 1986). Figure 2-14 shows the components of a behaviour in a behaviour network.

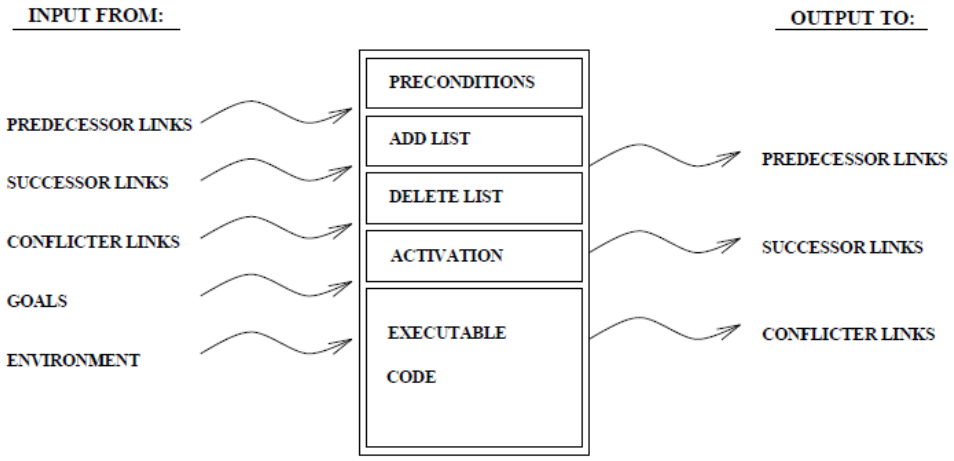

Figure 2-14 - Components of a behaviour

This diagram depicts the inner components of a behaviour in a behaviour network, also showing the inputs and outputs of a behaviour (Brooks 1986, Maes 1991, Tyrrell 1994).

A behaviour node in a network can have many types of input coming from goal nodes, environment nodes or other behaviour nodes. The input to a node is passed via a range of different types of link. Those links either increase a node's activation or reduce it. A behaviour may contain a list of preconditions that will all need to be met in order for the behaviour to be activated. A behaviour will also contain (the 'add list') a list of conditions that if the behaviour was executed it would make true and (the 'delete list') a list of conditions that if the behaviour was executed it would make false. The behaviour network operates on the basis that each behaviour has a certain amount of activation energy. This is an arbitrary value that is used to manage the amount of activation a behaviour holds. When the activation spreading process is started (the process to move activation energy between behaviours to identify the most appropriate behaviour in a

given situation), the amount of activation in a behaviour is either increased (from inputs) or decreased (through outputs). Finally, a behaviour will also store something that can be used to execute the process that it represents. Maes (Maes, 1991b, 1991c; Tyrrell, 1994) uses executable code for this component of a behaviour. This component can also be a hierarchy of behaviour networks (Nicolescu and Matarić, 2002; Lee and Cho, 2014).

There are two types of behaviour in a behaviour network; consummatory behaviours and appetitive behaviours. The consummatory behaviours are more beneficial to a system than appetitive behaviours as they provide an immediate effect (towards achieving a goal) whereas the appetitive behaviours assist the consummatory behaviours to become selected for activation (appetitive behaviours do not achieve goals, instead they enable consummatory nodes). Appetitive behaviours should lead into a consummatory behaviour, creating an appetitive-consummatory sequence. Using an example from (Maes, 1991d), there could be a goal called 'Hungry' ('Food OK') and to achieve this goal the agent will need to consume some food. The immediate consummatory behaviour to achieve this goal would be 'Eat Food' in a specific location. This consummatory behaviour would then have an appetitive behaviours 'Go to Food' as this behaviour would help to make the consummatory behaviour active and available for activation.

2.2.4.2.2 Components

The primary component of the behaviour network is behaviours, discussed in 2.5.2.1. Those behaviours are typically at a lower level than the system level and are also a higher level than the atomic actions. Figure 2-13 shows a basic example of the structure of a behaviour network and how the behaviours might be connected to one another. It shows that the behaviours are connected by external nodes; goals and

environment nodes. The goal nodes depict what the agent will be trying to achieve and are connected to the consummatory behaviours that will achieve the goal passing excitation to those nodes. If there are behaviours that can undo a goal, then the goal will inhibit those behaviours. This is based on a goal been part of the delete list of behaviours. The environment nodes derive from the processed sensor data that the robot receives and are connected typically to appetitive behaviours in the system.

As well as the connections from the goals and the environment nodes to the behaviour nodes, there are also different internal connections between the behaviours themselves. The internal connections of the network encode different types of relationship (Tyrrell, 1994):

- Consummatory-Appetitive relationships between nodes
- Conflict relationships between nodes
- Goal-achieving relationships between nodes and goals
- Goal-counteracting relationships between nodes and goals
- Situation-dependency relationships between nodes and environment sensors

Figure 2-14 shows the components of a behaviour and the input and output links, discussed in (Brooks, 1986; Maes, 1991a; Tyrrell, 1994), are specified as follows:

- Predecessor links: If proposition X is false and proposition X is a precondition of node A and proposition X is in the add list of node B (i.e., if B can help A to become executable), then there is an active predecessor link (excitatory) from A to B.
- Successor links: If proposition X is false and proposition X is in the add list of node A and proposition X is a precondition of node B and node A is executable (i.e., if A can help B to become executable), then there is an active successor link (excitatory) from A to B.

- Conflictor links: If proposition X is true and proposition X is a precondition of node A and proposition X is in the delete list of node B (i.e., if B stops A from becoming executable), then there is an active conflictor link (inhibitory) from A to B.
- Environment links: if proposition X about the environment is true and proposition X is in the precondition list of node A (i.e., if A is at least partially appropriate to the current situation), then there is an active link (excitatory) from the sensor of proposition X to node A.
- Goal links: If goal Y has an activation greater than zero and goal Y is in the add list of node A (i.e., if A is likely to achieve goal Y), then there is an active link (excitatory) from goal Y to node A.
- Protected goal links: If goal Y has an activation greater than zero and goal Y is in the delete list of node A (i.e., if A is likely to undo goal Y or to stop it from being achieved), then there is an active link (inhibitory) from the goal Y to node A.

2.2.4.2.3 Action Selection

The basic principle of a behaviour network resolves around an action selection mechanism (discussed in some detail in 2.4), which assists in the selection of the best behaviour (action) for an agent to do next at any given time. The challenge of action selection mechanisms and behaviour networks is to be able to select the correct level of abstraction for a behaviour. e.g., a behaviour of 'pick up object' is far less abstract than a behaviour of 'tidy table'. Nicolescu and Matarić (Nicolescu and Matarić, 2002) used a hierarchy of abstract levels of behaviours to solve this challenge. This method allowed a high-level behaviour network to make decisions with abstract behaviours and used a low-level behaviour network to work out how to complete the selected behaviour.

For an action to be eligible for selection, all of its preconditions must be true and its activation level must also be greater than a global threshold (Tyrrell, 1994)**,** it is not fully clear why a global threshold was used in the supporting text. When there are multiple nodes that are executable then the node with the highest activation is selected. After a node has been executed its activation is reset to zero while other nodes retain their activation levels. To calculate if an action is eligible for selection the following procedure is executed:

1. The external input is calculated from the goal nodes and the environment nodes into the network of behaviours.
2. The excitation is spread around the network via predecessor, successor and conflictor links.
3. The activation in all nodes is normalized so that the average activation is equal to π. It is also unclear as to why there is a need for normalization as the behaviour network will always select the node with the greatest activation level.

4. If any of the nodes are executable, the node with the highest activation that is over the global threshold is selected and executed. The nodes activation is then reset to zero.
5. If there are no nodes that are eligible for selection, then the global threshold is reduced and the cycle is repeated. As previously mentioned, this appears to be a pointless step. The threshold will continue to drop until a behaviour's activation exceeds it, at which point the behaviour network will execute the behaviour with the greatest activation (whether that is a normalised value or not).

2.2.4.2.4 Activation Spreading and The Division Rule

The process of spreading activation energy around the network is based on a concept called the division rule, that is explained in (Tyrrell, 1994), an emergent flaw from the original work of Maes (Maes, 1991a). The division rule refers to one of the steps of a behaviour network. The network must calculate how much energy excitation needs to be spread from one behaviour to the next. The original behaviour network (Maes, 1991a; Tyrrell, 1994) used the following constant parameters:

Global Parameters

(θ) The initial value of the global threshold

(ϕ) A constant used to determine the weighting of environmental inputs and successor links

(γ) A constant used to determine the weighting of goal inputs and predecessor links

(δ) A constant used to determine the weighting of protected goal inputs and conflictor links

The global parameters are constant values that are used in the behaviour network as input multipliers to calculate the amount of activation that is to be spread from one node in the network to another. The different link types in the network use the previously defined global parameters for the input multiplier calculations shown below.

Input Multipliers

(ϕ) Environmental sensors

(γ) Goals

(δ) Protected Goals

($\frac{\phi}{\gamma}$) Successor Links

($\frac{\gamma}{\gamma} = 1$) Predecessor Links

($\frac{\delta}{\gamma}$) Conflictor Links

The input multipliers originated in (Maes, 1991a) and were further detailed in (Tyrrell, 1994). These input multipliers use the constant global parameters and are used to define the proportion of activation energy to move between behaviours. To assist in this calculation, a division rule is also applied to ensure an even amount of activation energy is passed to nodes with multiple inputs / outputs and is defined below.

Division Rule

(N) Number of outputs of a given type

(M) Number of inputs of a given type

Example:

Given there is a situation where there is a predecessor link from node X to node Y. The proportion of energy passed from node Y to node X is given by

$$= \frac{\gamma \alpha}{\gamma NM} \quad (1)$$

$$= \frac{\alpha}{NM}$$

where α is the activation of node Y.

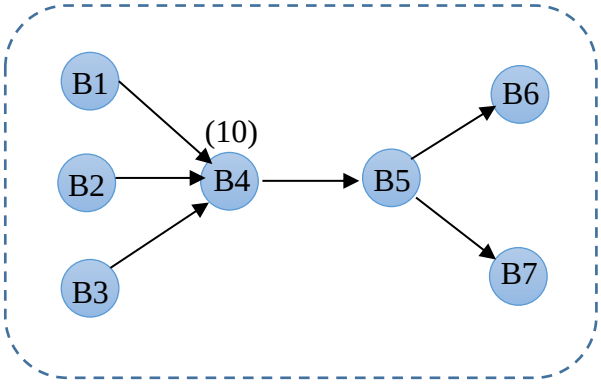

Figure 2-15. First step of the activation spreading function.

Figure 2-15 shows an example network made up of nodes and predecessor links. In this example, activation energy will move from B4 to B5. Node B4 currently contains an activation energy value of 10. It has 3 input nodes and 1 output node. Using the previously defined formula from (Tyrrell, 1994):

$$= \frac{10}{1*3} = 3.33$$

This would move a value of 3.33 from node B4 to node B5. Node B4 will retain a large proportion of energy because it has a large number of inputs of the same type. Further research into the input multipliers and the division rule was conducted by (Tyrrell, 1994) and is discussed later in this section.

Tyrrell (Tyrrell, 1994) found that there was an unbalanced competition between consummatory nodes caused by the division of the number of outputs. For example; given there is a situation where there is a one-to-one connection between a behaviour (B1) and a goal (G1) then during each cycle B1 will receive an input of (γ). Whereas; given there is a situation where there is a one-to-many connection between a goal (G2) and behaviours (B2 and B3) then during each cycle B2 and B3 will receive an input of ($\gamma/2$). This shows that the more behaviours that are connected to the same goal the more diluted the excitation those behaviours receive in each cycle. This is shown in Figure 2-16. This shows that behaviour B1 will be favoured compared to the other behaviours in the network.

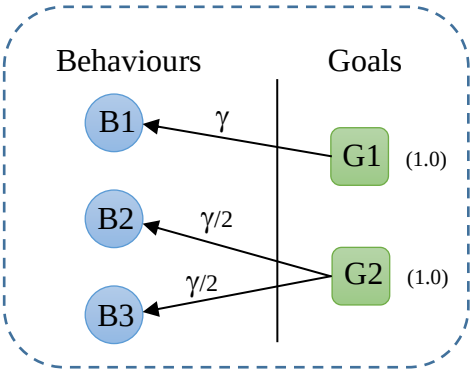

Figure 2-16 - Demonstrates the unbalanced competition between the nodes due to the division of the number of outputs.

Tyrrell (Tyrrell, 1994) then changed the division rule by removing (N) altogether and found a new problem with the division rule; namely that there was a prejudice against nodes with many inputs of the same type. For example; given there is a situation where there is a one-to-one connection between a behaviour (B1) and a goal (G1 with 1.0 energy) then during each cycle B1 will receive an input of (1.0 * γ). Whereas; given there is a situation where there is a one-to-many connection between a behaviour (B2) and multiple goals (G2 = 1.0 and G3 = 0.2) then during each cycle B2 will receive an input of (0.6 *γ). This is shown in Figure 2-17.

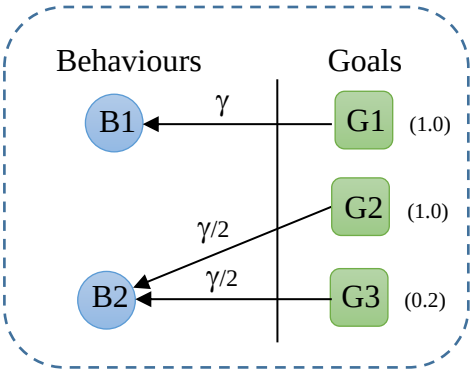

Figure 2-17 - Demonstrates the unbalanced competition due to the division of the number of inputs.

Tyrrell (Tyrrell, 1994) made some further changes to the division rule in order to allow the nodes that can achieve multiple goals to get more activation. He removed the division by goal, predecessor, protected goal and conflictor links. This method also proved to be unsuccessful in scenarios where a node has multiple predecessor links. Instead of being penalised with the division rule, these nodes received far more activation than their competition. This shows that there is a problem with and without a division rule in some instances. Table 2-1 shows the different problems that can occur with the division rule. Ideally there would be a division rule that is appropriate in all situations and enables the agent to successfully complete its goals.

Division Rule	**Deficit in Action Selection**
Division by N	Penalizes nodes sharing inputs
No division by N	No deficit
Division by M (successor and environment links)	No deficit
Division by M (goal and protected goal links)	Penalizes nodes with high M
No division by M (goal and protected goal links)	No deficit
Division by M (predecessor and conflictor links)	Penalizes nodes with high M, where inputs are due to different goals
No division by M (predecessor and conflictor links)	Favors nodes with high M, where inputs are due to the same goal.

Table 2-1 - The effects of the different variations of the division rule (Tyrrell 1994).

Tyrrell (Tyrrell, 1994) proposed a solution to this problem: to take the average of the input value both with and without division. The predecessor input thus becomes:

$$I_P = \frac{1}{2}\left(\alpha + \frac{\alpha}{M}\right) \quad (2)$$

In addition, the formula for conflictor input becomes:

$$I_C = \frac{1}{2}\left(\alpha\frac{\delta}{\gamma} + \frac{\alpha}{M}\frac{\delta}{\gamma}\right) \quad (3)$$

A side effect of this implementation is the possibility of a feedback loop of both appetitive nodes and consummatory nodes. It was shown that the appetitive nodes were selected more often than consummatory nodes after multiple iterations of the spreading activation process. Tyrrell (Tyrrell, 1994) suggests that there is no general division rule for predecessor links that works for all situations; this is "because there is no way of knowing whether or not the inputs derive from the same goal". Tyrrell (Tyrrell, 1994) also stated that it was not possible to solve the division rule problem without making "radical changes to the architecture and central concepts of MASM". Chapter 4.1 details a potential solution to this problem.

2.2.4.3 Factors of Favourability

The primary benefit of behaviour networks is that they are designed to work in dynamic unstructured environments. They use a reactive architecture that is able to select the best action for a robot to take in the existing situation. Typically, there is no planning system involved in a behaviour network; the culmination of behaviours over time are hoped to achieve the global goals set. A plan can thus be seen as an emergent property of these behaviours over time and could be used as a template for future behaviours. Lee and Cho (Lee and Cho, 2014) were able to extend the behaviour network by adding a deliberative control layer on top of a behaviour network. This allows the behaviour

network to subsume the commands of the planning layer when a new situation is presented, and a different action should be executed.

The concept of dynamic action selection is a valid and essential tool for robot planning because, in the real world, environments are constantly changing making it difficult for hard-coded solutions to remain successful.

2.2.4.4 Liabilities

Behaviour networks have not seen a great deal of research in recent years. There are a variety of possible reasons for this, each stemming from the liabilities of behaviour networks described previously. Behaviour networks do not plan, being reactive systems, and more recent work has focused on robot architectures (Knepper, Srinivasa and Mason, 2010; Weser, Off and Zhang, 2010) using planning, such as implementing a planning architecture on top of a behaviour network (Lee and Cho, 2014). Others have favoured learning algorithms, such a neural networks (Lin and Lee, 1991; Miljković *et al.*, 2013) as behaviour networks do not learn from their actions or about their environments.

The supporting texts on behaviour networks define some core liabilities with the system (Maes, 1991a; Tyrrell, 1994). The main liability is the division rule which is explained in the previous section. Other liabilities with behaviour networks include the binary nature of behaviour node achievement. Nodes can either achieve a goal or not achieve a goal; there is no mechanism in a behaviour network that allows a behaviour to partially achieve a goal. The same applies to negating a goal, the behaviour nodes either counteract a goal or they do not. For example; a robot could partially complete a goal of ‘Clean Kitchen’ by ‘washing the dishes’. However, this would counteract the goal

of ‘Cook Meal’ to some degree because the pots currently being used by this task might be cleaned by the other.

Tyrrell (Tyrrell, 1994) explains that there is a lack of persistence present in behaviour networks. This is because the original behaviour network (Maes, 1991a) will remove all energy in a behaviour that has been activated and the energy in all other behaviours will remain. The behaviour network needs to make a decision in each time-step and any residual activation energy could negatively influence the next selected behaviour. For example, over many time steps an undesirable behaviour could accumulate enough activation energy to become the next desired behaviour for activation. This is a huge liability for behaviour networks as it can either cause undesirable behaviours to be favourable or cause the agent to oscillate between two or more behaviours. For example, an agent could be in a situation where it is navigating a room and needs to decide which direction to go. Moving to the right will help it bring it closer to achieving one goal while moving to the left will help it to achieve another. In one time step the agent moves to the right and the energy in the ‘Move Right’ behaviour is reset to 0. However, the ‘Move Left’ behaviour will retain some energy from that time step. In the next time step, both the ‘Move Left’ and ‘Move Right’ behaviour is given some more activation energy but the combination of two time steps worth of energy in the ‘Move Left’ behaviour overcomes the fresh energy in the ‘Move Right’ behaviour. In this example the agent is now stuck in an oscillating loop.

It was found during the implementation of the behaviour network that the possibility of connection loops can exist in the behaviour network. This is explained in more detail in section 4.3, and is caused when multiple behaviours are reliant upon one another. When energy is passed into this loop, it will then continue to be passed infinitely as there are no documented methods to break the loop.

This section has detailed many liabilities within behaviour networks, some of which could be a contributing factor to why they have not seen a lot of interest in recent years. These liabilities range from; problems with the internal workings of the behaviour network (such as the division rule), a lack of persistence for making decisions and the possibility of the energy spreading mechanism getting stuck in a loop. In the following sections some of these liabilities shall be addressed and potential solutions suggested.

2.2.4.5 Improvements to Behaviour Networks

The previous sections discussed the current limitations with behaviour networks. It is possible that if these limitations were improved, then behaviour networks could see more research and implementations in the future. The lack of planning in behaviour networks has already seen a potential solution. Lee and Cho (Lee and Cho, 2014) added a deliberative control layer on top of a behaviour network. The deliberative control layer handled the planning side of a robot, while the behaviour network reacted to dynamic changes in the environment. This showed that a top-level planner could make the long term plans for a robot while a behaviour network operated as a reactive low-level layer enabling the robot to function in dynamic environments.

The division rule is a problem that has yet to be solved. Tyrrell (Tyrrell, 1994) showed a variety of different solutions but later explains that there is no perfect solution for the division rule. Section 4.1 shows a potential solution to this problem by embedding data into the links connecting the nodes to allow a smarter, more dynamic division rule.

The previous section discussed the lack of persistence present in behaviour networks and the problem associated with it, selecting inappropriate behaviours. One solution involves removing all of the energy in the network every time a behaviour is

activated and another solution is to give the selected behaviour some energy equal to the average amount of energy in the network.

Finally, a method to escape the feedback loop of a behaviour network is presented in section 4.3. This method involves creating data packets to travel through the behaviour network allowing more data to be collected about the source and destination of the data, preventing repetition.

2.2.4.6 Recent Work

While behaviour networks are no longer a heavily-researched topic, there have been some recent publications that have implemented behaviour networks into their solutions. Chae and Cho (Chae and Cho, 2014) proposed a hybrid control system that integrated a behaviour network, to cope with dynamic environments and a STRIPS planning module. The control module was used to create a sequence of reactive behaviours of which the behaviour network could take advantage. This system was tested with 30 different experiments and showed that the agent was able to complete both simple and complex tasks. The combination of STRIPS and behaviour networks allowed it to cope with unstable environments while achieving goals.

Paikan et al. (Paikan, Metta and Natale, 2013) describes a method for developing a reactive system by exploiting the connection between behaviours. Here a publish-subscribe methodology is applied to send data messages between the behaviours using connection points. The result of this methodology was tested on a humanoid robot and demonstrated that the final behaviour in a sequence could be built / composed from other behaviours in the system. However, there were some documented limitations to this approach. Firstly, this approach required parameters and rules to be defined specifically for each experiment. Second, this approach struggled with scalability as the

system could only handle a limited number of behaviours and rules. Finally, it was documented that there are difficulties in monitoring the current state of the system. This is caused by the publish-subscribe mechanism and could be solved by displaying the information passed between the modules.

Lee and Cho (Lee and Cho, 2014) also proposed a hybrid architecture that implements a planning control layer with a behaviour network. Instead of using STRIPS, a hierarchical planning layer is used to adjust the sequence of behaviours by using information from user feedback and sub-goals. The planning layer sits on top of the behaviour network to control and manage the whole sequence of behaviours taking into account sub-goals. This combination is designed to allow a robot to react in dynamic environments and achieve global goals. This methodology was shown to reduce the time between executing tasks by 17.5%, allowing the robot to select and execute relevant behaviours more efficiently. This was implemented in both a simulated environment and in the real world. However, the experiments were limited and the results from the hybrid method were not compared with original implementations of a behaviour network.

Kertész (Kertész, 2012) developed a variant of the behaviour network to handle computational resources of a robot more efficiently. In this study they proposed a system where a behaviour network is able to create and destroy behaviours in the network at run-time. This is a variant of a dynamic behaviour network. The benefit of this approach is that the behaviour network can remain simple and even complex goals can be broken down into smaller sub-systems. One limitation of this work (similar to others) is that each behaviour is still predefined for a particular problem / goal.

2.3. Alternative Approaches to Robot Control

2.3.1. Hierarchical Mechanisms

This section will discuss the different hierarchical mechanisms for action selection. It will begin by looking into the Rosenblatt and Payton architecture, which was one of the first hierarchical architectures developed as a response to the subsumption architecture. The expansion to the Rosenblatt and Payton architecture developed by (Tyrrell, 1994) will be discussed, concluding with some recent examples of hierarchical mechanisms that are being used.

Rosenblatt and Payton (Rosenblatt and Payton, 1989) created a layered architecture motivated by the principles of (Brooks, 1986)'s subsumption-based architecture. This layered architecture built upon the concept of creating a robot control system using successive layers of competence. This is achieved by breaking down the task achieving behaviours into small decision-making units. This allowed for more information to be available to other units and allowed the units to be fully accessible. One of the limitations of the subsumption-based architecture is that the behaviours are closed and are inaccessible from any of the other behaviours in the system. That is not the case in this architecture as there is no such communication barrier between behaviours. The small decision making units are defined by how they are connected to one another. They each take input data and transform them into output data for other units.

Rosenblatt and Payton (Rosenblatt and Payton, 1989) also explored another limitation of the subsumption-based architecture; that there is no cooperation between behaviours. Figure 2-18 shows an example scenario where each behaviour has the choice of five possible actions A – E. Each behaviour will select the action that is best for

itself. The diagram below shows that for behaviour 1, action A best fits its requirements, however action B and E would be satisfactory. For behaviour 2, action D would be its best choice with action E being satisfactory.

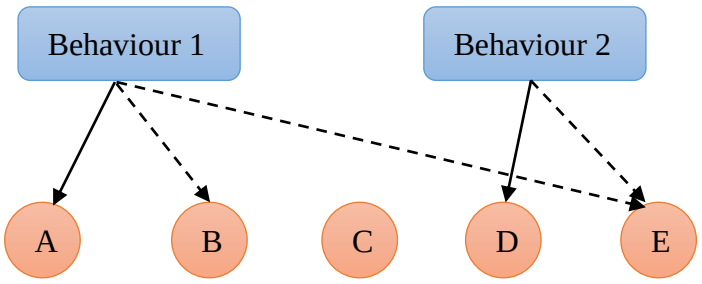

Figure 2-18 - Lack of cooperation in action selection

The diagram shows that the preferences for individual behaviours are not combined when considering an action.

The next figure (Figure 2-19) shows the above configuration in a subsumption architecture; it shows that the output in this scenario would be A if behaviour 1 were suppressing behaviour 2 otherwise, action D would be selected. The alternative behaviours in Figure 2-18 are lost as neither behaviour has a means to communicate with the other behaviours to come to the compromise of selecting action E.

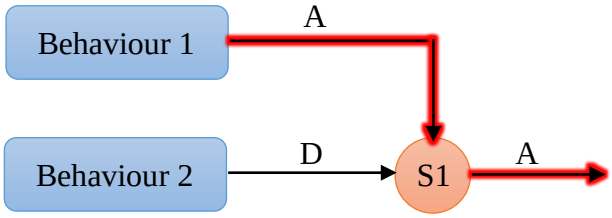

Figure 2-19 - Lack of cooperation in subsumption architecture

This diagram shows that neither behaviour in a subsumption architecture have a method to communicate so there is a loss of preference for which action a behaviour would like to execute.

The solution to some of the problems of the subsumption architecture (see section 2.2.2.) as explained by (Rosenblatt and Payton, 1989) was to make the behaviours as 'fine-grained' as possible to ensure that no module in the system had inaccessible internal states. In this new system, behaviours are broken down into a collection of simple decision-making units. Each of these units receive weighted inputs

from the other units in the system and from external data sources such as the robot's sensors. The unit then computes an activation level and sends out a single output.

Figure 2-20 extends the problem shown in Figure 2-18 and Figure 2-19. Here two behaviours again have the choice of five actions A – E. The amount of activation for each unit's choice is symbolised by the diameter of the circle. Green circles represent positive values and red represents negative (inhibition) values. The output of each behaviour is sent to a command unit, which combines the given input from the other units. The command unit selects the action with the highest activation (E) as the best compromise for both behaviour 1 and behaviour 2.

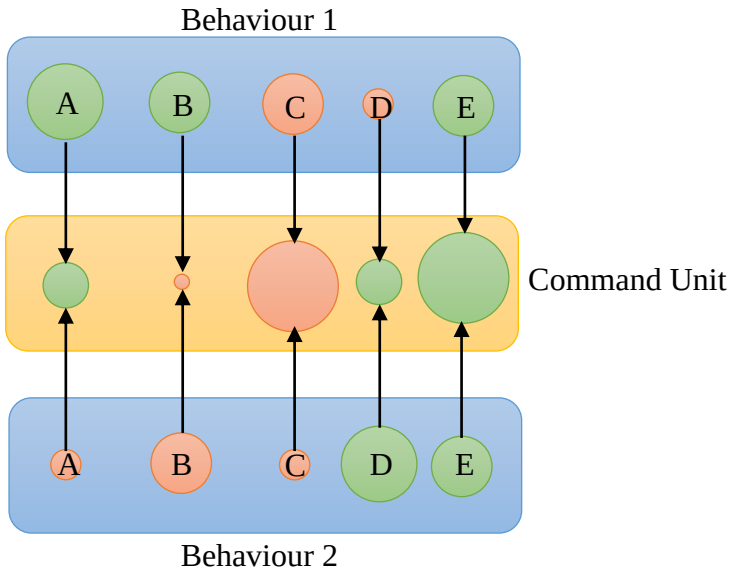

Figure 2-20 - Cooperation in the Rosenblatt and Payton

In this diagram multiple behaviours send their preferences to a command unit to decide an agent's actions.

Each behaviour in the system is distributed among several units allowing each unit to have a choice in the overall output of the behaviour.

This new architecture also solved the problem of modularity in the subsumption architecture as new behaviours that are added to the system do not subsume any of the existing behaviours or prevent those behaviours from sending their outputs. The new

behaviours express their decisions on what appears to be the best command to execute and leave it to the command unit's output to provide the best decision for all behaviours.

Rosenblatt and Payton (Rosenblatt and Payton, 1989) demonstrated the different levels of competence in their architecture. At the lowest level was an 'Avoid Obstacle' behaviour, which used a vehicle model trajectory (VMT) to indicate how far in a certain direction that a robot can go before it meets an obstacle, this behaviour was broken down into many fine-grained units. Some examples of these units include; a trajectory selection unit, which, is used to determine which trajectories are hazardous and which are safe, and a trajectory speed unit to determine the appropriate speed to travel down a trajectory. These units all work together, sharing data to give the basic level of competence to enable the robot to avoid obstacles. The next level of competence that (Rosenblatt and Payton, 1989) demonstrated in their system implemented a gradient field to indicate an optimal route from its current location to a goal location. This layer is again broken down into multiple 'fine-grained' units, which are able to assert an influence in the direction the robot has to go.

This approach expanded upon the early subsumption architecture by adding both modularity (adding more behaviour nodes) and a mechanism to allow the agent to show a degree of preference between the different behaviour nodes. This allowed the system to make more informed and 'smarter' decisions.

2.3.2. Reinforcement Learning

Reinforcement learning is a technique which uses a reward-based system (scalar evaluation) to allow an agent to select actions in a given environment with the aim of maximising its reward score (Kaelbling, Littman and Moore, 1996; Sutton and Barto, 1998; Kober, Bagnell and Peters, 2013). The overall goal is to select the action which will yield the highest reward, based on past experience. This technique has been heavily researched and applied to a variety of areas including robotics, game theory and multi-agent systems.

Smart and Kaelbling (Smart and Kaelbling, 2002) claimed that reinforcement learning was well suited for mobile robots and implemented a variant which they called Q-learning (Watkins and Dayan, 1992). Q-learning is a reinforcement technique that learns an action-value function for each possible world state and selects an action based on the value of each possible world state it can move to. In Smart and Kaelbling (Smart and Kaelbling, 2002) implementation it was assumed that the world could be described by a set of predefined states and that the robot could only choose from a fixed number of actions. At each time step the robot would observe the environment (state of the world) and choose an action to take. After the action had been executed the robot would receive a reward, based on how successful that action was. The robot would continue to perform this sequence for a predetermined number of iterations, in the hope that it had learned the best actions to take in each world state.

Recent work in the area of reinforcement learning includes (Mnih *et al.*, 2015) who combined reinforcement learning with neural networks to create a deep Q-network. This Q-network was tested on a variety of computer games to show that a reinforcement algorithm can work with high dimensional sensor data. In this study the

Q-network received pixels and the game score as inputs and the results show that the Q-network could out-perform previous algorithms. Kretzschmar et al. (Kretzschmar *et al.*, 2016) describes using a technique for modelling the behaviour of pedestrians to assist the navigation of a mobile robot. An inverse reinforcement learning algorithm was implemented. The inverse reinforcement algorithm aims to discover the reward function from a given behaviour model, in this case pedestrian movements. This technique was successfully implemented in a mobile robot and showed that it was able to navigate in an office environment with human obstacles.

