DISTRIBUTED COGNITION AS THE BASIS FOR ADAPTATION AND HOMEOSTASIS IN ROBOTS

JAMES HENRY STOVOLD

Doctor of Philosophy

University of York
Computer Science

July 2016

ABSTRACT

Many researchers approach the problem of building autonomous systems by looking to biology for inspiration. This has given rise to a wide-range of artificial systems mimicking their biological counterparts—artificial neural networks, artificial endocrine systems, and artificial musculoskeletal systems are prime examples. While these systems are succinct and work well in isolation, they can become cumbersome and complicated when combined to perform more complex tasks. Autonomous behaviour is one such complex task. This thesis considers autonomy as the complex behaviour it is, and proposes a bottom-up approach to developing autonomous behaviour from cognition. This consists of investigating how cognition can provide new approaches to the current limitations of swarm systems, and using this as the basis for one type of autonomous behaviour: artificial homeostasis.

Distributed cognition, a form of emergent cognition, is most often described in terms of the immune system and social insects. By taking inspiration from distributed cognition, this thesis details the development of novel algorithms for cognitive decision-making and emergent identity in leaderless, homogenous swarms. Artificial homeostasis is provided to a robot through an architecture that combines the cognitive decision-making algorithm with a simple associative memory. This architecture is used to demonstrate how a simple architecture can endow a robot with the capacity to adapt to an unseen environment, and use that information to proactively seek out what it needs from the environment in order to maintain its internal state.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF CODE LISTINGS

ACKNOWLEDGMENTS

I'd like to thank my supervisors, Jon and Simon, for their valuable advice and guidance, and for putting up with my ridiculous ideas each week; my parents, who have supported me—both financially and emotionally—throughout my time at university; my friends, who helped support me while I propped up the bar; the Deramore Arms in Heslington, for providing the bar; and the University of York Cycling Club, who occasionally reminded me that I should probably be writing and not riding my bike.

Finally, I'd like to thank Emma, for just being incredible and so very patient while I was writing.

DECLARATION

The work presented in this thesis is the author's own work. The ideas and results surrounding identity formation presented in chapter 3 have been published in (Stovold et al., 2014). The ideas and results surrounding cognitive decision-making and artificial homeostasis in robots in chapters 4 and 5 are about to be submitted for publication (Stovold et al., 2016).

This work has not previously been presented for an award at this, or any other, university. All sources are acknowledged as references.

INTRODUCTION

1.1 MOTIVATION

Since its initial proposal (Beni and Wang, 1993), swarm robotics has piqued the interest of the research community, providing a sense of excitement that is yet to wane nearly a quarter of a century later. There are many potential advantages of a decentralised collection of robots, working together like a colony of social insects (Beni, 2005). These advantages include the ability to withstand the failure of individuals without disrupting the behaviour of the collective (*robust*), the potential to form groups and apply different teams to different areas of a problem (*distributed*), and the inherent ability to scale up (*scaleable*) (Şahin, 2005).

Despite the potential advantages, swarm robotics are not yet able to operate reliably in a real-world environment. This stems from a combination of the relative infancy of the field and the difficulty of engineering self-organising behaviour, a problem that is exacerbated in a changing environment (Stepney et al., 2006). The work presented in this thesis investigates the fundamental basis of collective intelligence by considering distributed cognition and its role in adaptivity and homeostasis in robots. Distributed cognition—such as that found in social insect colonies and the human immune system—provides a mechanism through which autonomous behaviour can emerge from the interactions between a robot and its internal and external environments.

The original assumption that, by mimicking the behaviour of social insects, robots can inherit their robustness turns out to have counter-examples (Bjerknes and Winfield, 2013). The emergent high-level behaviour of social insects is dependent on the low-level interactions of each insect, both with its environment and with other insects. A minor perturbation at a low level can cause a significant change in behaviour at a high level. As demonstrated by the 'anchoring' problem in (Bjerknes and Winfield, 2013), this can have a major impact on even the simplest swarm robotics algorithms, with the failure of a single robot slowing the progress of the swarm, and further failures causing the entire swarm to stall. The failed robots act as an 'anchor' for the rest of the swarm.

Swarms, as distributed systems, do not have a natural boundary where one swarm ends and another might begin. While the distributed nature of swarm robotics can provide a key advantage over centralised systems, the lack of such a boundary can pose problems when the system is scaled up. In the same way that the assumption of robustness has counter-examples, swarms are not necessarily as scaleable as originally assumed. As a swarm is scaled up, the lack of a natural boundary can pose problems where multiple swarms interfere with each other and often combine into one swarm. This is evident in even a basic algorithm such as Reynolds' (1987) boids, where multiple swarms can struggle to remain separate from each other.

There has been a recent drive towards improving the fault tolerance of swarm robots due to such issues as the anchoring problem (Bjerknes and Winfield, 2013). This drive to improve the fault tolerance of swarm robotics is combined with the need to improve the adaptivity of the robots. In the anchoring example described above, if the swarm can recognise the failure of a robot, adapting the behaviour of the swarm accordingly would allow the swarm to continue. Timmis et al. (2016) provide a mechanism through which this is possible, taking inspiration from granuloma formation in the immune system to help recover from the anchoring problem. This kind of adaptivity is qualitatively similar to homeostatic behaviour: altering the high-level behaviour of the system to maintain a specific environment at the lower level (Ashby, 1960).

Artificial homeostasis—e. g. homeostasis in robots—has been developed previously, most notably by Neal and Timmis (2003); Vargas et al. (2005) and Schmickl et al. (2011), using neuroendocrine networks to provide adaptive behaviour. By altering the production of hormones that affect the response of a neural network to the environment, the neuroendocrine network is able to alter the high-level behaviour of the system according to low-level signals from a sensor. While this approach is effective, the range of behaviours is limited to those already trained into the neural network, and re-training takes time.

This thesis considers how a model of distributed cognition can help provide homeostasis and adaptivity to robots. There are similarities between homeostasis and cognition—the ability to make decisions, along with the ability to use previous experiences to influence future actions—and this similarity can help with the development of a cognitive-inspired homeostatic system:

DECISION-MAKING Current collective decision-making algorithms make use of distributed sensing, and so rely on individuals traversing the environment and aggregating at a particular point in space to make a decision (Schmickl and Hamann, 2011; Garnier et al., 2005). This aggregation poses a problem when making multiple, successive decisions. The process of aggregation removes the distributed nature of distributed sensing, requiring the robots to re-distribute themselves before they can make a second decision. By developing a decision-making algorithm that does not require distributed sensing, this problem can be solved, as the entire swarm reacts together, and remains together after the decision is made.

EXPLOITING PREVIOUS EXPERIENCE The ability to adapt the high-level behaviour of a robot, without the slow process of re-training the control architecture, would allow a robot to rapidly respond to new variations in a dynamic environment. By making use of previous interactions with the environment to influence the future behaviour of a robot, associative memory neural networks offer a rapid method of learning and applying experience in order to maintain the internal state of the robot.

1.2 OUTLINE AND CONTRIBUTIONS OF THE THESIS

The previous section reports the motivation for the thesis. There are a number of limitations in the function of swarm robotics for which distributed cognition can provide solutions:

- Swarms have no ability to distinguish themselves from other swarms due to the lack of a natural boundary. This means that multiple swarms are unable to work in the same area without interfering with each other.
- Collective decision-making algorithms struggle to make successive decisions, due to the use of aggregation in a distributed system.

- Current artificial homeostatic systems have no ability to rapidly apply new information learned from the environment without re-training.

These limitations of swarm robotics can be solved by developing an adaptive, homeostatic system, based on the ideas of distributed cognition. This system has the potential to work for an extended period of time in a changing environment, while also providing a platform for investigating the fundamental basis of collective intelligence. As such, the aim and objectives of the thesis are listed below.

THESIS AIM

To investigate whether distributed cognition can be used as the basis for adaptivity and homeostasis in autonomous robots.

OBJECTIVES

1. Determine a method of providing a distributed system with the ability to form a boundary.
2. Derive a form of collective decision-making that does not require distributed sensing.
3. Build an architecture that uses distributed cognition to provide adaptivity and homeostasis to an autonomous robot.
4. Test the ability of the autonomous robot to adapt to, and sustain itself in, previously-unseen environments.

The remaining chapters in this thesis, and the objectives they address, are outlined below:

- Chapter 2 reviews the literature surrounding swarm robotics, artificial homeostasis, and distributed cognition. This chapter also introduces concepts that are required to understand the approach taken in both this thesis and the work of other researchers. The following three results chapters also review literature that is directly relevant to the results presented.
- Chapter 3, the first of three results chapters, presents a model for emergent identity formation in a swarm. The results presented in this chapter provide the ability for swarms to form a boundary, allowing multiple swarms to work in the same

environment without negatively interfering with each other. This chapter directly addresses **Objective 1**.

- Chapter 4 presents a model of cognitive decision-making, based on the ideas of distributed cognition proposed by Cohen (2000). This second results chapter provides a mechanism through which swarms are able to collectively make decisions without distributed sensing, allowing for multiple decisions to be made without an external force intervening. The model constructed and tested in this chapter is used as the basis for the homeostatic architecture developed in chapter 5. This chapter directly addresses **Objective 2**.

- Chapter 5 presents an architecture for adaptive homeostatic behaviour in robots. The architecture harnesses the decision-making abilities of the model presented in chapter 4, and combines it with an associative memory neural network. This allows robots to learn about the environment and rapidly apply the learned knowledge to alter their high-level behaviour, providing a mechanism for proactive homeostatic behaviour. This chapter directly addresses **Objectives 3 and 4**.

- Chapter 6 discusses the work presented in this thesis, puts it back in context of the literature, and considers different avenues of future research. This chapter then concludes the thesis.

- Appendix A presents a description of the materials and methods used throughout this thesis. This includes statistical analysis, aleatory uncertainty analysis, parameter sensitivity analysis, and experimental setups.

CONTRIBUTIONS The work presented in this thesis shows that:

- A synchronisation algorithm can be used to form emergent identities in swarms, allowing multiple swarms to operate in the same area.

- Cognitive decision-making provides the ability to make multiple successive decisions without distributed sensing or leaders within the swarm.

- Distributed cognition can be used to provide homeostatic and adaptive behaviour to a robot.

LITERATURE REVIEW

As described in the previous chapter, this thesis asks whether distributed cognition can provide homeostasis and adaptivity to autonomous robots. This chapter provides the background in swarm intelligence, distributed cognition, and neuroendocrine networks required to understand both the approach taken in the later chapters, and the approaches taken by other researchers. The three results chapters (chapters 3, 4, and 5) also review literature that is directly relevant to the presented results.

This chapter is organised as follows: section 2.1 introduces the ideas of swarm intelligence and self-organising behaviour, which are key aspects in distributed cognition. Section 2.2 reviews swarm robotics, an application of swarm intelligence to robotics, and the context in which the work in this thesis is undertaken. Section 2.3 presents a review of existing approaches to homeostasis in robots. Section 2.4 discusses distributed cognition and collective decision-making. Finally, section 2.5 concludes the chapter with a brief summary, highlighting the holes in the reviewed literature.

2.1 SWARM INTELLIGENCE

Nature has, over the preceding few million years, solved a large number of problems. Evolutionary forces provide a continual drive for natural systems to adapt to changing environments. While these systems are just doing their utmost to survive for as long as possible, the resulting mutations appear to be surprisingly creative solutions to some otherwise difficult problems. Bio-inspired algorithms are an attempt to use these solutions for our own means. By taking inspiration from biological approaches, we can harness millions of years of evolution and exploit the natural solution to solve the problem we are facing.

A concrete example is the foraging behaviour of ant colonies. While some species of ant forage in huge raids (Burton and Franks, 1985; Deneubourg et al., 1989), most ants forage by sending out individual scouts that return to the nest when they find food (Deneubourg et al., 1990). By laying pheromone on the return trail, the scouts are able to lead other ants to the food source they have found. These new recruits help to reinforce this pheromone signal by laying their own pheromone on the same trail as the scout has done. Due to the self-reinforcing nature of this process, the food source with the shortest path will become the strongest pheromone trail, encouraging the majority of the ants to follow this path until the food source is diminished. Dorigo et al. (1996) took inspiration from how ants find this shortest path and developed an algorithm called 'Ant System' that can be used to solve optimisation problems. Most notably, Dorigo and Gambardella (1997) show that the NP-hard travelling salesman problem is particularly amenable to this approach and due to the nature of ant colony optimisation (ACO) techniques, the algorithm is able to maintain sub-optimal paths as well as the shortest path. This feature is an ideal situation for dynamic graphs, such as those in telecommunication networks (Schoonderwoerd et al., 1997), as it allows for very rapid re-routing of data if the shortest path is blocked.

2.1.1 *Self-organising behaviour*

While a large amount of work has focussed on the ability of ant colonies to search for food, a lot more is happening within the colony. Ants in the colony will be performing one of a number of tasks at any given point. These tasks typically include foraging, nest maintenance, fending off predators, and clearing away refuse (Gordon, 2002). How each ant knows how to perform each task, and how it decides which task to perform is a result of *self-organising behaviour*. Self-organisation can be described as "*a process in which [a] pattern at the global level of a system emerges solely from numerous interactions among the lower-level components*" (Camazine et al., 2002, p. 8). The emergence of high-level patterns from low-level interactions typify complex systems (Anderson, 1972), and as such are ubiquitous throughout the natural world. For example, in ant colonies, each ant will react to disturbances in its environment (e. g. discovering a damaged part of the nest) and will alter its behaviour accordingly. It will also change behaviour depending on how many other ants are doing a particular task. If it encounters a large number of ants all in 'nest maintenance' mode, it is more likely to switch to nest maintenance as well (Gordon, 2002). This self-organising behaviour allows the colony to react to external stimuli without the individual ants having sufficient cognitive capacity to do so.

In considering the intelligence that emerges from the self-organising behaviour in ant colonies, Hofstadter (1979) compares ants in a colony to neurons in a brain. There are many similarities, neurons react to those closest to them (local interactions), and fire only when a sufficient number of their neighbours fire (simple rules). They are also remarkably robust, with individual cell death—apoptosis—a comparatively regular occurrence.[1] Even with this cell death, we still have exemplary cognitive function. Similarly, social insect colonies are able to lose many individual members without the high-level abilities suffering.

By comparing the intelligence of ant colonies and brains, Hofstadter (1979) implies that intelligence is not necessarily substrate-specific. In other words, if both ants and neurons can support intelligence, it might be possible for alternative substrates (e. g. silicon) to support intelligence as well. This is the idea of *liftable intelligence*.

2.1.2 *Liftable intelligence*

Liftable intelligence is the idea that intelligence is substrate-agnostic. If it was possible to develop an architecture that precisely mimicked the behaviour and interactions of all the cells and biochemical processes in the brain, liftable intelligence says that this artificial brain would be capable of real intelligent thought. This does, however, ignore how long it takes to learn and what is available for the brain to work with during development, in the sense of Edelman's primary repertoire (Edelman and Mountcastle, 1978; Edelman, 1987). As such, Hofstadter (1979) posits that intelligence is a software property, not a hardware property. It matters not which substrate is used, what matters is the high-level behaviour of the system. Hofstadter (1979) describes this concept exceptionally well using an example involving a range of rather eccentric characters and a copy of a Dickens novel. The example presented here removes the (albeit very entertaining) characters and replaces them with three people who all encounter Charles Dickens' "The Pickwick Papers" in different ways:

The first person reads this book at the 'letter-level', i. e. they read each letter individually, rather than as words or sentences. When

1 it has been estimated that as adults, we lose around 9000 neurons a day through apoptosis.

reading the book at this level, the reader needs to associate each letter with a concept in their mind before continuing. Following Hofstadter's example, instead of reading 'the' as 't', 'h', 'e', this person considers each letter as their associated definite concept before continuing, hence:

> "Hmm... You mean that every time I hit a word such as *the*, I have to think of three definite concepts, one after another, with no room for variation?"
>
> "Exactly. They are the *t*-concept, the *h*-concept, and the *e*-concept—and every time, those concepts are as they were the preceding time." (Hofstadter, 1979, p. 326, emphasis added)

Due to the nature of reading the book at this level, it would not be possible to understand the plot of the book. Regardless of which concepts are associated with each letter, they would not map properly to the real-world in such a way as to allow the reader to understand the plot, it is only the interactions between these concepts that convey the wide range of meanings required.