While reinforcement learning can be applied to a variety of different areas, the algorithm does have some limitations which can limit the extent that it can be used in robotics. Reinforcement learning begins with the robot having no knowledge of the environment that it is in or the outcomes of its actions on the environment. The robot must learn, typically by trying random actions, how to function in an environment. Since it does not know how to function in an environment to begin with, the robot will perform many incorrect actions, such as hitting obstacles. This could take some time for the robot to find a reward and begin to improve. In other areas this initial process could be simulated and after thousands of iterations the agent would learn the mapping function to the environment. Another major limitation of using reinforcement learning for robotics is that it is difficult to describe the world state in a way that the reinforcement algorithm can use as input. This is because world states are large in complexity and can also change in dynamic ways. Techniques such as the value-function approximation (Boyan and Moore, 1995) can be used to reduce the dimensionality of the world state for the reinforcement algorithm to use. Current work in reinforcement learning for robotics focus on basic tasks such as walking down a corridor or avoiding obstacles.

2.3.3. Neural Networks

Neural networks are a popular machine learning technique that can be used in a variety of different fields, including data science and robotics. The aim is to mimic one of the best learning mechanisms currently available, the human brain. Work in the field of mathematics in the early 1940s (McCulloch and Pitts, 1943) paved the way for research in neural networks in the early 1950s (Kleene, 1951). Due to the limitations in the processing power of computers at that time, advancements in neural networks were limited. In the early 1990s research in this area increased dramatically as the computational power of machines allowed them to handle larger and more complex neural networks (Specht, 1991; Lawrence *et al.*, 1997; Haykin and Network, 2004).

Neural networks consist of three main components; the neuron, the connectome and the activation function. The neuron, shown in Figure 2-21, takes information from other neurons or from independent variables via a synapse (link), combines it and outputs to other neurons. The inputs to the neural network are from a single observation (such as a world state or row in a database). The input values must then be standardized or normalised so that each of the values are in a similar range, this is to allow the neuron to sum the values correctly. The output of a neural network can be a continuous value, binary values or even a robot action.

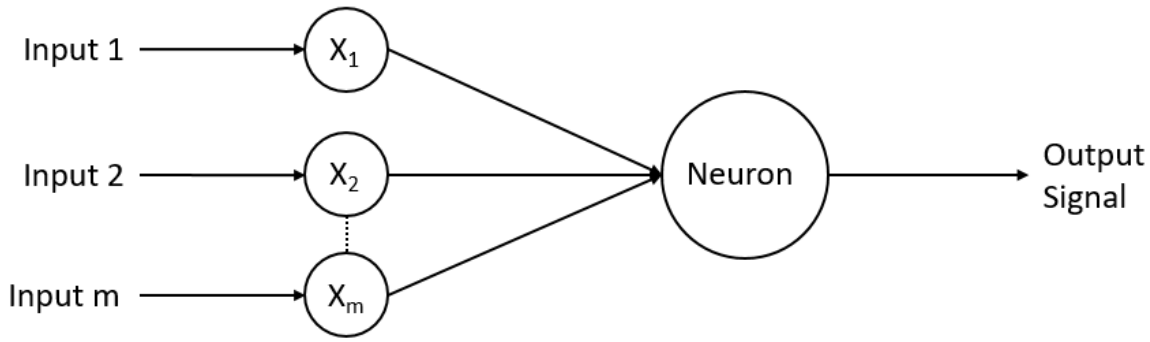

Figure 2-21 - The neuron of a neural network.

The synapses (links) that connect the input nodes and neurons to other neurons each have their own weighting mechanism. The way that neural networks learn is by adjusting the values in these weights until the desired outcome is achieved. This will typically use techniques such as gradient descent or back propagation to make these adjustments.

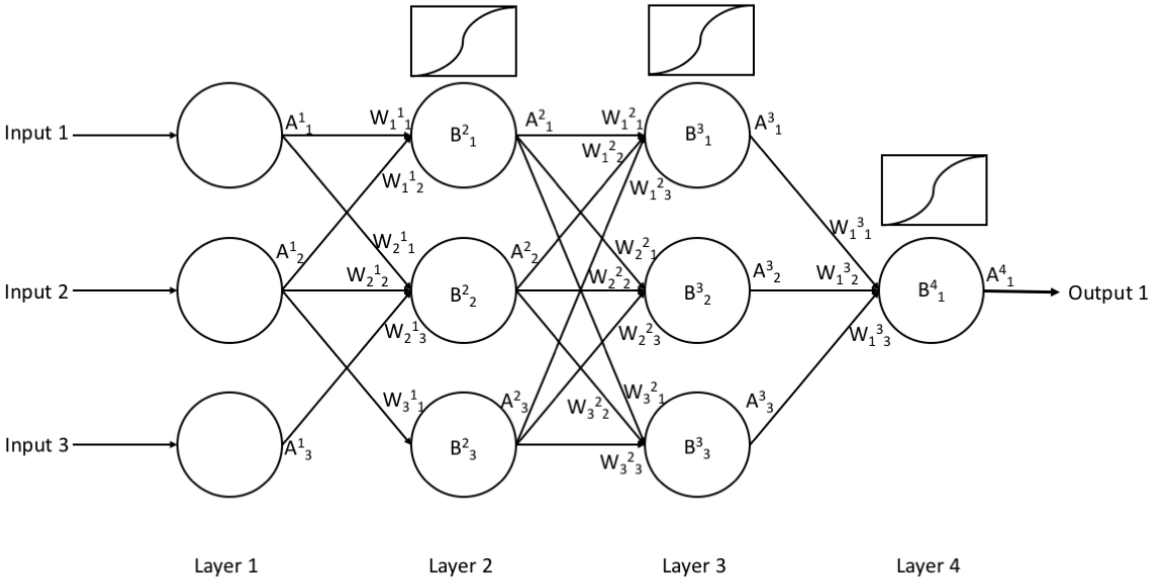

Figure 2-22 - The components of a Neural Network

Where A represents the input from a neuron, B represents a neuron and W represents the weighting on the synapse.

When data comes into a neuron, the first task is to calculate the weighted sum of all of its inputs. An activation function is then applied to this value and depending on the result of this function will determine what signal, if any, is passed on either to other neurons as an input or as the outcome of the neural network. This process may occur hundreds or thousands of times depending on the size of the network.

There are different types of activation function that can be applied in the neuron of the neural network. Which activation function should be used depends on the desired outcome.

1. Threshold Function

   This is an activation function that returns a binary result. If the value, after applying the threshold function, is zero then the outcome is also zero. If the result is one, then the outcome will also be one. Shown in the figure below.

2. Sigmoid Function

   This activation function uses a sigmoid formula on the sum of the weighted inputs. If the value approximates towards 1 then a signal will be passed on. This activation function is common in the final layer of the neural network as it can be used for returning probabilities.

3. Rectifier Function

   This is a common activation function found in the middle layers of the neural network. If the input to this function is 0 or less, then it will not pass that onto the next layer. As the value of the result increases, so does the output value.

4. Hyperbolic Tangent (tanh)

   This function is similar to the sigmoid function with the difference that it can return values ranging from -1 to 1 instead of 0 to 1.

A common structure for a neural network would be to have the inner hidden layers using a rectifier activation function and the final layer to use an activation function based on the desired output. For example, if the output is a binary value then either the threshold activation function or the sigmoid activation function could be used. The threshold activation function would output a 1 or a 0 and the sigmoid activation function would return the probability of the output being a 1.

In order for the neural network to learn, it will use a training set of data so that it can adjust the weights of the synapses (links) to get the best results. It does this by taking the output of the neural network and comparing that value to the actual value and applying a cost function on the difference. The goal is to minimise the cost function as much as possible, by returning and adjusting the weights of the synapses (links) using techniques such as back propagation and gradient descent.

Neural networks have been used in robotics for many years. Lin and Lee (Lin and Lee, 1991) proposed and demonstrated a neural network model using a fuzzy logic controller and a decision making system. This used supervised learning algorithms with a self-organising model which was shown to be superior to traditional back propagation learning models in traditional neural networks. Miller (Miller, 1994) used a neural network to teach a bipedal walking robot how to walk and make decisions in real-time. The robot was able to learn how to move its weight in order to take steps without falling over. Yang and Meng (Yang and Meng, 2000) applied a neural network to robot motion planning. The neural network was able to learn how to navigate a maze by taking in environment information as input to the neural network.

Recently Miljković et al. (Miljković *et al.*, 2013) used a neural network-based reinforcement learning controller (Q-learning) to observe the environment and select actions to control the robot's actuators. It would take the observed changes in the

environment and issue rewards to the actions depending on how well they performed. He et al. (He, Chen and Yin, 2016) designed a robotic system with full-state constraints, implementing a neural network. The neural network was able to handle uncertainties and disturbances with the robot.

The activation functions of a neural network could also be applied to the action selection mechanism of a behaviour network. The behaviour network is based on a simple summing function which could be replaced with a transfer function, as defined previously in this section, and could lead to future research on this topic.

Similar to other techniques discussed in the thesis, neural networks have limitations that reduce their applicability to some scenarios. For example, neural networks require training in order to function optimally and it can be difficult to train a system to handle uncertainty in dynamic environments. Training also takes time and can be computationally expensive (depending on the specifications of the robot). Finally, it is common for neural networks to be hard coded to accept predetermined inputs and to output expected results. This makes it difficult for a robot to learn new skills when it interacts with new objects.

This chapter has discussed a variety of different technologies and techniques that have been used in AI and robot control over the years. One limitation still holds for the majority of these and that is they have difficulties functioning accurately in dynamic and uncertain environments. The focus of this study is to explore and expand upon the technical feasibility of using behaviour networks in these dynamic and uncertain environments.

3. Proposed Approach and Architecture

In the previous chapters we have identified and discussed the possible architectures for robot control. Siciliano and Khatib (Siciliano and Khatib, 2016) explains the importance of selecting the correct approach and architecture in a robot system. Before selecting or designing the architecture, a set of requirements for the agent need to be defined. Firstly, an agent needs to be capable of navigating and functioning in a dynamic environment. Secondly, an agent needs to be capable of learning how to handle new (unknown) objects dynamically. Finally, the agent needs to be able to complete goals in a timely manner. The architecture will need to be able to support the agent to meet those requirements.

Based on the requirements, a hybrid architecture with deliberative modules was chosen. Behaviour networks have been proven to function in dynamic and unstructured environments (Maes, 1991a; Tyrrell, 1994); however, they do suffer from the limitation of being hard coded for particular goals. This thesis proposes a solution to that limitation by introducing wireless tags to store data about the objects in the environment. This introduction of wireless tags will require a robot architecture which can process the data embedded on wireless tags and update the behaviour network at runtime. This chapter details the proposed architecture, which is designed to incorporate wireless tags and dynamic behaviours into a behaviour network.

3.1. Overview

This section will go into detail, discussing the workings of the proposed architecture. It will begin with an overview of the whole architecture before going into detail for each

of the components and how they all integrate. The areas of the architecture that will be discussed include:

- Wireless tag manager
- Behaviour network manager
- Environment manager
- User interaction

Figure 3-1 shows the overall architecture of the system. The diagram shows the structure of the proposed architecture for the dynamic behaviour network. The agent uses it's on board sensors to take information from the environment. This information is processed and passed on to the deliberative control layer to act upon. The deliberative control layer also takes information about the wireless tags, from an internal database and from the input of users to create a behaviour network at runtime. The behaviour network can then decide on an action to take and pass that decision to the actuators to complete. Each component of this architecture is detailed in the following sections.

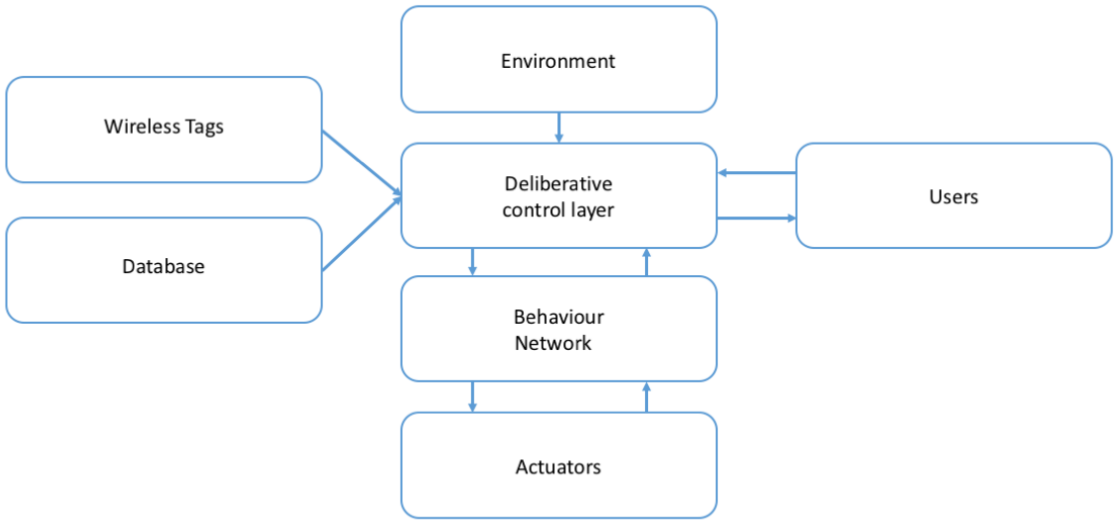

Figure 3-1 - Overall structure of the proposed architecture for the dynamic behaviour network.

The diagram in Figure 3-2 shows each of the internal components of the deliberative control layer and how they interact with one another. The wireless tag manager processes nearby wireless tags and passes information to the behaviour network manager. The environment manager processes observations in the world and also passes information to the behaviour network manager. Finally, the web manager takes input from users and passes information to the behaviour network manager.

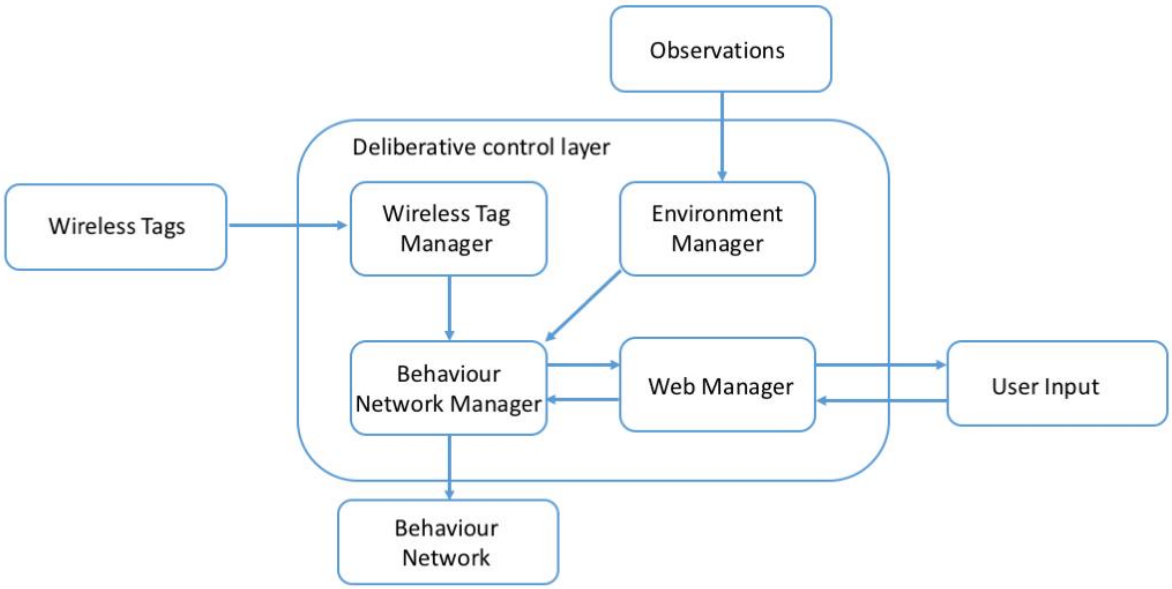

Figure 3-2 - Internal components of the deliberative control layer

When a wireless tag is located, the Wireless Tag Manager (Section 3.2) will download the data in a JSON format, process the information and add all found behaviours and goals to the agent's local database. The Environment Manager (Section 3.3) will take information (at each time step) about the environment (either from sensors or simulated environment). This information is processed and used to update the agent's representation of the world state. Should a user desire any goal to be completed first or increase/decrease its priority, then the user can adjust the motivation for that goal (using the proposed web UI in Section 3.5) and that motivation will be updated in the agent's local database. Finally, at each time step, the Behaviour Network

Manager will create a new behaviour network using the information (behaviours/goals and motivation) from its local database and the information about the environment in the world state. The behaviour network will be executed and output an action for the agent to take, sending instructions to the actuators to perform the selected action. Using this method the agent can react to changes in the environment, react to instructions from users and grow its behaviour network dynamically from wireless tags found in an environment.

3.2. Wireless Tag Manager

The wireless tag manager is the area of the architecture that is responsible for interacting with wireless tags in the environment. When a wireless tag is discovered, the information on that tag is downloaded and parsed into information that the system can use. This could include enough information to generate goals, behaviours and environment nodes that can be passed on to the behaviour network manager.

The wireless tag manager is designed to work on a real-world robot utilising its external sensors to connect to embedded wireless tags and to process any information that it found. For simulated tests of the behaviour network, the wireless tag manager was restricted to scanning for objects within a defined radius of the robot and then processing data files that were linked to that object. Figure 3-3 defines the structure of processing data files that were linked to that object.

```
{
        "ObjectId":,
        "ObjectName":,
        "GoalList":[{
                "goalID":,
                "goalName":,
                "activation":,
                "completed":,
                "Preconditions":,
                "GoalState":,
        "BehaviourList":[{
                "BehaviourID":,
                "Name":,
                "Activation_Threshold":,
                "Preconditions_Met":,
                "Currently_Executing":,
                "Priority":,
                "Energy":,
                "GoalNames": [],
                "Preconditions":[{
                        "PreconditionID":,
                        "Precondition_Name":,
                        "Value":,
                        }],
                "AddLists":[{
                        "AddListID":,
                        "Name":,
                        "Value":,
                        }],
                "DeleteLists":[{
                        "DeleteListID":,
                        "Name":,
                        "Value":}],
                }]
}
```

Figure 3-3 - Data structure of a wireless tag for use in a behaviour network

The data of the wireless tag directly correlates to the structure of a behaviour node described in the previous section.

3.3. Environment Manager

The environment manager has been designed to work on a robot utilising its external sensors to monitor, create, delete and manage the environment nodes (Figure 3-4 shows the structure of an environment node). The environment manager is the area of the architecture that is responsible for interpreting data from the sensors of a robot and converting that information into environment nodes. It will manage a list of these, some of which may come from data found on wireless tags. It will then have the task of using the robot's sensors to validate or disprove the state the nodes are in. For example; a wireless tag belonging to a Yellow Block may have an environment node 'Near Yellow Block', in its list of environment nodes. The environment manager is then able to use the sensor data from a robot such as sonar and cameras to determine the state of that environment node; in this example whether or not the robot is near the yellow block.

```
World State [{
        Observation Name: "Near Yellow Block",
        Observation Value: True
}]
```

Figure 3-4 - Structure of Environment Node.

For the simulated tests on the behaviour network, the environment manager was modified to take inputs from the simulated environment. Although the simulated environment processed the environment information differently, the core concept of the environment manager remained the same, to use this information to manage the environment nodes. These nodes are passed to the behaviour network manager to be added to the behaviour network.

3.4. Behaviour Network Manager

The behaviour network manager is the area of the architecture that is responsible for interpreting the data from the other areas of the architecture and converting that information into a behaviour network. When the system initialises, the behaviour network manager will connect to an on-board database and read in all of the initial behaviours and goals that the robot will start with and create an initial behaviour network (shown in Figure 3-5).

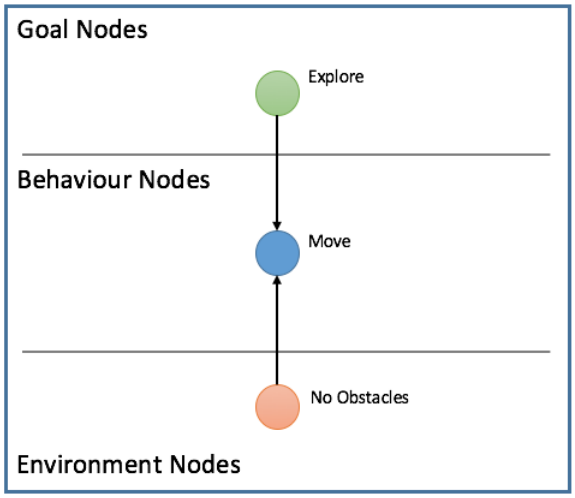

Figure 3-5 - Initial behaviour network that the robot will start with.

|This diagram displays a basic behaviour network that is used to start the robot in a new environment. As the robot explores a new environment, new goals and behaviours will be discovered and added to this network.

As shown in Figure 3-5, the robot will begin with a goal 'Explore' and a behaviour 'Move'. The robot will explore its environment until the wireless tag manager is able to find a wireless tag. At this point, the information on that wireless tag will be parsed and the behaviour network manager will receive a list of goals and behaviours to add to its on-board database. These goals will be reported to web GUI to notify the users that there are new goals available. The behaviour network will function automatically but new goals may require the authorization from the users before being added to the behaviour network as an achievable goal. At each time step a new behaviour network will be created from the behaviours and goals located in the on-board database. The

behaviour network manager also receives constant updates from the environment manager and is responsible for adding, removing and updating all of the environment nodes in the network. This is also done at each time step and is achieved by referring to the agent's interpretation of the world state (that the Environment Manager is responsible for updating) and generating new environment nodes. This method of changing the number of behaviours in a network has been investigated by Kertész (Kertész, 2012), who implemented a behaviour network that could initialise relevant behaviours at runtime.

Finally, the behaviour network manager is also responsible for starting the activation spreading process on the current behaviour network which can then return appropriate behaviours for execution. In a real-world situation, the behaviour network would select a high-level behaviour (high-level behaviours are abstract behaviours which can be completed by executing a variety of actuators) and use that to generate a low-level behaviour network to select the appropriate actuators to achieve it (Lee and Cho, 2014). In a simulated environment this step can be bypassed as each high level behaviour can have the appropriate executable code attached to the behaviour.

3.5. User Interaction

The ability to allow a user to interact with the behaviour network is one of the key methods to allow a behaviour network to become dynamic. Figure 3-6 shows an example screenshot of the proposed web GUI for user interactions with the behaviour network. Here the user is able to select from a list of goals that the robot is able to achieve and set the motivation values (how desirable it would be for the robot to complete that goal). If those goals are not already in the behaviour network, then the behaviour network manager will not only add the goal to the network but also any

behaviours that can help to achieve it. Users are also able to cancel or interrupt any goals that they do not want the robot to achieve. When a goal is cancelled or its motivation is reduced to zero then not only is the goal removed from the behaviour network but all of the behaviours that can only achieve that goal are also removed. This allows the behaviour network manager to manage the size of the network as the number of behaviours in a network can affect the performance of the network.

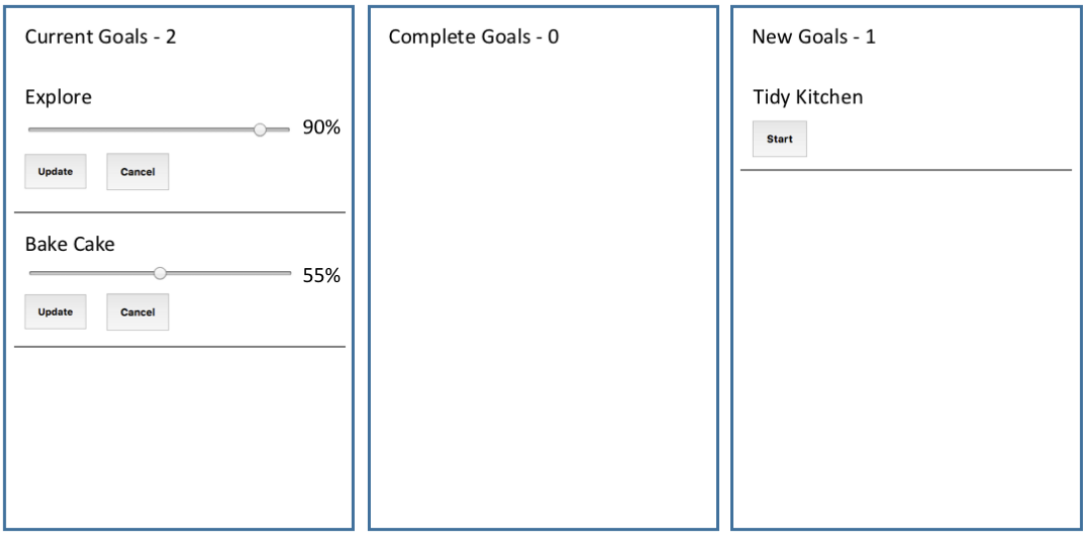

Figure 3-6 - Proposed Web GUI to allow users to interact with the behaviour network

This diagram shows the web GUI proposed to allow users to interact with the agent. The web GUI displays the current goals the agent is executing, with the current motivation associated with those goals. The web GUI will also display completed goals and new goals that become available to the agent. This proposed web GUI allows the users to dictate how the behaviours interact in the behaviour network. For example, increasing the motivation for a goal will affect which behaviours are selected for execution. This is a new element for a dynamic behaviour network.

3.6. Benefits of proposed architecture

The main benefit of the proposed architecture is that it employs a modular system in which each section of the architecture has its own purpose. Each section will act independently, not caring or knowing any details about the other modules. This allows the modules to ensure that they are able to perform their tasks and send messages to other modules that they are connected to. The modules will need to be synchronised to ensure that there is no lost information and that they are able to function concurrently. For example, the Wireless Tag Manager module is connected the to the Deliberative Control Layer and sends messages containing new behaviours to be processed. The Wireless Tag Manager is not concerned with the other modules such as the Database module or the Environment Manager.

The modularity of the proposed architecture also allows it to be extendable. As each section of the architecture is in charge of its own tasks. New modules can be added with little effort to add more functionality to the system. Although this study is focused on behaviour networks and it is at the core of the architecture, it could be 'swapped out' for an alternative action selection mechanism.

3.7. Limitations of proposed architecture

While this architecture has been designed around the behaviour network, taking into account previous work in this field (Tyrrell, 1994) and other architectures (Siciliano and Khatib, 2016), there are still some limitations to the proposed architecture. Firstly, for our experiments, it is assumed that each module is able to complete its tasks without errors. Table 3-1 shows an example of the types of error that can be exhibited in the architecture and which modules would be affected. There is no error handling functionality in the proposed system, this is because the architecture was not the main

focus of the study and was developed to enable testing of the behaviour network. Should a single module fail, e.g, the Environment Manager, the Deliberative Control Layer will continue to send messages to the Behaviour Network which will continue to attempt to select behaviours to execute. The experiments in the following sections assume that these types of errors do not occur.

Type of error	Module effected	Result of error
Inability to detect objects	Environment Manager	Cannot interact with objects. Potential for collisions. Some behaviours in the network will not be selectable.
Broken wireless tags	Wireless tag manager	New tags will not be added to network. Objects will be treated as tag less objects.
Failed Goals	Behaviour Network Manager	If a goal fails, then the network would have to reset the behaviours needed to achieve the goal. Could lead to a loop of behaviours failing to achieve the goal.
Failed behaviour / Action	Behaviour Network Manager	Actions can be failed, e.g. a robot could fail to pick up an object. Will need to reset the behaviour, to try again. Could lead to a loop of failing behaviours.
Failed module	Module specific	A module could fail completely and be unresponsive. Depending on the module the system could continue with limited functionality.
Failed action selection	Behaviour Network Manager	If the agent fails to make a decision on which action to take then the system will be locked in a loop and will not advance the world state.

Table 3-1 - Types of errors that can be exhibited by the architecture.

4. Behaviour Network

This chapter will provide a deeper explanation of behaviour networks. Following on from previous chapters which detail what a behaviour networks are and how they work, it will begin to address some issues found during the development process.

4.1. A Solution to the Division Rule

The division rule is a concept described in (Tyrrell, 1994), where the aim is to ensure that an appropriate amount of activation energy is distributed between the behaviours to allow appropriate behaviours to be selected and achieve goals. The division rule uses the number of input and outputs of the same type in order to calculate the proportion of energy to transfer. Tyrrell (Tyrrell, 1994) has demonstrated and explained the possible issues with the division rule as described in (Maes, 1991a) and has also shown potential solutions (also detailed in section 2.2.4.2.4). None of these solutions works perfectly as for each implementation (Tyrrell, 1994) there are scenarios in which that solution fails.

One problem resides in whether or not the calculation of energy spread should involve division by the number of predecessor and conflictor links. As an example, Figure 4-1 shows a possible scenario in which node 'B4' has three input links, two of which help to achieve the same goal. Tyrrell's approach would set M (the number of inputs) to three but this would lead to errors in the energy spreading calculations because nodes with low M would receive too little activation and can lose out against those with high M on occasions when it should be the preferred behaviour. A proposed solution is to store data in the links detailing which goal(s) each behaviour can achieve. This meta data (information about which goals behaviours contribute to) would allow for a more

accurate division rule as the division can then be based on either the number of goals a behaviour can achieve or the number of inputs to that behaviour.

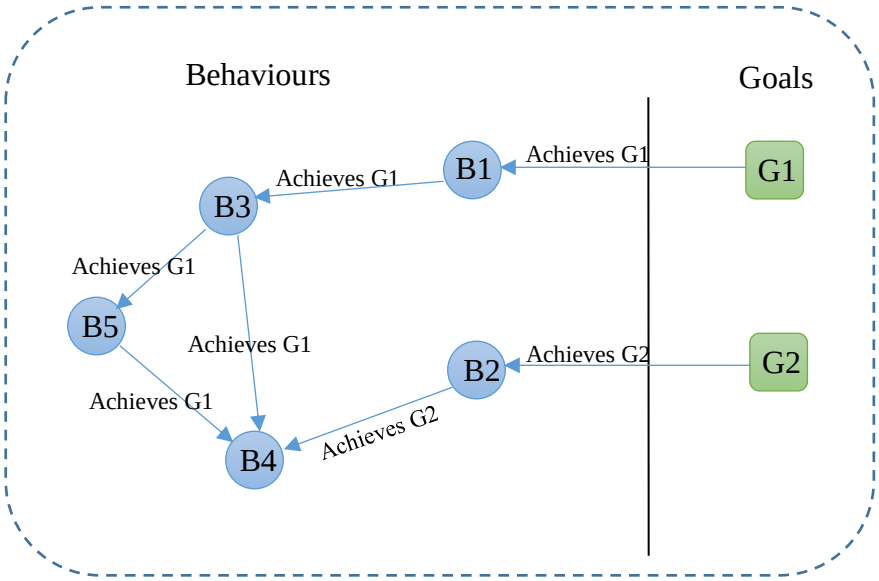

Figure 4-1 - Scenario where a node can have multiple inputs from different goals.

This diagram shows that behaviours in a behaviour network can help to achieve multiple goals. This is pertinent information for calculating how the flow of energy should be spread through the network.

If the links can be represented as objects with the following information (Figure 4-2). It is also easy to generate a visual representation of the behaviour network by linking the source and destination to behaviours in the network (Figure 4-3 shows this network automatically drawn from the metadata).

Link

Source = “B3”,
Destination = “B4”,
Goals = “G1”,
Type = “Predecessor”

Figure 4-2 - The data that can be stored in each link.

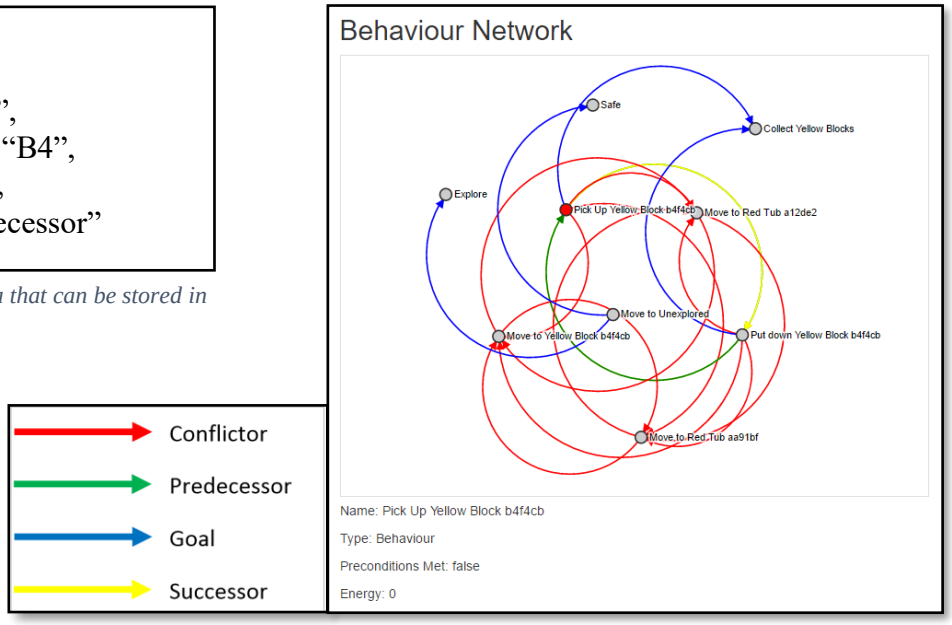

Figure 4-3 - An example of a behaviour network displayed automatically from metadata.

Other information that a link should have is the type of the link; this allows the algorithm to group links together before selecting the appropriate division rule. Finally, each link should store enough data that it is possible to determine to which goals it contributes. This will be useful information for the division rule of both predecessor and conflictor links. Using this information, the following algorithm was developed (Figure 4-4):

```
Algorithm
1: For each Link of type 'Goal' in Links
2:     Get all links of type 'Predecessor' and type 'Conflictor' where the link's target is the same as the goal link target
3:     For each link in the given query
4:         Call a recursive method passing in the link and goal information
5:
Method
6: Add goal information to Link
7: Find next link matching the source of the current link and the target of the next link
8: For each link in the given query
9:     Call this method passing in the next link and the goal information
```

Figure 4-4 - Pseudocode for embedding goal information to the links of a behaviour network

This algorithm will get the list of all available goal links and, for each goal link, it will look for other links, which are either of type predecessor or conflictor, and match the source of the link to the target of the other link. Then for every match a recursive method is called that will assign a goal to the link and search for the next link in the tree. Figure 4-1 shows an example of the algorithm passing goal information along the links between nodes.

It becomes clear to derive which goal(s) a behaviour can achieve by reading the data that is embedded in each link. Using that information, a more meaningful division rule can be applied for both predecessor input and conflictor input.

Predecessor input formula:

$$I_P = \frac{1}{2}\left(\alpha + \frac{\alpha}{MAX(M,G)}\right) \quad (4)$$

Conflictor input formula:

$$I_C = \frac{1}{2}\left(\alpha\frac{\delta}{\gamma} + \frac{\alpha}{MAX(M,G)}\frac{\delta}{\gamma}\right) \quad (5)$$

Global parameters:

(γ) A constant used to determine the weighting of goal inputs and predecessor links

(δ) A constant used to determine the weighting of protected goal inputs and conflictor links

Input Multipliers:

($\frac{\delta}{\gamma}$) Conflictor Links

Division Rule:

(G) Number of goals a behaviour can achieve

(M) Number of inputs of a given type

MAX(M, G) defines which division rule will be applied

In the new division rule, the use of the number of output (N) has been removed as Tyrrell (Tyrrell, 1994) detailed that there are no drawbacks in that alteration. If the behaviour can achieve multiple goals, then we divide by the number of goals that it can achieve. This will ensure that a behaviour that can achieve multiple goals will retain a strong proportion of energy as it will be preferred over other behaviours. If that behaviour is not appropriate then the environment nodes will reduce the energy in that behaviour significantly. Otherwise we divide by the number of inputs of the same type. If the division is high, then less energy will be passed through the network limiting any bias that may exist.

4.2. *Results of the Division Rule*

An example network was used to test the proposed division rule, shown in Figure 4-5. In this network, there are three goals and ten behaviours. Some simplified assumptions were made to test the proposed changes to the division rule: each of the links in the network are of the same type (predecessor), and the order of the nodes selected to spread energy is predefined. Section 4.3 explores the effects of different orders of energy spreading in a behaviour network.

The network in Figure 4-5 uses core principles from the examples in (Tyrrell, 1994) to show the different structures the network can have. In this network the three goals create small sub networks. Goal 1 creates a network with one behaviour having multiple inputs that originate from a single goal. Goal 2 creates a small chain of behaviours, with one behaviour having inputs that originate from two goals. Finally, Goal 3 creates another small chain of behaviours, with one behaviour taking a single input but indirectly achieves multiple goals.

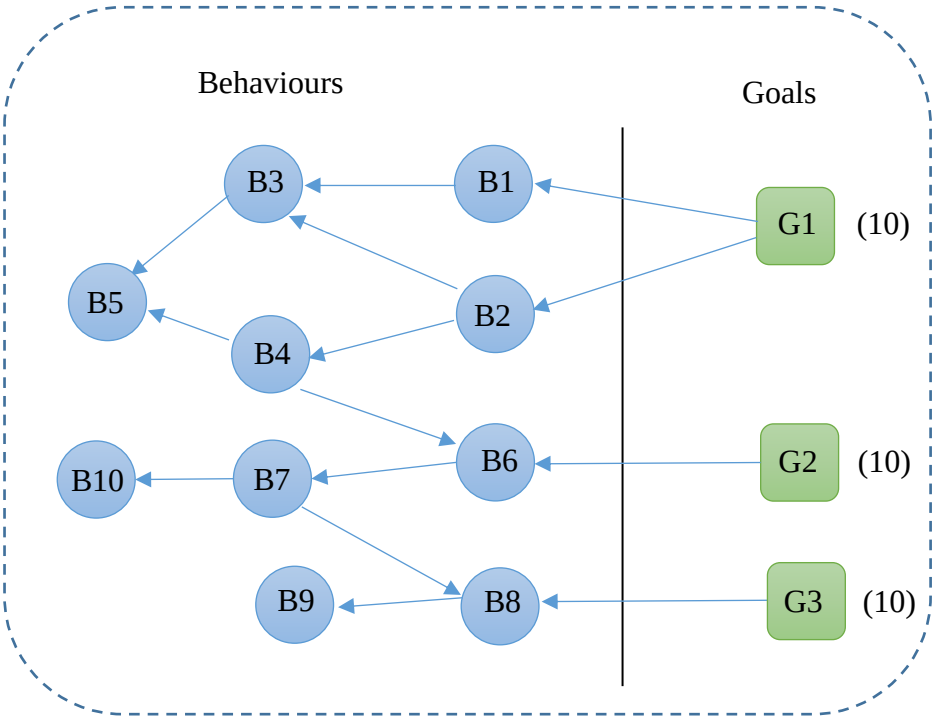

Figure 4-5 - A behaviour network to test the division rule.

For testing, each goal was given the same motivational value to pass into the network. The network ran for multiple iterations to show where the conversion of energy finishes. The first test used the same equations as in (Tyrrell, 1994) for spreading the energy between different behaviours and the second test used the proposed division rule. The results of the first test are shown in Figure 4-6.

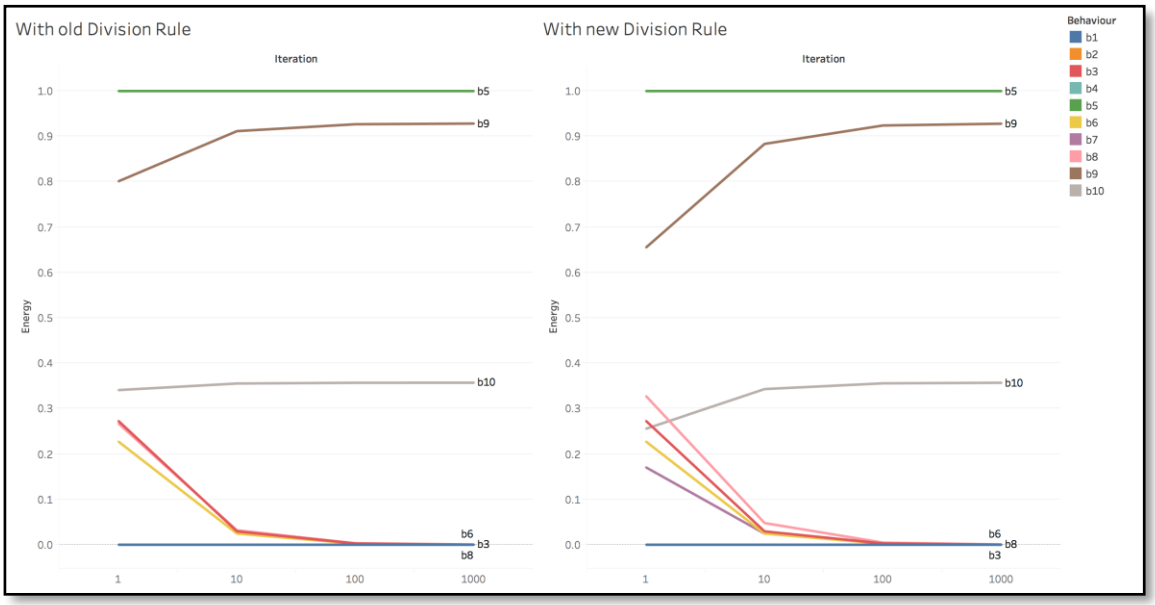

Figure 4-6 - Results from testing the original division rule and the new division rule for spreading energy in a behaviour network.

The chart on the left shows the energy in each behaviour over time (normalised with a min max scaler) using a traditional division rule. The chart on the right shows the energy in each behaviour over time (normalised with a min max scaler) using the proposed division rule.

Figure 4-6 shows the results from running the behaviour network (in Figure 4-5). This experiment used the same division rule as described in (Tyrrell, 1994). The chart on the left shows the value of energy in each behaviour over multiple iterations. The amount of energy in each behaviour was normalised with a range between 0 and 1 over each time step. It shows that behaviour (b5) starts and finishes with the highest distribution of energy, compared to the other behaviours in the network. The three behaviours with the largest amount of energy (b5, b9 and b10) are each the final behaviours of the respective goals in the network. These results are expected from this network as it is designed with a single link type and no environment nodes to affect the

flow of energy in the network. The chart also shows that over time, the distribution of energy gradually increases in those three behaviours with behaviour (b5) retaining the dominance over all other behaviours. The chart also shows that over time the distribution of energy in the network does not remain constant and that it needs multiple iterations of energy spreading to converge on the solution.

Figure 4-6 also shows the results from running the behaviour network (in Figure 4-5) with the updated division rule. This experiment applied the division rule as described in section 4.1. The chart again shows the value of energy in each behaviour over multiple iterations. It shows that behaviour (b5) and behaviour (b9) start with a wider range of values, behaviour (b5) also finishes with the highest distribution of energy. This is good as it shows network is consistent in that behaviour (b5) is best behaviour to select over the others after the first iteration of energy spreading. However, that does change after more iterations as the difference between the behaviours lessens. The three behaviours with the largest amount of energy are still the appetitive behaviours (b5, b9 and b10). The chart also shows that over time, the distribution of energy gradually increases in those three behaviours with behaviour (b10) retaining its dominance over all other behaviours. This again shows that over time the distribution of energy in the network is not constant and that over time the behaviours change position and settle after many iterations.

The results shown in Figure 4-6 show little improvement compared to the original approach also shown in Figure 4-6. However, the experiment was also successful in identifying the most important behaviour for the behaviour network in Figure 4-5. The following sections highlight the benefits and limitations of this new approach and the next section (Section 4.3) shall investigate further, energy spreading in a behaviour network.

4.2.1. Benefits of the new division rule

The new division rule was first mentioned in (Tyrrell, 1994) at a conceptual level. Tyrell concluded that in order to have an accurate division rule, each node would need to know exactly how many goals its inputs could achieve. The primary benefit of adding this metadata to the links of the behaviour network is that this information is now easily accessible. The metadata is also useful for debugging the behaviour network. There is no information on the evaluation side of behaviour networks, making it difficult to know whether the network is functioning correctly. The metadata makes it possible to check that the network has been correctly built at run-time.

4.2.2. Limitations of the new division rule.

The new division rule was developed to solve some of the issues described in (Tyrrell, 1994) and section 4.1. The results, however, have the disadvantage that at different time steps the behaviour network would select different behaviours. Only after multiple iterations have passed does the behaviour network decide upon the same behaviour consistently. This could be due to the sample behaviour network (in Figure 4-5) and a more complex network may be needed to better represent the changes. However, in the original design of the behaviour network, it would run for multiple iterations until the amount of energy in any node surpassed a set threshold. Meaning that the network would converge on the correct behaviour over time.

Another limitation to the new division rule is that the added overhead for adding this metadata to the links of the network may reduce performance. At each time step the behaviour network will re-create the links of the network. Once the network is created, it is passed to a recursive function that traverses through the entire network

adding the goal information to the links. This carries an inevitable overhead, which may limit the real-time responsiveness of a large network.

4.3. Energy Spreading in the Behaviour Network

During the implementation process of the behaviour network, some key issues were noted that have not been raised in any previous work. One such issue affects the process of spreading energy around the network. Tyrrell (Tyrrell, 1994) explains that energy must first enter the network via the goal nodes using the following formulas.

Energy moved from goal =

$$\gamma \frac{\alpha}{NM} \quad (6)$$

Energy moved from sensors=

$$\phi \frac{\alpha}{NM} \quad (7)$$

Global Parameters:

ϕ = A constant used to determine the weighting of environmental inputs and successor links

γ = A constant used to determine the weighting of goal inputs and predecessor links

Input Multipliers:

(ϕ) Environmental sensors

(γ) Goals

Division Rule:

(N) Number of outputs of a given type

(M) Number of inputs of a given type

The energy is passed between the nodes via the predecessor, successor and conflictor links. However, there is little discussion on the temporal order in which the behaviour nodes should pass energy between themselves. For any given network there may be many possible options for the order in which the energy should be spread and this can affect the outcome. This section will go into detail of the possible options followed by an analysis of the performance of these options. This section will also show the effects of selecting different temporal orders for spreading energy in a behaviour network.

Each of the following methods adopted a general set of rules prior to selecting the order that the behaviours should spread the energy between themselves. The external nodes, such as the goals and environment nodes, would spread energy into the network first before the internal behaviours could spread energy. This is to ensure that there is sufficient energy in the network to begin with. The links which increase other nodes energy levels (such as predecessor and successor links) were selected over inhibition nodes (such as protected goal and conflictor links) again to ensure that there was energy in the nodes prior to spreading the energy. The following examples show a behaviour network consisting of goal links and predecessor links to allow the network to be tested in more detail.

Tyrrell (Tyrrell, 1994) states that a behaviour network will run for multiple iterations of energy spreading until a behaviour's energy level has increased above a set threshold. The supporting text does not explain exactly how to determine the threshold feature. During the implementation process of the behaviour network, the first iteration of energy spreading it was found that depending on the order, the results could be different. This could yield undesired results either due to a loop or nodes not having all of the possible energy prior to spreading its energy. If the nodes do not have their full

allocation of energy prior to spreading, then an incorrect portion will be sent, and this would prevent the network from being in a stable situation. It is possible, however, that multiple iterations of the same sequence of nodes could average out the values and show different results. Table 4-1 shows that during one iteration of the spreading process that behaviour (B6) is preferable to behaviour (B8) however; Table 4-1 shows that after multiple iterations the energy value in the behaviours plateau making behaviour B8 more favourable than behaviour B6. Behaviours (B1) to (B4) end the iteration cycle with 0 energy as they send a full complement to the next behaviour.

After 1 iteration	**After 10 iterations**	**After 100 iterations**	**After 1000 iterations**
B1 (0)	B1 (0)	B1 (0)	B1 (0)
B2 (0)	B2 (0)	B2 (0)	B2 (0)
B3 (0)	B3 (0)	B3 (0)	B3 (0)
B4 (0)	B4 (0)	B4 (0)	B4 (0)
B5 (19.83)	B5 (171.45)	B5 (1696.16)	B5 (16943.22)
B6 (8.67)	B6 (65.16)	B6 (636.93)	B6 (6354.57)
B7 (16.00)	B7 (160.00)	B7 (1600.00)	B7 (16000.00)
B8 (7.50)	B8 (123.38)	B8 (1266.91)	B8 (12702.21)

Table 4-1 - Results from using a random order of behaviour nodes.

The order was: G1, G2, B1, B2, B3, B4, B8, B5, B6

This table shows the results from spreading activation energy through the behaviour network shown in Figure 4-6. The energy spreading process was performed multiple times to test whether the number of iterations made any difference to the results.

Figure 4-7 shows the normalised distribution of energy in the network over time (iterations). It shows that during the early iterations of the energy spreading process, the network has not yet converged on a solution.

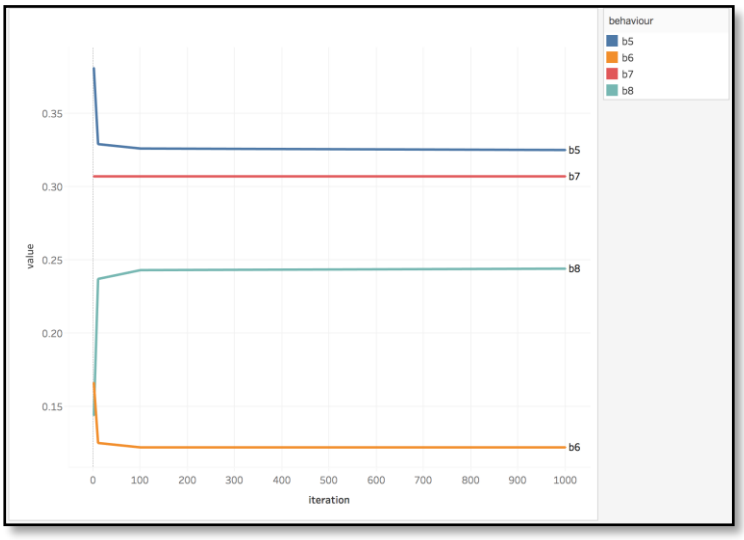

Figure 4-7 - Results from using an arbitrary order of behaviour nodes. The order was: G1, G2, B1, B2, B3, B4, B8, B5, B6

Two different approaches were tested. The first approach was to take a random order of the behaviour nodes and spread the energy from the selected node to the next. Figure 4-8 shows a potential situation for the behaviour network and where this method yields unsatisfactory results. Here if behaviour node (B6) is selected to spread energy to behaviour node (B5) and (B6) has a value of 5, then following the formula:

$$I = \frac{1}{2}\left(\alpha + \frac{\alpha}{MAX(M,G)}\right) \quad (8)$$

It will move 2.5 to B5. However; if behaviour node (B6) has not received all of its inputs from other behaviours in the system then it will be moving an incorrect portion of energy to the receiving nodes. For example, the behaviour B6 may have received some energy from B2 and sent a portion of energy to B5, however; it may not have received some energy from B3 meaning that an incorrect portion of energy had been sent.

Table 4-1, Table 4-2, and Table 4-3 each show that the order of the behaviour nodes, directly affects how much energy remains in certain behaviours after each iteration. Each of the random selections were simulated over multiple iterations to test whether the values would converge if the energy was spread around a number of times. This proved to be inconclusive as the final behaviour to be selected in each was trapped in a loop between behaviours B5, B6 and B8. Here the initial choice of behaviour directly affected which behaviour would have the most energy at any given number of iterations. For this test, there was no correct or incorrect behaviour to select. The aim is to show that the different behaviours are selected depending on the order that energy is spread.

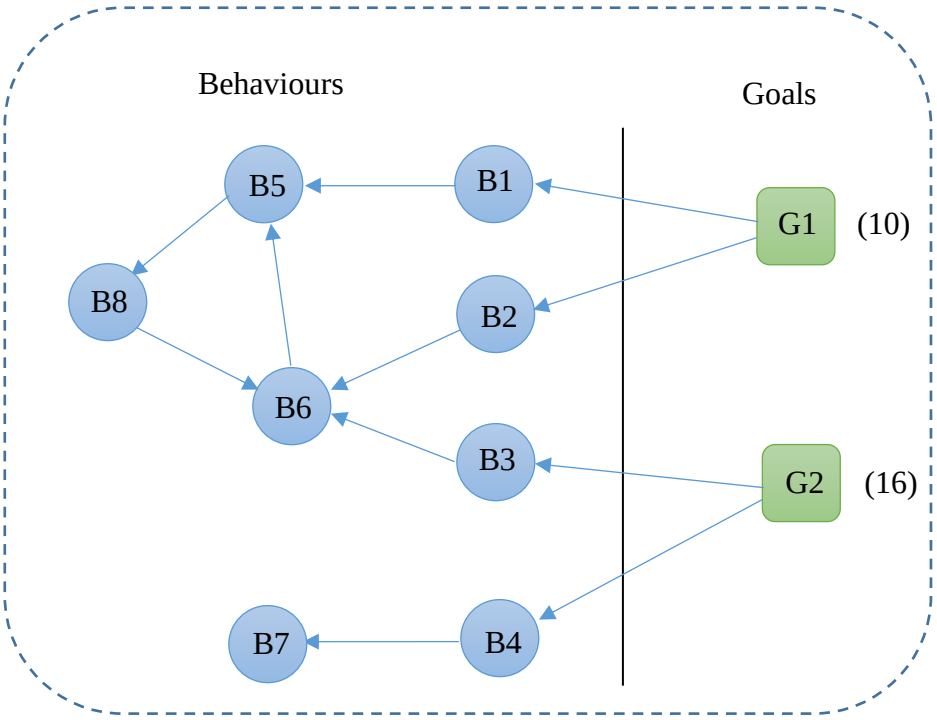

Figure 4-8 - An example behaviour network consisting of two goals and eight behaviours, each connected with either goal or predecessor links.

This figure displays a basic behaviour network that was used to test the energy spreading mechanism. This behaviour network also includes a feedback loop to show a potential flaw in current behaviour networks.

After 1 iteration	After 10 iterations	After 100 iterations	After 1000 iterations
B1 (0)	B1 (0)	B1 (0)	B1 (0)
B2 (0)	B2 (0)	B2 (0)	B2 (0)
B3 (0)	B3 (0)	B3 (0)	B3 (0)
B4 (0)	B4 (0)	B4 (0)	B4 (0)
B5 (19.83)	B5 (171.45)	B5 (1696.16)	B5 (16943.22)
B6 (16.17)	B6 (188.55)	B6 (1903.84)	B6 (19056.78)
B7 (16.00)	B7 (160.00)	B7 (1600.00)	B7 (16000.00)
B8 (0.00)	B8 (0.00)	B8 (0.00)	B8 (0.00)

Table 4-2 - Results from using an arbitrary order of behaviour nodes. The order was: G1, G2, B1, B2, B3, B4, B5, B6, B8

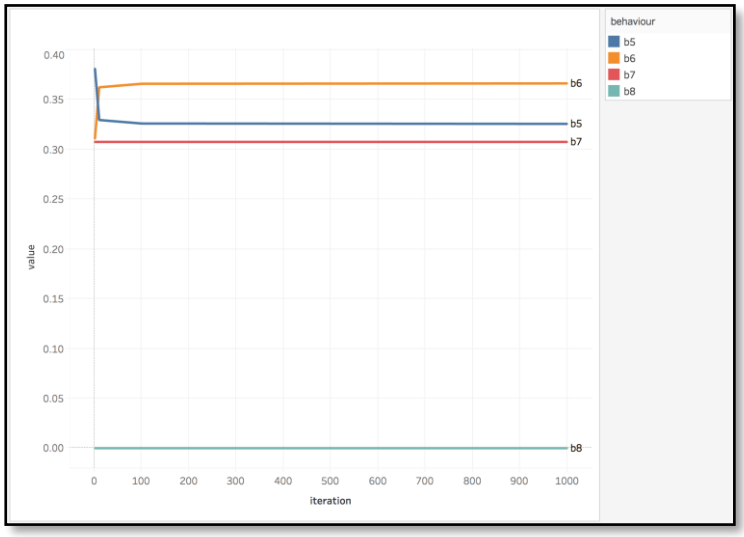

Figure 4-9 - Results from using an arbitrary order of behaviour nodes. The order was: G1, G2, B1, B2, B3, B4, B5, B6, B8

Table 4-2 shows the results from spreading activation energy through the behaviour network shown in Figure 4-8. The energy spreading process was performed multiple times to test whether the number of iterations made any difference to the convergence of the energy. Figure 4-9 shows the normalised distribution of energy in the network over time (iterations). It shows that during the early iterations of the energy spreading process, similarly to the previous result, the network has not yet converged on a solution. However, unlike the previous result the convergence of the energy has finished on different behaviours. In this order, behaviour (B8) concludes the iteration cycle with 0 energy because it is the last behaviour selected to send energy.

After 1 iteration	After 10 iterations	After 100 iterations	After 1000 iterations
B1 (0)	B1 (0)	B1 (0)	B1 (0)
B2 (0)	B2 (0)	B2 (0)	B2 (0)
B3 (0)	B3 (0)	B3 (0)	B3 (0)
B4 (0)	B4 (0)	B4 (0)	B4 (0)
B5 (6.83)	B5 (65.77)	B5 (654.86)	B5 (6545.77)
B6 (8.67)	B6 (96.93)	B6 (980.56)	B6 (9816.93)
B7 (16.00)	B7 (160.00)	B7 (1600.00)	B7 (16000.00)
B8 (20.50)	B8 (197.31)	B8 (1964.58)	B8 (19637.31)

Table 4-3 - Results from using a random order of behaviour nodes. The order was: G1, G2, B1, B2, B3, B4, B6, B8, B5

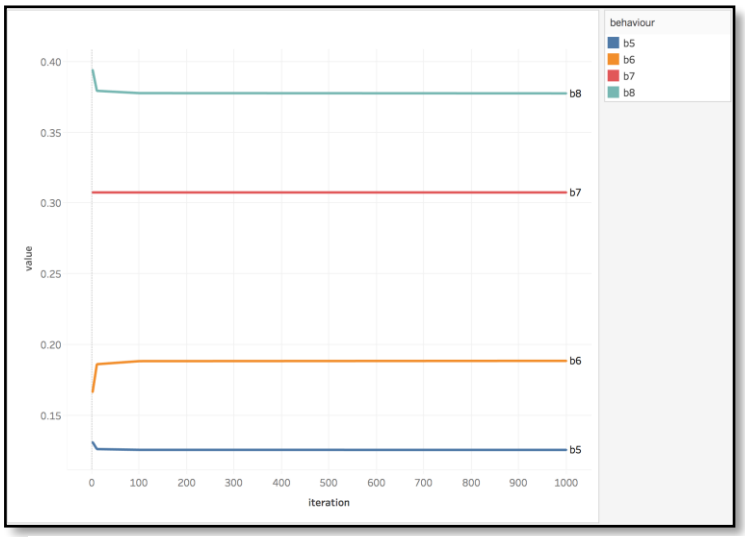

Figure 4-10 - Results from using a random order of behaviour nodes. The order was: G1, G2, B1, B2, B3, B4, B6, B8, B5

Table 4-3 shows the results from spreading activation energy through the behaviour network shown in Figure 4-8. The energy spreading process was performed multiple times to test whether the number of iterations made any difference to the convergence of the energy. Figure 4-10 shows the normalised distribution of energy in the network over time (iterations). It shows that regardless of the number of iterations of the energy spreading process, the result stays constant and the energy converges on the same solution. However, unlike the previous results the convergence of the energy has finished on different behaviours.

The second approach of selecting the order of the behaviours was to spread energy from each connected behaviour in sequence following the links from each node. Inspired by how depth first search can be used to search a given tree, the behaviours were selected by following links to the end of the tree and then backtracking to an unexplored node. For example, if behaviour B5 was selected to spread energy then the next behaviour would be B8 followed by B6. This soon demonstrated the same problem as mentioned previously: the system does not know whether a behaviour has received all its inputs or not before sending energy to the next node.

After 1 iteration	After 10 iterations	After 100 iterations	After 1000 iterations
B1 (0)	B1 (0)	B1 (0)	B1 (0)
B2 (0)	B2 (0)	B2 (0)	B2 (0)
B3 (0)	B3 (0)	B3 (0)	B3 (0)
B4 (0)	B4 (0)	B4 (0)	B4 (0)
B5 (24.83)	B5 (260.35)	B5 (2616.71)	B5 (26180.35)
B6 (11.17)	B6 (99.65)	B6 (983.29)	B6 (9819.65)
B7 (16.00)	B7 (160.00)	B7 (1600.00)	B7 (16000.00)
B8 (0)	B8 (0.00)	B8 (0.00)	B8 (0.00)

Table 4-4 - Results from using the second approach. The order was: G1, G2, B1, B2, B3, B4, B5, B8, B6

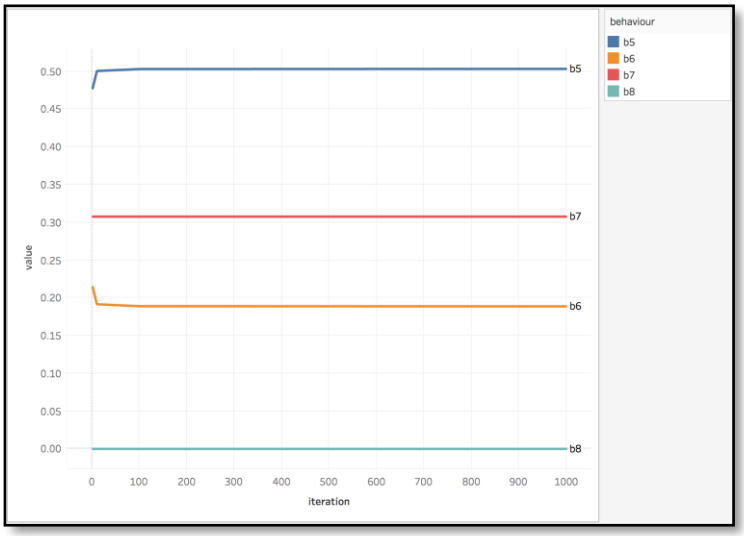

Figure 4-11 - Results from using the second approach. The order was: G1, G2, B1, B2, B3, B4, B5, B8, B6

The results from this approach are shown in Table 4-4 and Figure 4-11. These results were similar to one of the random selection approaches. It is clear from the results, from the different methods of spreading energy, that the order of nodes makes a difference in which behaviour accumulates the most energy. It is worth also noting that the behaviour network in Figure 4-8 contains a loop amongst nodes B5, B8 and B6. Here, each behaviour feeds into one another which, makes the order of the nodes a determining factor for the selected behaviour. The other determining factor would be the number of iterations that are run before the behaviour networks stops and selects a behaviour for execution. The purpose of the threshold feature would appear to be to have a fixed point when the iteration process stops. The aim here is to showcase that there are many different approaches to spreading energy through a behaviour network and show that the results from these methods differ and that a solution is required.

To solve the problem of potential loops and to ensure that the order of nodes does not determine where the energy converges. One solution is to ensure that each node cannot send its energy through the network unless it has received all of its input energy. However, the given situation in Figure 4-6 has a loop amongst behaviours B5, B6 and B8. Each of these nodes has an input which must first be executed prior to that node being able to spread energy. It is thus impossible to determine when one of these nodes has received all its inputs. The concept of using energy packets to spread energy between nodes was implemented to solve this problem.

4.3.1. Energy Packets

To compensate for the situations where a feedback loop occurs or for when the system needs to know whether a node has received its inputs, a new mechanism for spreading the energy between the nodes was devised. When a behaviour needs to send its energy

to the next behaviour it will create a packet to send (Figure 4-12). A packet contains data about itself: a list of all of the previous behaviours that it has travelled to, the target behaviour that it is about to travel to, its source and the amount of energy it contains.

Storing data relating to the locations where the packet has been (the sources list) allows the behaviours to know where the packet has been and where it originated from. This ensures that a packet of data will never contribute energy to the behaviour that it originated from and by doing so will prevent any loops from occurring.