Two further people, however, read the book at higher levels: the 'word-level', and the 'sentence-level'. As before, each word and sentence is associated with a definite concept, but due to the interactions between letters to make words—and again to make sentences—the range of concepts required is sufficiently large that the plot is easily comprehended by both readers.

As these readers are compared, the resolution at which they interpret the story is increasingly broad, although the plot remains the same. The idea of resolution and scope are discussed at length by Ryan (2007) in the context of emergent systems.

Finally, consider the case of two further people who have not yet read this book, who shall be referred to as Person A and Person B. Person A is bilingual, fluent in both French and English, and reads The Pickwick Papers in English. Person B is a French national, and does not speak a word of English. If Person A, having read the book in English, describes in extensive detail the plot of the book in French to Person B, such that Person B has as much an understanding of the plot as the other readers described above, then Person B has 'read' the book at the 'plot-level'. At this stage, the underlying substrate of the book (previously words and sentences in ink on paper), has no bearing on the ability of Person B to understand the plot of the book. The plot is *substrate-agnostic*.

Liftable intelligence implies that, regardless of whether the substrate is natural or artificial, it should be possible to produce a system that exhibits intelligent behaviour. Most importantly—for the work in this thesis at least—this implies that it should be possible to develop intelligent behaviour in robots. Liftable intelligence is just another name for the philosophical concept of *functionalism*—the idea that an object is characterised by the function it performs, not the substance from which it is made (Putnam, 1960). The downside to using the term 'functionalism' is that it is a contentious concept, and therefore is difficult to present an argument without the possibility that the reader might colour the discussion with their preconceived notion of functionalism.

2.2 swarm robotics

Swarm robotics is a field of study that uses inspiration from swarm intelligence to produce certain behaviours in groups (or 'swarms') of robots.

Şahin (2005) presents a review of swarm robotics, collecting together the various terms used in the literature in order to produce a reliable definition of swarm robotics. Şahin proposes the following five criteria for identifying a 'swarm robotics' system:

- Autonomous robots
- Large number of robots
- Few homogenous groups of robots
- Relatively incapable or inefficient robots
- Robots with local sensing and communication capabilities

Interestingly, these five properties do not include self-organising behaviour. The large number of algorithms that make use of self-organisation shows how effectively emergent effects can be used to produce behaviour from an otherwise simple control architecture. This is addressed by Dorigo and Şahin (2004), by suggesting that criteria such as these should not be *"used as a checklist for determining whether a particular study is a swarm robotics study or not. Instead, they should be used as yardsticks for measuring the degree to which the term 'swarm robotic' might apply"* (p. 111).

While swarm robotics do not require the use of self-organising algorithms, the decentralised nature of swarms makes self-organising behaviour very attractive. There are a few examples of swarm robotics with non-self-organising behaviour (Flushing et al., 2012; Dorigo et al., 2013), but the advantages of using self-organisation is substantial enough for the majority of projects to make use of it. Engineering self-organising behaviour is difficult (Stepney et al., 2006), so many researchers opt for mimicking biology in order to make the design of self-organising behaviour easier.

Swarm robots have been used for a number of very interesting toy examples, but as yet have not found a reliable real-world application. The SWARMIX project (Flushing et al., 2012) is the closest to have come to a real-world application, with experiments run outside of a controlled laboratory environment (albeit still in relatively quiet areas). The SWARMIX project consists of an attempt to provide a dynamic ad-hoc communication network to a heterogenous search-and-rescue team working in the mountains. The swarm of drones form a dynamic chain between mountain rescue dogs and a human search party. As the distance between the dogs and the humans varies, the chain varies between a lattice formation (Olfati-Saber, 2006) and a stretched-out line of drones, in order to maintain connectivity across the network.

The lack of real-world applications for swarm robotics, however, stems predominantly from the fragility of self-organising algorithms. Contrary to early ideas that a large number of robots would give inherent robustness to the swarm (Şahin, 2005), a single robot failing can be enough to disrupt the swarm, and further failures cause it to stall completely, resulting from the lack of fault tolerance in swarm robotics systems (Bjerknes and Winfield, 2013).

The only viable methods of fault tolerance consist of using external tracking systems (Millard et al., 2014) or using aggregation to try and counter the effects of a failed robot (by re-charging the failed robot, for example) (Timmis et al., 2016). Despite the reliability issues of swarm robot algorithms, algorithms are still being developed. Holland and Melhuish (1999) describe an algorithm for emergent sorting behaviour in a swarm based on the stigmergic behaviour of ant colonies. This is similar to the construction mechanisms used by termites, which make use of pheromones as well as stigmergic communication. Werfel et al. (2014) have used termites as inspiration to develop collective construction behaviour for swarms of robots (Petersen et al., 2011; Werfel et al., 2014). Ijspeert et al. (2001) present an experiment in producing collective behaviour. The experiment consists of a series of sticks that must be removed from their holes. The only way to remove a stick is if two robots both pull the stick at the same time. Gauci et al. (2014b) present a method for self-organised aggregation based on a single binary sensor, which is extended to the clustering of objects in an environment (Gauci et al., 2014a).

2.2.1 *Swarm taxis*

The 'swarm taxis' algorithm (Bjerknes et al., 2007) is often cited as one of the quintessential swarm robotics algorithms. This algorithm allows the swarm to locate an infrared beacon without any individual having the ability to calculate which direction the beacon is in.

The swarm taxis algorithm consists of two main behaviours, *coherence* and *avoidance*. If a robot strays from the swarm, then the coherence behaviour will rotate the robot 180° to head back towards the swarm. Robots avoid colliding with other robots by maintaining an 'avoid radius' around them (avoidance behaviour). When not in either of these states, the robots randomly move, turning often to help with cohesion (Bjerknes et al., 2007).

When these behaviours are implemented across multiple robots in a swarm they aggregate the swarm into one group. In order for this aggregate to perform taxis, an infrared beacon is added to the environment. While the robots have no capacity to detect the direction of the beacon, they can detect the directionless signal from it. Using this information, Bjerknes et al. alter the robots such that they have a slightly larger 'avoid radius' when they are receiving this infrared signal, compared to when they are not. As such, the shadow cast by those robots closest to the beacon prevents those robots farther away from receiving the signal (see fig. 2.1). The 'illuminated' robots, with a larger avoid radius, try to move away from those 'shadowed' robots. The shadowed robots, however, still have the smaller avoid radius, so they move towards the illuminated robots in response. This provides the swarm with information regarding the direction of the beacon, even though the individuals have no knowledge of this information. Over time, the swarm moves across the environment to the beacon.

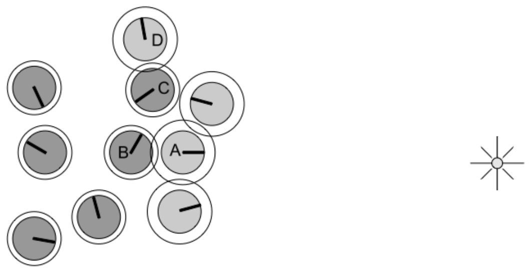


Figure 2.1: Diagram showing the operation of a swarm taxis algorithm. Robots B and C do not receive the signal from the beacon, and so try to stay close to the other robots. Robots D and A do recieve the signal from the beacon, and so move farther away from those robots near them. The effect is that the swarm slowly moves across to the beacon. Diagram from (Bjerknes et al., 2007).

The emergent behaviour is what makes algorithms like the swarm taxis algorithm so challenging to design. In single robots, the robot would have a sensor to detect the direction of the beacon, and this would be included in the robot controller. In swarms, however, the behaviour of the collective is qualitatively different from the behaviour of the individual.

2.2.2 *Limitations of swarm robotics*

Due to the difficulties in engineering self-organising behaviour (Stepney et al., 2006), some researchers instead focus on computational methods of producing controllers. Evolutionary algorithms have been used widely in developing collective behaviour for robots. By placing constraints on the individual controllers, and setting the fitness function to something only collective behaviour can achieve, evolutionary algorithms are able to produce self-organising swarm algorithms (Floreano and Mondada, 1998; Nolfi et al., 2000). By simulating the behaviour of the swarm, evolutionary algorithms are able to test the collective behaviour of a controller and optimise the behaviour of the individual according to this high-level behaviour. There are, however, problems with this approach. Jakobi et al. (1995) present the idea of the 'reality gap', where controllers optimised in the perfect, noise-free world of a simulation will likely fail upon bridging the reality gap to noisy, imperfect, real-world experiments.

Further to the problems introduced by the reality gap, due to the nature of swarm robots to react to the environment and other robots around them, it becomes a reasonable concern that an external party might be able to subvert the behaviour of the swarm. By introducing a robot to the swarm that alters the high-level behaviour, a single robot could prevent the swarm from completing its objective. The 'anchoring' problem in the swarm taxis algorithm (Bjerknes and Winfield, 2013), where a single robot failing can disrupt the swarm, and further failures can anchor the swarm to a single point, could easily be achieved by subverting robots. By responding to any signals it receives but refusing to move, a subverting robot could trigger the stalling of the entire swarm. While this is currently not an actively researched area (most likely due to swarm robotics algorithms still being tested in laboratory environments), this is a serious concern when considered in terms of real-world applicability.

Finally, one of the most prominent limitations of swarm robotics is that they are implemented on relatively incapable platforms. This is actually one of the points Şahin (2005) identifies as a defining property of swarm robots. The problem of developing a robot controller with self-organising behaviour is difficult on an idealised platform, but when the platform is limited in its abilities it becomes even more challenging. For example, the small capacity for charge in swarm robots is exacerbated by any advanced controller that attempts to use too much processing power. As such, emergent behaviour that makes use of the simplest possible controllers will help to increase the battery life of the robots. Eventually, however, the batteries will run out of charge. Before swarm robotics can be moved to the real world, the ability to detect and alter the behaviour of the robot/swarm to find a recharge station is essential. This adaptivity of behaviour is commonly referred to as *artificial homeostasis*.

2.3 homeostasis and adaptivity

Homeostasis, "*the automatic regulation of physiological functions*" (Vernon, 2015, p. 94), is an essential part of keeping biological systems functioning. It provides the ability to keep a specific variable within certain limits around a set point, allowing the organism to maintain an optimal working environment for its biochemical reactions. As discussed in section 2.1.1 above, minute changes in low-level behaviour can have large effects on high-level behaviour. The capacity to adapt the high-level behaviour of the system in order to maintain an optimal working environment for the lower levels is a worthwhile expenditure of energy, resulting in a system that can survive in a wider range of environments. For example, if humans struggled to survive once the surrounding temperature lifted above 25°C, it is highly unlikely that we would be the most dominant species on the planet. This approach to homeostasis through adaptivity is a useful way to think of the problem, and is one that is used throughout this thesis, essentially considering the problem of homeostasis as a combination of adapting to the environment, and making decisions about what to do based on previous experiences.

Artificial homeostasis is the problem of maintaining an appropriate working environment for artificial systems such as robots. Swarm robotics systems, often assumed to be inherently robust, have recently been shown to be surprisingly delicate (Bjerknes and Winfield, 2013). The failure of a single robot is sufficient to disrupt the entire system. The difference with swarm systems from other robotic systems is that they have another level where minor variations could be magnified. The swarm can be viewed at a number of different levels (micro-, meso- and macro-scopic, corresponding to the sensor/controller level, individual robot level, and swarm level, respectively). Variation at any of these levels will directly affect the other levels. At the microscopic level (i. e. sensor/controller level), a small discrepancy between a sensor value and real value can have large effects on the behaviour of the algorithm at the macroscopic level, but might only result in almost imperceptible variation at the mesoscopic level (Jakobi et al., 1995). Such is the nature of complex systems (Ryan, 2007). Before swarm robotics can be exploited for real-world applications, the ability to autonomously alter the macroscopic behaviour to maintain the micro- and meso-scopic states must be developed.

A simple example of how this might happen is to consider a robot in a swarm that is about to run out of charge. The change at the microscopic level is that the charge sensor has been steadily reducing for an extended period of time, such that it finally drops below a threshold. At the mesoscopic level, this could result in the single robot failing as it has too little charge to perform any tasks. At the macroscopic level, the entire swarm is disrupted, and possibly fails entirely due to the anchoring effect described by Bjerknes and Winfield (2013). As such, a small signal in one of many robots is indicative of a problem that prevents the entire swarm from performing its task.

A popular method for approaching these complex problems is to look to biology for inspiration (Floreano and Mattiussi, 2008). By mimicking the biological approach to homeostasis, researchers have developed methods for providing artificial homeostasis to robots (Neal and Timmis, 2003, 2005; Vargas et al., 2005; Timmis et al., 2009; Stradner et al., 2009; Moioli et al., 2009; Schmickl et al., 2011). The approach taken by these researchers is to develop artificial neural, endocrine, and immune networks to mimic the biological processes at play within the body. This approach has its merits, but often produces controllers that are complicated, and difficult or slow to adapt. Neuroendocrine systems often use a number of separate neural networks designed to perform specific tasks, and the ability to adapt present only in the option to switch between pre-trained neural networks, or to completely retrain the network.

2.3.1 *Neural and neuroendocrine networks*

Artificial neural networks are a simplified computational model of collections of neurons in the human brain (Rosenblatt, 1958). By passing data between layers of the network, neural networks are able to approximate any mathematical function required (provided sufficient neurons are available) (Hornik et al., 1989). In the context of robotics, using a model of the brain provides a simple, intuitive way of thinking about how information passes through the controller. With sensor inputs lined up on one side of the network, and actuators on the other, the processing occurs on the journey between one and the other. The simplest method of using a neural network for a robot controller is Braitenberg's vehicles (Braitenberg, 1986). These vehicles had a small number of sensors and wheels. By connecting the sensors to the wheels directly in different arrangements, the vehicles exhibit very different behaviour. Some flee from light, while others appear to cautiously approach it.

In order to use a neural network to control a robot, the network needs to be trained to exhibit the correct behaviour when provided with certain stimuli. Neural networks can be trained through a process called 'backward propagation of error' (Rumelhart et al., 1986). This process consists of presenting a series of input–output pairs to the network and adjusting the weights between neurons in order to minimise the error between the desired and actual output. While this method does work, it is not the most efficient or the best at generalising the problem. Overfitting often causes problems where the network does not produce the desired output when provided with inputs that it has not been trained on.

More recently, evolutionary algorithms have been used to not only train the network, but also to alter the structure of the network in order to meet the constraints of the problem (Floreano and Mondada, 1998). Reinforcement learning is also a method that can be adopted to train a network, and is based on the dopamine reward system in the brain, where correct behaviour is rewarded and incorrect behaviour is not (Watkins and Dayan, 1992). Over time, the system learns which behaviours are correct by altering the weights of the neural network (Williams, 1992).

Neuroendocrine networks take inspiration from the interaction between the nervous system and the endocrine system in the body. The endocrine system produces and stores hormones that help to regulate the behaviour of the nervous system. In doing so, they provide a mechanism for signalling a change in behaviour to all tissues and cells in the body. They are able to alter the response of the system to certain stimuli. The well-known fight-or-flight mechanism relies on stress hormones such as cortisol, noradrenaline, and adrenaline to prepare the body to rapidly respond to a threat. Part of this response is decreased reaction times that come from more efficient neural signalling, a classic example of how the endocrine network can affect the behaviour of the nervous system.

The hormones in artificial neuroendocrine networks are used to alter the response of a neuron to stimuli. This is achieved by adding a new weight term to the inputs of each neuron that is affected by any hormone present. In doing so, the hormone can increase or decrease the response of the neuron to that stimulus. If the network is set up in such a way that two paths through the network have complementary responses to the hormone, then the endocrine network is able to switch between behaviours by varying the release of hormone.