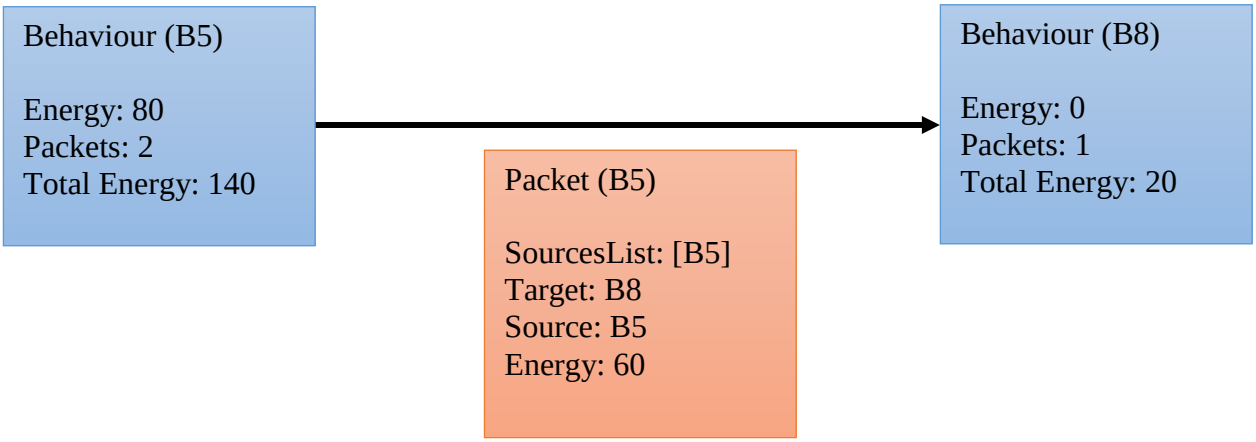

Figure 4-12 - The contents of a behaviour and a packet

The behaviour nodes are also modified to accept energy in the packets. Each node records its energy, a list of all of the packets that it has received and a total energy which is a sum of its energy and the energy of all of the packets that it contains.

When a behaviour needs to send energy, it will create a packet and put a proportion of its own energy into the packet (defined in the previous section). The packet will then travel along a link of the same type (e.g., predecessor). When it arrives at a new behaviour the packet will be stored in that behaviours list of packages. The node will then check if it needs to send energy farther (defined by the outputs of the same type) and, if so, will create another packet and send a proportion of its own energy to the next behaviour but only if that behaviour is not in the list of behaviours which have contributed to that packet. This process of creating and sending packets will

continue until the packet cannot travel any further (e.g., it does not have any output of the same type).

4.3.3.1. Experiment 1

The first experiment was to test the data packet approach in the behaviour network from the previous example (Figure 4-5). This experiment will show how the data packet approach can handle problems such as loops and the order of selection for energy spreading that the traditional behaviour network could not deal with. The behaviour network in Figure 4-8 was used with the data packet approach and the results are shown in Table 4-5 and Figure 4-13.

After 1 iteration	After 10 iterations	After 100 iterations	After 1000 iterations
B1 (0)	B1 (0)	B1 (0)	B1 (0)
B2 (0)	B2 (0)	B2 (0)	B2 (0)
B3 (0)	B3 (0)	B3 (0)	B3 (0)
B4 (0)	B4 (0)	B4 (0)	B4 (0)
B5 (6.83)	B5 (108.29)	B5 (1105.22)	B5 (11074.4)
B6 (16.17)	B6 (164.68)	B6 (1660.06)	B6 (16613.83)
B7 (16.00)	B7 (160)	B7 (1600)	B7 (16000)
B8 (13)	B8 (87.03)	B8 (834.72)	B8 (8311.61)

Table 4-5 - Results from using the packet approach in experiment 1.

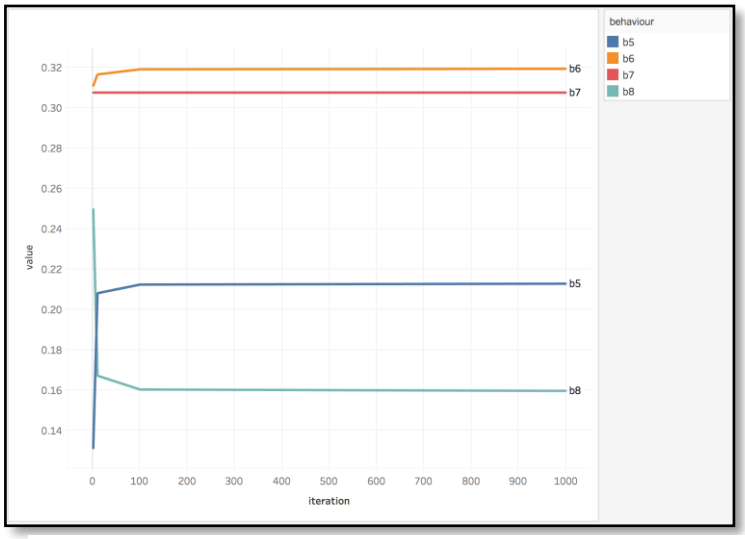

Figure 4-13 - Results from using the packet approach in experiment 1.

This experiment was tested multiple times with a variety of different orders and the results remained constant. In the experiment above, instead of selecting a set order for the behaviours to spread energy, each iteration used a different randomly selected order. This was done to show that this new approach is impervious to the problems caused by the executing order. It shows that a loop in a system can be tolerated and that it does not matter if a behaviour has received all of its inputs before sending energy out. The outcome of these results show that behaviour B6 is the best behaviour to execute, this is correct as behaviour B6 would help two behaviours (B2 and B3) become executable and can aid in achieving two different goals.

The packet approach was able to demonstrate some positive results, however; this was shown using only predecessor and goal links in the network. The next test was to implement this approach with actual behaviours and multiple different types of links. This test was run multiple times and similar results were presented.

4.3.3.2. Experiment 2

The second experiment was to test the data packets in a behaviour network with a simple real-world scenario. This scenario is shown in Figure 4-14 where the agent has three goals; 'Avoid Obstacles', 'Explore' and 'Collect Yellow Blocks'. The aim of this experiment is to track the data packets as they travel through the network, to ensure that the correct amount of energy is left in each behaviour after each iteration and to observe any erroneous behaviour. In this experiment, it is assumed that the robot only has one hand to pick up objects with. If the hand is full (holding an object) then the robot cannot pick up any other object.

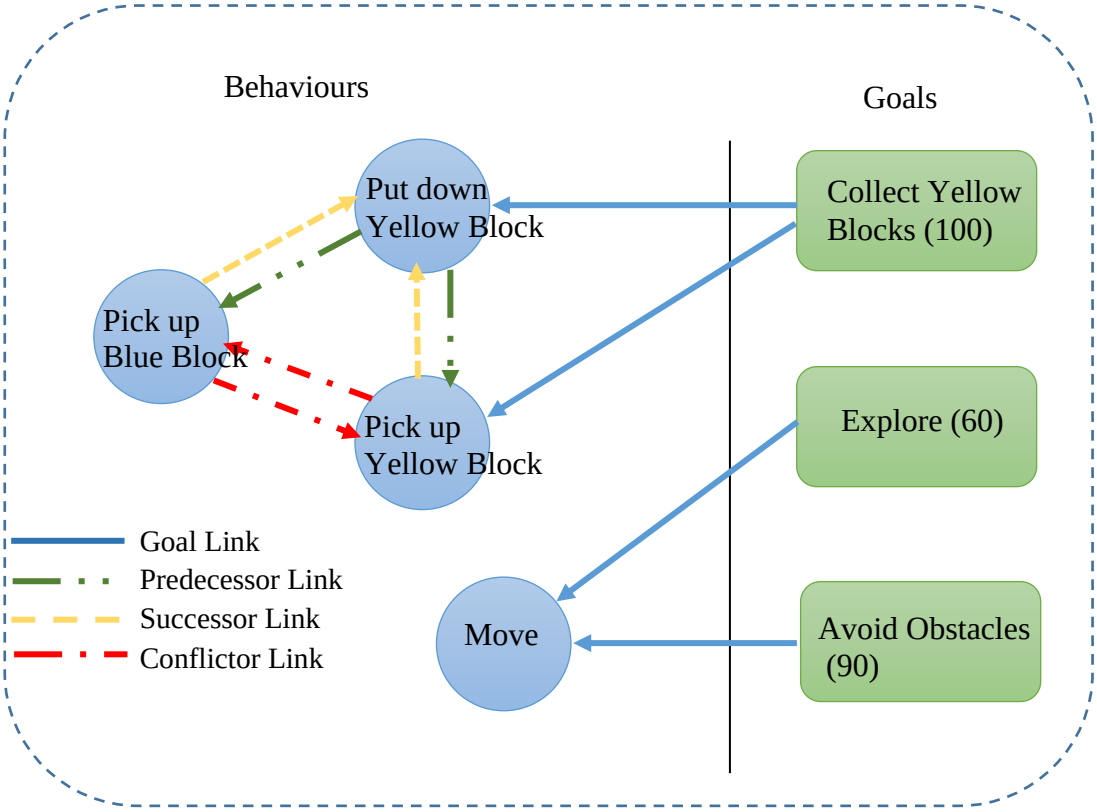

Figure 4-14 - An example real world situation shown as a behaviour network consisting of three goals and four behaviours, each connected with different link types.

Here there are three goals a robot could have; 'Explore' will have the robot explore and learn its environment and this will be achieved with the 'Move' behaviour. The 'Avoid Obstacles' goal will ensure the robot can avoid obstacles, again with the 'Move' behaviour. Finally, the 'Collect Yellow Blocks' goal would have the robot find and move Yellow blocks to a different location. 'Avoid Obstacles' and 'Explore' both have a behaviour 'Move' which will have the robot move in a random direction. 'Pick up Yellow Block', 'Put down Yellow Block' allow the robot to interact with the yellow block to achieve the goal of 'Collect Yellow Blocks'. Finally; 'Pick up Blue Block' was added to allow for conflict in the system. The situation of the environment is that the robot is in front of a Yellow Block (ready to be picked up), the robot's hand is empty, and it is too far from a Blue block to interact with it. Using these conditions, the appropriate links are used to connect the behaviours, shown in Figure 4-14.

The packet approach was applied to this network. The first implementation followed the same approach as (Tyrrell, 1994) where energy was sent into the network first via the goal links. The predecessor links were then used to send energy around the network followed by successor links and concluding with inhibiting energy following the conflictor links. The results from this implementation are shown in Table 4-6.

Behaviour	Energy	Total Energy
Move	120	120
Pick up Yellow Block	20	60
Put down Yellow Block	0	60
Pick up Blue Block	0	27.5

Table 4-6 - Results from using the packet approach with different types of links.

The results in Table 4-6 show that the system believes that the best action to take would be to perform the 'Move' behaviour. This is because of the behaviour achieves two different goals and is separated from the other behaviours. This means that the 'Move' behaviour does not need to pass its energy to any other behaviour and that it will not be inhibited either. This is a problem if the other systems in the behaviour network are what are preferred for the robot to do, more than exploring the environment. This is not a problem with the network but in motivation (amount of energy in the goals) of 'Explore' and 'Avoid Obstacles'. These would need to be reduced or have some other inhibition setting that could be applied to it.

The results in Table 4-6 also show that both the 'Pick up Yellow Block' and the 'Put down Yellow Block' finished with the same amount of energy even though the most appropriate behaviour would be 'Pick up Yellow Block' given the situation the robot is in. By included a threshold (Tyrrell, 1994), it was observed that the behaviour network was able to select a behaviour after a single iteration. This is because multiple iterations are only needed if the threshold hold is not hit and the network need more energy to converge on a solution.

To test whether the reasoning for these results is due to the order in which the energy is passed via the links (the results in Table 4-6 followed the order, predecessor, successor and conflictor) the same experiment was executed with a different order. Table 4-7 then shows the results for when this order is changed to; successor, predecessor and conflictor.

Behaviour	Energy	Total Energy
Move	120	120
Pick up Yellow Block	20	60
Put down Yellow Block	0	60
Pick up Blue Block	0	27.5

Table 4-7 - Results from using the packet approach with different types of links.

Table 4-7 shows the same results as in Table 4-6, showing that the order that the links are executed in do not make any difference to the end results. The method that the behaviours use to pass energy to other behaviours is to send a proportion of energy based on its current energy level. This could explain the unsatisfactory results as when a behaviour sends energy to another behaviour via a predecessor link, it is reducing the amount of energy that behaviour has. When that same behaviour then has to send energy via a different link type it may find that its energy level is lower than expected or empty. For example; behaviour 1 has an energy of 50 and it sends 40 energy to behaviour 2 via a predecessor link. Behaviour 1 then has 10 energy left for when it needs to send energy to behaviour 3 via a successor link. However; behaviour 1 could have also received a packet of energy from another behaviour so making behaviour 1 have 10 energy and a packet of 30 energy. The current solution of the system has the behaviour only send energy from its own source and not from any packets it may have received. This worked fine when there was only one type of link to consider (shown in Table 4-5) but when multiple links are introduced the solution needs to be modified.

The next approach involved merging the energy in each of the packets with the remaining energy stored in each behaviour for each type of link. For example, following from the previous example, behaviour 1 has sent 40 of its 50 energy to behaviour 2 via a predecessor link and it has received a packet from another behaviour. Previously it would create a packet using the remaining 10 energy and send that via a successor link, instead, before it creates a new packet it will merge the energy in its current list of packets with the 10 energy it has remaining and send that proportion.

Behaviour	Energy
Move	120
Pick up Yellow Block	23.75
Put down Yellow Block	120
Pick up Blue Block	-8.75

Table 4-8 - Results from using the packet approach with merging using different types of links.

Table 4-8 shows the results from using this approach with a link order of; predecessor, successor and conflictor. It shows that the 'Pick up Blue Block' has a negative value, this value is accurate as it is a behaviour that does not benefit the system. The 'Put down Yellow Block' has the most energy with 120, even though it is a behaviour that cannot be executed based on the current situation that the robot is in. Finally; the 'Pick up Yellow Block' has a value of 23.75, which again based on the current situation is incorrect. The order of the links was then changed to successor, predecessor and conflictor, and the results are shown in Table 4-9.

Behaviour	Energy
Move	120
Pick up Yellow Block	46.25
Put down Yellow Block	0
Pick up Blue Block	13.75

Table 4-9 - Results from using the packet approach with merging using different types of links.

Table 4-9 shows that the results from using the packet approach with energy merging following the link order of; successor, predecessor and conflictor. Here the 'Pick up Blue Block' behaviour has a value of 13.75 which is low in comparison to the values in other behaviours. This means that this behaviour would be unlikely to be chosen for activation. The 'Put down Yellow Block' behaviour has a value of 0, which again is correct given the current situation the robot is in. Finally, the 'Pick up Yellow Block' behaviour has the most energy in that subsystem with a value of 46.25, making this the most likely behaviour to be selected. It is worth noting that the 'Move' behaviour has a value of 120 as in this implementation the weighting for the 'Explore' and 'Avoid Obstacles' goal had not been changed.

In summary, this experiment has tested the data packet approach in a real-world scenario with multiple link types. It was demonstrated that the energy packets were successful in navigating the network. This experiment was run multiple times with different parameters and the overall results showed that packet merging with a fixed link order yielded more domain accurate decisions. The network in Figure 4-14 did not make use of environment nodes and these will be included in the later experiments to further test this approach.

4.3.3.3. Experiment 3

The third experiment extends the concepts implemented in experiment 1 and experiment 2 to further test the data packets in a more complex situation. The aim is to test a situation in which the order of the tasks that the agent needs to execute is important. This is to ensure that the agent is selecting the most appropriate behaviours to achieve its goals. The situation (shown in Figure 4-15) is a behaviour network with the goal to bake a cake. Logically, baking a cake involves multiple steps, some of which need to be executed in a particular order and others which can be executed at any time. This experiment tests whether using data packets enables selecting the correct order of behaviours to execute.

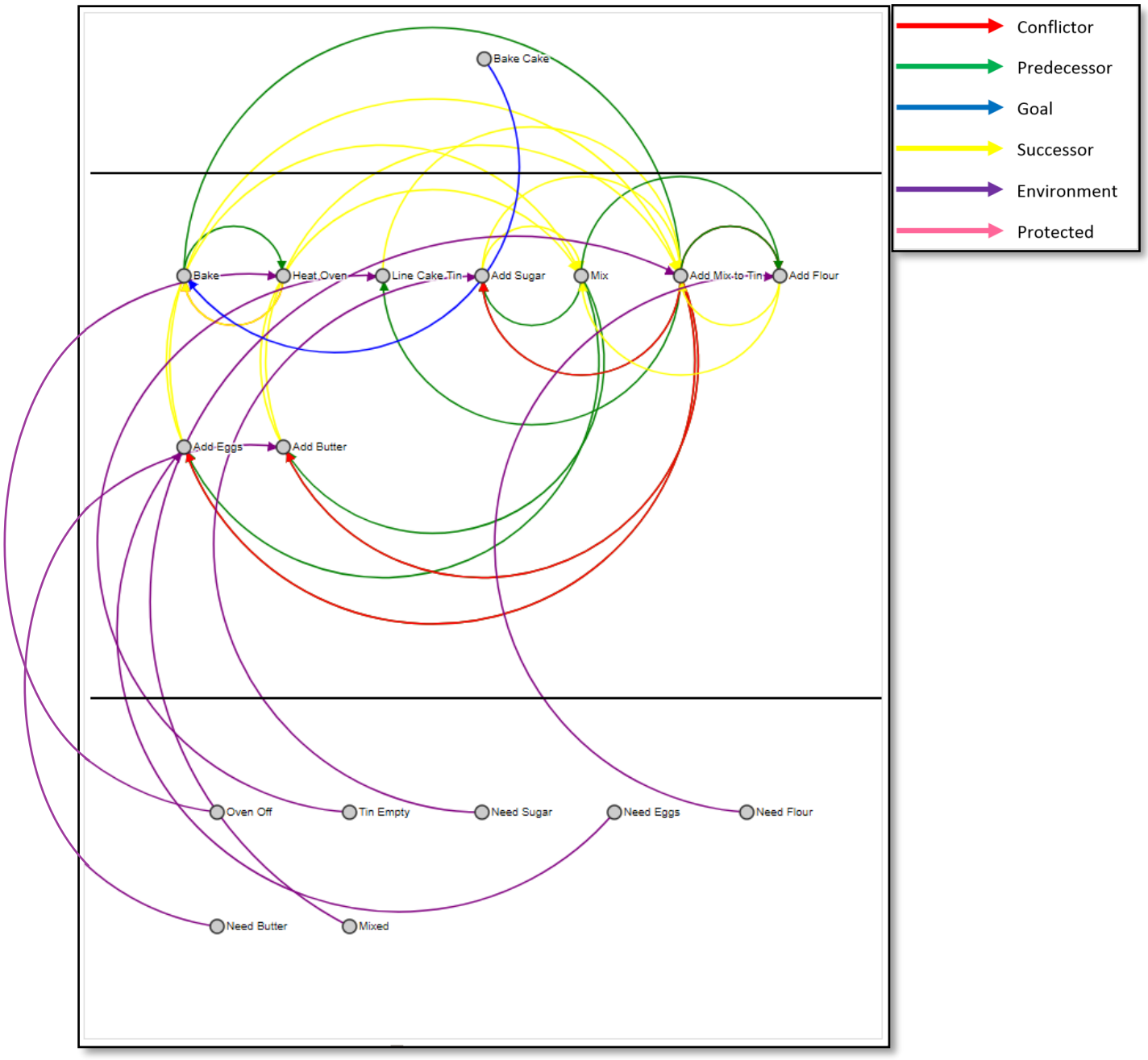

Figure 4-15 - Behaviour Network - Experiment 3

Table 4-10, Table 4-11 and Table 4-12 show the individual properties of the behaviours that are included in the network shown in Figure 4-15. Here the agent has one consummatory node 'Bake', which achieves the goal of 'Bake Cake'. It also has six appetitive nodes, which help to make the 'Bake' behaviour executable. The agent can execute the 'Heat Oven' and 'Line Cake Tin' behaviour at any time without greatly affecting the outcome of the goal. The 'Mix' behaviour is an appetitive behaviour for the 'Add mix to tin' behaviour and should logically be selected after one of the 'Add Eggs', 'Add Flour', 'Add Butter' or 'Add Sugar' behaviours have executed. This behaviour achieves the precondition 'Mix in Tin' (The precondition needed to achieve the behaviour 'Bake'). If the sequence that the behaviours is executed in are different to the desired sequence, then the end result would be a failed bake. The chronological sequence for the expected order of behaviours is:

- Heat Oven (any time)
- Line Cake Tin (any time)
- Add Sugar → Mix
- Add Butter → Mix
- Add Flour → Mix
- Add Eggs → Mix
- Add Mix to Tin (only after adding ingredients, mixing and lining tin)
- Bake (final behaviour)

The Heat Oven and Line Cake Tin behaviour may be selected in any order as both of these behaviours have an immediate contribution to the goal. For this experiment a successful cake can be baked regardless of the order of the added ingredients and it is expected that the order of added ingredients will depend on which the agent locates first. The Mix behaviour should be selected after each ingredient is added. Once all ingredients are added then the Add Mix to Tin behaviour should be selected followed by the final Bake behaviour to achieve the goal.

Behaviour	Precondition List
Heat Oven	Oven Off
Line Cake Tin	Tin Empty
Add Sugar	Need Sugar
Add Eggs	Need Eggs
Add Flour	Need Flour
Add Butter	Need Butter
Mix	Unmixed
Add Mix to Tin	Tin Lined, Sugar, Flour, Eggs, Butter, Mixed
Bake	Mix in Tin, Oven On

Table 4-10 - Precondition list for Experiment 3

Behaviour	Add List
Heat Oven	Oven On
Line Cake Tin	Tin Lined
Add Sugar	Unmixed, Sugar
Add Eggs	Unmixed, Eggs
Add Flour	Unmixed, Flour
Add Butter	Unmixed, Butter
Mix	Mixed
Add Mix to Tin	Mix in Tin
Bake	Bake Cake

Table 4-11 - Add list for Experiment 3

Behaviour	Delete List
Heat Oven	Oven Off
Line Cake Tin	Tin Empty
Add Sugar	Mixed, Need Sugar
Add Eggs	Mixed, Need Eggs
Add Flour	Mixed, Need Flour
Add Butter	Mixed, Need Butter
Mix	Unmixed
Add Mix to Tin	Mixed
Bake	*None*

Table 4-12 - Delete list for Experiment 3

This scenario does not take into account the lower level atomic actions that the agent would need to perform, but instead focuses on the high level behaviours needed to achieve the goal. For example, in order to add eggs to the cake mix the agent would need to select actions to position a robot arm near an egg and to apply the correct amount of pressure to crack the egg. Work has been done to incorporate a hierarchy of behaviour networks to solve a particular task (Nicolescu and Matarić, 2002). This concept could be used to extend this scenario, so when the agent selects a high level behaviour to execute a new behaviour network would be created with that high level behaviour as the goal and some low level atomic actions as the behaviours.

Table 4-143 shows the first scenario in experiment 3. Here the preconditions have been met for behaviours; 'Heat Oven', 'Line Cake Tin' creating the network shown in Figure 4-15. Each of these behaviours is a valid action to begin the process of baking a cake. Table 4-134 shows the results from the first energy spreading process. Here the agent believes that the most appropriate behaviour to execute is the 'Heat Oven' behaviour, followed by behaviours; 'Mix' and 'Line Cake Tin'. The agent selected the 'Add' ingredient behaviours to be the most inappropriate behaviours to execute at this time. The baking a cake example was selected as there is a sequence of behaviours that are expected to be followed. Currently the agent is selecting an expected order of behaviours.

World State	Values
Oven On	False
Oven Off	True
Tin Empty	True
Tin Lined	False
Mixed	True
Unmixed	False
Have Sugar	False
Have Flour	False
Have Eggs	False
Have Butter	False
Mix in Tin	False

Table 4-13 - Experiment 3, Scenario 1

Behaviour	Energy
Bake	-11.5
Heat Oven	15.5
Line Cake Tin	5.9
Add Sugar	2.9
Mix	8
Add Mix to Tin	12
Add Flour	2.9
Add Eggs	2.9
Add Butter	2.9

Table 4-14 - Experiment 3, Scenario 1 - Results

Based on the results from scenario 1 (Table 4-14), the agent executes the 'Heat Oven' behaviour turning on the oven and meeting one of the preconditions for the 'Bake' behaviour. Behaviours; 'Mix' and 'Add Mix to Tin' may have the most energy but their preconditions are not met. These behaviours pass a portion of their energy to the appetitive behaviours that will meet their preconditions. This leads to scenario 2 shown in Table 4-15 and also the results of the next round of energy spreading in Table 4-16. This shows that behaviour 'Line Cake Tin' is now the most appropriate behaviour for the agent to execute, followed by the add ingredient behaviours.

Preconditions	Values
Oven On	**True**
Oven Off	**False**
Tin Empty	True
Tin Lined	False
Mixed	True
Unmixed	False
Have Sugar	False
Have Flour	False
Have Eggs	False
Have Butter	False
Mix in Tin	False

Table 4-15 - Experiment 3, Scenario 2

Behaviour	Energy
Bake	0
Heat Oven	0
Line Cake Tin	15.46
Add Sugar	9.56
Mix	17.06
Add Mix to Tin	27.60
Add Flour	9.56
Add Eggs	9.56
Add Butter	9.56

Table 4-16 - Experiment 3, Scenario 2 - Results

The agent now selects the 'Line Cake Tin' behaviour (based on the results from Table 4-16) which lines the cake tin ready for the mixture and meets a precondition of 'Add Mix to Tin' behaviour. This leads to scenario 3 shown in Table 4-17 and also the results of the next round of energy spreading in Table 4-18. This shows that the agent believes that the best option is one of the add ingredient behaviours.

World State	Values
Oven On	True
Oven Off	False
Tin Empty	False
Tin Lined	**True**
Mixed	True
Unmixed	False
Have Sugar	False
Have Flour	False
Have Eggs	False
Have Butter	False
Mix in Tin	False

Table 4-17 - Experiment 3, Scenario 3

Behaviour	Energy
Bake	0
Heat Oven	0
Line Cake Tin	0
Add Sugar	11.25
Mix	8
Add Mix to Tin	8
Add Flour	11.25
Add Eggs	11.25
Add Butter	11.25

Table 4-18 - Experiment 3, Scenario 3 - Results

Based on the results from Table 4-18, the 'Add Sugar' behaviour is now selected for execution. This starts the chain of behaviours needed to create the cake mix. Executing this behaviour leads to scenario 4 shown in Table 4-19 and also the results of the next round of energy spreading in Table 4-20.

World State	Values
Oven On	True
Oven Off	False
Tin Empty	False
Tin Lined	True
Mixed	True
Unmixed	False
Have Sugar	**True**
Have Flour	False
Have Eggs	False
Have Butter	False
Mix in Tin	False

Table 4-19 - Experiment 3, Scenario 4

Behaviour	Energy
Bake	0
Heat Oven	0
Line Cake Tin	0
Add Sugar	0
Mix	13.25
Add Mix to Tin	16
Add Flour	13.25
Add Eggs	13.25
Add Butter	13.25

Table 4-20 - Experiment 3, Scenario 4 - Results

Table 4-20 shows that currently the most appropriate behaviour to execute is the 'Mix' behaviour. The agent then follows the chain of adding ingredients and performing the mix behaviour until all of the ingredients have been added and mixed. This leads to scenario 5 shown in Table 4-21 and also the results of the next round of energy spreading in Table 4-22.

World State	Values
Oven On	True
Oven Off	False
Tin Empty	False
Tin Lined	True
Mixed	**True**
Unmixed	False
Have Sugar	**True**
Have Flour	**True**
Have Eggs	**True**
Have Butter	**True**
Mix in Tin	False

Table 4-21 - Experiment 3, Scenario 5

Behaviour	Energy
Bake	24
Heat Oven	0
Line Cake Tin	0
Add Sugar	-11.25
Mix	0
Add Mix to Tin	45
Add Flour	-11.25
Add Eggs	-11.25
Add Butter	-11.25

Table 4-22 - Experiment 3, Scenario 5 - Results

The results shown in Table 4-22 show that the most appropriate behaviour to execute is the 'Add Mix to Tin' behaviour and the most inappropriate behaviours are to add more ingredients to the mix. This leads to scenario 6 shown in Table 4-23 and also the results of the next round of energy spreading in Table 4-24.

World State	Values
Oven On	True
Oven Off	False
Tin Empty	False
Tin Lined	True
Mixed	True
Unmixed	False
Have Sugar	True
Have Flour	True
Have Eggs	True
Have Butter	True
Mix in Tin	**True**

Table 4-23 - Experiment 3, Scenario 6

Behaviour	Energy
Bake	39
Heat Oven	0
Line Cake Tin	0
Add Sugar	7.5
Mix	0
Add Mix to Tin	0
Add Flour	7.5
Add Eggs	7.5
Add Butter	7.5

Table 4-24 - Experiment 3, Scenario 6 - Results

The agent now believes that the most appropriate behaviour to execute is the 'Bake' behaviour and in executing this behaviour the agent is able to achieve the goal of 'Bake Cake'. The whole sequence of selected behaviours are shown in Figure 4-16, here you can clearly see which behaviours the agent selected at each time step. It also shows that at certain time steps there were behaviours with greater energy than the selected behaviour, however these behaviours did not have their preconditions met.

This experiment shows that the current implementation of the behaviour network is able to successfully and accurately, select the most appropriate behaviours in a correct sequence leading to the completion of a complicated goal. The test scenario was a closed test with only one system of behaviours. To further test that this is the best implementation of a behaviour network a final test scenario was devised incorporating multiple systems into a single behaviour network and a simulated environment.

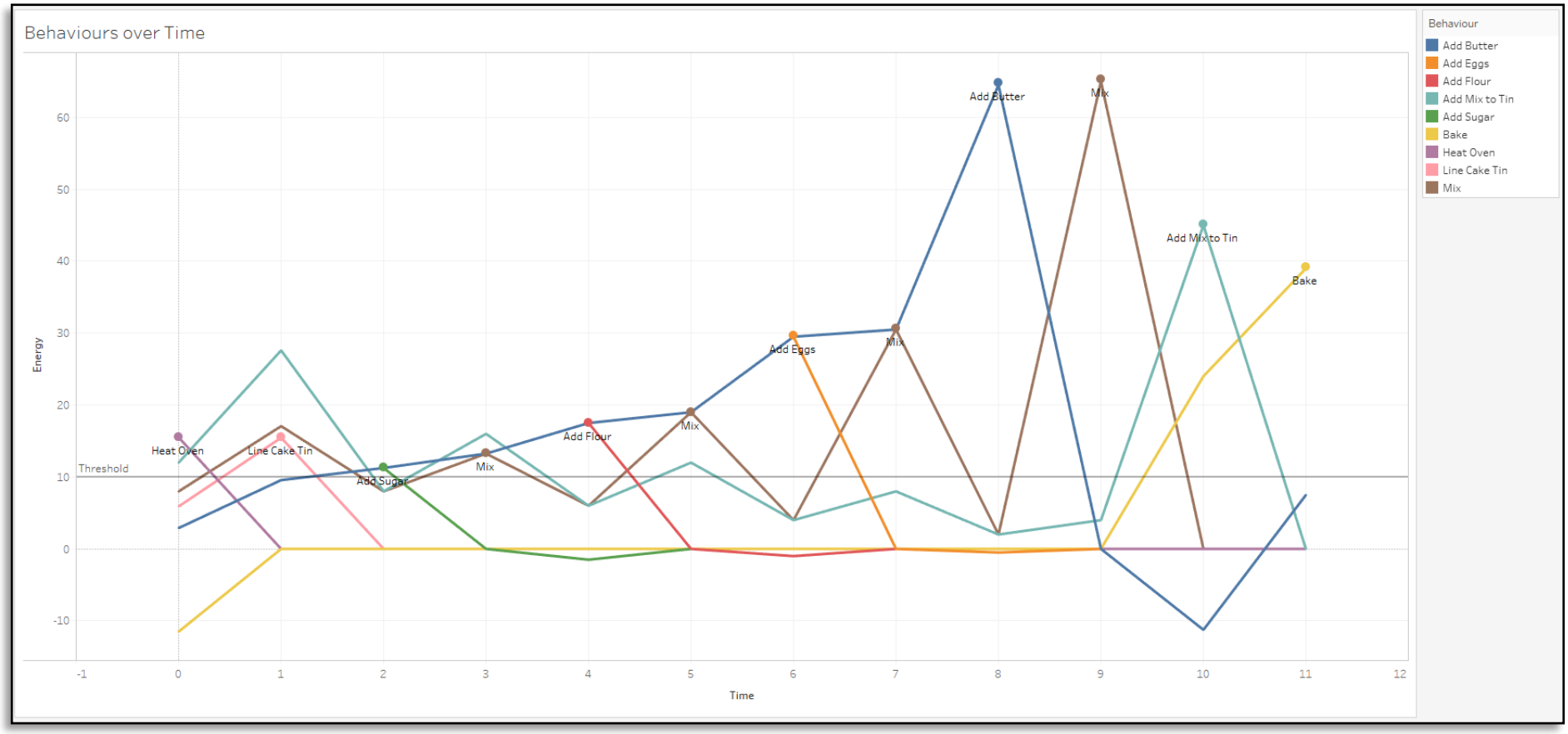

Figure 4-16 - Experiment 3 - Results
Each labelled point represents the selected behaviour for that time step.