The architecture developed in (Neal and Timmis, 2003, 2005; Vargas et al., 2005) makes use of a neuroendocrine network to control the behaviour of a robot. This consists of a neural network that is linked to sensor inputs, but is regulated using artificial hormones. As these hormones vary, the behaviour of the neural network changes. For example, by connecting the hormone production controller to a distance sensor that fires when the robot approaches an object, the robot can avoid objects in the environment. As the hormone concentration increases, the high-level behaviour of the robot will alter such that the robot diverts. This can be achieved by having complementary responses to the hormone on each wheel. As such, the presence of the hormone will drive each wheel in opposite directions, turning the robot away from the object in the environment. Once diverted, the hormone production will decrease again due to the lack of an object to drive it. As such, neuroendocrine systems can provide an alternative to re-training a single neural network for multiple behaviours. Similar results can also be achieved without the neural network, relying instead on the interactions between hormones and actuators to perform collision avoidance (Stradner et al., 2009; Schmickl et al., 2011).

This approach to adaptivity in an artificial system is effective so long as the required behaviour is already trained into a neural network for the endocrine network to switch between. In order to train new behaviour into the robot, a method for altering the function of the network is required. In (Neal and Timmis, 2005) this is provided by an artificial immune system that is able to change the structure and weights of the neural and endocrine networks. This tries to mimic the behaviour of the body, where the interactions between the neural, endocrine, and immune systems give rise to homeostasis (Neal and Timmis, 2003).

One of the main problems with this approach to adaptivity is the speed at which the robot can adapt. The network will adapt quickly to behaviours that the neural network is already trained for, but new behaviours would require a training phase in order to learn about the new environment. The work presented in chapter 5 provides a mechanism through which a robot is able to rapidly adapt to an unseen environment and continually learn about the environment to provide adaptivity. The system presented achieves this through associative memory neural networks, which are described below.

2.3.2 *Associative memory*

Associative memory is a form of memory that associates stimuli with responses. In biological systems, the memory often results from temporal associations that emerge between two sets of interacting neurons (Palm, 1980, 1981). As the stimulus is presented to the first set of neurons, the response of those neurons is sent to the second set of neurons, which subsequently respond. This second response is similar every time, indicating that the same response will occur if given the same stimulus.

Willshaw et al. (1969) first proposed a method for the brain to store memories inspired by ‘correlograms’. Correlograms are produced by shining light through two pieces of card with pinholes in. The correlations between the two sets of holes are captured on a third piece of card. Inverting this pattern would allow either of the original sets to be reconstructed from the inverted correlation pattern and the other original card.

The correlogram was simplified to the ‘associative net’ (Willshaw et al., 1969), helping to remove the imperfections of the original setup. By simultaneously presenting a series of input–output pairs to the associative net, the associations between stimulus and response are encoded into the network. The stored associations are retrieved by re-presenting one of the original stimuli.

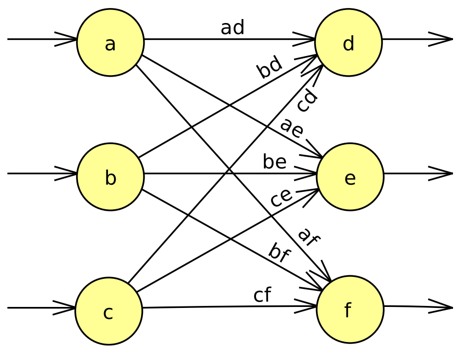
The correlation matrix memory (CMM) (Kohonen, 1972) is a matrix-based representation of an associative net. The matrix represents the binary weights from a fully-connected, two-layer artificial neural network (one input layer, one output layer; see fig. 2.2b). As such, the network in fig. 2.2b would be represented by the CMM, $\mathcal{M}$, with k input–output pairs $\mathcal{I}$ and $\mathcal{O}$ in fig. 2.2a.

Before training, the initial matrix $\mathcal{M}$ is filled with zeros (as there are no associations stored in the network). As the k binary-valued input–output pairs are presented to the network, the associations are built up in the matrix, $\mathcal{M}$. These associations are stored as 1s in the matrix, corresponding to coincident 1s in both input and output vectors. For example, in fig. 2.2a, if a and e were both 1, then ae would be set to 1 on training.

This approach to training is in direct contrast to a ‘typical’ artificial neural network. The typical feedforward neural network will train the connections between neurons based on the stimulus–response pairs provided to it, as done here, but the difference is the presence of a ‘learning rate’. This learning rate represents how quickly the network learns about the patterns presented to it, and is typically very small ($\epsilon = 0.05$ is an appropriate starting point). The use of a learning rate helps to prevent overfitting of the data, by relying on multiple examples of similar data to reinforce the signal, building on previous examples until it recognises the patterns in the data. This is not the approach taken with CMMs. While feedforward neural networks recognise patterns through repeated presentation of examples, CMMs

$$\begin{array}{cc|cc} \begin{pmatrix} \mathcal{I} \end{pmatrix} & \begin{pmatrix} \quad \mathcal{M} \quad \end{pmatrix} & \begin{pmatrix} a \\ b \\ c \end{pmatrix} & \begin{pmatrix} ad & ae & af \\ bd & be & bf \\ cd & ce & cf \end{pmatrix} \\ & & & \\ & \begin{pmatrix} \quad \mathcal{O} \quad \end{pmatrix} & & \begin{pmatrix} d & e & f \end{pmatrix} \end{array}$$

(a) Basic CMM architecture (left), where $\mathcal{M}$ represents the matrix of binary weights, $\mathcal{I}$ and $\mathcal{O}$ represent the input–output pair corresponding to the neurons a, b, c and d, e, f in (b), respectively.



(b) CMM associative memory neural network. The binary weights between the two layers are represented by the matrix $\mathcal{M}$ in (a).

Figure 2.2

recognise patterns through a process called 'generalisation', which consists of applying a threshold to the response from the network.

The downside to this approach is that a CMM currently will immediately learn every example presented to it—this is what contributes to its exceptionally-quick training speeds. By immediately learning every example, the CMM makes the assumption that every example presented to it must have happened (i. e. there are no false examples in the training set). From the perspective of robotics, where stimulus data values are typically taken directly from a sensor, this is an appropriate approach. Even allowing for noise in the sensor values, these are representative of the values that would be presented during real sensor readings.

To recall, we present an input pattern $\mathcal{I}_r$, and we get the following: (Haykin, 1994)

$$\mathcal{O} = \mathcal{M}\mathcal{I}_r$$

where $\mathcal{I}_r$ is the input pattern, and $\mathcal{O}$ is the output from the network trained with associations stored in $\mathcal{M}$. The desired output pattern,

$\mathcal{O}_r$ is currently combined with noise from the other patterns stored in the network, e_r, hence:

$$\mathcal{O} = \mathcal{O}_r + e_r$$
$$e_r = \sum_{\substack{j=1 \\ j \neq r}}^{k} (\mathcal{I}_j^T \mathcal{I}_r)\, \mathcal{O}_j$$

Thresholding the output vector $\mathcal{O}$ leaves the desired output vector $\mathcal{O}_r$. Different thresholding strategies offer different advantages in terms of the ability of the network to generalise from noisy or incomplete patterns to a correct output (Austin and Stonham, 1987).

The ability of the network to generalise noisy inputs suggests a range of applications in real-world environments, where a noisy signal is far more common than a clean signal. If a CMM is distributed across a swarm of robots, the swarm could potentially survive the failure of a single robot by generalising the signal that would have been contributed by the failed robot. This concept is beyond the scope of this thesis, but is discussed as part of the further work section (section 6.3).

2.4 DISTRIBUTED COGNITION

The collective behaviour of ants is one of the most fascinating areas of the natural world, and has been studied at great length (Franks, 1989; Franks et al., 2002; Couzin et al., 2005). By researching the underlying mechanisms behind distributed cognition (also known as colony-level, swarm, or collective cognition), researchers are able to develop analytical techniques and theories that can be applied in other cognitive systems (Marshall and Franks, 2009). For example, taking techniques for analysing collective decision-making processes in ant colonies and comparing them to similar processes in the visual activity of primates (Marshall et al., 2009; Franks et al., 2013). Passino et al. (2008) use the ideas from distributed cognition to compare the nest-site selection behaviour of honey bees with the brain. The famous 'waggle dance' behaviour, and the associated positive and negative feedback mechanisms give rise to very efficient collective decision-making processes (Visscher and Camazine, 1999; Camazine et al., 1999; Seeley et al., 2012; Marshall et al., 2015).

Cognition is a concept that is notoriously difficult to define (Boden, 2008). There are a wide range of different definitions of cognition, each based on evidence and ideas from different sources. Due to the sparse nature of these definitions, it is inevitable that each definition is 'incorrect' in some way, for the application at hand. The lack of a single definition of cognition gives rise to the problem of which definition to use. Most researchers will pick a definition of cognition that fits closest with the application or source of inspiration used.

This thesis focusses on distributed cognition. This particular definition is used because of its foundations in the immune system (Cohen, 1992a,b, 2000) and social insects (Mitchell, 2005), and because when considering swarm systems, it makes sense to use a definition for cognition that is distributed, making use of the interactions between underlying agents.

The cognitive view of the immune system differs from the conventional clonal selection theory (Cohen, 1992a,b). It suggests that the various cells of the immune system adapt to the presence of pathogens by maintaining an imprint, or 'internal image', of the environment that it has been exposed to. This imprint is stored primarily in the antigen receptors, and influence any interactions that may take place.

Cohen (2000) defines (distributed) cognition as:

> "Cognitive systems, I propose, differ strategically from other systems in the way they combine three properties:
>
> 1. They can exercise options; *decisions*.
> 2. They contain within them images of their environments; *internal images*.
> 3. They use experience to build and update their internal structures and images; *self-organization*." (p. 64, emphasis original)

This definition serves as a measuring stick for the work presented in this thesis, and is used as the framework around which we construct an architecture for cognition in robots. It is, however, worth noting that Cohen's final point, referring to the use of experience to update internal structures of the agents within a cognitive system can be thought of as a learning mechanism. At its most basic level, any learning system will use previous experience to alter future behaviour, which is exactly what the immune system is doing during what Cohen has termed 'self-organization'. In order to prevent too much confusion of terms, given the extensive use of the term 'self-organisation' in the swarm intelligence literature, when referring to self-organisation throughout the rest of this thesis, it is the swarm intelligence definition that is being intended, unless specifically stated otherwise.

2.4.1 *Collective decision-making*

Of the three points laid out by Cohen (2000) in section 2.4 above, we have covered internal images (associative memory, section 2.3.2) and self-organisation and adaptation (sections 2.1.1 and 2.3). Decision-making is one of the key features of intelligent systems (Pfeifer and Scheier, 2001). The ability to change behaviour based on previous experience and the current situation allows for the wide repertoire of behaviours that typify cognitive systems. This could be through adaptation, re-learning, or just through altering the parameters in a control architecture. These action selection mechanisms vary between implementations, but are still decisions about what action to take next (Pfeifer and Scheier, 2001).

One of the most heavily researched natural decision-making processes is the nest-site selection of the honey bee (genus *Apis*), including the famous 'waggle dance' (von Frisch, 1967). The waggle dance has many similarities to the behaviour of neurons in the brain (Passino et al., 2008; Seeley et al., 2012). It is a process through which scout bees are able to provide information to the rest of the hive about potential nest sites. Each scout reports the quality and location of the nest site it has found to the hive through a variety of 'waggles'. As multiple scouts dance for their respective sites, there are positive and negative feedback mechanisms through which the lower-quality sites are filtered out (Seeley and Buhrman, 1999; Seeley et al., 2012). The negative feedback mechanisms are provided through stop signals from scouts that have encountered something on their scouting route (e. g. a predator) and are discouraging other sites in the area (Nieh, 2010). This is provided through the scout vibrating, then butting her head against the bee she wants to inhibit (Nieh, 1993).

Cross-inhibition helps to drive the overall system towards a decision faster, through positive and negative feedback mechanisms. This process is analogous to behaviour seen in the brain. As a neuron fires (or bee dances), it inhibits other neurons around it, making it more likely to be the most prominently-firing neuron at that point. If all neurons behave in this way, inhibiting others around them, then whichever neuron is firing the fastest will be inhibited less, and inhibit others more. This results in a self-reinforcing process, after which the less-prominently-firing neurons are inhibited by the most-prominent neuron, and a decision is made. Through cross-inhibition, distributed decision-making systems are able to increase the speed at which they reach consensus (Marshall et al., 2015).

The idea of self-organising collective decision-making has been implemented in a number of bio-inspired algorithms. The BEECLUST algorithm (Schmickl et al., 2007) takes inspiration from the behaviour of young honey bees in a temperature gradient to produce collective aggregation in a swarm of robots. This is extended in (Schmickl et al., 2009; Schmickl and Hamann, 2011) in the form of a collective decision-making algorithm where the aggregation behaviour of the robots decides between two sources with different temperatures (in this case, selecting the higher temperature). Similarly, Garnier et al. (2005) implements an algorithm based on the collective behaviour of cockroaches that decides between two shelters. The cockroaches preferentially decide on the larger of two shelters without any mechanism for measuring their size. These two systems work in very similar ways, with individuals pausing for lengths of time proportional to some property of the environment. For BEECLUST, this is the temperature of the environment, for the cockroaches, it is how many other cockroaches are present. These methods allow for effective aggregation of individuals in order to make a decision, but rely on the behaviour of distributed individuals sampling the environment. This approach is an effective method in biology as it allows the colony to sample a wider area of the environment before settling on a decision. In real-world applications of swarm robotics, on the other hand, this approach is unlikely to be as effective, as described below.

The aggregation approach allows the swarm to interact with the environment (through the distributed sampling of each individual). This provides information regarding any gradients that exist, allowing for a collective decision to be made. The alternative is for the swarm to traverse the environment as a group, and react to the gradient as a group. This is similar to the approach that army ants take when foraging, where they move as a group and react to the presence of prey in a way that is appropriate to the distribution of prey in the environment (Deneubourg et al., 1989). While each approach has its merits (distributed sensing gives wider coverage of the environment, group sensing keeps the swarm together), when it comes to constructing artificial decision-making systems, practicality becomes an important issue. For example, can the swarm make multiple decisions repeatedly? How efficiently can the swarm make the decision? In the sense of heuristic decision-making systems, real-world systems often do not require the global optimum and a local optimum will suffice.

The 'best-of-n' problem is the problem of selecting the best option out of n alternatives. It is a way of formalising the decision-making process, making it amenable to formal mathematical analysis. When considering bio-inspired decision-making systems, it is often helpful to think in terms of the best-of-n problem, if only to highlight how they differ from other decision-making processes. For example, Ant Colony Optimisation (ACO) (Dorigo et al., 1996; Dorigo and Gambardella, 1997) makes a collective decision between n alternative routes from nest to a food source (assuming single food source and single nest). ACO does not, however, just select the global optimum and ignore all the other routes: it selects the global optimum from n alternatives, and maintains a network of local optima as well. This allows the algorithm to rapidly adapt to a changing environment, such as a blocked route on the network, switching to the next best option. This alternative formulation is sometimes referred to as the κ-best-of-n problem—as in, selecting the κ best options out of n alternatives (Louchard and Bruss, 2015).

There are alternative methods of decision-making in a collection of distributed agents that do not rely on self-organising behaviour. Zavlanos and Pappas (2008) present a method of using an auction between agents that allows decisions to be made regarding the removal of communication links within the swarm itself. One of the main drawbacks of this approach is that, by introducing a very small communication delay to the network, it is possible to disrupt the algorithm to such an extent that a link can be removed before all agents have been consulted. This problem can be circumvented through the inclusion of a timestamp during the voting process, and delaying the end of auction until all votes contain the same timestamp, but this adds further delays to an already slow process.

For self-organising decision-making algorithms, robots are an effective method of testing an algorithm. The noise that is included in simulation rarely approximates real-world environments (Jakobi et al., 1995), so implementing the algorithm on a real-world, robotic platform offers unique insight into how well the algorithm functions.