4.3.3.4. Experiment 4

Experiment 4 is an extension of the previous experiment. In the previous experiment the goal of baking a cake and the behaviours associated with that goal was introduced. In that experiment there were some assumptions such as:

- The robot was located in the kitchen with the oven and the cake tin.
- All of the ingredients were within the range of the robot.
- Low-level behaviours ('go to' and 'pick up') were included in the higher-level behaviours.

In order to do the testing for Experiment 4, a simulated environment has been implemented. The simulation consists of a grid of cells with each cell containing a value representing an object in the environment. The conditions for this environment are as follows:

- Movement – the agent can only move from one cell to an adjacent cell.
- Sensors – the agent can only scan the cells immediately next to it.
- Memory – the agent maintains a memory of every cell that it has scanned.

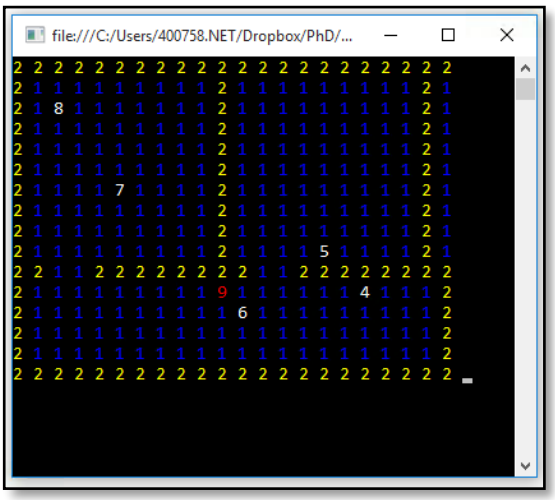

Figure 4-17 - The simulated environment for experiment 4

Object in Environment	Value Representation
Unexplored area	0
Explored area	1
Wall	2
Ingredient	4
Ingredient	5
ingredient	6
ingredient	7
Kitchen (oven)	8
Robot	9

Table 4-15 - Value representation of the simulated environment.

In this environment (Figure 4-17), there are two rooms; one room contains the oven represented by the value ‘8’ and the other room is where the robot will begin (represented by the value ‘9’). Table 4-15 defines what each value represents in the environment.

This experiment uses the following assumption: that the robot will begin the experiment with a full complement of behaviours and goals. The robot will start with an ‘Explore’ goal and an associated ‘Move to Unexplored’ behaviour, where the robot will choose the closest unexplored location (represented by ‘0’ in Figure 4-18) and navigate to that location. The goal is achieved when there are no more unexplored locations for the robot to navigate to. The robot will also begin with the ‘Bake Cake’ goal and all the associated behaviours demonstrated in Experiment 3. In addition to these behaviours the robot will have a ‘Pick Up’ behaviour, this is necessary as each of the ingredients needed to bake the cake are distributed around the environment (shown in Figure 4-17). It is again assumed that the robot can only carry one object a time. The robot will need to explore the environment in order to locate the distributed ingredients. In this experiment, the robot’s sensor range is limited to the areas directly adjacent to itself. Once found, the robot will need to collect the ingredients and bring them to the kitchen, which will enable the robot to perform additional behaviours.

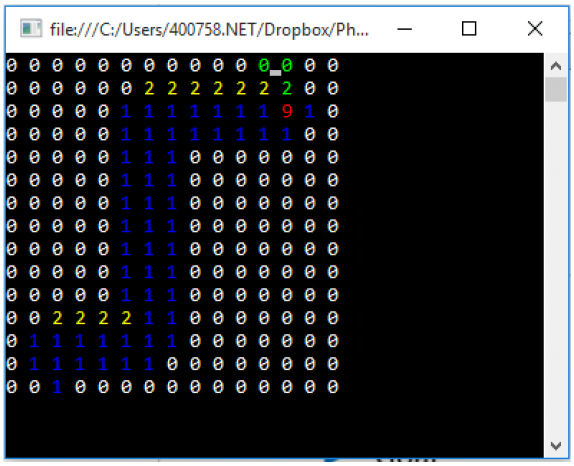

Figure 4-18 - Internal memory map for the agent in the simulated environment

This experiment will make full use of the proposed architecture, described in Chapter 3. The robot will monitor the environment of the simulator and store the information in its own memory map (Figure 4-18). At each time step, the environment manager will take the values of each space around the robot and update the memory map. At the same time the environment manager will update the world state in the robot's memory. The behaviour network manager has access to the world state in the robot's memory and will use this information to update the precondition, add list and delete list for each behaviour. This ensures that the robot will always have an up-to-date behaviour network for the environment that it is in.

Behaviour	**Precondition List**
Bake	Mix in Tin, Oven On, Near Kitchen
Heat Oven	Oven Off, Near Kitchen
Mix	Unmixed, Near Kitchen
Line Cake Tin	Tin Empty, Near Kitchen
Add Mix to Tin	Tin Lined, Sugar, Flour, Eggs, Butter, Mixed, Near Kitchen
Add Sugar	Need Sugar, Have Sugar
Go to Sugar	Sugar Found
Pick up Sugar	Sugar Found, Near Sugar, Hand Empty
Add Eggs	Need Eggs, Have Eggs
Go to Eggs	Eggs Found
Pick up Eggs	Eggs Found, Near Eggs, Hand Empty
Add Flour	Need Flour, Have Flour
Go to Flour	Flour Found
Pick up Flour	Flour Found, Near Flour, Hand Empty
Add Butter	Need Butter, Have Butter
Go to Butter	Butter Found
Pick up Butter	Butter Found, Near Butter, Hand Empty
Go to Kitchen	Kitchen Found

Table 4-16 - Precondition list for Experiment 4

For this experiment the robot will not use the wireless tag manger in the architecture as it will begin this experiment with all of the behaviours that it will need to achieve the goal. In this experiment there will not be any user interaction to interfere with the robot's decision making process and so the web manager from the architecture will not be used either.

Behaviour	Add List
Bake	Bake Cake
Heat Oven	Oven On
Mix	Mixed
Line Cake Tin	Tin Lined
Add Mix to Tin	Mix in Tin
Add Sugar	Unmixed, Sugar
Go to Sugar	Near Sugar
Pick up Sugar	Hand Full, Have Sugar
Add Eggs	Unmixed, Eggs
Go to Eggs	Near Eggs
Pick up Eggs	Hand Full, Have Eggs
Add Flour	Unmixed, Flour
Go to Flour	Near Flour
Pick up Flour	Hand Full, Have Flour
Add Butter	Unmixed, Butter
Go to Butter	Near Butter
Pick up Butter	Hand Full, Have Butter
Go to Kitchen	Near Kitchen

Table 4-17 - Add list for Experiment 4

Behaviour	Delete List
Bake	*None*
Heat Oven	Oven Off
Mix	Unmixed
Line Cake Tin	Tin Empty
Add Mix to Tin	Mixed
Add Sugar	Mixed, Need Sugar
Go to Sugar	*None*
Pick up Sugar	Near Sugar, Hand Empty
Add Eggs	Mixed, Need Eggs
Go to Eggs	*None*
Pick up Eggs	Near Eggs, Hand Empty
Add Flour	Mixed, Need Flour
Go to Flour	*None*
Pick up Flour	Near Flour, Hand Empty
Add Butter	Mixed, Need Butter
Go to Butter	*None*
Pick up Butter	Near Butter, Hand Empty
Go to Kitchen	*None*

Table 4-18 - Delete List for Experiment 4

Table 4-16, Table 4-17 and Table 4-18 show the individual properties of the behaviours that are included in the network shown in Figure 4-19; these are the precondition, add list and delete list for each behaviour.

Table 4-19 shows the initial world state that the robot will begin this experiment in. At this initial point, the robot can only perform explore behaviours until it either finds some ingredients or the kitchen. The behaviour network in Figure 4-19 shows the initial state of the network at the start of the experiment.

Observation	State
Oven On	FALSE
Oven Off	TRUE
Tin Empty	TRUE
Tin Lined	FALSE
Need Sugar	TRUE
Sugar	FALSE
Need Eggs	TRUE
Eggs	FALSE
Need Flour	TRUE
Flour	FALSE
Need Butter	TRUE
Butter	FALSE
Mixed	TRUE
Unmixed	FALSE
Mix in Tin	FALSE
Sugar Found	FALSE
Near Sugar	FALSE
Hand Empty	FALSE
Eggs Found	FALSE
Near Eggs	FALSE
Flour Found	FALSE
Near Flour	FALSE
Butter Found	FALSE
Near Butter	FALSE
Hand Full	FALSE
Have Sugar	FALSE
Have Eggs	FALSE
Have Butter	FALSE
Have Flour	FALSE
Unexplored	TRUE
Explored	FALSE
Kitchen Found	FALSE
Near Kitchen	FALSE

Table 4-19 - Initial world state for Experiment 4

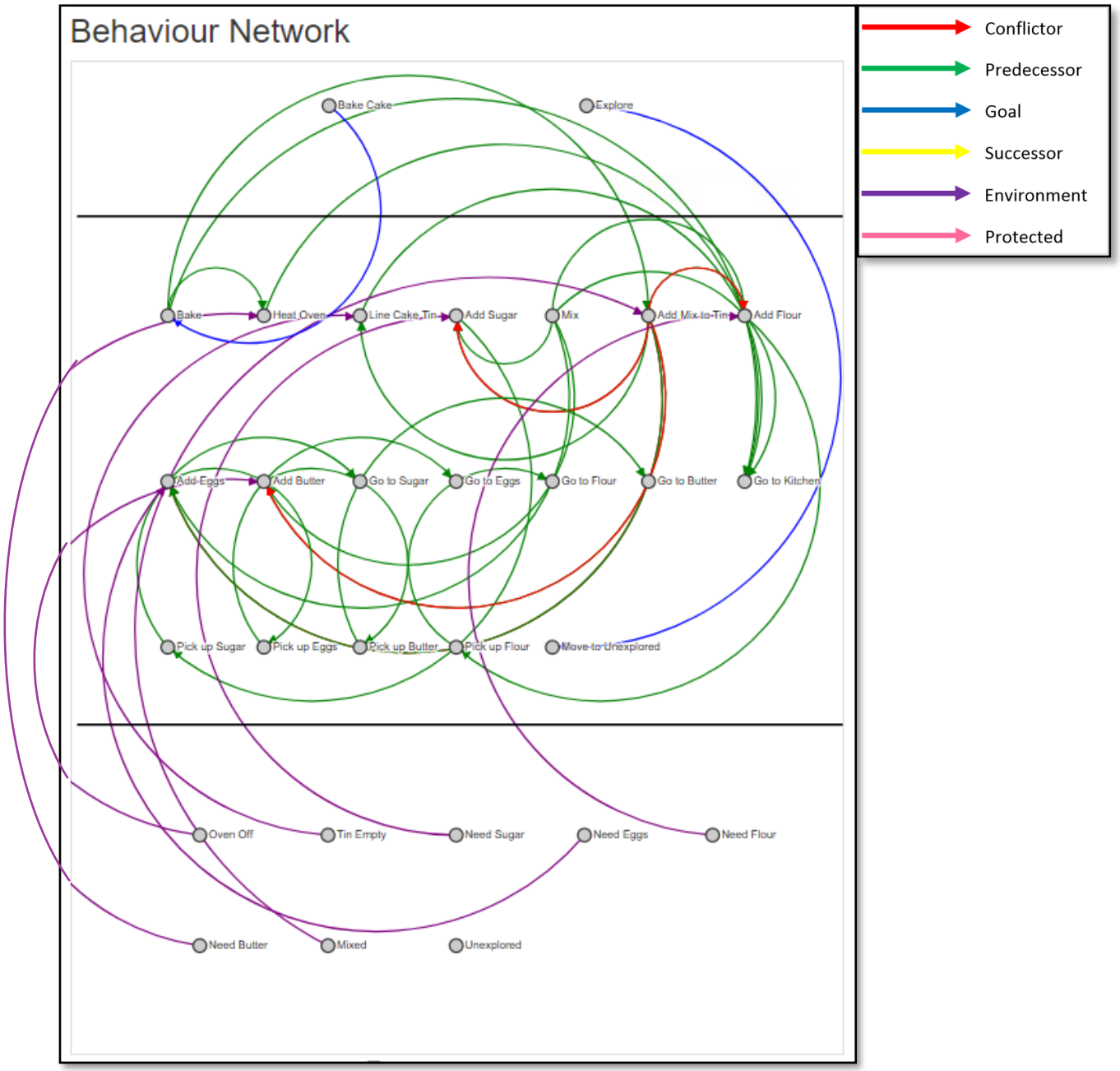

Figure 4-19 - Behaviour Network - Experiment 4

During the early time steps, the robot will select the 'Move to Unexplored' behaviour and navigate the simulated environment. When the robot finds an ingredient, the world state will be updated (by the Environment Manager) allowing the behaviours associated with that object to be selectable. The robot will either interact with the ingredient or continue until it locates the 'Kitchen' object. Figure 4-20 shows the results for this experiment.

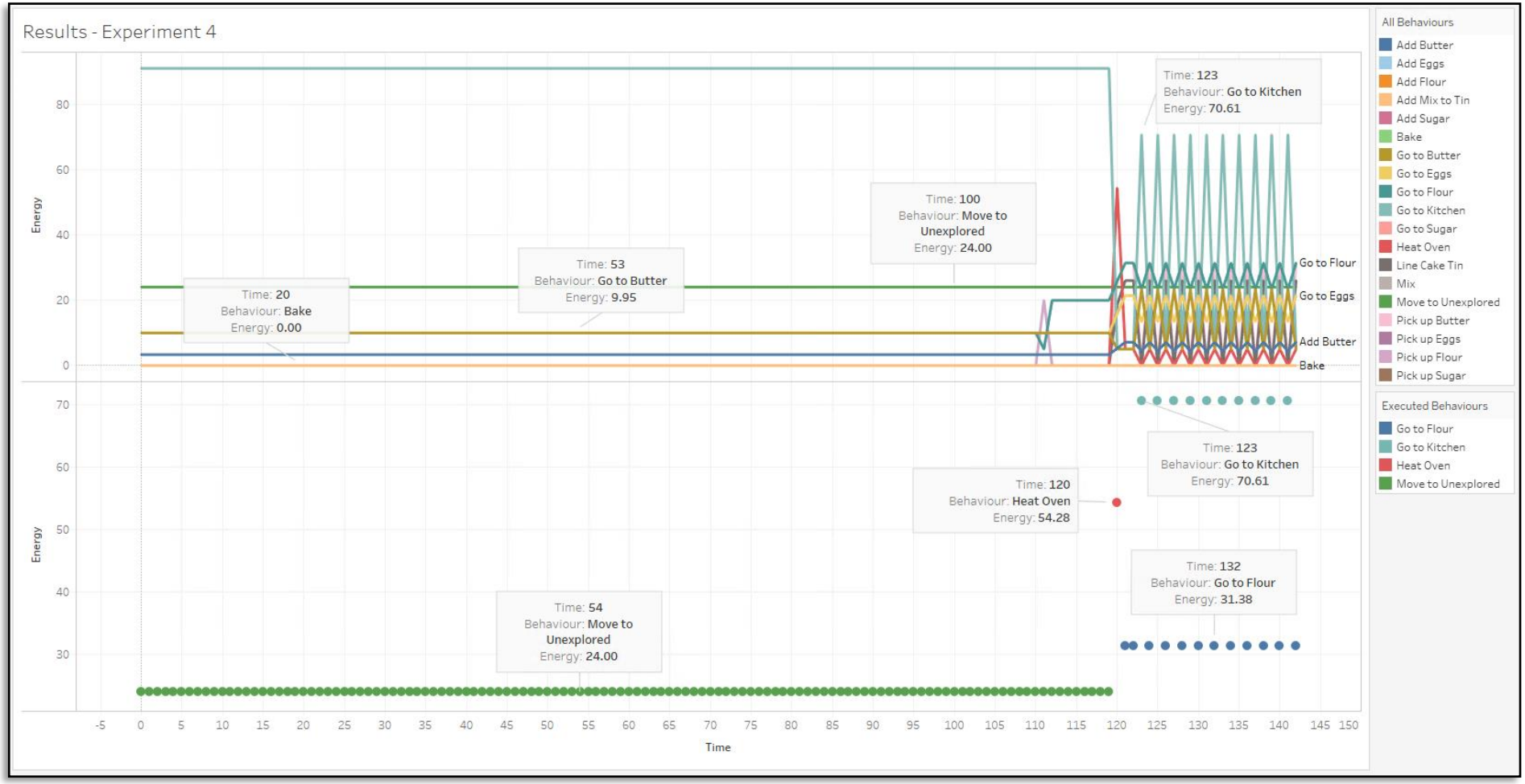

Figure 4-20 - Experiment 4 - Results

Figure 4-20 shows the results from this experiment. The top graph shows the movement of energy over time and the bottom graph shows the selected behaviours at each time step. During the first 100 time steps the robot explores the environment until time step 110 when the robot starts to locate some ingredients. The change in the distribution of energy in the network (from finding the ingredients) was not sufficient to outperform the 'Move to Unexplored' behaviour.

The robot continues to explore the environment until time step 120, where the robot locates the 'Kitchen'. Although the robot had found some ingredients to interact with, the associated behaviours were not selected for execution as they did not have sufficient energy to out-compete the 'Explore' behaviour. This is because those behaviours had a diluted portion of energy, due to the high number of behaviours needed to pass energy to the 'Pick up' behaviours. When the robot is nearby to the kitchen, it is capable of executing behaviours, such as; 'Heat Oven' and 'Line Cake Tin'. The 'Heat Oven' behaviour is selected, however; the robot decides that the 'Go to Flour' behaviour would be the next best behaviour to select over the 'Line Cake Tin' behaviour. Table 4-20 shows the world state when the robot decides to leave the proximity of the kitchen.

The path that the robot has selected is now to navigate to the flour, pick up the flour, return to the kitchen and add the flour to the bowl. The reason why the robot chooses to the leave the kitchen, even though there are still behaviours to select, is because of the numerous inputs to that behaviour has overpowered the other options. It is observed in Figure 4-20 that once the robot leaves the proximity of the Kitchen, leading to the world state shown in Table 4-21, that the distribution of energy moves to the 'Go to Kitchen' behaviour.

Observation	State	Observation	State
Oven On	TRUE	Hand Empty	FALSE
Oven Off	FALSE	Eggs Found	FALSE
Tin Empty	TRUE	Near Eggs	FALSE
Tin Lined	FALSE	Flour Found	TRUE
Need Sugar	TRUE	Near Flour	FALSE
Sugar	FALSE	Butter Found	TRUE
Need Eggs	TRUE	Near Butter	TRUE
Eggs	FALSE	Hand Full	FALSE
Need Flour	TRUE	Have Sugar	FALSE
Flour	FALSE	Have Eggs	FALSE
Need Butter	TRUE	Have Butter	FALSE
Butter	FALSE	Have Flour	FALSE
Mixed	TRUE	Unexplored	FALSE
Unmixed	FALSE	Explored	TRUE
Mix in Tin	FALSE	Kitchen Found	TRUE
Sugar Found	FALSE	Near Kitchen	TRUE
Near Sugar	FALSE		

Table 4-20 - World state when the robot is in the proximity of the kitchen (Experiment 4).

Observation	State	Observation	State
Oven On	TRUE	Hand Empty	FALSE
Oven Off	FALSE	Eggs Found	FALSE
Tin Empty	TRUE	Near Eggs	FALSE
Tin Lined	FALSE	Flour Found	TRUE
Need Sugar	TRUE	Near Flour	FALSE
Sugar	FALSE	Butter Found	TRUE
Need Eggs	TRUE	Near Butter	FALSE
Eggs	FALSE	Hand Full	FALSE
Need Flour	TRUE	Have Sugar	FALSE
Flour	FALSE	Have Eggs	FALSE
Need Butter	TRUE	Have Butter	FALSE
Butter	FALSE	Have Flour	FALSE
Mixed	TRUE	Unexplored	FALSE
Unmixed	FALSE	Explored	TRUE
Mix in Tin	FALSE	Kitchen Found	TRUE
Sugar Found	FALSE	Near Kitchen	FALSE
Near Sugar	FALSE		

Table 4-21 - World state when the robot leaves the proximity of the kitchen.

It is observed at this point that the robot will continue to select the behaviours: 'Go to Kitchen' and 'Go to Flour' consecutively and not advance the world state. The expected outcome would be for the agent to navigate to the ingredient, perform the pickup behaviour and bring the ingredient to the kitchen. The loop that the agent demonstrated is undesirable and showed no indication of stopping. This negative result was unintentional and was caused by the unfair competition (where behaviours receive energy from more behaviours than others) (See section 2.2.4) of the inputs to those behaviours at each time step. A similar situation was also described in (Tyrrell, 1994), where unfair competition can lead to incorrect behaviours being selected for execution.

A variety of techniques have been tried to bypass this limitation (Tyrrell, 1994; Lee and Cho, 2014). The first technique was to use intermediary behaviours to filter the flow of energy to those behaviours with high inputs. This concept would use an intermediary behaviour to hold the energy from the various inputs and send a proportion of that energy to the consummatory behaviour. This yielded poor results as the proportion of energy given to the consummatory behaviour was still greater than the energy in all other behaviours in the system.

Another technique was to make changes to the global weighting parameters. A variety of different values were used for the parameters of the behaviour network. Those values remained between 0 and 1 and were chosen at random. Some parameter changes caused the robot to select inappropriate behaviours and some changes limited the flow of energy through the network. There are parameter tuning algorithms (such as genetic algorithm or grid search) that could be used to find the best global weights for the network. However, these algorithms work well in constrained situations and should the situation / environment change then the results would be much worse than optimal.

One potential solution to the problem shown in this experiment would be to add additional motivation to goals / behaviours that the agent is currently performing. For example, if the agent performs a 'Go To' object behaviour then on the next time step additional motivation would be added to the same 'Go To' behaviour until the agent is able to navigate to the object. The amount of motivation should be strong enough to allow the behaviour to be selected again but also weak enough that is does not over power other behaviours from ever been selected.

Another potential solution (explored in section 5.1.2) is to abstract the behaviours to a higher-level of detail. The theory is that it would reduce the complexity of the behaviour network, which would result in fewer inputs to each node. If there are fewer inputs to a node then there would be less chances of bias in the network (as shown in this experiment). E.g. instead of the behaviours; 'Go To Object', 'Pick up Object' and 'Put down Object', they could be grouped into a single higher-level behaviour. If the higher level behaviour is selected for execution then additional steps to select the action to take can then be explored. Lee and Cho (Lee and Cho, 2014) demonstrated how a high-level behaviour network can start smaller behaviour networks to achieve a high-level behaviour.

4.3.3.5. Overview of the results

The data packet approach has been tested with four different experiments. The aim of these experiments is to test the data packet approach with incrementally more complex / difficult situations. The first experiment (Section 4.3.3.1) incorporated a simple behaviour network with a single link type. This experiment showed that the data packet approach, regardless of order, was able to evenly distribute activation energy through a behaviour network. The second experiment (Section 4.3.3.2) expanded from

the previous experiment to simulate a real-world scenario with different goals and behaviours and a variety of link types. The behaviour network was again, simple and the results showed that it was able to evenly distribute the energy through the network and complete the goal. The third experiment (Section 4.3.3.3) incorporated a much more complicated real-world scenario, baking a cake. The assumptions for this scenario was that the agent was within the range of each object/ingredient and the aim was to see if the behaviour network could select the actions in the correct order to complete the goal. This experiment, again had a variety of different link types and used the data packet approach to distribute the energy through the network. The results showed that the agent was able to select the correct order of actions and complete the goal. To extend this experiment further, experiment 4 (Section 4.3.3.4) incorporated a simulated environment for the agent to navigate. The test scenario was the same as experiment 3 except that each object was randomly placed in the environment. The agent then needed to explore the environment to locate each object/ingredient in order to complete the goal. The behaviour network was expanded with new behaviours to enable the agent to navigate and interact with objects. The results from this experiment were undesirable as the agent was trapped in a loop of selecting conflicting behaviours. This is discussed in detail in Section 4.3.3.4. The next experiment (Section 5.1.1) shall expand upon this experiment to test the behaviour network with distributed behaviours/ goals. One of the aims of this experiment is to see if the agent's behaviour will change if it receives the behaviours iteratively from the environment and if that will allow it to complete the goal successfully.

The technical aims described in Section 1.3.1 discussed the exploration and experimentation of behaviour networks and their activation spreading mechanisms, which have been achieved in this Section. These experiments were necessary to assist

with answering the hypothesis of the study; as an agent needs to be able to successfully complete goals in dynamic and both unstructured and structured environment. The next section shall continue with this objective as it explores the dynamic behaviour network with distributed behaviours and goals.

5. Dynamic Behaviour Network

This chapter will begin with a discussion on dynamic behaviour networks, starting with an introduction to the disadvantages to standard behaviour networks and concluding with the motivation for using a dynamic behaviour network.

Chapter 4 covers a detailed explanation of a standard behaviour network and how it can be used in certain situations. Traditionally, a behaviour network will be hand-designed (predefined), with each behaviour being hard-coded to work in the environment of the agent. The network will allow the agent to react to changes in its environment and select the best behaviour for the situation it is in. The main disadvantage of the behaviour network is the hard-coded nature of its behaviours. This limitation prevents the agent from learning new behaviours or goals or reacting to situations that it was not pre-programmed to handle. For this, a dynamic behaviour network is required. A dynamic behaviour network is defined as one which can learn new behaviours over time and can modify itself based on internal and external factors, while keeping some of the same traits as the static behaviour.

Chapter 3 discussed the proposed architecture to be used for both a standard behaviour network and a dynamic behaviour network. In this architecture, the behaviour network manager is the component responsible for managing the network of behaviours. It takes input from other areas of the system and uses that to update the

behaviour network, either by adding new behaviours or goals or removing unwanted behaviours.

To enable a dynamic behaviour network to learn new behaviours and goals, it is proposed to embed wireless tags onto objects in a real-world environment. When a robot enters the environment, it will be able to read the wireless tags and download the necessary behaviours and goals needed to interact with items in that environment. Chapter 3 discussed the Wireless Tag Manager and how that would be used to read data from wireless tags in the environment. When the robot reads a wireless tag, the information is parsed and passed to the Behaviour Network Manager to add those behaviours and goals to the existing behaviour network. It is worth noting that the experiments were conducted in a virtual environment and the wireless tags were simulated in the experiments.

Another element to a dynamic behaviour network is the ability to allow users to interact with the decision-making process. Traditionally, a standard behaviour network will have predefined goals, each with their own predefined motivations. In this situation, there is no method for altering those goals or motivations. In the proposed architecture for this system, the method for how users will be able to influence the motivations of the goals for an agent are discussed. The Behaviour Network Manager will again take this input and make changes to the behaviour network at run time.

5.1. Test Cases for the Dynamic Behaviour Network

This section describes all of the test cases that were created to test the dynamic behaviour network. Each test case showcases a particular problem with which a static behaviour network would have difficulty and that a dynamic behaviour network can solve.

The first experiment will introduce distributed behaviours to a behaviour network. The primary aim for this experiment is to show that a behaviour network can grow dynamically, adding new behaviours and goals at run-time. This part of the experiment will be successful if the agent can successfully navigate a simulated environment and read in new behaviours / goals dynamically and perform new behaviours that it did not originally begin with. The secondary aim for this experiment is to extend the experiment (4) from Section 4.3. To explore whether a dynamic behaviour network can perform better than the static behaviour network with the same experiment. This part of the experiment will be successful if the agent is able to overcome the problems found in Section 4.3 and complete the goal (‘Bake Cake’).

The second experiment will test the dynamic behaviour network with multiple competing goals. The previous experiments demonstrated the agent starting with a single default goal ‘Explore’ along with a larger goal such as ‘Bake Cake’. This experiment will test the dynamic behaviour network with multiple goals that are distributed in the simulated environment. This experiment will be successful if the agent is able to successfully explore the environment, add all found goals / behaviours to its network and complete all of the goals.

The third experiment will test how a dynamic behaviour network can respond to user input. The aim is to show that there are many ways in which a behaviour network

can be dynamic and that by changing the motivation for goal can affect the decisions of the agent.

5.1.1. Distributed Behaviours

The primary benefit of a dynamic behaviour network is the ability to distribute behaviours in an environment and to read those behaviours at a later time. This allows an agent to begin with a limited pool of behaviours, allowing for a constrained decision-making process. The agent can then locate and add new behaviours at run-time, giving the agent more functionality than it was originally programmed with.

To test the functionality of a dynamic behaviour network with distributed behaviours, experiment 4 from Chapter 4 was extended. The baking a cake example was used and the behaviours and the goals for this scenario were embedded into the objects in the environment.

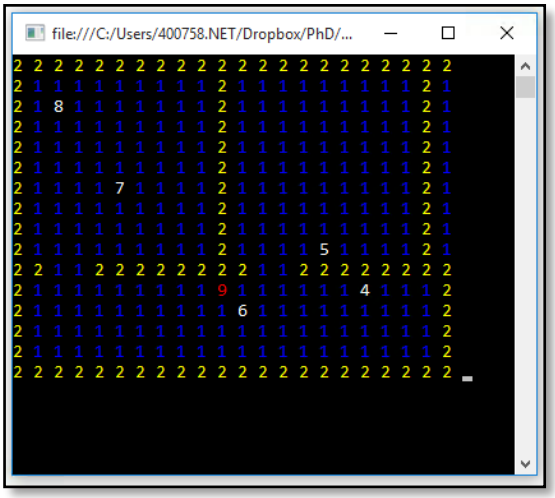

Figure 5-1 - Simulated environment with distributed behaviours

The robot needs to continue to navigate an unstructured environment, an environment that has not been predefined (shown in Figure 5-1) to locate each of the key ingredients needed to bake a cake. Each of the ingredients has associated behaviours, which the robot will read when nearby (simulating reading from a wireless

tag). Those behaviours will be added to the existing behaviour network and are selected for activation when appropriate. Table 5-1 shows the associated behaviours and goals that are embedded in each object in the environment (Figure 5-1).

Object in Environment	Value Representation	Behaviour(s) / Goal(s)
Unexplored area	0	N/A
Explored area	1	N/A
Wall	2	N/A
Ingredient (Sugar)	4	Behaviour: 'Pick up Sugar'
		Behaviour: 'Go to Sugar'
Ingredient (Eggs)	5	Behaviour: 'Pick up Eggs'
		Behaviour: 'Go to Eggs'
Ingredient (Flour)	6	Behaviour: 'Pick up Flour'
		Behaviour: 'Go to Flour'
Ingredient (Butter)	7	Behaviour: 'Pick up Butter'
		Behaviour: 'Go to Butter'
Kitchen (Oven) (Recipe for making a cake)	8	Goal: 'Bake Cake'
		Behaviour: 'Bake'
		Behaviour: 'Heat Oven'
		Behaviour: 'Line Cake Tin'
		Behaviour: 'Add Mix to Tin'
		Behaviour: 'Add Sugar'
		Behaviour: 'Add Eggs'
		Behaviour: 'Add Flour'
		Behaviour: 'Add Butter'
		Behaviour: 'Go to Kitchen'
Robot	9	N/A

Table 5-1 - Value representation of the simulated environment with associated distributed behaviours and goals.