Some of the algorithms we discussed above, specifically the cockroach (Garnier et al., 2005) and BEECLUST algorithms (Schmickl et al., 2007, 2009; Schmickl and Hamann, 2011) are implemented on robots in either original papers, or follow-up papers. Further examples include Parker and Zhang's approach to the problem of binary nest-site selection (i. e. best-of-2 problem) by providing light sources for the robots to decide between (Parker and Zhang, 2009, 2011). This is contrasted by the work in (Trianni and Dorigo, 2005) which prevents the swarm from distributing itself. Instead, they consider the problem of collective decisions in physically-connected robots, using an experimental setup that consists of the swarm collectively deciding whether it can traverse different-sized troughs in the ground. In larger-scale experiments, Valentini et al. (2015a) consider the speed–accuracy trade-off in decision-making using the binary discrimination problem (best-of-2 problem), implemented using 100 kilobots[2] and analysed from a theoretical standpoint to determine what effect different parameters have on the speed–accuracy trade-off (Valentini et al., 2015a,b).

2.5 SUMMARY

This chapter has introduced some potentially new concepts, and reviewed important areas of the literature. The three results chapters (chapters 3, 4, and 5) also review the literature that is directly relevant to the results presented. The literature presented in this chapter has been focussed on the ideas of distributed cognition, and the possibility of using distributed cognition to develop homeostatic behaviour for robots.

Through the review of the literature presented in this chapter, it is evident that there are currently holes in the literature. First, at present there is no way of determining the boundary of a distributed system. One of the main side-effects of swarm robot systems being tested in isolation is that they rarely encounter anything other than the controlled environment of the lab. While this is essential to ensure that the results gathered are viable, it means that swarms are not being tested in the same environment as other swarms. Providing a method for forming a distributed boundary would help to isolate one swarm from another, while still offering the potential for interactions between the two. Chapter 3 presents such a method.

2 Exceptionally-small swarm robots for use in large swarms (Rubenstein et al., 2014)

Second, current decision-making algorithms (such as BEECLUST (Schmickl and Hamann, 2011)) rely on distributed sensing, where robots are distributed across an environment in order to collectively sample all areas of the environment, and then aggregate on the best area to make the decision. While effective, this method relies on aggregation, which makes successive decisions slow, as after aggregating, the robots will need to re-distribute themselves around the environment. This also requires the robots to know when a decision has been reached, or an external force would need to intervene. Alternative approaches consist of using flocking algorithms to maintain a coherent collection of robots, but this approach currently requires 'leaders' within the swarm to guide it towards a decision (Yu et al., 2010). The leader-based approach again implies that an external force is required to impart this knowledge into the few leaders within the swarm. Chapter 4 presents a decision-making algorithm that takes inspiration from Cohen's (2000) discussion of distributed cognition in the immune system. It makes decisions using a flocking mechanism, but without the requirement of having leaders to guide the swarm.

Third, the reliance on biomimetic approaches to artificial homeostasis (Neal and Timmis, 2003; Vargas et al., 2005; Timmis et al., 2009) results in complicated controllers that are often slow to adapt. Taking a bottom-up approach to the problem, chapter 5 presents a controller inspired by distributed cognition, and shows that artificial homeostasis and adaptation can be implemented through the interactions of a decision-making algorithm with a learned imprint of the environment.

EMERGENT FORMATION OF IDENTITY IN DISTRIBUTED POPULATIONS

The previous chapter presented a review of the literature surrounding distributed cognition, homeostasis and adaptation, and swarm robotics. This chapters directly addresses objective 1 of the thesis (*Determine a method of providing a distributed system with the ability to form a boundary*). The ability of a swarm to form a boundary helps with the problems associated with scaling the system up. If a swarm is able to differentiate itself from other swarms in the same environment, then multiple swarms are able to work together without interference. In order to increase the speed with which results can be gathered, rather than make use of real-world robots, simulated swarms of generic 'agents' are used in this chapter. Details of these agents are provided in appendix A, but they are essentially massless points in a virtual environment.

This chapter presents the 'identity algorithm', which provides a mechanism for swarms to form a boundary, in order to function more effectively in the presence of other swarms. The identity of a swarm emerges from the interactions between agents and between swarms. The firefly synchronisation algorithm (Tyrrell et al., 2006) provides the ability for swarms to distinguish those agents that are members of the same swarm from those in other swarms. Linking the firefly algorithm with the control algorithm for the agents provides a mechanism through which the agents are more likely to respond to agents that they are synchronised with. This process provides a swarm with the ability to differentiate itself from other swarms, by reacting less strongly to the agents in the other swarms. This is achieved by weighting the behaviour of agents towards those agents that are synchronised together. This self–not-self distinction is key for multiple swarms to be able to work in the same area without interfering with each other. Although the physical members (individuals) of the swarm could (and often do) vary, the identity of the swarm remains for an extended period of time, and even while interacting with other swarms.

The results presented in this chapter address a number of specific questions about swarms and their interactions with each other. Specifically whether swarms can form identities and remain as coherent swarms for an extended period of time, while in the presence of other swarms. By providing swarms with the ability to form identities, swarms are able to distinguish themselves from other swarms. Mitchell (2005) considers this idea in the context of self-awareness and distributed cognition in ant colonies and the immune system. Both Mitchell and Cohen (2000) consider how, for example, the immune system can determine what is part of the body, and what is an intruder. This self–not-self distinction is a key part of developing a collective identity.

The rest of the chapter is organised as follows: section 3.1 discusses the idea of collective identity, section 3.2 presents the identity algorithm and the analysis performed on the simulation software. Section 3.3 investigates three specific questions that, when combined, provide sufficient evidence to reason that the system meets the requirements for collective identity in a swarm. Finally, section 3.4 concludes the chapter by reconsidering the definition of collective identity from the perspective of the algorithm presented.

3.1 COLLECTIVE IDENTITY

Melucci (1995) defines identity in a sociological context:

> "The term identity... implies the notion of unity, which establishes the limits of a subject and distinguishes it from all others; it implies a relation between two actors that allows their (mutual) recognition." (p. 45)

This definition implies that identity is essential for determining the limits of a subject (in this chapter, the limits of the swarm) and hence how it can act in relation to an environment that also contains other (similar) agents and swarms.

One of the key challenges when considering the identity of a swarm rather than an individual is the question of what happens when the members of the swarm change. In the simplest case, what happens when a single agent or robot leaves the swarm: does that swarm have the same identity as before? This is a well-known and oft-debated philosophical problem, commonly referred to as the *Ship of Theseus* problem (Chisholm, 2004), or colloquially as the *Trigger's*

Broom problem.[1] The Ship of Theseus problem questions whether replacing a single panel in a ship changes the identity of the ship, or if it remains the same ship. This is directly analogous to the question of what happens to the identity of a swarm when an individual leaves the swarm. While many others have debated this issue, an in-depth discussion on the topic is outside the scope of this thesis. The work presented makes the assumption that, provided there is no more than a 10% change in the members of the swarm at once, the swarm retains the same identity. This 10% value has been determined through a qualitative examination of the behaviour of the presented simulation. By setting this to 10%, the swarm tracking system in the simulation is able to reliably follow sub-swarms as the members of the swarms vary.

By introducing an accepted 10% change in the members of a swarm, it is possible to see where different questions could be raised regarding the identity of the swarm. For example, what happens if all the members of the swarm are eventually changed for other members, 10% at a time? Hofstadter (1979) considers this problem in his famous *. . . Ant Fugue* where he considers how groups of ants redistribute themselves across the colony. These ants form into groups (termed 'signals'), and move across the colony to the point where they are required. Throughout this process, the ants could slowly be replaced by other ants, but the signal still remains:

> *Tortoise:* . . . Does a signal, from its creation until its dissolution, always consist of the same set of ants?
>
> *Anteater:* As a matter of fact, the individuals in a signal sometimes break off and get replaced by others of the same caste, if there are a few in the area. Most often, signals arrive at their disintegration points with nary an ant in common with their starting lineup. (Hofstadter, 1979, p. 323)

Another important part of Hofstadter's depiction of ant signals is a description of what happens when two signals meet in the colony at the same time. In the colony, the two signals pass directly through each other, without impeding the progress of the other signal. This is ascribed to being a result of the caste distribution (the distribution

1 From the British sitcom "Only Fools and Horses", where the character Trigger proclaims "Maintained it for 20 years. This old broom's had 17 new heads and 14 new handles in its time".

of jobs within the colony), where two signals can cross each other because they are heading to different parts of the colony to do different jobs, and so they fail to recognise the other signal because they are performing a different job. As described in our identity algorithm later, this is represented in our system by the synchronised flashing of a swarm of agents: if two swarms are not flashing in time with each other, they do not react to each other as strongly.

Returning to Melucci's (1995) definition of identity, there are a number of key characteristics that are adaptable to work with swarms instead of humans. Specifically, Melucci describes three features that identity always refers to:

- Continuity through time
- Separation from others
- Recognition of and by others

This definition implies that a swarm needs to be able to form and maintain an identity through time (continuity through time), needs to be able to maintain an identity through space (separation from others), and be able to react to other swarms in the area (recognition of and by others).

Through the use of the firefly algorithm (Tyrrell et al., 2006), the work in this chapter introduces a method of forming emergent identities in a simulated swarm of agents. By linking the synchronisation effect of the firefly algorithm to the alignment vector in Reynolds' boids algorithm (Reynolds, 1987), small swarms of agents are shown to behave independently of other swarms, and embody the three features of identity formation described by Melucci above.

The firefly algorithm is used to vary the sensitivity of the agent to its neighbours. Specifically, the sensitivity affects the weighting of the alignment vector provided by the boids algorithm (Reynolds, 1987). This results in a system where those agents that are synchronised will be more likely to remain aligned to each other, providing the positive feedback necessary to hold the swarm together over both time and space.

Three specific questions are asked of the behaviour of the identity algorithm which, when combined, provide evidence towards the system meeting the requirements laid out in Melucci's (1995) definition of identity:

RQ1: *Do distinct sub-swarms form and behave independently?*

RQ2: *Can sub-swarms preserve their identity for an extended period of time?*

RQ3: *Can sub-swarms preserve their identity across space?*

3.2 IMPLEMENTATION

This section describes how the algorithm for collective identity is implemented in a simulated environment. As described in section 3.1, the emergent synchronisation effects of the firefly algorithm (Tyrrell et al., 2006) are linked to Reynolds' boids model, which provides the control logic (Reynolds, 1987).

Tyrrell et al.'s firefly algorithm (shown in code listing 3.1) synchronises agents as an emergent effect of each agent running a simple procedure. Each agent increment an internal counter up to a predetermined threshold over time. Once it reaches that threshold, the agent flashes a light and resets to zero. If one agent sees another agent flash, it will immediately increment its own counter by a set amount (referred to as *jumping*). As a result, all agents that can see each other for an extended period of time will synchronise to the same frequency. Fig. 3.1 shows one counter reaching its threshold and resetting (left), and also the jumping effect of seeing another agent flashing in its vicinity (right). Over time this is sufficient to synchronise all the counters such that they reach the threshold at the same time, and so flash in time with each other.

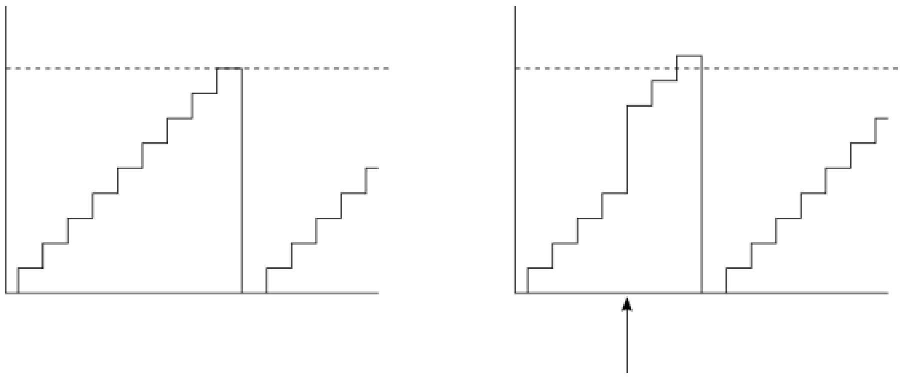


Figure 3.1: Graphical depiction of the firefly algorithm. Left: the internal counter on a robot counts up to the threshold, before flashing and returning to its baseline level. Right: After four steps, the internal counter jumps as a result of stimulation from a different robot, resulting in it reaching the threshold level earlier than before. Over time, this process synchronises the flashing of all the robots in the area.

Listing 3.1: Pseudo-code for firefly synchronisation behaviour

```
foreach (agent in agents)
{
  agent.jump = false;
  neighbours = agent.getAgentsWithin(radius);
  if (flashingNow(neighbours))  agent.jump = true;

  agent.counter++;
  if (agent.jump)  agent.counter += JUMP_RATE;

  if (agent.counter >= FLASH_THRESHOLD)
  {
    agent.shouldFlashNext();
    agent.counter = 0;
  }
}

setFlashForNextTimestep(agents);
```

In the boids algorithm (Reynolds, 1987), (shown in code listing 3.2) each agent, or 'boid', follows three simple rules: *alignment*, *coherence*, and *separation*. At each discrete timestep t, each of the agents calculates its unit velocity vector, $\mathbf{v}$, for time $t+1$ by considering those other agents in its immediate vicinity (see fig. 3.2). If there are any agents in the region of separation, it will turn away from its closest neighbour. If there are no agents in this region, then it will align itself to all those agents in the region of alignment by taking the mean of their headings, and combining it with an adjustment to turn towards the closest agent in the region of coherence.

Listing 3.2: Pseudo-code for boids flocking algorithm

```
foreach (boid in boids)
{
  neighbours = boid.getBoidsWithin(radius);
  foreach (neighbour in neighbours)
  {
    distanceToNeighbour = boid.distanceToNeighbour(neighbour);
    if (distanceToNeighbour <= AVOID_RADIUS)
    {
      boid.turn_away_from(neighbour);
    }
    else
    {
      if (distanceToNeighbour <= ALIGN_RADIUS)
      {
        boid.align_with(neighbour);
      }
      else
      {
        boid.turn_towards(neighbour);
      }
    }

  }
}
moveForwards(boids, 1.0);
```

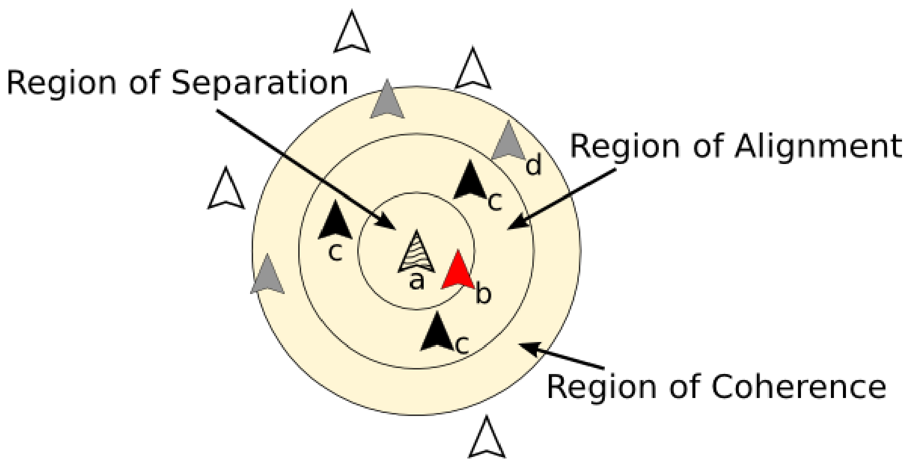


Figure 3.2: The agent in question (*a*) turns away from those agents in the region of separation (*b*), maintaining a set amount of space around them at all times. If agent *b* is not present, the agent turns towards the average heading of those in the region of alignment (agents *c*), and turns towards the closest agent in the region of coherence (*d*).

The boids algorithm is used as a control algorithm for the agents as it only requires local interactions, in a similar way to a real robotics platform, and offers a cheap method of controlling agents so that the focus can be on the larger-scale effects of linking a synchronisation algorithm with a control algorithm.