The aim of this scenario is for the robot to navigate the unstructured environment, locating each of the key ingredients needed to bake a cake. When the robot locates the kitchen, it will find the recipe with the goal for baking a cake. The recipe will contain a list of ingredients and the instructions to achieve the goal. This will then trigger the chain of behaviours needed to complete the goal.

Behaviour	Precondition List
Move to Unexplored	Unexplored

Table 5-2 - Precondition List for distributed behaviours scenario

Behaviour	Add List
Move to Unexplored	Explored

Table 5-3 - Add List for distributed behaviours scenario

Behaviour	Delete List
Move to Unexplored	Unexplored

Table 5-4 - Delete List for distributed behaviours scenario

Table 5-2, Table 5-3 and Table 5-4 show the individual properties of the behaviours that are included in the network shown in Figure 5-2. These form the precondition, add list and delete list for each behaviour. Table 5-5 shows the initial world state that the robot will begin this experiment in. At this initial point, the robot will only be able to perform the 'Move to Unexplored' behaviour until it has been able to navigate the environment enough to find the additional behaviours. The behaviour network in Figure 5-2 shows the initial state of the network at the start of the experiment.

Observation	State
Unexplored	TRUE
Explored	FALSE

Table 5-5 - Initial world state for the distributed behaviours scenario.

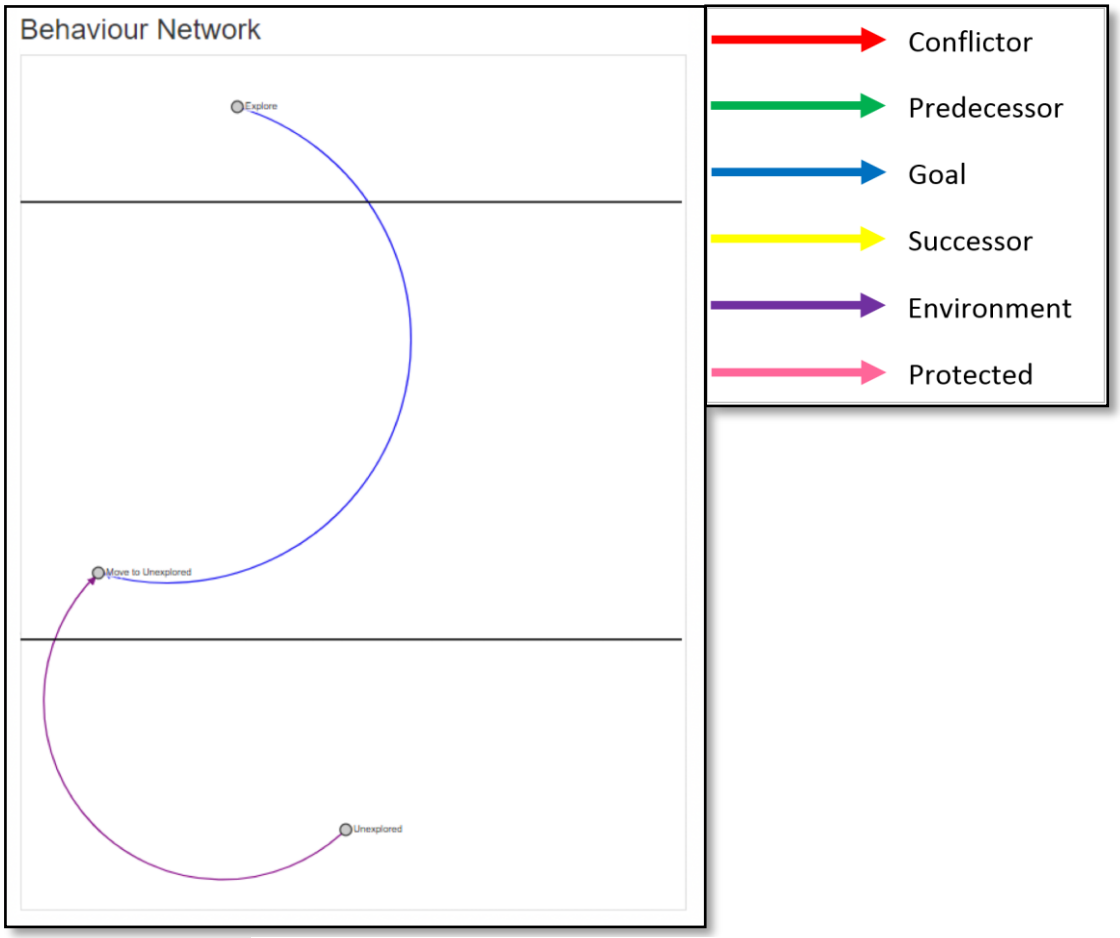

Figure 5-2 - Behaviour Network – distributed behaviours scenario

The experiments are conducted through simulated time steps. At each timestep the robot will make a decision on which action to perform and that action will be executed. During the early time steps, the robot will select the 'Move to Unexplored' behaviour and navigate the simulated environment. When the robot finds an ingredient, the world state will be updated (by the Environment Manager) allowing the behaviours associated with that object to be selectable. The robot will either interact with the ingredient or continue until it locates the 'Kitchen' object where it will obtain the 'Bake Cake' goal. In this scenario the robot will learn about the different ingredients scattered around the environment but it will not have any motivation to interact with those objects. When the robot enters the kitchen (represented by an '8') it will read the data from the wireless tag, telling the robot how to use those ingredients it found earlier on in the environment. Figure 5-3 shows the results for this experiment.

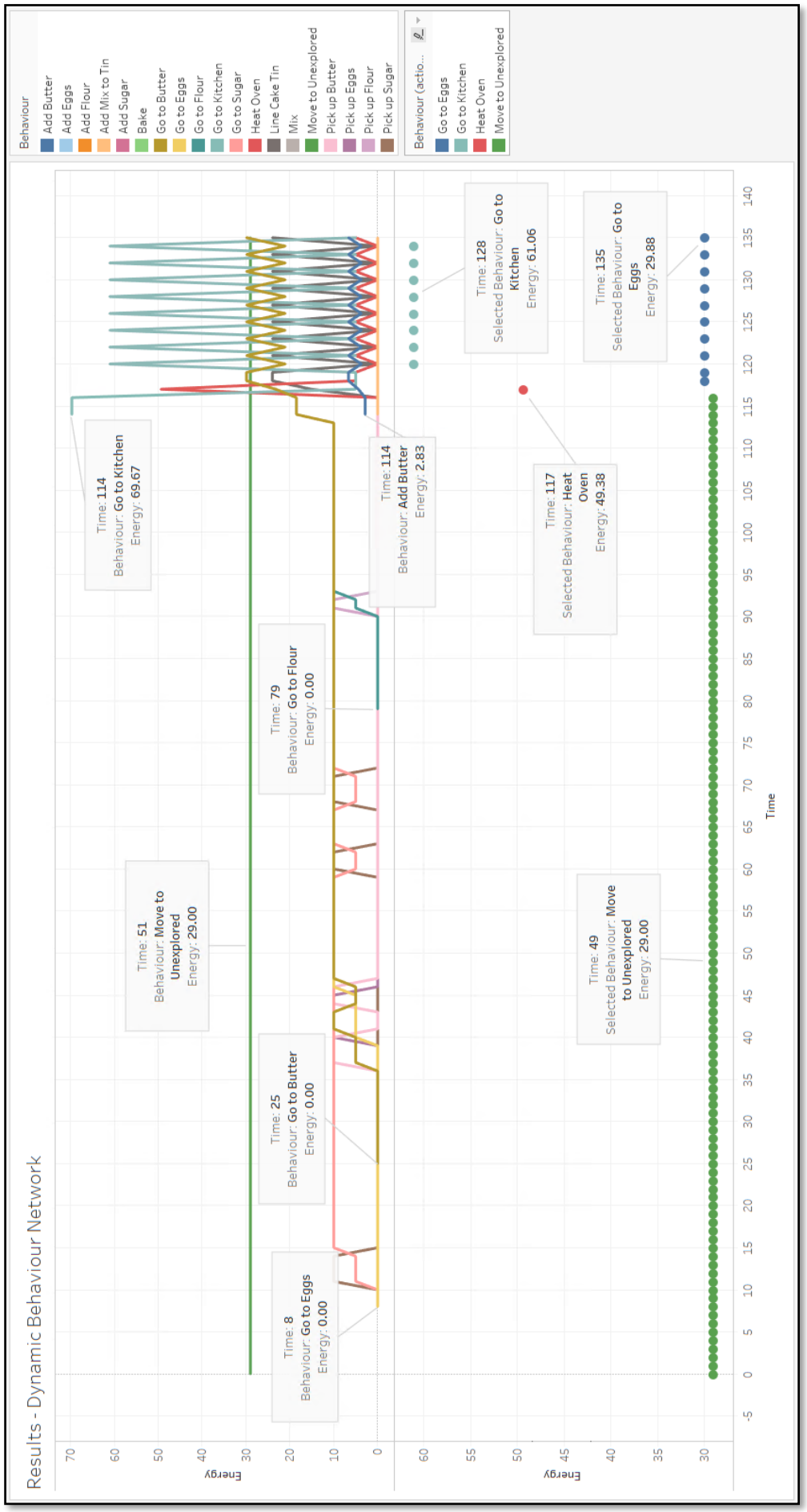

Figure 5-3 - Results for distributed behaviours – Bake Cake Scenario

Figure 5-3 shows the results from this experiment. The top graph shows the movement of energy over time and the bottom graph shows the selected behaviours at each time step. While the results for this experiment are similar to those in experiment 4 (Figure 4-20), the top half of the chart shows the dynamic nature of the network. This shows that during the initial time steps of the experiment, only the 'Move to Unexplored' behaviour received any input. At time step 8 the robot locates the first object (Eggs). The Wireless Tag Manager receives the wireless tag data for that object, parses it and passes that to the Behaviour Network Manager to add to the existing network. It is observed that this process of locating objects and building the behaviour network continues until time step 114, when the robot locates the kitchen.

Figure 5-4 shows the evolution of the behaviour network over time for this scenario. The first part of the behaviour network contains the initial goal and behaviour to allow the robot to explore the environment. The second part of Figure 5-4 shows that the behaviour network has grown and new behaviours have been added in real-time. Although the robot has new behaviours in its network, it cannot utilise those behaviours until a goal is added and those behaviours receive some motivation. The third part of Figure 5-4 shows the completed behaviour network towards the end of the simulation. Here the robot has located the kitchen and found a wireless tag containing a new goal ("Bake Cake"). This allows all of the behaviours that the robot has found on the different ingredients in the environment to begin to receive motivation.

The complexity of the behaviour network over time is challenging to define. The complexity will depend of the number of environment variables, the number of goals, the number of behaviours and the number of links. The number of links in the network will depend on the state of the environment and the conditions of the behaviour. In the worst case scenario the complexity of the network will be O(n).

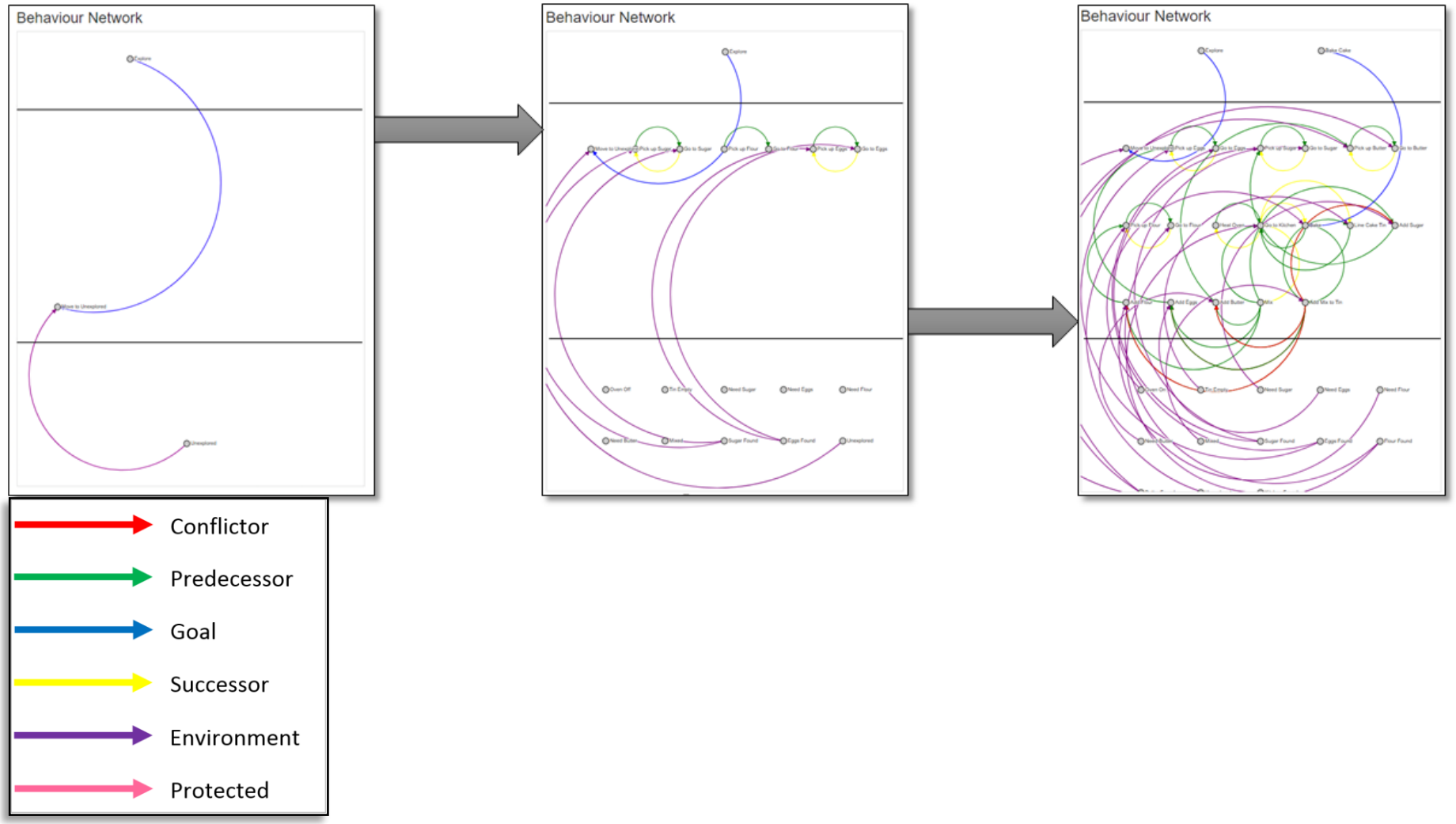

Figure 5-4 - Distributed behaviours added to behaviour network over time.

The first network shows the initial configuration of the behaviour network.

The second network shows the behaviour network after it has found some wireless tags and has incorporated the information into the network.

The final network shows the final configuration when all of the wireless tags have been found.

When the robot locates the 'Kitchen', similarly to in experiment 4 (Chapter 4), the 'Heat Oven' behaviour is first selected. Again; the robot decides to go to an ingredient instead of selecting the 'Line Cake Tin' behaviour. Table 5-6 shows the world states when the robot decides to leave the proximity of the kitchen.

Observation	State	Observation	State
Oven On	TRUE	Hand Empty	FALSE
Oven Off	FALSE	Eggs Found	TRUE
Tin Empty	TRUE	Near Eggs	FALSE
Tin Lined	FALSE	Flour Found	TRUE
Need Sugar	TRUE	Near Flour	FALSE
Sugar	FALSE	Butter Found	TRUE
Need Eggs	TRUE	Near Butter	FALSE
Eggs	FALSE	Hand Full	FALSE
Need Flour	TRUE	Have Sugar	FALSE
Flour	FALSE	Have Eggs	FALSE
Need Butter	TRUE	Have Butter	FALSE
Butter	FALSE	Have Flour	FALSE
Mixed	TRUE	Unexplored	TRUE
Unmixed	FALSE	Explored	FALSE
Mix in Tin	FALSE	Kitchen Found	TRUE
Sugar Found	TRUE	Near Kitchen	TRUE
Near Sugar	FALSE		

Table 5-6 - World state when the robot is in the proximity of the kitchen (Distributed Behaviour Scenario).

The robot has now selected to navigate to the Eggs. It is expected that the next sequence of behaviours will be to pick up the Eggs, return to the kitchen and add the Eggs to the bowl. The reason why the robot chooses to the leave the kitchen, is again because of the numerous inputs to that behaviour has overpowered the other options. This is a fundamental problem with behaviour networks that is discussed in the following chapter. It is observed in Figure 5-3 that once the robot leaves the proximity of the Kitchen, leading to the world state shown in Table 5-7, that the distribution of energy moves to the 'Go to Kitchen' behaviour.

Observation	State	Observation	State
Oven On	FALSE	Hand Empty	FALSE
Oven Off	TRUE	Eggs Found	TRUE
Tin Empty	TRUE	Near Eggs	FALSE
Tin Lined	FALSE	Flour Found	TRUE
Need Sugar	TRUE	Near Flour	FALSE
Sugar	FALSE	Butter Found	TRUE
Need Eggs	TRUE	Near Butter	FALSE
Eggs	FALSE	Hand Full	FALSE
Need Flour	TRUE	Have Sugar	FALSE
Flour	FALSE	Have Eggs	FALSE
Need Butter	TRUE	Have Butter	FALSE
Butter	FALSE	Have Flour	FALSE
Mixed	TRUE	Unexplored	TRUE
Unmixed	FALSE	Explored	FALSE
Mix in Tin	FALSE	Kitchen Found	FALSE
Sugar Found	TRUE	Near Kitchen	FALSE
Near Sugar	FALSE		

Table 5-7 - World state when the robot leaves the proximity of the kitchen (Distributed Behaviours Scenario).

It is observed at this point that the robot will continue to select the behaviours: 'Go to Kitchen' and 'Go to Eggs' consecutively and not advance the world state. Again, there is an undesired emphasis to certain inputs with respect to others (discussed in previous chapters as unfair competition) at each time step. This is shown in Figure 5-5, where the robot is caught between two paths.

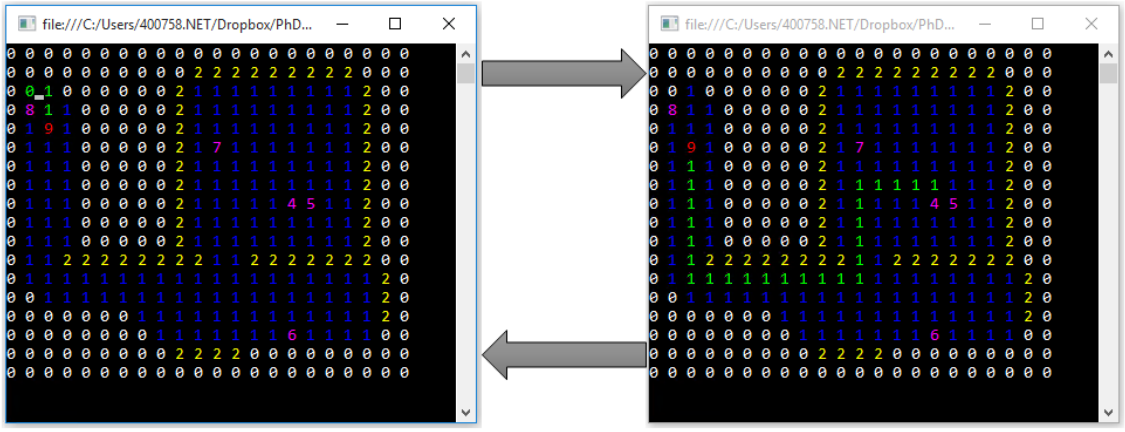

Figure 5-5 - The robot is trapped between selecting two different paths.

The screenshot to the left of Figure 5-5 shows the robot executing the 'Go to Kitchen' behaviour and has a route plotted (represented by the green path) to the kitchen. The screenshot to the right of Figure 5-5 shows the robot executing the 'Go to Eggs' behaviour with the route plotted. This shows that when the robot tries to navigate to the kitchen, and becomes in the proximity of the kitchen it will then plot a route to an ingredient. Then when it is outside of the proximity of the kitchen, the robot will plot a route back to the kitchen.

This experiment is the same experiment as in section 4.3.1.3 with the addition of distributed behaviours. In the first experiment the agent was given all of the behaviours and goals at the beginning of the simulation. In this experiment the agent begins the experiment with only the 'Explore' behaviour. The aim of this experiment was to test the distributed behaviours in an unstructured environment. The results are positive, as it shows that the agent is able to navigate the environment and collect new goals and behaviours. It shows that the behaviour network was successful in incorporating new behaviours into an already running behaviour network and that it was able to make decisions towards achieving new goals. Unfortunately, it also shows the same negative result as the previous experiment and the agent was stuck in the same loop as before.

The results show that a dynamic behaviour network with distributed goals and behaviours can work. The robot was able to navigate an unstructured environment with a basic behaviour network and add new goals and behaviours dynamically. Although the robot was unsuccessful in completing the goal that it had found, the robot was able to successfully build a new behaviour network dynamically. As discussed previously and also discussed in (Tyrrell, 1994) the network demonstrated unfair bias between some of the behaviour in the network leading to incorrect behaviours been selected. The next experiment will aim to test the dynamic behaviour network without any bias. This

experiment will take inspiration from Section 4.3.1.3, where the behaviour network will contain higher-level behaviours. These would have fewer inputs resulting in less possible bias in the network. This experiment will also test the dynamic behaviour network with multiple goals. The robot will again need to navigate an unstructured environment and complete all of the goals.

5.1.2. Distributed behaviours with multiple goals

In the previous section, the distribution of behaviours in the environment using the previous experiment from chapter 4 (baking a cake) was tested. In this section the concept of distributed behaviours is further tested by presenting a new scenario, one with multiple goals. In this scenario the robot begins the experiment with the same conditions as in the previous experiment. The robot is placed in the same simulated environment shown in Figure 5-1, but each object will be unique and contain different goals and behaviours. As before, once the robot is within the proximity of the objects, it will read the data from the wireless tags and build upon the initial behaviour network.

In this scenario the robot will need to navigate and explore the unstructured environment, locating each object that has been added. This scenario will put the robot in an environment with several different goals that the robot will need to accomplish. Object 5 represents a plant in an environment that needs some water. When the robot locates the plant, it will receive the goal: 'Water Plant' and the behaviours: 'Go to Plant and 'Water the Plant'. The robot will need to explore the environment to find some water in order to solve this goal. Object 6 represents some water, which the robot can use to water the plant (Object 5). When the robot locates the water, it will receive the behaviours: 'Go to Water' and 'Pick up Water'. If the robot does not locate the plant first, then there will be no motivation to interact with the water object. The aim for this

goal is to have the robot locate both objects, collect the water and bring it to the plant to be able to water the plant.

Object 7 represents some mess in the environment that the robot will need to tidy. The idea of this scenario is that the behaviours are all higher-level behaviours, meaning that the robot will not be expected to perform the complicated lower-level behaviours needed to interact with the object. The assumption is that the when the higher-level behaviours are selected, that they are immediately executed successfully. Given this assumption the goal: 'Clean Mess' and behaviours: 'Go to Mess' and 'Tidy Mess' are embedded in the mess object.

Finally, another object (Object 8) which represents a dishwasher in the environment, is included. The idea for this object is that the robot will need to turn on the dishwasher in order to wash some pots. For simplicity, it is assumed that the world state is such that the pots are already in the dishwasher and the robot only needs to switch it on. This is to ensure that the robot has a complicated behaviour network, with many different goals and behaviours. It will also limit the number of behaviours in each sub system so that neither behaviour is overly popular in the network, causing an unfair distribution of energy (such as that seen in the previous experiment). Table 5-8 shows the associated behaviours and goals that are embedded in each object in the updated simulated environment.

Object in Environment	Value Representation	Behaviour(s) / Goal(s)
Unexplored area	0	N/A
Explored area	1	N/A
Wall	2	N/A
Plant	5	Goal: 'Water Plant'
		Behaviour: 'Go to Plant'
		Behaviour: 'Water the Plant'
Water	6	Behaviour: 'Pick up Water'
		Behaviour: 'Go to Water'
Mess	7	Goal: 'Clean Mess'
		Behaviour: 'Go to Mess'
		Behaviour: 'Tidy Mess'
Dishwasher	8	Goal: 'Wash Pots'
		Behaviour: 'Turn On Dishwasher'
		Behaviour: 'Go to Dishwasher'
Robot	9	N/A

Table 5-8 - Value representation of the simulated environment with associated distributed behaviours and goals.

This scenario uses the same initial behaviours as the previous example and the properties are shown in Table 5-2, Table 5-3 and Table 5-4. Those behaviours are included in the network shown in Figure 5-2: the precondition, add and delete list for each behaviour. Table 5-5 shows the initial world state in which the robot begins this experiment. At this initial point, the robot will only be able to perform the 'Move to Unexplored' behaviour. When it has been able to navigate the environment enough to find the additional behaviours and goals, other possibilities will arise. The behaviour network in Figure 5-2 shows the initial state of the network.

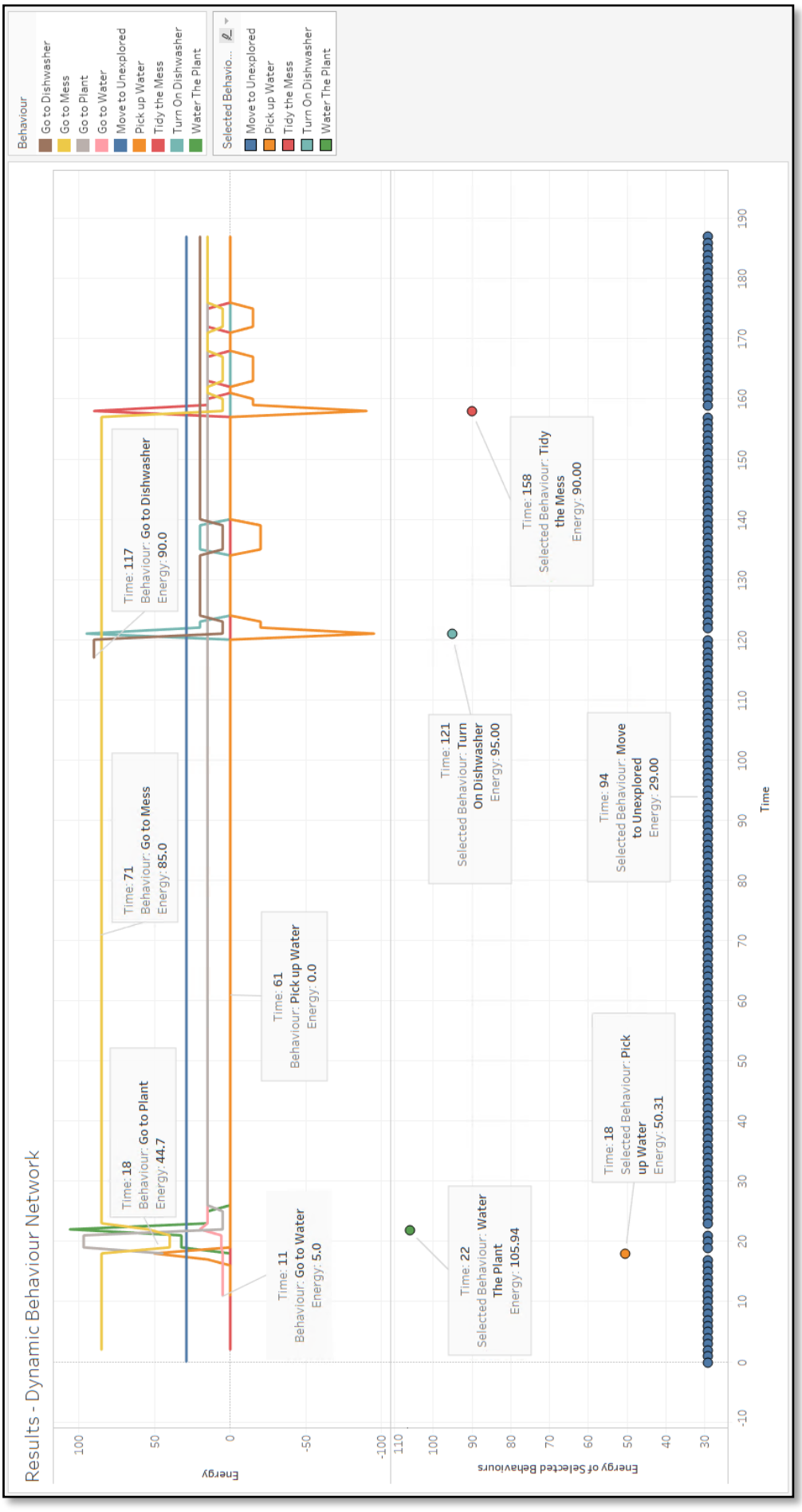

Figure 5-6 - Results for distributed behaviours

Figure 5-6 shows the results from this experiment. The top graph shows the movement of energy over time and the bottom graph shows the selected behaviours at each time step. The results highlight the dynamic nature of the behaviour network as it shows new behaviours added throughout the experiment. It shows that the robot was able to navigate the environment until it found the water object. At time step 18, the robot picked up the water object and at time step 22 the robot executed the 'Water the Plant' behaviour fulfilling the goal 'Water Plant'. The robot then continues to explore the environment until it finds the dishwasher, which allows it to execute the 'Turn on Dishwasher' behaviour that completes the goal 'Wash Pots'. Finally, the robot continues to explore until it finds the mess in the environment, which allows it to execute the 'Tidy the Mess' behaviour and achieve the final goal.

Unlike the previous experiment, the robot was able to explore the entire environment, read all of the wireless tags and execute all of the goals. In this scenario the robot was able select the behaviours which helped it to successfully achieve the goals without the any of those behaviours having an unfair advantage over any other behaviour.