The firefly algorithm provides a swarm with the ability to distinguish between individuals that are members of the swarm and individuals that are not, simply by whether they are flashing at the same time. Each agent combines the boids and firefly algorithms by increasing the sensitivity of the *alignment* vector in the boids algorithm when aligning to agents that are flashing in time with itself.

The dynamics of this model are simulated using NetLogo (Wilensky, 1999). The code used in the simulation is based on two existing simulations of the boids and firefly algorithms (available in the NetLogo models library (Wilensky, 1997, 1998)). The alignment vector in the boids algorithm is linked to the firefly algorithm by defining a new form of neighbourhood in each agent: while previously the agents would search for 'flockmates' (or 'neighbours') in their vicinity, and use the headings of those flockmates to calculate a new heading for themselves, the new system introduces 'flashmates' as well. Flashmates are defined as those agents in the immediate vicinity that have flashed at the same time as the agent in question. The algorithm turns each agent towards the mean of its flashmates, rather than the flockmates used in the original system. A pseudo-code description of the identity algorithm is provided in code listing 3.3.

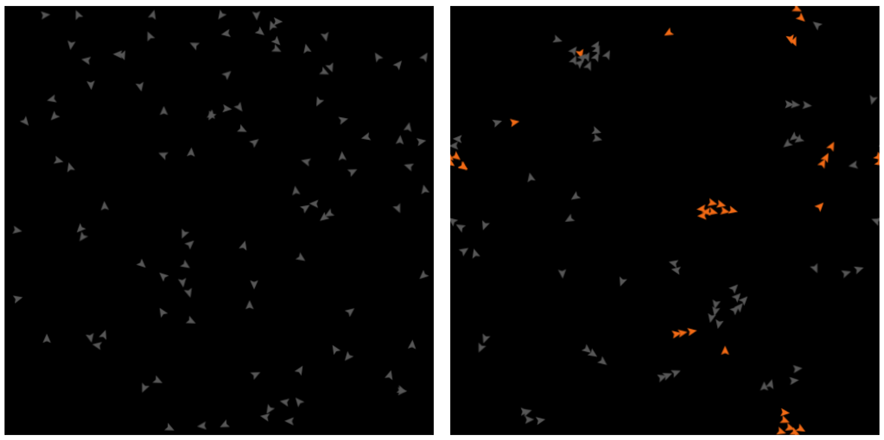
Fig. 3.3 shows two screenshots of the system, with fig. 3.3a showing an initial, random start position, and fig. 3.3b shows the system after it has been run for 1000 timesteps, where small distinct swarms are clearly visible.

3.2.1 *Tracking system*

A tracking system is developed as part of the above simulation that identifies swarms between timesteps. The tracking system identifies swarms in the simulation by taking an agent, then looking at each of its flashmates in turn and adding them to a group. This process is then repeated until all the flashmates of the flashmates have been identified—the resulting group is considered a single swarm. Once a swarm is identified by the tracking system, it is assigned a numerical group ID and the number of agents in the swarm

Listing 3.3: Pseudo-code for full identity algorithm

```
foreach (boid in boids)
{
  boid.jump = false;
  neighbours = boid.getBoidsWithin(radius);

  if (flashingNow(neighbours))
  {
    boid.jump = true;
  }

  foreach (neighbour in neighbours)
  {
    distanceToNeighbour = boid.distanceToNeighbour(neighbour);
    if (distanceToNeighbour <= AVOID_RADIUS)
    {
      boid.turn_away_from(neighbour);
    }
    else
    {
      // if they last flashed at the same time, then align
      if (distanceToNeighbour <= ALIGN_RADIUS &&
            neighbour.counter == boid.counter)
      {
        boid.align_with(neighbour);
      }
      else if (distanceToNeighbour <= COHERE_RADIUS)
      {
        boid.turn_towards(neighbour);
      }
    }

  }
}

moveForwards(boids,1.0);
increaseAllCounters(boids);
setFlashForNextTimestep(boids)
```



(a) Random start position. (b) System state after 1000 timesteps.

Figure 3.3: Screenshots of the simulation, showing the initial, random start position (a), and the state of the system after 1000 timesteps (b), showing small, distinct swarms.

is stored alongside it. In the subsequent timesteps the number of agents is updated as agents join and leave the swarm. The parameter **`group-comparison-threshold`** tells the tracking system how large the percentage change in swarm size can be between timesteps. If the swarm size changes more than this, then the swarm has changed sufficiently to be considered a new swarm and a new group ID is assigned. This approach assumes that it is the number of agents, not the specific agents that make up a swarm. This means that if one agent is replaced with another agent within the same timestep, the swarm is considered the same swarm. This is directly analogous to the ideas presented by Hofstadter (1979), where the ant signal can be made up of different ants throughout the process of traversing the colony.

3.2.2 *Assumptions in the simulation*

There are a number of assumptions made during the construction of this model. First, the agents in the simulation are modelled as a single point in space; second, the agents do not have a restricted field of view; third, the environment has a periodic boundary.

Of the three assumptions listed, the first will have the largest effect on the behaviour of the system. Having all agents modelled as points means that the firefly algorithm will potentially be able to see more agents within its radius of vision (**`vision`** parameter) than it would be able to otherwise (as they could be blocked by other agents). This concern is negated by adjusting the firefly algorithm so that it will only jump once if it sees other agents, rather than jumping once for each other agent it sees. In addition, there will also be an implication for the boids algorithm, as multiple agents could occupy the same point in space, whereas real robots would block each other if travelling in different directions.

The other two assumptions have a very small impact on the behaviour of the system: the 360° visibility is unrealistic in terms of natural systems, but is viable in artificial systems. The periodic boundary on the environment might have an impact on how the swarms form in the simulation, but this is unlikely. If the swarms were constrained to an environment without a periodic boundary, then—with sufficient space—similar swarms would form and the system would continue as it would before. A periodic boundary provides this sufficient space without requiring a simulator that is too slow to run.

One further concern is that the agents have perfect vision (something that is completely unrealistic in the real world)—if an agent flashes in the vicinity of another agent, it is guaranteed to see that flash. This could easily contribute towards widening the reality gap (Jakobi et al., 1995) between our proof-of-principle and an implementation on a real robotics platform. To combat this, we introduce a simple noise term into the firefly algorithm, where each time the algorithm runs, there is a small probability that it sees a flash that is not actually there. This probability is parameterised as **`noise-rate`** in the simulation. While it may seem as though this is contradicting what we are aiming to do (introducing more flashes, rather than restricting the likelihood of seeing existing flashes), the result is the same: the firefly algorithm is destabilised. Whether that destabilisation occurs from the an agent not jumping when it sees a flash, or from an agent jumping when it does not see a flash, the result is that the synchronisation process is perturbed.

A list of relevant parameters, including typical values/ranges is given in table 3.1.

3.2.3 *Simulation analysis*

A major problem with stochastic simulations is how to ensure the effects observed are a result of varying the parameters and not a

Parameter	Value
`population`	100 (50 in fixed setup)
`vision`	3.0 patches
`max-align-turn`	2.50 degrees
`max-cohere-turn`	1.75 degrees
`max-separate-turn`	1.00 degrees
`cycle-length`	4
`flash-length`	1
`jump-level`	1
`noise`	true / false
`noise-rate`	0.00005
`control`	true / false
`group-threshold`	10
`group-comparison-threshold`	90%

Table 3.1: Parameters used in the identity algorithm simulation, and their typical values.

result of the inherent stochasticity within the system (Read et al., 2012). Aleatory uncertainty analysis, provided through the `spartan` package (Alden et al., 2013) provides a method of determining how many replicate runs are required to prevent this problem. Fig. 3.4 shows that 350 runs are required for the simulation.

In order to quantify the effect that each parameter has on the behaviour of the system, `spartan` offers multiple options for parameter sensitivity analysis. The option used here is known as 'one-at-a-time' analysis, relating to its approach of holding all parameters steady at their baseline levels and varying one parameter at a time, measuring the effect it has on the macroscopic behaviour of the simulation. Multi-dimensional options, often used in conjunction with one-at-a-time analysis, sample combinations of parameters using 'Latin Hypercube' sampling (Marino et al., 2008) and run the analysis on these combinations. This provides a mechanism through which more complex interactions between parameters can be detected. In the work in this chapter (and throughout this thesis), only one-at-a-time analysis is performed, as more complex interactions are highly unlikely to occur in such simple models, especially given how unsensitive the simulation is to the parameters through one-at-a-time analysis.

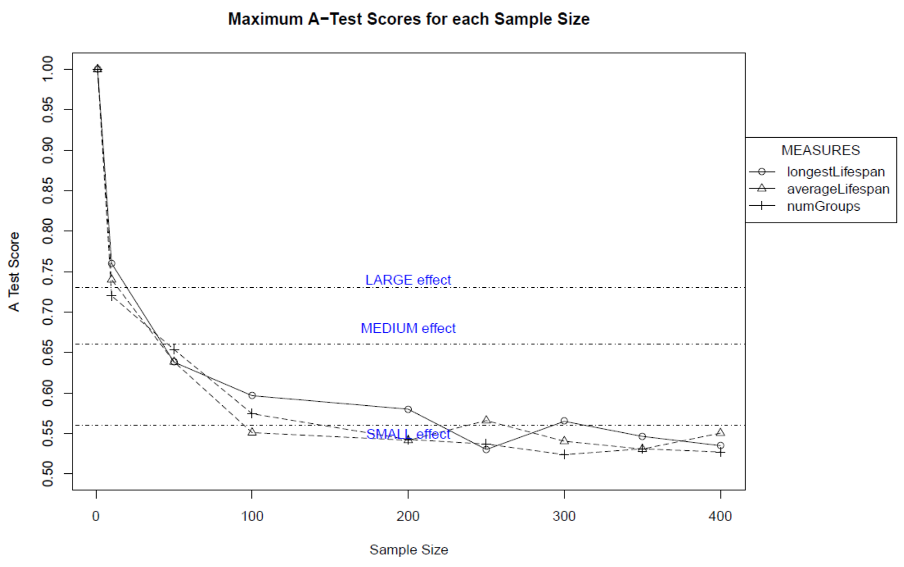


Figure 3.4: Aleatory uncertainty analysis, as generated by `spartan`, showing that we require at least 350 replicate runs to ensure the effects we are seeing are a result of any changes we make rather than from the inherent stochasticity within the simulation.

One-at-a-time analysis is run for each of the parameters listed in table 3.2, with the results for the **`population`** and **`noise-rate`** parameters shown in figs. 3.5 and 3.6 respectively. All other parameters tested did not reveal that the simulation was sensitive to the parameters.

Parameter	Baseline	Min value	Increment	Max value
`population`	100	50	10	150
`max-align-turn`	2.5	0.5	0.5	10.0
`noise-rate`	0.00005	0.00000	0.00001	0.00010
`cycle-length`	4	2	2	16
`jump-level`	1	1	1	5

Table 3.2: Parameters varied during the 'one-at-a-time' parameter sensitivity analysis.

The graph in fig. 3.5 shows the sensitivity of the system to the **`population`** parameter. The number of distinct swarms and the average lifespan of the swarms are both significantly sensitive to the change in **`population`** (positively and negatively correlated, respectively).

The **`noise-rate`** parameter in fig. 3.6 shows negatively-correlated significant sensitivity to both longest swarm lifespan and average lifespan, but not to the number of distinct swarms.

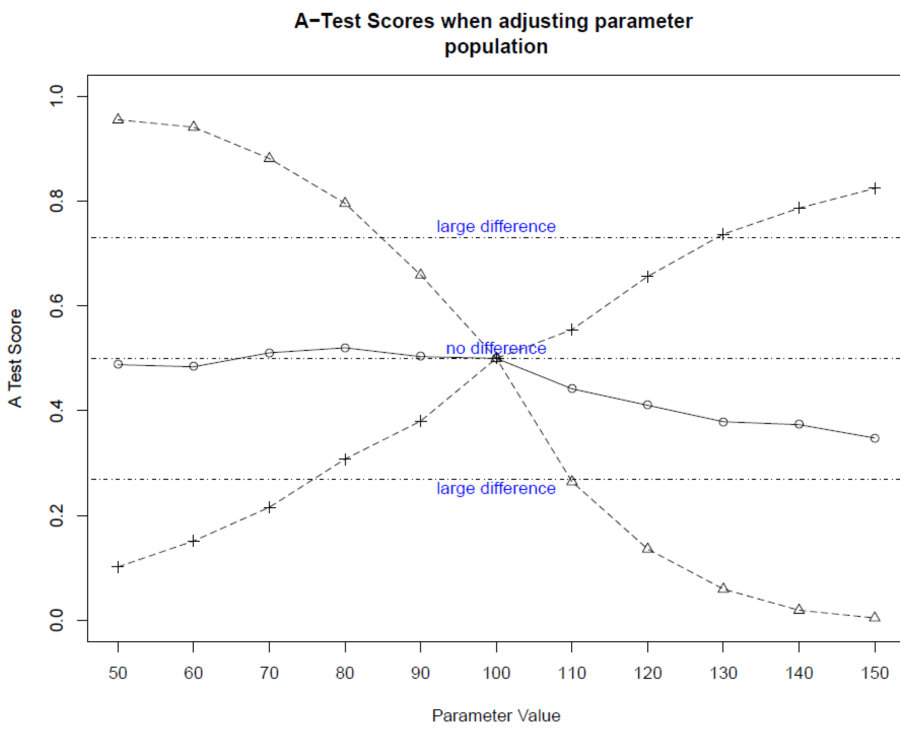


Figure 3.5: One-at-a-time parameter sensitivity analysis for the **`population`** parameter. The cross represents the number of groups in the simulation, the triangles represent the average lifespan of a swarm, and the circles represent the longest lifespan of the swarms.

While both the **`noise-rate`** and **`population`** parameters show significant sensitivity, the system is not so sensitive that the selected baseline values are too much of a behavioural niche to be useful. The **`population`** parameter, for example, is only significantly different from the baseline with a change of 10–15 agents, meaning that for minor fluctuations in the population, the behaviour remains steady.

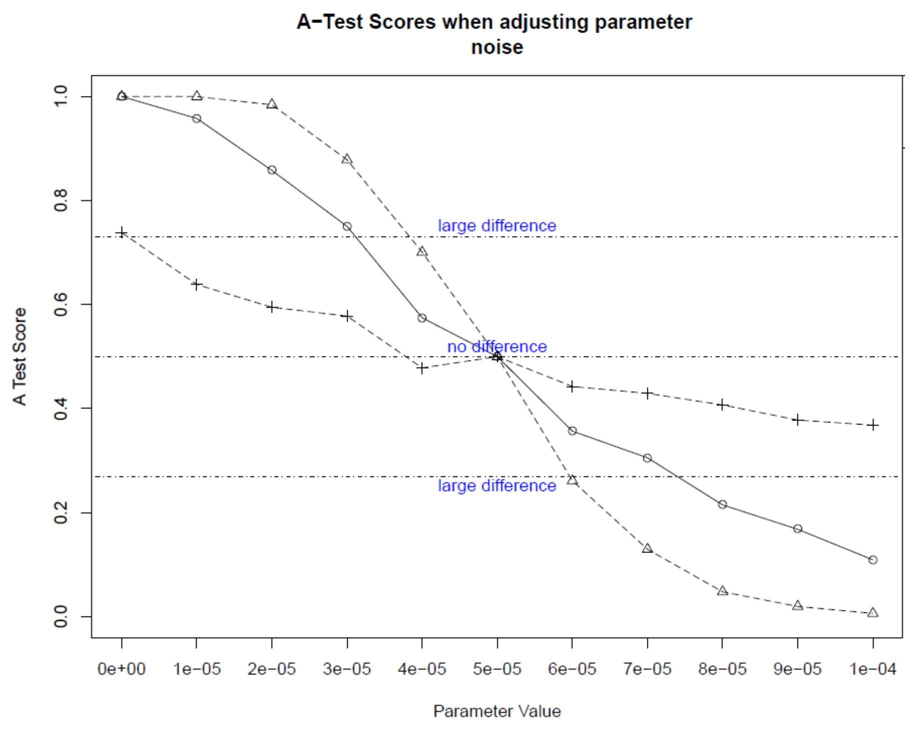


Figure 3.6: One-at-a-time parameter sensitivity analysis for the **`noise-rate`** parameter. The cross represents the number of groups in the simulation, the triangles represent the average lifespan of a swarm, and the circles represent the longest lifespan of the swarms.