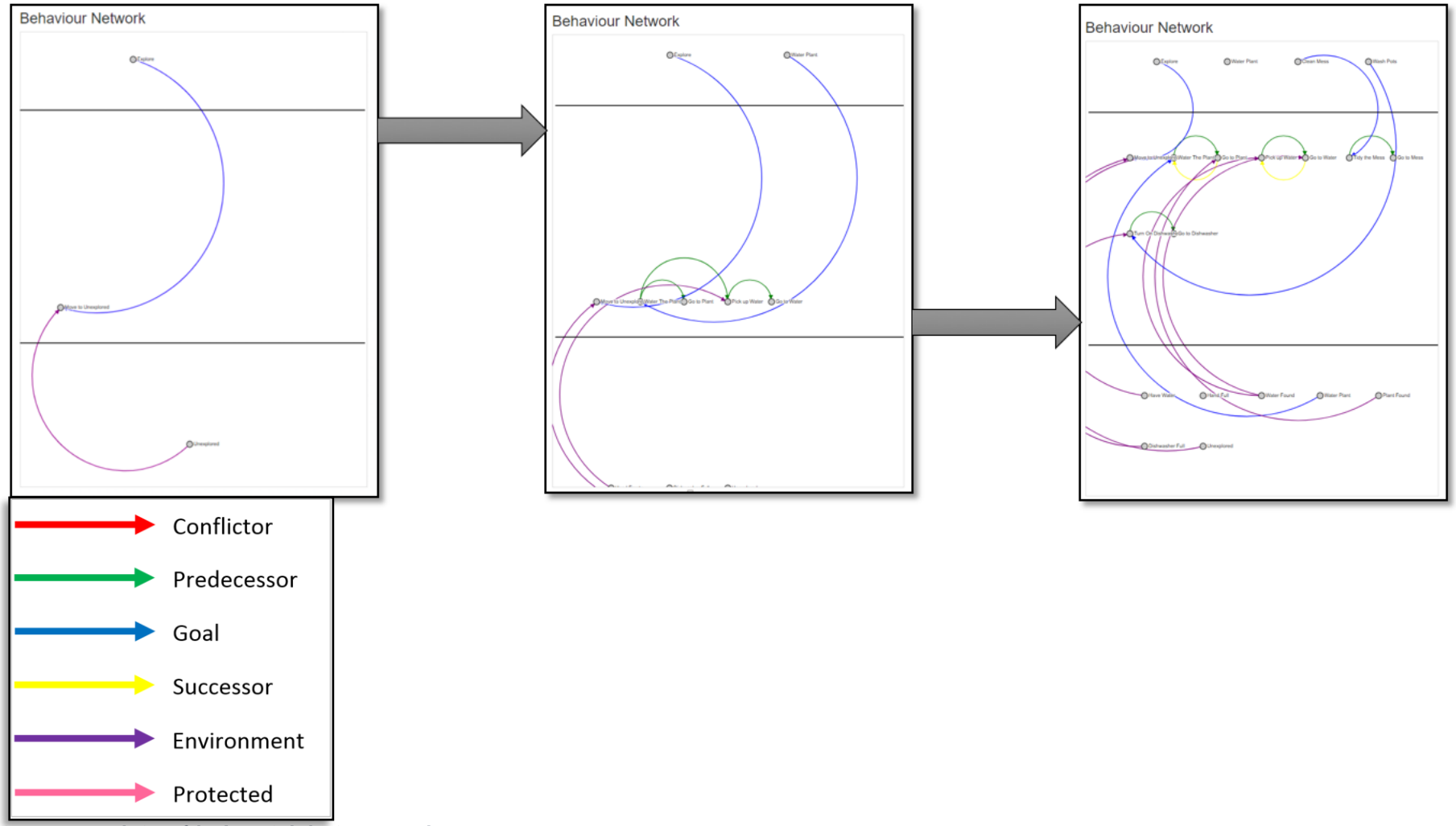

Figure 5-7 - Evolution of the dynamic behaviour network

Figure 5-7 shows the evolution of the behaviour network over time for this scenario. The first part of the behaviour network contains the initial goal and behaviour that allow the robot to explore the environment. The second part of Figure 5-7 shows that the behaviour network has grown and new behaviours have been added in real-time. The third part of Figure 5-7 shows the completed behaviour network towards the end of the simulation. Here the robot has located all of the objects in the environment and has completed one of the goals. At this point, each of the behaviours in the network are all competing with one another for activation.

The key difference between this scenario and the bake a cake scenario is the reduction of inputs to each behaviour. This scenario was designed to limit the number of behaviours in each sub-system (goals) to prevent the number of inputs to each behaviour. It was shown in the previous experiment that the behaviours with a large number of inputs from other behaviours had a greater chance of being selected for execution. While this was expected, it prevented the robot from advancing the world state and accomplishing the goals. The behaviour that the system exhibited as a whole was at fault. In this scenario, each behaviour only had a few inputs which allowed the behaviour network to select the appropriate behaviours and complete each goal.

This experiment was also run numerous times and although the objects (with the associated goals/behaviours) were spawned in different locations, the results were same. The agent was able to read the information, build and add to a behaviour network and achieve all of the goals. As the objects were placed in random locations in the environment, the order of executed behaviours was different. The time that it took the robot to complete the goals also differed; however, the robot was always able to successfully achieve the goals.

This experiment was conducted on a simpler test scenario, where the behaviours in the behaviour network, were at a higher-level than in the previous experiment. This is to remove the bias found during that experiment (Section 4.3.3.5) and to show that a dynamic behaviour network can successfully complete new goals. This approach was also conducted in (Nicolescu and Matarić, 2002), were a behaviour network used abstract behaviours, which fed into a lower-level behaviour network. This experiment makes the assumption that the higher-level (abstract) behaviour that is selected is completed successfully. Future work (Section 7.3), can involve extending this to incorporate the generation of the lower-level behaviour network. This experiment was successful in demonstrating the dynamic nature of the dynamic behaviour network using sub-systems without bias between the behaviours.

5.1.3. User Interactions

One of the benefits of the proposed architecture is that users can interact with the behaviour network by changing their preferences for different goals. In a traditional behaviour network, the goals of the system have their motivations predefined. The behaviour network will run, selecting and executing behaviours until those goals are completed. There are no options in these systems to restart completed goals or to cancel a goal without stopping the behaviour network and restarting the system. In the proposed architecture, a user interface is included to allow for dynamic manipulation of the behaviour network. The user interface is detailed in section 3.5.

The user interface is demonstrated using the previous, ‘bake cake’ scenario. In this experiment the transition when the robot navigates the environment and locates the kitchen with the wireless tag embedded is observed. When the Wireless Tag Manager reads the wireless tag, it will parse the information and send that to the

Behaviour Network Manager. When the Behaviour Network manager receives this information, it will send a send a message to the UI Manager with the information about the new goals that it has received. In real-time the users will be able to see new goals appear that the robot will be able to achieve. Figure 5-8 shows the initial state of the UI as the robot begins to navigate the environment.

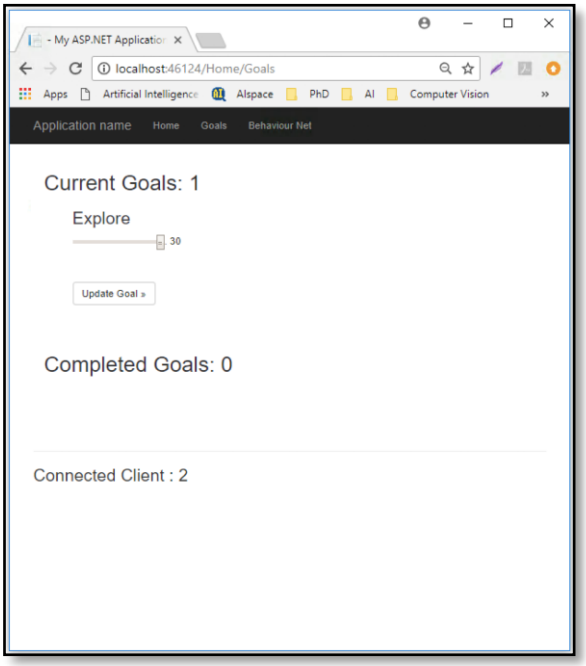

Figure 5-8 - Initial state of the Web GUI

When new goals are added to the web UI, the users can then manipulate the motivation for those goals. In the behaviour network the motivation for a goal indicates how much energy will be sent from the goal to the behaviours in the system. Reducing this value will decrease the amount of energy added to the system and allow other systems (such as the explore system) to become more likely to be achieved. Similarly, increasing the motivation value will make that system more likely to be selected over other systems.

Figure 5-9 shows the changes that the web UI will make when new goals are found in the environment. The first part of the figure shows the initial state of the UI when the experiment is first run. This shows that the only goal that the robot has is to explore the environment. The second part of the figure shows the new goal ‘Bake Cake’

has been added and the user now has the options to manipulate the motivation for that goal. The final part of the figure shows the status of the Web UI when the robot has completed the 'Bake Cake' goal. Here, the goal is moved to the completed column and the user now has the options to restart the goal and reset the motivation for that goal.

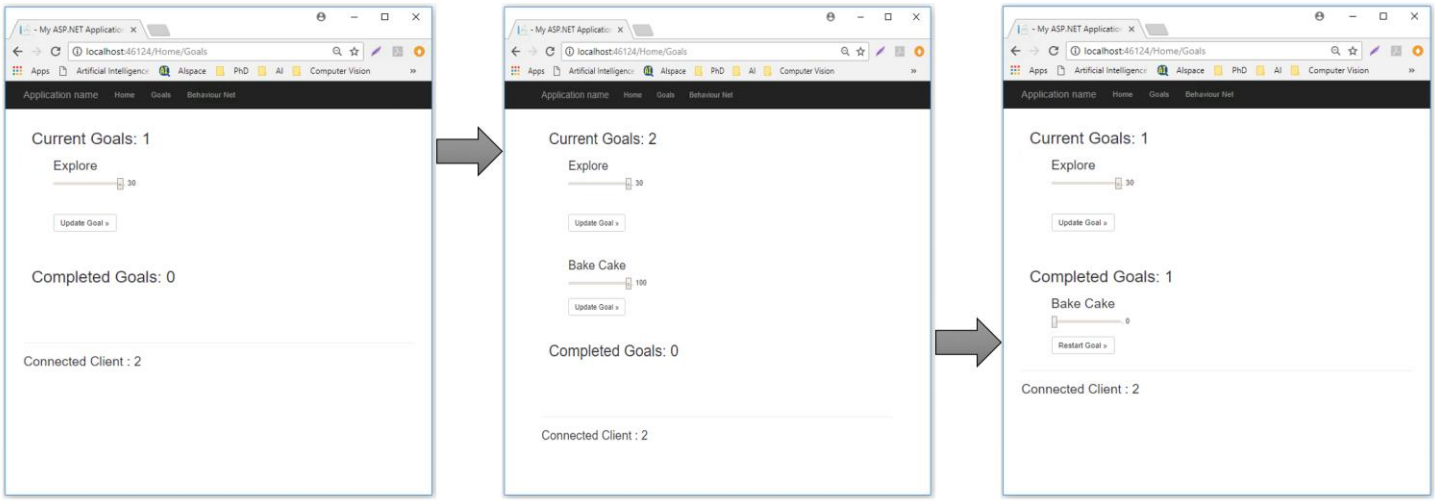

Figure 5-9 - Evolution of the Web GUI as the robot finds new goals and when goals are completed.

The functionality and effects of the web UI are further demonstrated to the dynamic structure of the behaviour network by changing the motivation for the goal 'Bake Cake'. When the goal is added to the web UI, the motivation is then significantly

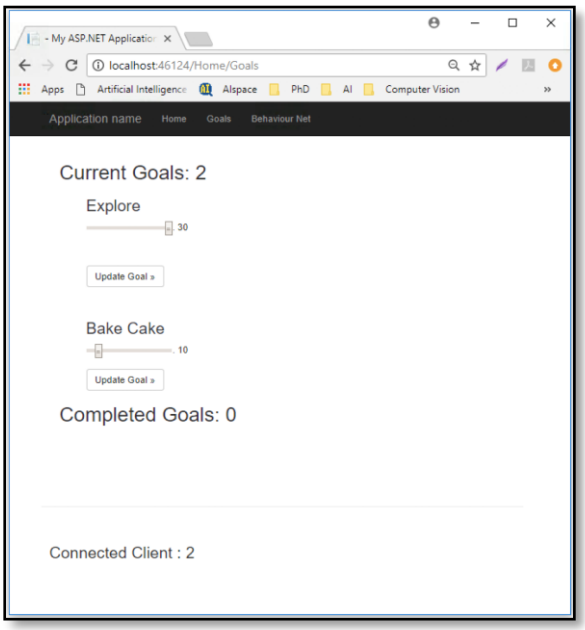

Figure 5-10 - Web GUI - Updating the motivation for the 'Bake Cake' goal.

lowered as shown in Figure 5-10. The new motivation is sent through the architecture from UI Manager to the Behaviour Network Manager, where the behaviour network is updated in real-time.

Figure 5-11 then shows the results of the experiment by making this change to the motivation of the 'Bake Cake' goal. Figure 5-11 shows that the robot (similar to previous experiments) continues to explore the environment until it locates the kitchen object. At this point the new goal is shown to be added to the network and the robot begins to execute some behaviours towards this new goal. At time-step 130 the motivation is altered as shown in Figure 5-11 and this is reflected in the robot's future action selections. Here the amount of energy in those behaviours is decreased and the robot favours the 'Explore' behaviour over the various behaviours in the 'Bake Cake' system.

The experiments in Chapter 4 acted as the controlled experiments for this test. From those experiments, it is clear how the behaviour network would run for this scenario without any user interaction or changes to the motivation of the goals at run time. Those results can be compared to the results from this experiment to show the difference. In the previous experiments, the robot would favour the behaviours in the Bake Cake system over the Explore system. When the motivation for the Bake Cake system is reduced then the behaviours in the Explore system is favoured. This experiment was run numerous times with different changes to the motivation at different times and again showed similar, expected results. This shows that there are different forms of a dynamic behaviour network. This dynamic network was able to change its preferences for which goals to achieve at run time which resulted in the whole system exhibiting different behaviours.

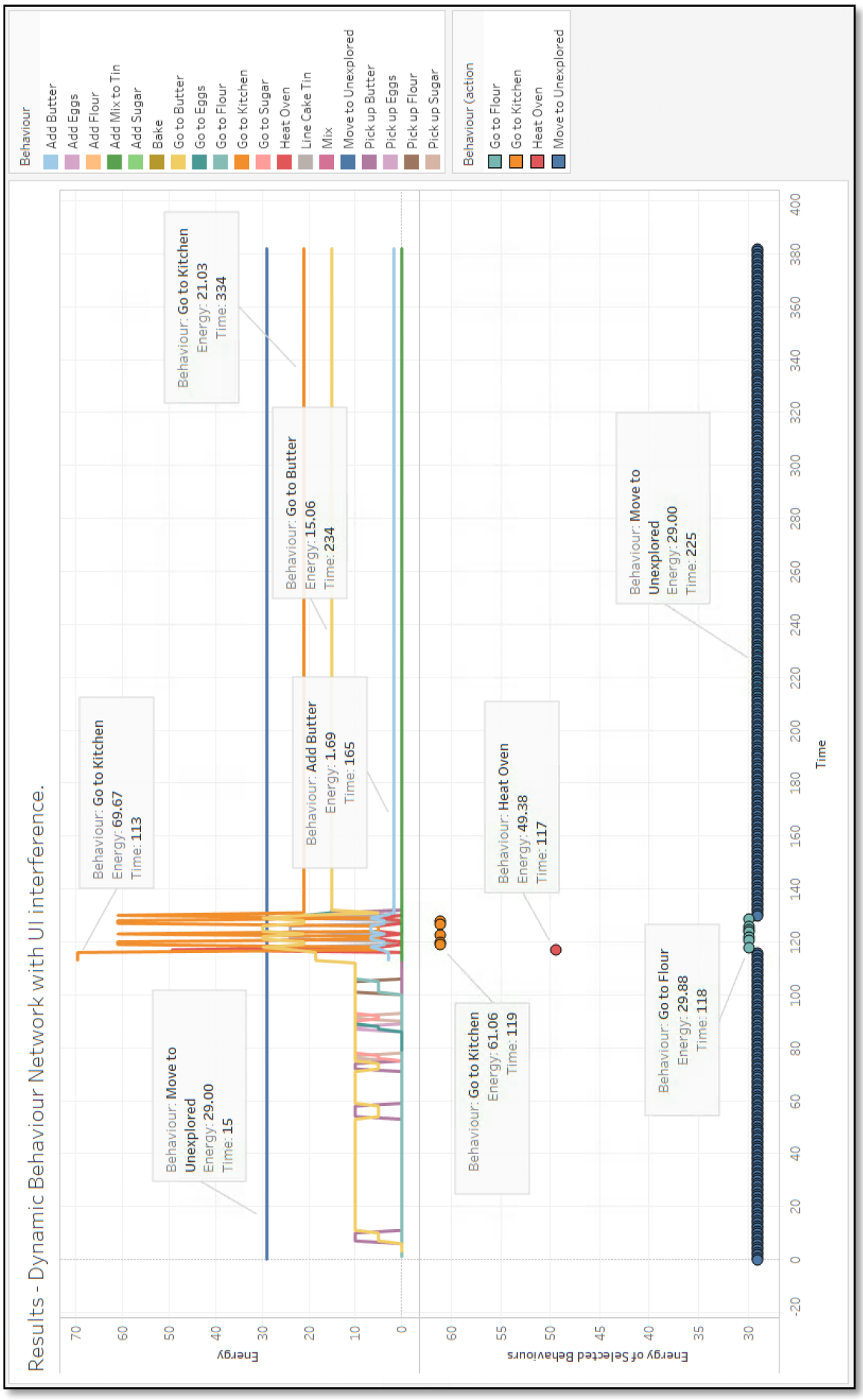

Figure 5-11 - Results from changing the motivation of a goal at run-time.

6. Discussion

In this chapter the contribution of this thesis and the potential for future work is discussed. The chapter begins with a discussion on behaviour networks, focusing on their benefits and disadvantages. This leads to a discussion on the proposed architecture to extend the standard and dynamic behaviour networks and how this has performed in testing. The changes that have been made to the standard behaviour network are then discussed and the results from the experiments reviewed. A discussion on the dynamic behaviour network then follows, discussing the concept and the experiments before reviewing the results. Finally, the potential future work that can expand upon the work in this thesis is discussed.

6.1. Overview of behaviour networks

The focus of this thesis is on how behaviour networks can be used and extended for unstructured environments (an environment that is not predefined and can change unexpectedly). A behaviour network is an action selection mechanism that consists of a variety of nodes, which are joined by a variety of links. The aim of the behaviour network is to pass activation energy between the nodes via the links until a behaviour(s) has its activation energy greater than a set threshold. The behaviour that has the largest proportion of energy and is also executable (has its preconditions met) is then selected for execution. A behaviour network can use a variety of different techniques to distribute activation energy around the network and this is discussed in detail in Chapter 2.

The behaviour network is a reactive system. At each time-step the agent will run the behaviour network, passing activation energy into the network from Goal and Environment nodes. That energy is then distributed through the behaviour nodes and

the behaviour that has accumulated the greatest amount of activation energy is selected for execution. The benefit of the behaviour network is the frequency with which the behaviour network is run; that is the frequency that energy is passed around the network. As this happens at each time step, the agent is able to make decisions based on the current state of the environment. Any change to the environment is taken into consideration and the agent can act accordingly.

While behaviour networks are capable of selecting behaviours in dynamic environments, they do have a limitation. The limitation is that each behaviour network needs to be predefined for the task that the agent is to accomplish. Each goal, behaviour and environment node are traditionally hard-coded to function together and allow the agent to make decisions in a predefined environment. This is because of the structure of the nodes in the network. For a node to connect with another node in the behaviour network, that node must have properties in the add list, delete list or precondition list of the other node. This relationship allows the activation energy to move from one node to the next and settle in the most appropriate node for the given situation. One of the goals of this thesis is to propose a potential solution to this limitation and show that behaviour networks do not need to be predefined for different tasks.

6.2. Proposed Architecture

The supporting literature on behaviour networks focuses on the structure and methodologies behind the behaviour network itself. Little focus has been spent documenting the architecture that can be used to support and enhance the functionality of a behaviour network. Chapter 3 presented an architecture that can support a behaviour network for both real-world and simulated agents. This architecture is broken down into the key modules that are used to move information from one component to

another. Chapter 3 concluded with a discussion on how those modules effect the functionality of the behaviour network.

6.2.1. Purpose of the proposed architecture

The aim of the proposed architecture is to enable the extension of the functionality of the standard behaviour network. The architecture is needed to support each requirement of the behaviour network to allow it to function correctly. For example, a behaviour network is designed for agents to interact in real-world or simulated environments. For this, an environment module is then included to manage the inputs from the environment and parse that information into something that the behaviour network can use. For the behaviour network, the environment module would need to manage the world state of the environment as that would affect the input from the environment nodes in the behaviour network. The world state would also need to stay updated as the different states would reflect which behaviours in the network could be selected.

6.2.2. Benefits of proposed architecture

The primary benefit is the concept of modularity that is used to divide the features of the behaviour network into separate modules. Each module is responsible for its own tasks and is capable of sharing information with other modules in the architecture. As each module is separate from the others, they are capable of parallel processing. This means that each module is executing synchronously, allowing the agent to process more information than if everything was in a single module / process. For example, the agent is able to process information from the environment whilst also taking input from other modules in the system.

As each module in this architecture is separated from the others, it does add an element of resilience to the behaviour network. Depending on the module, should one fail in the architecture, it is possible for the other modules to continue to function. Although the architecture will not have the same functionality without some of the other modules, depending on which module failed, it can continue to operate. For example, if the wireless tag manager failed, the other modules would continue to function. The robot would be able to explore the environment and achieve the goals it currently has but, it would not be able to learn anything new from the environment. Furthermore, during the implementation process, the modularity and resilience of the modules allowed for more accurate development of the behaviour network. It was simple to identify any areas of the system that were not functioning correctly and address those issues.

6.2.3. Disadvantages of proposed architecture

While the architecture was successfully implemented, and each module was able to add key features to the system, there are still some limitations to be addressed in the architecture. The main limitation that was observed during the implementation of the architecture was the different speeds that each module was able to perform their tasks at each time-step. It is important that each module works seamlessly with one another and no module is left waiting for another module to finish. The environment manager module was observed to take longer to process the information about the current state of the environment than other modules in the system. The result of this module taking longer to process information, was that the behaviour network manager module would create a behaviour network based on previous world states, which would lead to the behaviour network making inappropriate decisions.

The modules in the architecture communicated by sending the information to other modules when they had completed their tasks regardless of what the other modules where doing. To solve this issue, synchronization steps were taken to ensure that the modules would wait to receive all of the necessary information before executing.

6.2.4. Overview of the results

Each experiment of the behaviour network and dynamic behaviour network utilised some modules of the proposed architecture. The experiments for Chapter 4 utilised the environment manager and the behaviour network manager to create a basic behaviour network. During these experiments additional changes were made to the environment manager to ensure the modules were synchronised to prevent incorrect behaviours being selected. Chapter 5 introduced the remaining modules, including; the Environment Manager, the Wireless Tag Manager and the User Input manager for the dynamic behaviour network. Adding these modules showed the benefit of the modularity of the proposed system, as they were included with very minor changes being needed for the other modules.

Overall, the architecture was able to successfully support the standard behaviour network with the environment manager module and the behaviour network module. The architecture was able to also successfully support the dynamic behaviour network with very minor alterations being needed. The synchronicity of the modules proved to be a challenge for the architecture. This was because each module was able to complete its tasks at different speeds. To be able to pass information around the system during each time-step, some alterations were then made to the architecture. This was also

tested during the experiments in chapter 4 and chapter 5 and it was demonstrated that the modules were able to function synchronously and follow the same iteration.

One of the aims of this project, which was discussed in Section 1.3.1, was to develop an architecture that could be used to test behaviour networks and dynamic behaviour networks. The proposed architecture was defined in Section 3.1 and used for the experiments in Section 4 and Section 5. This section has discussed the benefits and limitations of the proposed architecture and concludes that this objective has been met.

6.3. Behaviour Networks

The behaviour network is the primary motivation of this thesis. This section begins with a discussion of the first contribution of the thesis, which is the modification to the division rule. An alternative method of sending data through a behaviour network, called the data packet approach, is then discussed. The results from the various experiments are reviewed and discussed. Finally, the limitations of behaviour networks that were found during these experiments are explored and the section concludes with potential future research.

6.3.1. Discussion of the division rule

Chapter 2 describes the inner workings of a behaviour network and explains the problems with the current methods for spreading activation energy through the behaviour network. As described in chapter 2, a portion of energy from a behaviour is passed through the various links to other behaviours in the network. The type of the link, predecessor, conflictor or successor (to name a few), affects which formula is used to calculate the proportion of energy to send. It was shown that the proportion of energy, distributed throughout a network, would depend on the number of inputs and outputs of a behaviour which led to unfair competition between the behaviours of the system.

The problem resides in whether or not the calculation of energy spread should involve division by the number of predecessor and conflictor input links. This concept of a division rule has worked successfully on other link types; however, it has been shown that the inclusion and exclusion of a division rule for predecessor and conflictor links can still lead to unfair distribution of energy in the network. Tyrrell (Tyrrell, 1994) discussed a variety of solutions to this problem and concluded that, at the time, there was no solution to this problem.

In chapter 4, a solution to the division rule was presented. The solution details how knowledge of how many goals that each behaviour can contribute to can be used to select a more accurate division rule dynamically. An algorithm for embedding this goal information onto the links of the network and the updated division rule formulae were presented and discussed. The new division rule was then tested on a basic behaviour network and the results were compared with the same experiment but without the division rule.

The aim of this experiment was to find a division rule that can be applied to any situation. It has been shown that different division rules can have different effects on the network (Tyrrell, 1994). An ideal division rule would ensure that there is no unfair competition between the behaviours of the network and that each behaviour has a fair chance of getting selected. The results from this experiment were indecisive as the first round of energy spreading would select different behaviours as opposed to if it was run for multiple iterations. The behaviour network should be able to select the most appropriate behaviour at any time step; it should not need to run for multiple iterations for it to converge on a solution. The supporting text (Tyrrell, 1994) does make use of a threshold parameter allowing the network to run for multiple iterations. It is argued that this was a fix to the problem of selecting the correct order of behaviours in which to

spread energy. To address this problem a new method of spreading energy through the network was also presented (the energy packet approach). One limitation of the division rule was not on the division rule itself but in the method of how energy is distributed through the network. The results showed that it needed multiple iterations of spreading energy to make a decisive decision. However, it is argued that regardless of the iteration, the behaviour network should always be selecting the best behaviour for the given situation. That decision should not change based on an arbitrary number of energy spreading iterations. The later experiments in chapter 4 and chapter 5 demonstrate that with the use of energy packets, correct behaviours can be selected after a single iteration of energy spreading.

6.3.2. Discussion on the data packet approach

During the implementation process of the behaviour network, it was noted that there had been little discussion of the order of energy spreading amongst nodes in the network and the impact of choosing an inappropriate node order. The supporting text has detailed how energy enters the network and how the energy should move around but not about the order. In chapter 4, the effect that the different orders of energy spreading in the behaviour network can have on the distribution of energy was explored. The results were presented by selecting a variety of different orders and showed that different orders led to different distributions of energy in the network. The results also showed that multiple iterations of the energy spreading process would affect the resulting distribution of energy in the network. The possibility of feedback loops were also discussed in this chapter, where there can exist a series of behaviours that each feed into one another. In the current version of the behaviour network, this would create a loop of energy passing that would not end.

A solution to the problem of loops was then proposed, the concept of data packets. Instead of simply sending an arbitrary value representing energy around the network, a data packet is sent in its place. The data packet will contain a variety of metadata including the proportion of energy to send from one behaviour to next. Each packet will also contain a history of where it has been in the behaviour network, allowing it to escape node loops and to traverse the full network. The goal of the data packets is to ensure that each behaviour can send a proportion of energy, that would achieve the desired results, to the next behaviour and ensure that there is no unfair competition in the distribution of energy.

This concept was tested with a variety of incrementally more complex experiments. It began with a behaviour network consisting of a single link type. The aim of the experiment was to distribute the energy packets through a behaviour network. This experiment showed that the energy packets were able to be successfully distributed through behaviour network. This experiment was then extended to one with multiple link types. The aim of this experiment was similar to the previous experiment, where the goal was to distribute energy packets through a behaviour network but one with multiple link types. Again, this experiment (using a basic behaviour network) was able to show similar results in that the energy packets were successfully distributed through the network. The next experiment in this section involved creating a new, more complicated test scenario. We presented the scenario of baking a cake as it is a logical problem with few correct sequences that an agent would need to follow. The aim of this experiment was to test the energy packets in a more complicated real-world scenario. To also show that the energy is distributed to behaviours would achieve the goals of the system. Once more, the results from this experiment were presented showing that the energy packets were successfully distributed, and the agent was able to select the behaviours to

complete the goal. The data packets were able to navigate the behaviour network fully and the behaviour network did not need to run for multiple iterations to converge on the solution. To achieve this, the data packets in each behaviour were merged to ensure the correct proportion of energy was distributed and the link order for sending the data packets was defined. This experiment was able to show that the data packet approach can be used in a real-world situation in behaviour networks to achieve goals.

The final experiment extended the baking a cake scenario further by utilizing the proposed architecture with a simulated unstructured environment. The goal of this experiment was to have the agent collect the distributed ingredients and perform the necessary behaviours to complete the goal. This experiment also used the data packet approach to distribute energy in the network. It was shown that this experiment yielded poor results as the agent was unable to accomplish the goal of baking a cake within the simulated environment. In this experiment the agent would explore the environment until it found the 'kitchen', at this point the agent was able to perform some behaviours towards the goal. The robot then needed to navigate to an ingredient to bring it to the kitchen, to then execute more behaviours with that ingredient. As soon as the robot left the kitchen, its next decision was to return back to the kitchen (before it had made it to the ingredient). This is because when it was no longer in the proximity of the 'kitchen' the motivation to go to the kitchen was too great and outclassed the other behaviours. When the robot was in the kitchen, there was little it could do but leave towards an ingredient.

This result shows a good example of one of the fundamental limitations of the behaviour network. As the behaviours in the network are selected when they have the most energy and one way to gain the most energy is to be the behaviour with the most inputs. The division rule that was previously discussed worked on the inputs of a node

and on restricting the output. However, if the node with the greatest number of inputs is the final destination of the activation energy then there is a greater chance of that behaviour being selected.

One potential solution to this problem could be to add a method to detect loops and break the agent out of the loop when it occurs. However, this does not solve the problem and the agent could return back to the loop soon after. The division-rule was discussed by (Tyrrell, 1994) as a technique to divide by the number of inputs or goals but, as previously discussed, was proven to be ineffective. Section 4.1 presented an updated division rule by incorporating goal information into the division rule formula. This also yielded unsatisfactory results as it would inhibit behaviours achieving multiple goals. Another solution would be to reduce the complexity of the network by using higher-level behaviours. A higher-level behaviour being a behaviour that includes many sub-behaviours. The aim is to remove the bias from the network which is caused by behaviours having many more inputs than other behaviours in the network. The final technique was to reduce the complexity of the network by using high-level behaviours and was suggested in section 5.1. This is method is further discussed in the next section. This limitation of behaviour networks will form a strong basis for future research to address which, is discussed in more detail in section 7.5.

6.3.3. Overview of the results

One of the aims of this study was to investigate behaviour networks (discussed in Section 1.3.1). Section 2.2.4 explored behaviour networks and identified some limitations to the methods that are used to spread activation energy through a network. Two new techniques were proposed; a new division rule (Section 4.1) and the data packet approach (4.3.1). As previously discussed, the results from the new division rule did not

show any significant improvements, however; embedding goal specific information to the links of a behaviour network did enable easier debugging of a behaviour network and inspired the second technique (data packets). The data packet approach (discussed in Section 4.3.1) was able to show better results. This method showed that this new approach could solve problems (such as; loops in the network and the order for spreading activation energy through a network). Although, in some experiments the agent was not able to fully complete the scenario (achieve all of the goals), the reasoning behind this was due to the structure of the network and not the activation spreading mechanism. The objective to investigate behaviour networks and the activation spreading mechanisms has been achieved and Section 7.3 details the future work that can expand upon the methods presented in this section.

6.4. Dynamic behaviour network

One of the main contributions of this thesis is the concept of the dynamic behaviour network, which is introduced in chapter 5. A dynamic behaviour network is defined as a behaviour network which can dynamically control and modify the goals and behaviours within the network. The numerous benefits that such a system would offer compared to the traditional behaviour network are discussed. The ability to allow a behaviour network to grow without any modification to the architecture will allow agents to achieve more than they were originally designed to accomplish. This is achieved by distributing behaviours and goals in an environment for an agent to find and add to their behaviour network. To distribute the behaviours into the environment, the use of wireless tags was proposed (although other technologies can also be used to demonstrate this concept) to embed information in to the objects in an environment. When the robot enters the environment, it will read the data stored on a wireless tag

and use that information to update its behaviour network allowing it to interact with those objects without prior knowledge.