3.3 RESULTS

The algorithm presented is able to form multiple swarms from randomly-initialised, simulated agents. These swarms are observable after only 1000 simulated timesteps. Using the simulation described in section 3.2, the three questions defined in section 3.1 are answered. The questions are repeated below:

RQ1: *Do distinct sub-swarms form and behave independently?*

RQ2: *Can sub-swarms preserve their identity for an extended period of time?*

RQ3: *Can sub-swarms preserve their identity across space?*

3.3.1 *Distinct sub-swarms remain highly-polarised through time while global polarisation varies*

To determine whether distinct sub-swarms are able to form and behave independently, a measure of coherence for sub-swarms is required. This can also be used to measure the global coherence of the swarm. Flock polarisation is a measure of the degree of alignment between swarm members (Couzin et al., 2002). This measure is extended to take into account local neighbourhoods, measuring the polarisation of smaller, distinct swarms.

To test RQ1, the following null hypothesis is defined:

NULL HYPOTHESIS (H_0): The identity algorithm has no effect on the polarisation of sub-swarms compared with the overall swarm.

The polarisation of the overall swarm, P_{global}, is a value between 0 and 1, where 0 implies the vector sum of the headings of all individuals is zero (e. g. two individuals heading east and west), and 1 implies the headings of all individuals are identical (e. g. two individuals both heading west) (Couzin et al., 2002):

$$P_{global}(t) = \frac{1}{N}\left|\sum_{i=1}^{N}\mathbf{v}_i(t)\right| \tag{3.1}$$

where N is the number of agents, and $\mathbf{v}_i(t)$ is the unit velocity vector of agent i at time t.

Our extended metric, P_{local}, measures the polarisation within each sub-swarm, such that a value of 0 implies all individuals in each sub-swarm are heading in opposite directions, and a value of 1 implies that all individuals in each sub-swarm are heading the same direction (but not necessarily the same direction as those in another sub-swarm).

$$P_{local}(t) = \frac{1}{K}\sum_{k=1}^{K}\left(\frac{1}{M_k}\left|\Sigma_{j=1}^{M_k}\mathbf{v}_j(t)\right|\right) \tag{3.2}$$

where K is the number of sub-swarms, M_k is the number of agents in sub-swarm k, and $\mathbf{v}_j(t)$ is the unit velocity vector of flashmate j at time t. When K is zero (i. e. there are no sub-swarms detected by the tracking system), P_{local} is set to zero (this can be seen in fig. 3.8b).

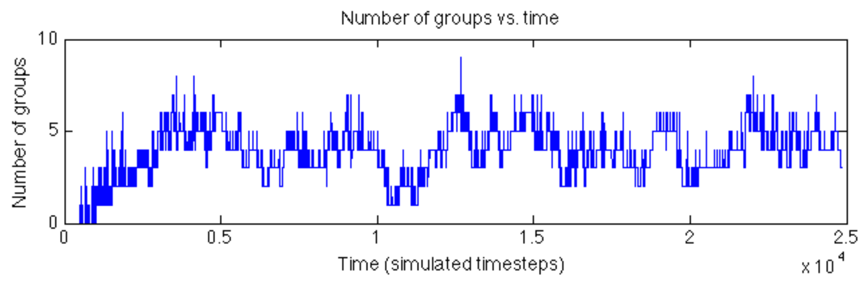
These two measures are compared to determine whether there are large numbers of aligned individuals in few swarms, or fewer aligned individuals in many distinct swarms. Calculating the ratio of local to global polarisation measures ($\frac{P_{local}}{P_{global}}$) provides a measure for comparing the effect of the algorithm at the local with the global level. A high ratio of local–global polarisation indicates that the sub-swarms are polarised more highly than the overall swarm, and the higher the ratio, the more the sub-swarms are behaving independently (by heading in different directions to each other). A low ratio of local–global polarisation, however, indicates the opposite, that the sub-swarms are not distinct or behaving independently compared to the overall swarm.

Fig. 3.7 shows the difference between local and global polarisation as sub-swarms form, and then repeatedly merge and split. It can be seen that, after the initial swarm formation and settling-down period, the local polarisation stays above 0.9 for the remainder of the experiment. This shows that within each swarm the individuals are well-aligned to each other, which is as we expect if the swarms are synchronised.

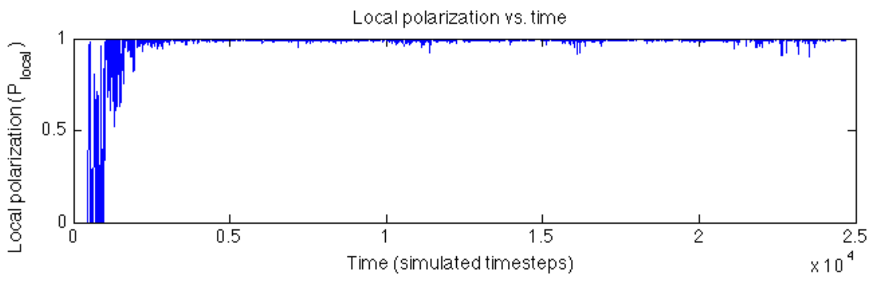
In comparison, the global polarisation varies considerably over time but remains predominantly below 0.4 throughout. We can infer from this that there were a number of swarms flocking together, rather than one large swarm, and because those swarms were aligned well to each other (as shown by the local polarisation), they were also synchronised with each other.

The same polarisation experiment is run with a control case, without the link between the firefly and boids algorithms. The effect of removing this link is shown in fig. 3.8, where it is evident that the entire population is consistently heading in a very similar direction throughout. This implies zero, or very few, sub-swarms, with the overall swarm heading in the same direction as the few sub-swarms that have formed.

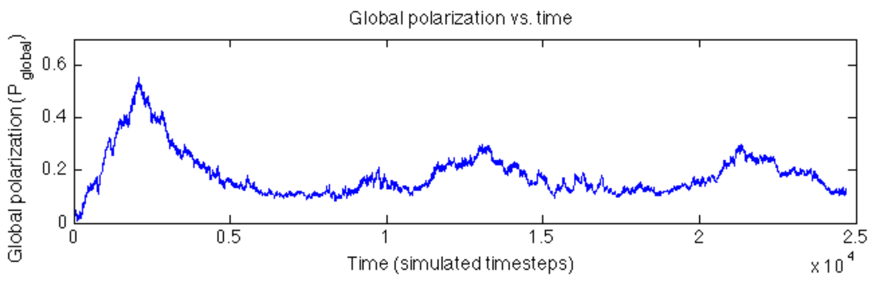
To reject H_0, the ratio of local to global polarisation is calculated. As mentioned above, this ratio provides an indicator of how different the behaviour is at the local level compared with the global level. Fig. 3.9 shows this ratio for the two experiments above. Using the Vargha–Delaney A-test, the polarisation ratio for the control case is shown to be is significantly higher than the test case ($A = 0.97$), indicating that H_0 can be rejected, and concluding that distinct sub-swarms can form and behave independently through the identity algorithm.



(a) The number of swarms varies over time as swarms split into sub-swarms, and merge back into super-swarms.

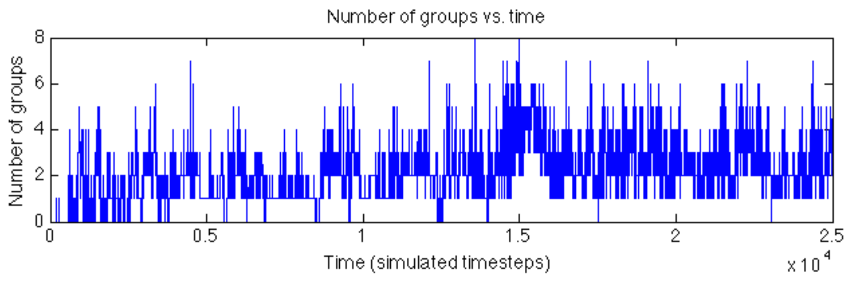


(b) Local polarisation (P_{local}) remains high over the course of the experiment (after the initial settling-down period as swarms initially form). This shows that all swarms in the experiment are coherent.

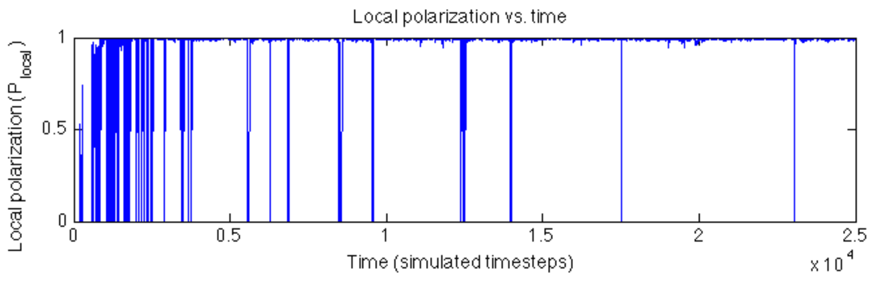


(c) Global polarisation (P_{global}) varies over the course of the experiment as the number of swarms (fig. 3.7a) varies, and remains low throughout. This shows that the swarms that have formed are not all travelling in the same direction.

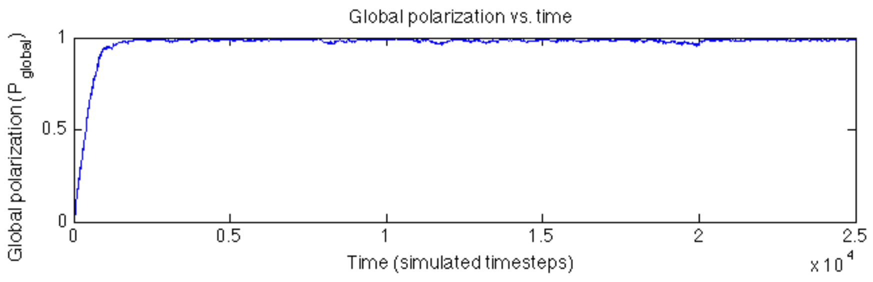
Figure 3.7: When the two algorithms are linked, local polarisation remains high throughout the experiment and the global polarisation varies, but remains low, while swarms split and merge. This shows that sub-swarms can travel in different directions (as shown by P_{global}) while still remaining as coherent sub-swarms (as shown by P_{local}).



(a) The number of swarms varies over time as swarms split into sub-swarms, and merge back into super-swarms.

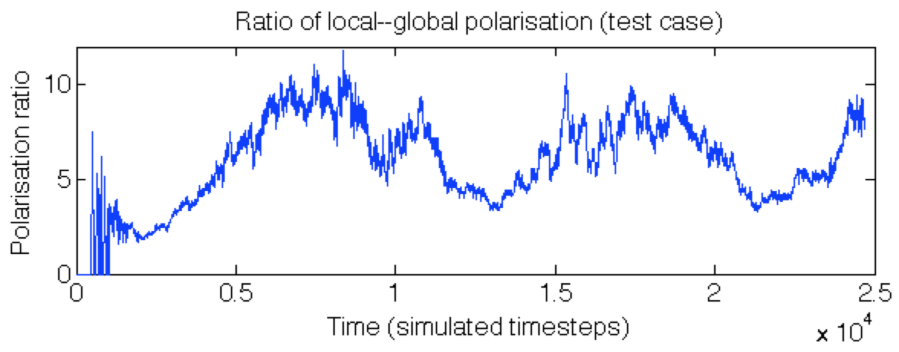


(b) Local polarisation (P_{local}) remains high throughout the course of the experiment. When the number of swarms (fig. 3.8a) is zero, P_{local} is also set to zero, which can be seen regularly through this experiment.

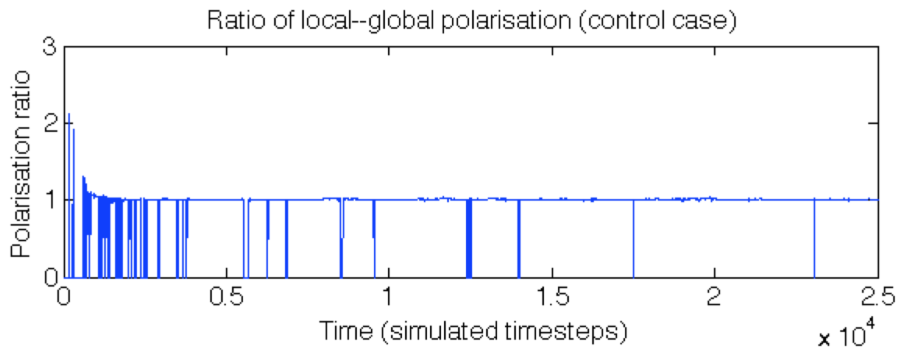


(c) Global polarisation (P_{global}) is consistently high throughout this experiment. This shows that the agents have formed into swarms that are all heading in the same direction, as in the classic boids algorithm (Reynolds, 1987).

Figure 3.8: When the two algorithms are unlinked, swarms are highly polarised, both globally and locally. This shows that all the swarms in the experiment are heading in approximately the same direction throughout.



(a) Test case polarisation ratio. A ratio above 1 indicates that the sub-swarms are more highly-polarised than the overall swarm. This shows that distinct sub-swarms have formed, and as the ratio increases, the sub-swarms are increasingly heading in different directions to each other, while remaining polarised within the sub-swarm.



(b) Control case polarisation ratio. A ratio that settles at 1 indicates that the sub-swarms and overall swarm are equally polarised. This shows that the sub-swarms are not distinct and are not behaving independently from each other.

Figure 3.9: Ratio of local–global polarisation measures ($\frac{P_{local}}{P_{global}}$) for test and control cases.

3.3.2 *Swarms preserve identity over an extended period of time*

The swarms formed by the system need to be able to persist through both time and space. To show this, specific swarms are tracked as they traverse the environment. These swarms remain intact for an extended period of time even if they occupy the same area in space as another swarm. The swarms are also resilient to small perturbations in the membership of the swarm (showing that the identity of the swarms are unaffected by individual robots switching swarms). Details of the tracking system are given in section 3.2.1.

To test RQ2, the null hypothesis is defined:

NULL HYPOTHESIS (H_0): The identity algorithm has no effect on how long swarms can survive.

Fig. 3.10 shows the length of time that each swarm exists for. There are a large number of transient swarms forming (this is a typical effect of the random start position), and fewer, longer-lasting swarms that are able to traverse the environment for an extended period of time (in this case, up to 6694 simulated timesteps, from a simulation lasting 25000 timesteps).

In the following results, the 'baseline' experiments consist of runs where the two algorithms are linked and there is no additional noise; the 'noise' experiments consist of runs where there has been additional noise added to the simulation (see section 3.2 for details of this noise); and the 'control' experiments consist of runs where the two algorithms are unlinked.

The most notable difference between the boxplots in fig. 3.10 is that the longest-lasting swarm in the baseline case is 3750 timesteps longer compared with the noise case, and 3991 longer compared with the control case. To confirm that this is typical behavior, the simulation is run again, and the longest-lasting swarm tracked for each. The results are presented in fig. 3.11, where the first two boxplots show the behaviour of the system when the two algorithms are linked, and the second two when the algorithms are unlinked (control cases).