To test this new concept, the experiments used in chapter 4 were extended. The bake a cake scenario was extended to incorporate the distributed behaviours. Instead of the robot having prior knowledge about each object, it now has to navigate the environment and find each object to learn how to interact with them. This aim of this experiment was to show that distributed behaviours can be added to a behaviour network at run-time. The robot began the experiment with the only ability to explore the environment, ensuring that the robot had to explore and learn the other behaviours and goals. The results of this experiment were, overall, positive as it was shown that the robot was able to successfully navigate the unstructured environment and learn all of the distributed behaviours and goals. All of these were successfully added to the robot's behaviour network in real-time and the robot was able to begin performing behaviours from the new selection. However, the robot was still unsuccessful in completing the new goal (bake cake) in the simulated environment. The results were similar to the previous experiment, where the robot was given the goal of baking a cake in the simulated environment. The results showed that once the robot was in the proximity of the 'kitchen', the same unfair bias was present, and the robot was stuck in a loop. When the robot entered the kitchen, it would perform a behaviour within the kitchen's proximity (turn on oven). The network would then select to move towards an object to interact with, as at that time it is the best thing to do. When the robot leaves the proximity of the kitchen, the network would choose to go back to the kitchen. This is because the behaviour to go to the kitchen has more inputs then the go to object behaviour. While this may not be the best behaviour to perform, the behaviour still receives a greater portion of energy. Section 6.5 discusses a potential solution to this problem.

To validate that the previous result was due to the unfair bias in the network, a new scenario to test the distributed behaviours was then proposed. In the next experiment the objects in the environment each had different goals and behaviours. The aim for this experiment was to test a more abstract network where the unfair bias in the previous experiment was no longer present. By having a behaviour network with more abstract behaviours it was theorised that those behaviours would have fewer inputs and outputs resulting in less bias in the network. In this experiment, the results showed that the robot was able to successfully navigate the environment and collect all of the behaviours and goals. The results also show that the robot was able to successfully select and execute each behaviour to accomplish all of the goals that it was able to find.

The primary aim of this study was to investigate behaviour networks with distributed behaviours and goals. This developed into the dynamic behaviour network which has been tested on a variety of test scenarios. As previously discussed, the results from this technique showed that a behaviour network can successfully grow and adapt new behaviours and goals into an existing behaviour network. While the agent was only able to complete the goals from a simplified network the core concept (distributed behaviours and goals) was shown to work successfully and opens the field of behaviour networks for future work.

6.5. Limitations and Future work

The research presented in this thesis has made significant contributions to the field of behaviour networks. However, there are still limitations in this field that open up new areas for further research. This section will discuss these limitations and present potential areas for future research.

6.5.1. Connection mechanism of internal behaviours

The behaviours of a behaviour network were described in detail in chapter 3. Behaviours in the network are connected via different types of link, depending on the state of the environment. For example, if there is an observation in the add list of one behaviour (a) and that same observation is in the precondition list of another behaviour (b) then there is an active predecessor link from behaviour (a) to behaviour (b). During the development process of the behaviour network it is observed how fragile this connection mechanism is. The values in the add list, delete list and precondition list must match exactly for a link to be created. If a new behaviour is introduced where these values do not match, then a link will not be created, and the behaviour network will show poor results. This is especially important when discussing the concept of the dynamic behaviour networks as there could be numerous creators of different behaviours. Those creators will need to know all possible world states to correctly create the add list, delete list and precondition list of a behaviour. For example, if a behaviour was designed to pick up an object and all possible world states were not defined, then the precondition list could miss a proposition on that world state. This would result in the behaviour being selected when it may not be an appropriate time to select the behaviour.

There is the potential for future research in the area of an ontology that all creators of the behaviours and goals can follow. This will ensure that the values in the add list, delete list and precondition list will always match and each link in the network is correctly created. Schlenoff et al. (Schlenoff *et al.*, 2012) is developing a standardised robotic ontology for knowledge representation and reasoning in robotics. Alternatively, fuzzy matching techniques could be adopted to ensure that the propositions in the add list, delete list and precondition list match what is in the world state or other behaviours.

Fuzzy-based matching is a popular technique for matching text that is similar but not exactly the same and has been applied to a variety of different domains (Hu and Liu, 2004).

6.5.2. Binary nature of behaviours

Behaviours in a traditional behaviour network are binary in nature. This refers to the observations / conditions in that behaviours add list, delete list and precondition list. Those observations can only be fully satisfied. For example, an observation could be 'near object' which is either true or false. There is no degree of freedom in this example, so an agent could be a few metres from an object in a different room but the observation will still be treated as false. Future research can be included in changing the fundamentals of a behaviour network to handle degrees of freedom for the observations. The environment nodes could pass more energy depending on the relevance of the situation. Following from the previous example, the environment nodes would pass more energy to the behaviours that are related to an object depending on how close the robot is to that object.

Additionally, the observations cannot be a combination of multiple observations. For example, a behaviour could be required to interact with either one object or another object but not necessarily both. In logic gate theory this would be represented by an OR gate, however behaviour networks can only handle AND gates. This is because, in the traditional behaviour network, every proposition in a behaviour's precondition list must be true for that behaviour to be selected. The precondition list cannot have one proposition to be true and one to be false and be selected for execution. Traditional behaviour networks are not designed for this type of selection and fundamental changes would need to be made to the structure of a behaviour. There is potential for future

research in the different types of logic gates and how they can be reflected as behaviours in a network.

Finally, it is not currently possible to partially satisfy a behaviour's or goal's preconditions. This is again because of the binary nature of behaviours. For example, there could be a goal to clean a room. The robot could have already spent some time working towards this goal and the room is now partially tidy, let us say the room is 80% clean. The robot is then given another goal to bake a cake with similar motivation. In this example, it is not clear which goal the robot would try to satisfy first. One goal is 80% complete while the other is 0% complete. There should be additional motivation added to the behaviours of the system that are close to being completed. This would allow the robot to complete goals in a timely manner and allow for multiple goals to exist concurrently without the possibility of the robot attempting to complete them all simultaneously. While criteria such as urgency, completeness and persistence have already been defined in other AI architectures (Di Rocco *et al.*, 2013), the fundamentals of the behaviour network would need to be modified to accommodate these criteria. For example, more motivation could be added to the behaviours of a network that help to achieve a goal that has already met some of its preconditions.

6.5.3. Parameter choosing for behaviour networks

The supporting text on behaviour networks detail the various parameters used when building a behaviour network and this was discussed in chapter 2. The supporting text details what parameters should be included in a behaviour network but not what values to use or how to set the values of the parameters. In the various experiments discussed throughout this thesis, the same values were used for the parameters of the network. These were decided by selecting a random set of values and testing a traditional

behaviour network. However, it is likely that the parameters used in these experiments could have been optimised and that each experiment could have used a different set of parameter values.

Future research in the area would involve testing a traditional behaviour network with a variety of different parameter tuning techniques and reporting on which technique would be best for a behaviour network. Some techniques that could be used include random search (Bergstra and Bengio, 2012), grid search (Lameski *et al.*, 2015) and genetic algorithms (Eiben and Smit, 2011). It is also possible that the parameters of the behaviour network cannot be perfectly tuned or adequately justified for all circumstances. This is because the behaviour network is designed to work in a changing environment and tuning the network to one environment may not work as well in another environment. This area should still be further explored as there could be a generic set of parameters that works well for most environments.

6.5.4. Additional metadata on wireless tags

Future research on the functionality and storage of wireless tags would enhance the work in this thesis on dynamic behaviour networks. This thesis currently details how wireless tags can be used to store goal and behaviour information about objects in an environment for a robot to access and utilise in real-time. However, there is the potential to store more information about objects on those tags. For example, a wireless tag could store meta-data relating to how an object can be interacted with. This meta-data could describe how to pick up an object, how much pressure to apply, the rotation of the object when placing it down and the associated weight of the object. This could be fed into the behaviours of the network and allow for smarter interaction of objects without any prior knowledge of those objects.

Further research can also be performed on the computer vision element of robotics and the data that can be stored on the wireless tags. The experiments conducted in this thesis were performed in a simulated environment where computer vision techniques were not need for the identification of objects in an environment. It is possible however to store data on the tags relating to the visualisation of the objects. Basic meta-data such as colour, shape and size can be used for simple object detection. Larger files, such as 3D models, could be stored on the tag (depending on storage) or references to location for those files (web) could be embedded on the wireless tags. Behaviours or code instructions could also be added to the wireless tag to assist with identifying the object in the environment. The benefit of doing so would be to allow the robot to identify the objects in the environment, even those of which it has no prior knowledge. It is also possible that by embedding this metadata in the wireless tags that it would also help the robot to identify objects faster than using conventional methods.

7. Conclusion

This chapter concludes the thesis by beginning with a summary of the whole thesis, the contributions of the thesis and then ending with a summary of potential future work. This chapter discusses how the outcome of this thesis achieves the research question detailed in Chapter 1.

7.1. Summary of thesis

The aim of this thesis is to demonstrate the technical feasibility and usefulness of embedding wireless tags to objects in a dynamic environment to assist the everyday tasks of a robot. To achieve this research question, the objective is defined to be exploring the use of embedded data in unstructured environments to improve the capabilities of a robot.

Behaviour networks were used for the action selection mechanism of the robot as the behaviour network is a reactive system capable of functioning in an unstructured environment. The concept of distributing object specific data into wireless tags in an environment was proposed. By adding new goals and new behaviours to the wireless tags, the work is able to show that a robot with limited functionality can be enhanced with data found in an environment. The fundamentals of the traditional behaviour network were further explored, and key limitations were identified before presenting the approaches to potential solutions. The next sub-chapter discusses the contributions of this thesis.

7.2. Contributions to research

This thesis aimed to demonstrate the technical feasibility and usefulness of embedding wireless tags to objects in a dynamic environment to assist the everyday tasks of a robot. Behaviour networks were chosen due to their reactive action selection mechanism for

use in dynamic unstructured environments. Within this thesis, behaviour networks were closely examined and tested in a variety of situations. The different areas of behaviour networks were tested, and improvements were made along the way. The contributions of this thesis are summarised below:

1. Updated division rule

   The limitations of the standard division rule in traditional behaviour networks are explored. This is discussed in chapter 3. Changes to the division rule in chapter 4 were proposed. The division rule was documented in (Tyrrell, 1994) and refers to an unfair distribution of energy in a network when a behaviour has multiple inputs and needs to transfer a proportion of energy to the next behaviour. It is shown that by embedding goal-related meta-data in to the links of the network that the system can select a more appropriate division rule. This method was tested and documented in chapter 4.

2. New approach for spreading energy through a network

   Another limitation with behaviour networks was identified. The order in which the nodes of the behaviour network are evaluated, affects the distribution of energy. It was demonstrated that multiple iterations of the energy spreading process also affected the distribution of energy in the network. In chapter 4, a new approach called the data packet approach was presented. This technique distributed energy in packets of data with associated meta-data allowing the behaviour network to distribute energy more accurately. This technique was tested in a variety of different experiments and the results were presented.

3. Proposed architecture for behaviour networks

   In chapter 3, a modular architecture for use with both traditional behaviour networks and the dynamic behaviour network was presented. The architecture was designed to work with both types of behaviours network in both simulated and real-world scenarios. The modularity of the different areas of the architecture allowed for quick transitioning between the different networks each with different requirements. The architecture was used in each of the experiments of this thesis and the results were documented.

4. Dynamic behaviour network for distributed behaviours

   The dynamic behaviour network is the main contribution of this thesis, aside from the improvements made to the traditional behaviour network. Chapter 5 detailed the dynamic behaviour network and the experiments that were used to test it within a simulated environment. The dynamic behaviour network was able to demonstrate that a robot could enter an unstructured environment and interact with new behaviours of which it had no prior knowledge to achieve goals that it learned within the simulated environment.

The findings from the various experiments conducted in this thesis have demonstrated the validity and the feasibility of distributing data in an environment to assist a mobile robot. Successful improvements to the traditional behaviour network have been provided and shown to give positive results. Finally, this thesis has provided an in-depth discussion into the current limitations of behaviour networks and provided suggested areas for future research.

7.3. Future Work

This thesis has shown how behaviour networks can be used to navigate unstructured environments and how embedding data into the environment can improve the capabilities of a robot. This thesis also discusses the limitations of behaviour networks in chapter 7. The following areas for future research are then recommended:

1. An ontology for behaviours

   It is especially important for dynamic behaviour networks to follow an ontology for the different terms used in a behaviour's add list, delete list and precondition list if new behaviours are to be developed from outside the system. This is because each distributed behaviour will need to follow a defined set of terms in order to create successful links between the behaviours in the network. Fuzzy matching could be applied to combine different terms for the same thing. A common grammar would also be needed as the order and meaning of the words / terms in the behaviours would greatly affect how they are matched together.

2. Utilizing the relevance of observations

   A behaviour network should take into consideration the relevance of an observation when determining a proportion of energy to send into the network. Observations of high relevance to a given situation can provide more energy than those with less relevance. The proportion of extra energy to be added and the factor of relevance to a situation would need to be further researched. A definition for relevance for a behaviour network will need to be described.

3. The use of logic gates in behaviour networks

   There are situations that were observed during the development of the behaviour network where the behaviour's observations could have been combined and where

a behaviour's preconditions could have been satisfied if only one of the listed observations were met. For example; a behaviour could be selected if an agent was in either (location a) or (location b).

4. Prioritising goals close to completion

   Research can be conducted into the effects of prioritising goals that are close to completion. When a goal is partially completed it can pass into the network a higher proportion of energy. This would make the goal more likely to be achieved than new goals that recently added. The benefit of this would be to ensure that the system is not left with many incomplete goals especially when new goals are added.

5. Parameter tuning techniques

   Chapter 6 discusses the limitation of the documentation on behaviour networks regarding choosing an adequate set of parameters. Further research can be conducted into the various different parameter tuning techniques for use with behaviour networks.

6. Additional data on tags for object recognition

   This thesis has demonstrated the technical feasibility of embedding data into an environment. The work in this thesis was conducted in a simulated environment however; should this be extended to a real-world experiment then further research will be needed for the recognition of objects in the environment. It is suggested that further research in the data that can be stored on a wireless tag (such as meta-data or 3D models) to assist with the object recognition of objects that a robot has no prior knowledge about.

References

Arkin, R. C. (1987) 'Path planning for a vision-based autonomous robot', in *Cambridge Symposium_Intelligent Robotics Systems*. International Society for Optics and Photonics, pp. 240–250.

Arkin, R. C. (1998) *Behavior-based robotics*. MIT press.

Asimov, I., MYSTERY, R. and TIEDEMANN, M. W. (1941) 'Three laws of robotics', *Internet].[cited 2015 Aug 3]. Available from: http://www. auburn. edu/~ vestmon/robotics. html*.

Asimov, I. (1963) *I, Robot*. Doubleday (Doubleday science fiction). Available at: https://books.google.co.uk/books?id=P9zPAAAAMAAJ.

Ballard, D. H. (1981) 'Generalizing the Hough transform to detect arbitrary shapes', *Pattern Recognition*, 13(2), pp. 111–122.

Bergstra, J. and Bengio, Y. (2012) 'Random search for hyper-parameter optimization', *Journal of Machine Learning Research*, 13(Feb), pp. 281–305.

Bolmsjo, G., Neveryd, H. and Eftring, H. (1995) 'Robotics in rehabilitation', *IEEE Transactions on Rehabilitation Engineering*, 3(1), pp. 77–83. doi: 10.1109/86.372896.

Boyan, J. A. and Moore, A. W. (1995) 'Generalization in reinforcement learning: Safely approximating the value function', *Advances in neural information processing systems*. MORGAN KAUFMANN PUBLISHERS, pp. 369–376.

Brooks, R. (1991) 'Intelligence without Reason', *Artificial Intelligence*, 47(1–3), pp. 139–159. doi: 10.1007/BF01538672.

Brooks, R. A. (1986) 'A Robust Layered Control System For A Mobile Robot', *IEEE Journal on Robotics and Automation*, 2(1), pp. 14–23. doi: 10.1109/JRA.1986.1087032.

Brooks, R. A. (1995) 'Intelligence without reason', *The artificial life route to artificial intelligence: Building embodied, situated agents*, pp. 25–81.

Bryson, J. (2000) 'Hierarchy and sequence vs. full parallelism in action selection', *Proceedings of the Sixth International Conference on Simulation of Adaptive Behavior*, pp. 147–156. Available at: http://books.google.com/books?hl=en&lr=&id=Q4w6EvRbySAC&oi=fnd&pg=PA147&dq=Hierarchy+and+Sequence+vs+.+Full+Parallelism+in+Action+Selection&ots=QmVrcv8DIf&sig=FUd0lKV5vboKh1E0qrwS7ldluDQ.

Carreras, M. (2004) *A PROPOSAL OF A BEHAVIOR-BASED CONTROL ARCHITECTURE WITH REINFORCEMENT LEARNING PhD Thesis A Proposal of a Behavior-based Control Architecture with Reinforcement Learning for an Autonomous Underwater Robot*, *Camera*.

Chae, Y.-J. and Cho, S.-B. (2014) 'Planning-driven behavior selection network for controlling a humanoid robot', in *Neural Networks (IJCNN), 2014 International Joint Conference on*, pp. 4244–4250.

Chow, T. S. (1978) 'Testing software design modeled by finite-state machines', *IEEE transactions on software engineering*, 4(3), p. 178.

Cifuentes Costa, S. (2013) 'Behaviour blending for multiple robot coordinated navigation through virtual potential fields. Integración de comportamientos para la navegación coordinada de múltiples robots mediante potenciales virtuales'.

Clarke, R. (1993) 'Asimov's laws of robotics: implications for information technology-part I', *Computer*, (12), pp. 53–61.

Clarke, R. (1994) 'Asimov's laws of robotics: Implications for information technology. 2', *Computer*, 27(1), pp. 57–66.

Collins, S. H. and Ruina, A. (2005) 'A bipedal walking robot with efficient and human-like gait', in *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, pp. 1983–1988.

Corporation, Ir. (2007) *Robots that make a difference - iRobot*. Available at: http://www.irobot.co.uk/ (Accessed: 26 January 2015).

Cutumisu, M. and Szafron, D. (2009) 'An Architecture for Game Behavior AI: Behavior Multi-Queues.', in *AIIDE*.

Diekmann, T., Melski, A. and Schumann, M. (2007) 'Data-on-network vs. data-on-tag: Managing data in complex RFID environments', *Proceedings of the Annual Hawaii International Conference on System Sciences*, pp. 1–10. doi: 10.1109/HICSS.2007.160.

Dolgov, D. and Durfee, E. H. (2002) 'Satisficing strategies for resource-limited policy search in dynamic environments', in *Proceedings of the first international joint conference on Autonomous agents and multiagent systems: part 3*, pp. 1325–1332.

Eiben, A. E. and Smit, S. K. (2011) 'Parameter tuning for configuring and analyzing evolutionary algorithms', *Swarm and Evolutionary Computation*. Elsevier, 1(1), pp. 19–31.

Erol, K., Hendler, J. A. and Nau, D. S. (1994) 'UMCP: A Sound and Complete Procedure for Hierarchical Task-network Planning', in *AIPS*, pp. 249–254.

Ferguson, D. and Stentz, A. (2007) 'Anytime, dynamic planning in high-dimensional search spaces', in *Robotics and Automation, 2007 IEEE International Conference on*, pp. 1310–1315.

Fikes, R. E. and Nilsson, N. J. (1971) 'STRIPS: A new approach to the application of theorem proving to problem solving', *Artificial intelligence*, 2(October), pp. 189–208. doi: 10.1016/0004-3702(71)90010-5.

Fitzgerald, C. (2013) 'Developing baxter', *IEEE Conference on Technologies for Practical Robot Applications, TePRA*. doi: 10.1109/TePRA.2013.6556344.

Gates, B. (2007) 'A robot in every home.', *Scientific American*, 296(1), pp. 58–65. doi: 10.1038/scientificamerican0208-4sp.

Gudwin, R. *et al.* (2018) 'An urban traffic controller using the MECA cognitive architecture', *Biologically Inspired Cognitive Architectures*. Elsevier.

Halal, F. and Zaremba, M. B. (2018) 'A Hybrid Architecture for Planning and Execution of Multi-Behavior Data Acquisition Missions', in *2018 IEEE International Conference on Computational Intelligence and Virtual Environments for Measurement Systems and Applications (CIVEMSA)*, pp. 1–6.

Harnad, S. (1990) 'The symbol grounding problem', *Physica D: Nonlinear Phenomena*, 42(1), pp. 335–346.

Hart, P. E., Nilsson, N. J. and Raphael, B. (1968) 'A Formal Basis for the Heuristic Determination of Minimum Cost Paths', *IEEE Transactions on Systems Science and Cybernetics*, 4(2), pp. 100–107. doi: 10.1109/TSSC.1968.300136.

Hayes, P. J. (1971) *The Frame Problem and Related Problems on Artificial Intelligence*. Stanford University, Computer Science Department.

Haykin, S. and Network, N. (2004) 'A comprehensive foundation', *Neural Networks*, 2(2004), p. 41.

He, W., Chen, Y. and Yin, Z. (2016) 'Adaptive neural network control of an uncertain robot with full-state constraints', *IEEE Transactions on Cybernetics*. IEEE, 46(3), pp. 620–629.

Hough, P. (1962) 'Method and means for recognizing complex patterns'. Google Patents.

Hu, M. and Liu, B. (2004) 'Mining opinion features in customer reviews', in *AAAI*, pp. 755–760.

IFR (2016) *World Robotics Report 2016 - International Federation of Robotics*, *Executive Summary World Robotics 2016 Industrial Robots*.

Jiang, P. *et al.* (2012) 'A behaviour network approach to support opportunity-based virtual enterprises in the internet', *Multiagent and Grid Systems*, 8(4), pp. 311–328. doi: 10.3233/MGS-120197.

Kaelbling, L. P., Littman, M. L. and Moore, A. W. (1996) 'Reinforcement Learning : A Survey', 4, pp. 237–285. doi: 10.1613/jair.301.

Kertész, C. (2012) 'Dynamic behavior network', in *2012 IEEE 10th International Symposium on Applied Machine Intelligence and Informatics (SAMI)*, pp. 207–212. doi: 10.1109/SAMI.2012.6208958.

Kleene, S. C. (1951) *Representation of events in nerve nets and finite automata*.

Klenk, M., Molineaux, M. and Aha, D. W. (2013) 'Goal-Driven Autonomy For Responding To Unexpected Events In Strategy Simulations', *Computational Intelligence*, 29(2), pp. 187–206.

Knepper, R. A., Srinivasa, S. S. and Mason, M. T. (2010) 'Hierarchical planning architectures for mobile manipulation tasks in indoor environments', in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 1985–1990.

Kober, J., Bagnell, J. A. and Peters, J. (2013) 'Reinforcement learning in robotics: A survey', *The International Journal of Robotics Research*. SAGE Publications Sage UK: London, England, 32(11), pp. 1238–1274.

Kragic, D. *et al.* (2018) 'Interactive, Collaborative Robots: Challenges and Opportunities.', in *IJCAI*, pp. 18–25.

Kretzschmar, H. *et al.* (2016) 'Socially compliant mobile robot navigation via inverse reinforcement learning', *The International Journal of Robotics Research*. SAGE Publications Sage UK: London, England, 35(11), pp. 1289–1307.

Lameski, P. *et al.* (2015) 'SVM parameter tuning with grid search and its impact on reduction of model over-fitting', in *Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*. Springer, pp. 464–474.

Lawrence, S. *et al.* (1997) 'Face recognition: a convolutional neural-network approach', *IEEE Transactions on Neural Networks*, 8(1), pp. 98–113. doi: 10.1109/72.554195.

Lee, Y.-S. and Cho, S.-B. (2014) 'A hybrid system of hierarchical planning of behaviour selection networks for mobile robot control', *Int. Journal of Advanced Robotic Systems*.

Lee, Y. S. and Cho, S. B. (2014) 'A hybrid system of hierarchical planning of behaviour selection networks for mobile robot control', *International Journal of Advanced Robotic Systems*, 11(1). doi: 10.5772/56088.

Leiva, A. J. F. and Barragán, J. L. (2011) 'Decision tree-based algorithms for implementing bot AI in UT2004', in *Foundations on Natural and Artificial Computation*. Springer, pp. 383–392.

Lin, C.-T. and Lee, C. S. G. (1991) 'Neural-network-based fuzzy logic control and decision system', *IEEE Transactions on computers*. IEEE, 40(12), pp. 1320–1336.

Maes, P. (1989) 'The dynamics of action selection', *Proceedings of the 11th international joint conference on Artificial intelligence*, 2, pp. 991–997.

Maes, P. (1991a) ‘A bottom-up mechanism for behavior selection in an artificial creature’, in *Proceedings of the first international conference on simulation of adaptive behavior on From animals to animats*. MIT Press, pp. 238–246.

Maes, P. (1991b) ‘A Bottom-up Mechanism For Behaviour Selection In An Artificial Creature’, pp. 238–246.

Maes, P. (1991c) ‘Learning to Coordinate Behaviours’, *Learning*.

Maes, P. (1991d) ‘The Agent Network Architecture (A N A)’, 2(4), pp. 115–120.

Maes, P. (1993) ‘Modeling Adaptive Autonomous Agents 1 Introduction 2 What is an Adaptive Autonomous Agent ?’, pp. 1–37.

Maes, P. and Brooks, R. A. (1990) ‘Learning to Coordinate Behaviors’, in *AAAI*, pp. 796–802.

McCulloch, W. S. and Pitts, W. (1943) ‘A logical calculus of the ideas immanent in nervous activity’, *The bulletin of mathematical biophysics*. Springer, 5(4), pp. 115–133.

Meeden, L., McGraw, G. and Blank, D. (1993) ‘Emergent control and planning in an autonomous vehicle’.

Miljković, Z. *et al.* (2013) ‘Neural network reinforcement learning for visual control of robot manipulators’, *Expert Systems with Applications*. Elsevier, 40(5), pp. 1721–1736.

Miller, W. T. (1994) ‘Real-time neural network control of a biped walking robot’, *IEEE Control Systems*. IEEE, 14(1), pp. 41–48.

Mnih, V. *et al.* (2015) ‘Human-level control through deep reinforcement learning’, *Nature*. Nature Research, 518(7540), pp. 529–533.

Murphy, R. and Woods, D. D. (2009) ‘Beyond Asimov: the three laws of responsible robotics’, *IEEE Intelligent Systems*. IEEE, 24(4).

Nagata, F., Otsuka, A. and Watanabe, K. (2012) ‘Network-based subsumption architecture for multiple mobile robots system’, *6th International Conference on Soft Computing and Intelligent Systems, and 13th International Symposium on Advanced Intelligence Systems, SCIS/ISIS 2012*, (January), pp. 187–192. doi: 10.1109/SCIS-ISIS.2012.6505001.

Nicolescu, M. N. and Matarić, M. J. (2002) ‘A hierarchical architecture for behavior-based robots’, *Proceedings of the first international joint conference on Autonomous agents and multiagent systems part 1 AAMAS 02*, pp. 227–233. doi: 10.1145/544741.544798.

Oland, E., Andersen, T. S. and Kristiansen, R. (2016) ‘Subsumption architecture applied to flight control using composite rotations’, *Automatica*, 69, pp. 195–200.

Paikan, A., Metta, G. and Natale, L. (2013) 'A port-arbitrated mechanism for behavior selection in humanoid robotics', *2013 16th International Conference on Advanced Robotics, ICAR 2013*. doi: 10.1109/ICAR.2013.6766466.

Pais, S. and Symonds, J. (2011) 'Data Storage on a RFID Tag for a Distributed System', *International Journal of UbiComp*, 2(2), pp. 26–39. doi: 10.5121/iju.2011.2203.

Peach, B. and Robinson, P. (2016) 'The Use of Data Packets in a Behaviour Network to Improve the Action Selection Mechanism', *PlanSIG 2016*.

Prescott, T. J. (2002) 'Comparing a brain-inspired robot action selection mechanism with'winner-takes-all'', in *From Animals to Animats 7: Proceedings of the seventh international conference on simulation of adaptive behavior*. MIT Press, p. 75.

Richards, N. M. and Smart, W. D. (2013) 'How should the law think about robots?', *Available at SSRN 2263363*.

Roberts, M. *et al.* (2014) *Iterative goal refinement for robotics*. DTIC Document.

Di Rocco, M. *et al.* (2013) 'Configuration Planning with Multiple Dynamic Goals.', in *AAAI Spring Symposium: Designing Intelligent Robots*.

Rosenblatt, J. K. and Payton, D. W. (1989) 'A fine-grained alternative to the subsumption architecture for mobile robot control', *International Joint Conference on Neural Networks*, (February 1989), pp. 317–323 vol.2. doi: 10.1109/IJCNN.1989.118717.

Rosenblatt, J. K. and Payton, D. W. (1989) 'A fine-grained alternative to the subsumption architecture for mobile robot control', in *Neural Networks, 1989. IJCNN., International Joint Conference on*. IEEE, pp. 317–323.

Saffiotti, A., Konolige, K. and Ruspini, E. H. (1995) 'A multivalued logic approach to integrating planning and control', *Artificial Intelligence*, 76(1–2), pp. 481–526. doi: 10.1016/0004-3702(94)00088-I.

Samsonovich, A. V (2010) 'Attention in the asmo cognitive architecture', in *Biologically Inspired Cognitive Architectures 2010: Proceedings of the First Annual Meeting of the BICA Society*. IOS Press, p. 98.

Schlenoff, C. *et al.* (2012) 'An IEEE standard ontology for robotics and automation', in *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 1337–1342.

Seraji, H. and Howard, A. (2002) 'Behavior-based robot navigation on challenging terrain: A fuzzy logic approach', *Robotics and Automation, IEEE Transactions on*, 18(3), pp. 308–321.

Siciliano, B. and Khatib, O. (2016) *Springer handbook of robotics*. Springer.

Simmons, R. *et al.* (1997) 'A layered architecture for office delivery robots', in *Proceedings of the first international conference on Autonomous agents*, pp. 245–252.

Singh, S. and Kelly, A. (1996) 'Robot planning in the space of feasible actions: Two examples', in *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, pp. 3309–3316.

Smart, W. D. and Kaelbling, L. P. (2002) 'Effective reinforcement learning for mobile robots', in *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, pp. 3404–3410.

Specht, D. F. (1991) 'A general regression neural network', *IEEE transactions on neural networks*. IEEE, 2(6), pp. 568–576.

Sutton, R. S. and Barto, A. G. (1998) *Reinforcement learning: An introduction*. MIT press Cambridge.

Swere, E. and Mulvaney, D. J. (2003) 'Robot navigation using decision trees', *Electronic systems and control division research*.

Turner, J. T., Givigi, S. N. and Beaulieu, A. (2013) 'Implementation of a subsumption based architecture using model-driven development', *SysCon 2013 - 7th Annual IEEE International Systems Conference, Proceedings*, pp. 331–338. doi: 10.1109/SysCon.2013.6549902.

Tyrrell, T. (1993) 'Computational mechanisms for action selection', pp. 1–218. Available at: http://w2mind.computing.dcu.ie/worlds/w2m.TyrrellWorld/tyrrell_phd.pdf.

Tyrrell, T. (1994) 'An evaluation of Maes's bottom-up mechanism for behavior selection', *Adaptive Behavior*, 2(4), pp. 307–348.

Vinyals, O. *et al.* (2015) 'Show and tell: A neural image caption generator', in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3156–3164.

Watkins, C. J. C. H. and Dayan, P. (1992) 'Q-learning', *Machine learning*. Springer, 8(3–4), pp. 279–292.

Weiser, M. (1993) 'Hot topics-ubiquitous computing', *Computer*. IEEE, 26(10), pp. 71–72.

Weiser, M. (1995) 'The computer for the 21st century', *Ieee*, pp. 933–940.

Weser, M., Off, D. and Zhang, J. (2010) 'HTN robot planning in partially observable dynamic environments', in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pp. 1505–1510.

William, J. (1983) 'The principles of psychology'.

Yang, K. M. and Cho, S. B. (2013) 'STRIPS planning with modular behavior selection networks for smart home agents', *Proceedings - IEEE 10th International Conference on Ubiquitous Intelligence and Computing, UIC 2013 and IEEE 10th International Conference on Autonomic and Trusted Computing, ATC 2013*, pp. 301–307. doi: 10.1109/UIC-ATC.2013.70.

Yang, S. X. and Meng, M. (2000) 'An efficient neural network approach to dynamic robot motion planning', *Neural Networks*. Elsevier, 13(2), pp. 143–148.