Corroborating the previous data, the inclusion of noise in the model reduces the lifespan of the longest-lasting swarm when compared to the baseline case. As anticipated, unlinking the two algorithms—as in the control cases—has a more drastic effect on the lifespan, reducing it much more significantly compared with the

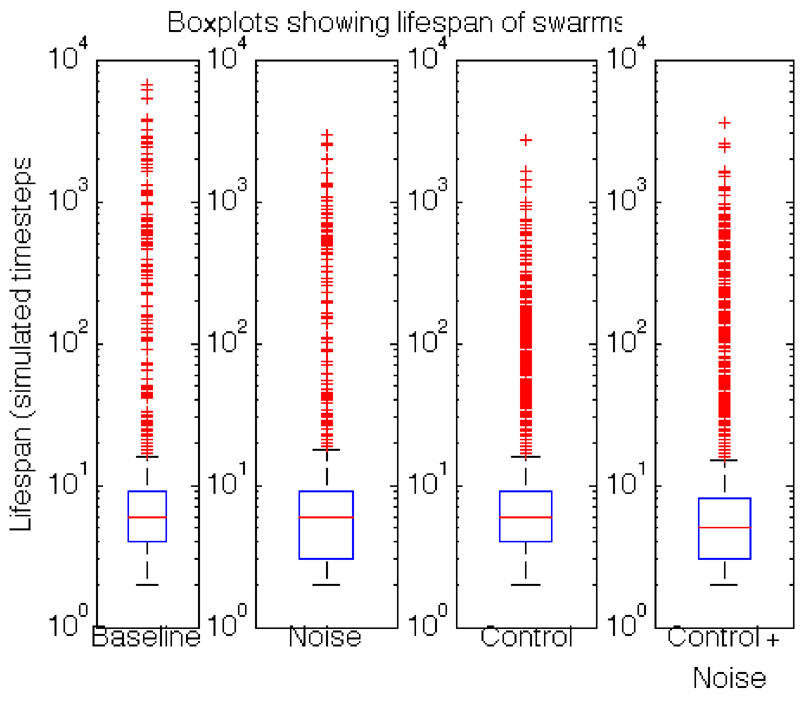


Figure 3.10: Logarithmic-scale boxplots showing the lifespan of every swarm ($\geq$ 10 agents) for baseline, control, and noisy experimental cases, over runs of 25000 timesteps. Comparing the baseline and control datasets gives $p = 0.126$ and $A = 0.519$; comparing baseline and noise gives $p = 0.0104$ and $A = 0.533$; comparing noise and control + noise gives $p = 0.0513$ and $A = 0.520$; comparing control and control + noise gives $p = 4.9e{-}14$ and $A = 0.569$. All A-Test values are below 0.66, implying that there is not sufficient evidence to suggest the data are from different distributions. Details of the tests are given in appendix A.

baseline case. When noise is included in the control case, however, there is some unexpected behavior: the inclusion of noise increases the lifespan of swarms when the two algorithms are unlinked.

This increase is due to the increased rate of firefly flashing in noisy, unsynchronised swarms: the tracking algorithm finds multiple 'swarms' in each unsynchronised swarm. This results from one swarm containing multiple 'sub-swarms' that are flashing in synchrony, rather than distinct swarms. This occurs even though there are other agents in the swarm that are not synchronised. This means that when noise is introduced, those individuals who would otherwise change swarms when they are affected by noise, or who would disrupt the synchronisation of the swarm (resulting in

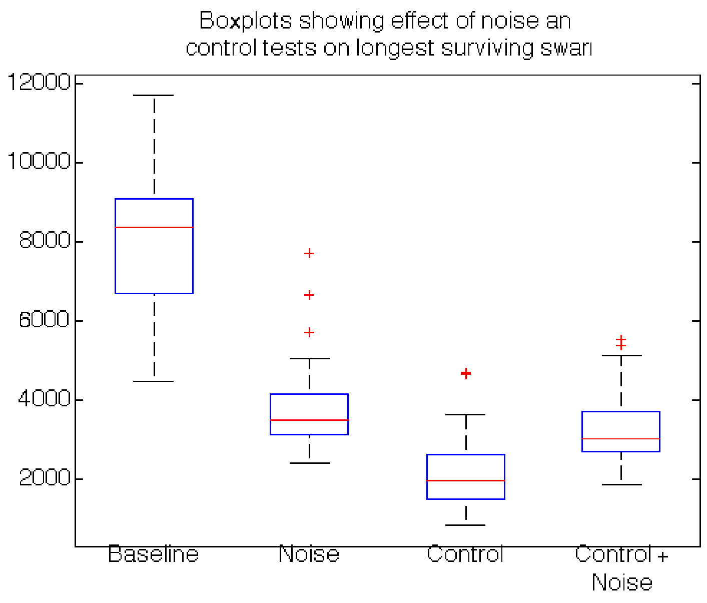


Figure 3.11: Boxplots showing longest lifespans of swarms. The algorithms are linked in the baseline and noise cases, and are unlinked in the control and control + noise cases. Comparing the distributions gives the following results: baseline and noise: $p = 1.70e{-}16$ and $A = 0.978$; baseline and control: $p = 8.98e{-}18$ and $A = 0.998$; noise and control + noise: $p = 0.0019$ and $A = 0.681$. This shows that, statistically, these are all from different distributions. Details of the tests are given in appendix A.

the swarm breaking up), will now remain in the same swarm but contribute to a different tracked swarm.

The above results show that the system consistently produces long-lasting swarms, even in the presence of noise. The control case shows that when the two algorithms are unlinked, the identity of a swarm does not persist for any extended period of time (median: 1934.5), compared to the baseline case (median: 8373), and the noisy case (median: 3472), which is as anticipated.

Considering the longest-lasting swarms in each setup (fig. 3.11), H_0 is rejected through the Vargha–Delaney A-test. The A-test results between baseline and control cases, and between noise and control + noise are both significantly different ($A = 0.998$ and $A = 0.681$, respectively). As both of these values is greater than 0.66 H_0 can be rejected, indicating that the identity algorithm does have an effect on how long swarms can survive, and so implying that sub-swarms can preserve their identity for an extended period of time.

3.3.3 *Swarms preserve identity across space*

For two swarms to preserve their identities through space, they must be able to occupy the same area in the environment at the same time without disintegrating. To measure this, the total overlap between swarms at each timestep is calculated, to determine the proportion of sub-swarms that are occupying the same area of space at the same time.

To test RQ3, the null hypothesis is defined:

NULL HYPOTHESIS (H_0): The identity algorithm has no effect on the coherence of a swarm, when encountering another swarm.

The overlap measure $O(t)$ at timestep t, is defined as the extent to which the areas of each swarm coincides with the area of another swarm:

$$O(t) = \frac{1}{K}\sum_{i=1}^{K}\left[\sum_{j=i+1}^{K}\left(\frac{o_{ij}}{0.5(A_i + A_j)} * 100\right)\right] \tag{3.3}$$

where K is the total number of swarms, A_i is the area of the swarm (calculated as the smallest rectangle that encapsulates the entire swarm), and o_{ij} is the overlap of the areas of swarms i and j.

Fig. 3.13 shows how the overlap varies as the swarms cross each other over time (for clarity, this is just a smaller section of the 25000 timesteps recorded). The plots in figs. 3.14a and 3.14b show the overlap from a fixed start point. The experiment was setup as in fig. 3.12 such that two swarms were already present, and set on paths that coincide with each other. This allows for an examination of the behaviour of the swarms without interference from stray individuals or from other groups. The plot in fig. 3.14a shows the overlap as the two swarms cross each other, with fig. 3.14b showing the same situation, but with the two algorithms unlinked. Both figs. 3.14a and 3.14b contain three plots pertaining to the median, 5th and 95th percentile of the data over 50 runs.

Finally, fig. 3.14c shows the Vargha–Delaney A-test value, comparing the test case and control case for each timestep. It is evident that between timesteps 40 and 60, there is a significant difference between the test and control cases. As this is immediately after the two swarms overlap, H_0 can be rejected, and conclude that the identity algorithm helps to maintain coherence in a swarm as it encounters other swarms, thus preserving its identity through space.

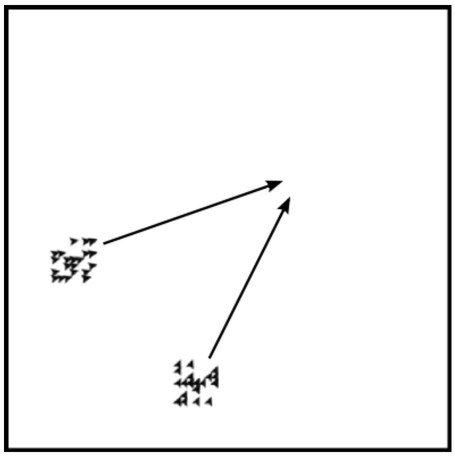


Figure 3.12: Start position of the fixed-start overlap experiment. The two swarms of 25 agents are randomly distributed within a radius of 4 patches from each start point.

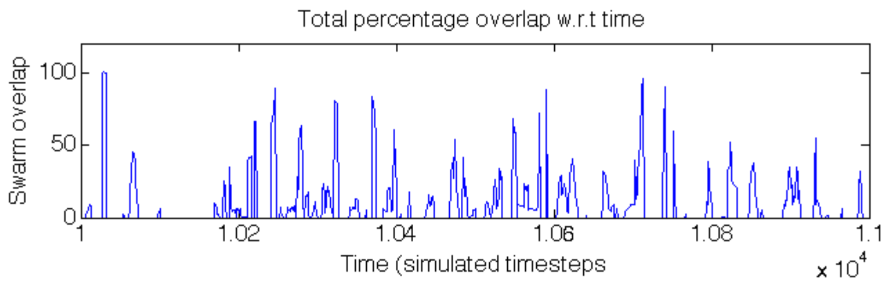
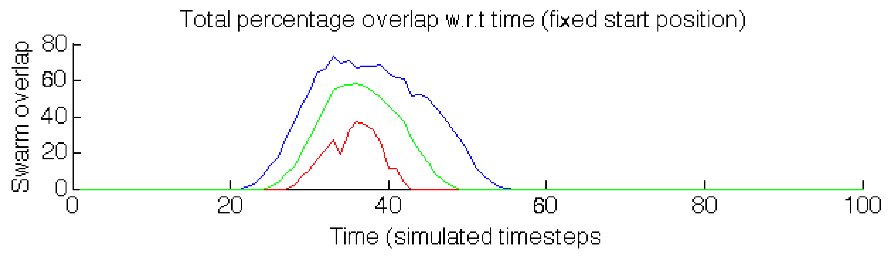
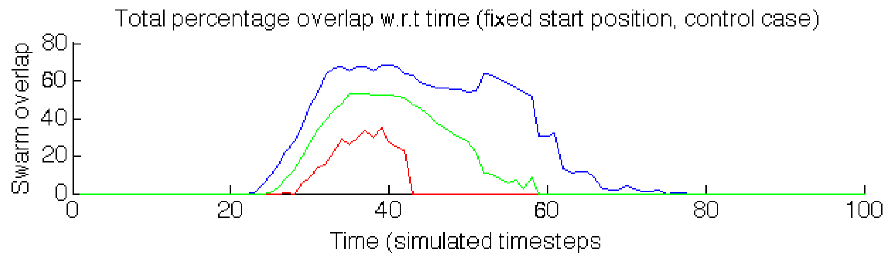


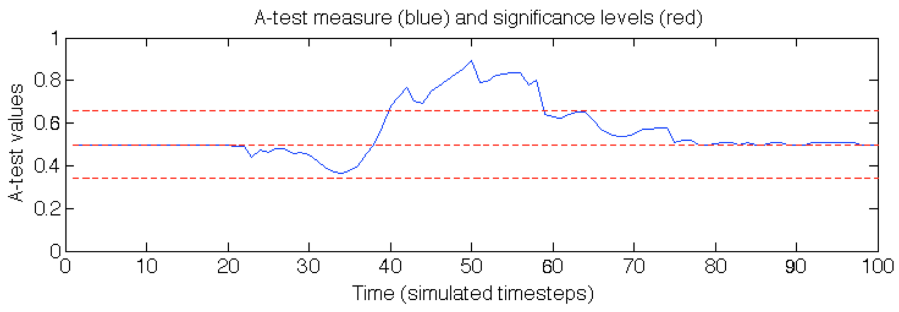
Figure 3.13: Plot showing the percentage of total overlap as swarms cross over each other. This is a smaller section of a 25000-timestep run, for the purposes of clarity.



(a) Plots showing median, 5th, and 95th percentile of overlaps from a fixed start position (over 50 runs). The consistent rise and fall of the overlap over a large number of runs shows that the swarms remain coherent while they pass across each other in space.



(b) Plots showing median, 5th, and 95th percentile of overlaps from a fixed start position (over 50 runs), with the two algorithms unlinked. The consistent rise, but inconsistent fall of the overlap over a large number of runs shows that the swarms are unable to remain coherent when they encounter another swarm in the area.



(c) Plot showing the Vargha–Delaney A-test value over time, comparing the test case and control case from (a) and (b) respectively. The dashed lines indicate no difference ($A = 0.50$), and significant difference ($A = 0.34$ and $A = 0.66$).

Figure 3.14: When the two algorithms are linked, the swarms are able to remain coherent while encountering other swarms in the environment.

3.4 DISCUSSION

Just as the cognitive behaviour of the immune system (Cohen, 2000) depends on its ability to distinguish self and not-self (Mitchell, 2005), for swarm robots to work reliably in the presence of other swarms, they require some method of determining the boundary of the system. The emergent identity provided by the presented algorithm offers this ability, making use of synchronisation between agents to allow swarms to form identities and maintain that identity over time and through space.

Providing swarms with the ability to form an identity helps to expand the possible application areas of swarm systems. Because swarms with identities are able to distinguish between self and not-self, yet still interact with other swarms in the environment, it enables multiple swarms to work in the same area without interfering with each other.

As described in section 3.1, Melucci's (1995) definition of identity consists of three criteria:

- Continuity through time
- Separation from others
- Recognition of and by others

This chapter has shown that by coupling a synchronisation method with the control architecture, swarms are able to meet all three criteria. Provided the synchronisation method is visible (in some form) to the other agents and swarms in the environment, then the specifics of the synchronisation method are of little consequence. For example, a more robust synchronisation method is likely to provide the swarm with an identity that will be less susceptible to perturbations.

The use of synchronisation in a swarm is not wholly original. It has been used previously to help guide flying robots towards a goal. By using a synchronisation method, the flying robots are able to loiter, preventing them from falling out the sky (Hauert et al., 2013). While this is different from the work presented here, the ideas are related: by using synchonisation, multiple agents/robots are able to perform tasks as a group more effectively than they would otherwise (the identity algorithm provides a mechanism for multiple swarms to work in the same area, while the loitering, flying robots use synchronisation to keep the swarm together without falling).

A specific set of parameters has been used for the results presented (given in table 3.1), but the system has been thoroughly analysed to determine how sensitive the system is to these parameter settings (see section 3.2). The most important parameter in the system in terms of scaleability is the cycle length of the firefly algorithm (i. e. how high the threshold is for each agent), as this parameter dictates how many different identities can exist simultaneously in the same area, and so explicitly imposes a limit on the system.

The results presented are entirely from computer simulations, the limitations of which are discussed in section 3.2. Further work would need to include the effect of physical robots on the algorithm, as physical robots can block the firefly signals if they are visual. This problem could potentially be alleviated by using sound or a Bluetooth communication system instead, but these could also introduce unexpected problems.

In summary, the firefly algorithm can be used to influence the control logic of agents, allowing a swarm to form an identity and maintain it through both time and space. The use of collective identity increases the scaleability of swarm systems, allowing multiple swarms to work in the same environment without interfering with each other.

The next chapter considers the problem of collectively making a decision without distributing robots across the environment. This is achieved through flocking algorithms, and so the identity algorithm presented in this chapter could easily be integrated alongside it.

COLLECTIVE DECISION-MAKING WITHOUT DISTRIBUTED SENSING

The previous chapter presented an algorithm that helps to increase the scaleability of swarm systems. By forming identities, multiple swarms are able to work in the same environment. This increases the robustness of the swarm by isolating each swarm from other swarms, but not from the environment. The work provides insight into the nature of collective systems, and suggests new methods for dealing with the problems associated with moving swarm robotics from laboratory test cases to real-world applications.

This chapter addresses objective 2 (*Derive a form of collective decision-making that does not require distributed sensing*). By not using distributed sensing, the speed with which successive decisions are made can be increased. The method used in this chapter is to collect the swarm into a flock that reacts to environmental cues collectively. As such, the work in chapter 3 can be applied to the decision-making algorithm presented in this chapter, in order to maintain multiple flocks of agents within the same decision-making environment. This would further increase the speed with which successive decisions can be made.

The work in this chapter considers the problem of how a homogenous swarm of agents are able to collectively make decisions. Contrary to previous work in collective decision-making that requires distributed sensing and the aggregation of individuals, the algorithm presented in this chapter uses flocking to keep the swarm together and the flock reacts as a whole to environmental cues, rather than as individuals. This results in an approach to decision-making that few other researchers have used. Yu et al. (2010) use this approach, but require leaders within the flock to guide it to a decision, the approach taken here removes the need for these leaders by providing a method for the swarm to generate its own (emergent) motive. As a result, this approach to decision making allows for multiple successive decisions to be made, without an external force (e. g. an experimenter) redistributing the swarm across the environment.

Addressing a number of specific questions about the ability of leaderless flocks to make cognitive decisions, this chapter asks how population size and the physical size of the flock affects the decision made, along with asking how effectively the flock can make decisions when confronted with multiple, opposing environmental cues.

The chapter is organised as follows: section 4.1 discusses different approaches to decision-making in natural and artificial systems; section 4.2 contains a description of a model for flock-based cognitive decision-making; section 4.3 presents a number of experiments investigating how the number of agents in a swarm, along with physical swarm size, can have an effect on the decision made; and section 4.4 presents initial results from an extension to the system that helps to increase the speed with which successive decisions can be made. Finally, section 4.5 concludes the chapter.

As in chapter 3, rather than make use of real-world robots, simulated swarms of generic 'agents' are used. The details of these agents are provided in appendix A, but they are essentially massless points in a virtual environment that have limited communication with each other and the environment. Using simulated agents increases the speed with which the algorithm can be tested, and—as shown in chapter 5—this does not affect the decision-making capacity of the system presented.

4.1 DECISION-MAKING

Both natural and artificial systems exhibit decision-making behaviour (Seeley et al., 1991; Garnier et al., 2005). The most prominent examples are of nest-site selection (Lindauer, 1957; Seeley and Visscher, 2004; Pais et al., 2013) and foraging behaviours (Burton and Franks, 1985; Gordon, 2002) in colonies of ants (Evison et al., 2012; Kaur et al., 2012) and bees (List et al., 2009; Makinson et al., 2010; Diwold et al., 2011; Schaerf et al., 2013).

The typical approach of collective decision-making systems is to have individual agents react to the environment and (either actively or passively) 'recruit' other agents to join them. The swarm-level decision is made through one of two methods: either (1) a quorum-sensing approach where once the number of agents with the same decision reaches some threshold, the entire swarm switches to that decision (Lindauer, 1957; Visscher and Camazine, 1999); or (2) an aggregation method, where individuals collect together at some point in the environment, and the number of agents at each point dictates the decision made (Jeanson et al., 2004; Garnier et al., 2005).

One of the most prominent examples in natural systems is the nest-site selection behaviour in bees (Lindauer, 1957). Scout bees search for a new nest site, and recruit other bees to follow them through the well-known waggle dance. This behaviour spreads from the original few scouts to others that have followed, with cross-inhibition between opposing scouts being used to prevent stalemate (Seeley et al., 2012). Eventually, sufficient scouts are recruiting for the same site that a quorum is reached, and the decision is made: the hive moves to a new site (Seeley and Visscher, 2004). This is a process that Passino et al. (2008) uses as an example for swarm cognition, and is often cited as one of the key cognitive decision-making processes in nature (Passino et al., 2008, 2010; Couzin, 2009; Goldstone and Gureckis, 2009; Reina et al., 2015a,b).

Franks et al. (2013) discuss the tradeoff between speed of the decision made and the cohesion of the swarm making the decision. From the perspective of artificial systems, cohesion of a swarm of robots after making a decision is an important factor, as it allows the system to continue working after making a decision—in real-world applications, the system is rarely expected to make a single decision and be reset. Previous collective decision-making algorithms (Kengyel et al., 2011; Parker and Zhang, 2011; Valentini et al., 2015b) do not maintain coherence of the swarm throughout the process.

By making use of flocks, coherence can be maintained throughout the decision-making process, with the entire swarm reacting together, rather than individually. The work in this chapter takes this flock-based approach, but unlike Yu et al. (2010), the work in this chapter does so without the use of 'leaders' in the swarm to guide it, allowing the swarm to generate its own (emergent) motive instead.

Cohen (2000) discusses the concept of decision-making in terms of distributed cognitive systems, and proposes a definition of cognitive decision-making:

> "[A] decision emerges... from a match between an environmental case and an internal motive. Decisions are associations." (p. 69)

He continues:

> "Decision-making is positive action; instead of passively receiving what the environment imposes, the cognitive system exerts its will... in choosing among alternatives" (p. 69)

This definition implies that for an agent to make a cognitive decision, it must be able to do more than simply react to the environment, it must be able to react differently to the same environmental situation based on the internal state of the agent.

For the cognitive decision-making algorithm in this chapter, the following terms are defined: 'context' is the state of the environment as perceived by an agent; 'motive' is the likelihood of picking certain actions, should an appropriate context arise. In an empty environment, therefore, the actions of the swarm can only be a result of the inherent predisposition of certain actions within the swarm, and is representative of the motive of the swarm.

In short, decision-making in a cognitive system is a function of context and motive:

$$Decision = f(motive, context)$$

Cognitive decision-making has been formally analysed from a 'bottom-up' perspective (Reina et al., 2015a,b), specifically focussing on the link between the microscopic (individual-level) behaviour and how it gives rise to a specific desired macroscopic (swarm-level) behaviour. The main drawback with this work is that, while Reina et al. have provided a design pattern for a single algorithm (collective decision-making through cross-inhibition), it is difficult to extrapolate from this to many other forms of decision-making systems, something that is typical of software engineering design patterns (Gamma et al., 1994).

The work presented in this chapter is a model of (microscopically-defined) swarm-level cognitive decision-making. This work differs from the typical collective decision-making approach described above, as individuals do not act alone in reacting to the environment. By collecting the swarm into a flock, the entire swarm is able to react to environmental cues. As the first agents in the flock react to an environmental cue, other agents in the flock will react to the first agents, rapidly passing the information about the cue across the flock. Through this mechanism, the entire flock is able to react very quickly to environmental cues. Previous flock-based decision-making systems have had to rely on informed individuals within the flock that guide the swarm (Yu et al., 2010). The work presented here allows the swarm to make decisions without extra information provided by leaders, while still retaining the property of cohesion (Franks et al., 2013). As such, the work presented in this chapter seeks to answer the following research questions:

> **RQ1**: Does the motive of the swarm increase as more members are added to the swarm?
>
> **RQ2**: Does the motive of the swarm increase as the physical size of the swarm increases?
>
> **RQ3**: Does adding more members to the swarm help the swarm to decide between equal options?

4.2 implementation

The previous section concluded with a list of research questions this chapter is seeking to answer. This section describes the experimental setup and simulation software that is used to answer these questions. As described in section 4.1 above, the approach to decision-making taken in this chapter is to use Cohen's (2000) definition of cognitive decision-making, combined with a flocking algorithm that allows the swarm to react quickly to environmental cues.

As discussed in section 3.2, Reynolds' (1987) 'boids' algorithm is an algorithm that produces flocking behaviour in simulated agents. Originally designed to provide realistic behaviour to animations, it is suitable for producing basic flocking behaviour, but agents in the flock do have a tendency to 'overlap' each other—a result of having massless agents that do not collide.

For the behaviour in the decision-making algorithm proposed here, agents that overlap will result in a slowing of information transfer across the flock, reducing the speed with which other agents react to the change in position. By using a flocking algorithm that provides more structure to the flock, this problem is alleviated. The algorithm proposed by Olfati-Saber (2006) pushes the agents into a semi-rigid lattice structure. The agents are less likely to overlap, as the force pushing the agents apart is sufficiently strong, without resulting in disintegration. As a result, information is able to transfer very quickly across the flock as it reacts to variations from the lattice structure.

The agents in the flock move according to three forces: the flock force, the motive force, and the context force. These three forces—described in detail below—combine to provide a final directional force that the agents follow. The flock force is provided by Olfati-Saber's (2006) algorithm, and works to keep the flock in its lattice

Listing 4.1: Pseudo-code for CDM algorithm

```
calcAgentForces()
{
    if (neighbours)
    {
        Ui = vectorAdd( calcFlockForce(),
                        calcMotiveForce(),
                        calcContextForce()
                      );
    }
    else
    {
        // if alone, move to attractor and wait to be picked
            up by a flock
        Ui = calcContextForce();
    }

    dX = Ui[0];
    dY = Ui[1];

    direction = fmodf(atan(dX, dY), 2 * PI);
    magnitude = 0.005 * sqrt(pow(dX, 2) + pow(dY, 2));
}
```

formation. The motive force is provided by flock-generated 'virtual goals' that are projected ahead of the flock. The context force provides environmental cues to the agents, resulting from a gradient towards an attractor in the environment.

The overall algorithm, combining each of the three forces listed above can be described by a series of vector-adjustments, u_i, for an agent i:

$$u_i = f_i^g + f_i^\gamma \tag{4.1}$$

where f_i^g acts to form a lattice structure amongst the neighbours in a swarm (flock force); and f_i^γ provides 'navigational feedback' towards a goal (combining motive and context forces). Our definition of u_i varies from Olfati-Saber's (2006) original definition through the removal of the directional term f_i^d, relying instead on the navigational feedback provided through f_i^γ. The algorithm is given in pseudo-code alongside the mathematical descriptions to aid in reproducibility.

FLOCK FORCE The flock force, f_i^g, provides the cohesive force that pushes the agents into a lattice formation. It is defined by Olfati-Saber (2006) as:

$$f_i^g = -\nabla_{q_i} V(q) \tag{4.2}$$

where $V(q)$ is an attractive/repulsive force based on the distance to the nearest neighbouring agents, and q_i is the two-dimensional position of agent i. f_i^g provides the force required to keep an agent i a set distance from its neighbouring agents. The full details of the flocking algorithm are provided in (Olfati-Saber, 2006), and pseudo-code provided in code listing 4.2.

By restricting the agents to only local interactions, the algorithm is applicable to both simulated and real (robotic) platforms. The use of only local interactions does, however, pose new challenges. Olfati-Saber's (2006) algorithm requires navigational feedback—in the form of a goal in the environment—in order to prevent the flock from disintegrating. With only local interactions, information about a global goal will not be easy to provide, so 'virtual goals' are used to provide this mechanism at a local level. As each agent moves through the environment, it periodically projects a new goal directly in front of it, at such a rate that it can never reach it. This virtual goal provides sufficient navigational feedback to the flock to prevent disintegration, while also providing motive to the agents (as defined in section 4.1).[1]

MOTIVE FORCE The motive force is provided through a 'virtual goal' in the environment. The motive force draws each agent toward its virtual goal, while the context force (described below) draws each agent towards any environmental attractors. The strength of the motive force varies according to how easily the virtual goal of an agent is influenced by environmental factors. If there is a particularly strong attractor in the environment when the agent re-calculates its goal, then the new goal will reflect that influence. At the same time, however, the other agents in the flock will also have an influence on the new position of the goal through the flock force, resulting in three balanced forces that depend on the number of agents and the parameters of the algorithm.

A virtual goal is calculated by each agent projecting forward by a predetermined distance **d** and using that position in the environment as its goal for a set period of time, $\mathbf{G^{update}}$. These parameters, **d**

1 The virtual goal is analogous to holding a carrot on a stick ahead of a mule

Listing 4.2: Pseudo-code for flock force calculations

```
calcFlockForce()
{
    cumulativeSum[] = {0, 0};

    foreach (neighbour in neighbours)
    {
      // calculate distance/direction to neighbour
      dist              = -distanceToAgent(neighbour);
      dir               = directionToAgent(neighbour);
      diffPose[]        = {dist * cos(dir),
                           dist * sin(dir)};

      // calculate denominator and divide...
      auxVecDivide      = 1 / sqrt(1 + eps * dist * dist);
      vecNij            = vectorMult(diffPose, auxVecDivide);

      sigmaNorm         = calcSigmaNorm(diffPose);
      phiAlpha          = calcPhiAlpha(sigmaNorm);
      result            = vectorMult(vecNij, phiAlpha);
      cumulativeSum     = vectorAdd(cumulativeSum, result);
    }

    fg = cumulativeSum;
    return fg;
}

// helper functions
calcSigmaNorm(double z[2])
{
    magSq               = pow(z[0], 2) + pow(z[1], 2);
    magSqPrime          = magSq * eps + 1;
    sigmaNorm           = (1 / eps) * (sqrt(magSqPrime) - 1);
    return sigmaNorm;
}

calcPhiAlpha(double z)
{
    // r and d are globally-defined parameters
    rSigma = calcSigmaNorm({r, 0});
    dSigma = calcSigmaNorm({d, 0});

    return calcRho(z / rSigma) * calcPhi(z - dSigma);
}

calcRho(double z)
{
    if (z < h)
      return 1;
    else if (z <= 1)
      return 0.5 * (1 + cos((PI * (z - h) / (1 - h) )));
    else
      return 0;
}

calcPhi(double z)
{
    y = z + c;
    sig = y / sqrt(1 + pow(y, 2));
    return 0.5 * ((a + b) * sig + (a - b));
}
```

and **G$^{\text{update}}$**, are set at the start of each simulation run, and do not vary throughout the run or between agents[2]. Between them, these parameters provide the above mentioned weighting towards motive or context for the agents.

Specifically, an agent i calculates its virtual goal, G_i, by taking the average heading of its neighbours, N_i, within a pre-defined radius, **r**. Let the average heading of the neighbours of agent i be ϕ_i, defined as:

$$\phi_i = \frac{1}{|N_i|} \sum_{j \in N_i} \theta_j \tag{4.3}$$

then projecting forwards by the predefined distance, **d**, gives the virtual goal for an agent i as the coordinate pair $G_i = (x_{G_i}, y_{G_j})$:[3]

$$x_{G_i} = x_i + d \cdot sin(\phi_i) \tag{4.4}$$

$$y_{G_i} = y_i + d \cdot cos(\phi_i) \tag{4.5}$$

The goal is recalculated periodically so that the motive reflects the current state of the agent, including influences from the environment. This update period is parameterised as the virtual goal update interval (**G$^{\text{update}}$**). As the interval is increased, the agent is weighted further towards the motive than the context (and vice-versa), as information from the context will influence the position of the virtual goal less often.

CONTEXT FORCE The environment is empty other than any attractor sources included. An attractor, j, exerts a force on all agents in the environment. The force at any point can be described by the Gaussian probability density function, with mean μ_j, standard deviation σ_j, and distance to the centre of the attractor from agent i, $(q_i - \mu_j)$.

$$f_i^{context}(\mu_j, \sigma_j, q_i) = \frac{1}{\sqrt{2\sigma_j^2 \pi}} e^{-\frac{(q_i - \mu_j)^2}{2\sigma_j^2}} \tag{4.6}$$

2 these could vary in a heterogenous swarm, but this is beyond the scope of this thesis.
3 the trigonometric functions *sin* and *cos* may appear to be backwards, but this is a result of the way NetLogo calculates its headings and, if appropriate, can be reversed for other simulation setups.

Listing 4.3: Pseudo-code for motive force calculations

```
calcMotiveForce(double aveHeading)
{
    if (shouldUpdateGoal())   calcVirtualGoalPos(aveHeading,
        virtualGoal);

    dist         = distanceToGoal();
    dir          = directionToGoal();

    diffPose[]   = {dist * cos(dir), dist * sin(dir)};
    qDiff[]      = {0,0};
    qDiff        = vectorMult(diffPose, c1);

    fmotive = qDiff;
    return fmotive;
}

calcVirtualGoalPos(double aveHeading, double returnVal[])
{
    virtXcor     = (xcor + (distToVirtGoal * cos(aveHeading)));
    virtYcor     = (ycor + (distToVirtGoal * sin(aveHeading)));
    returnVal[0] = virtXcor;
    returnVal[1] = virtYcor;
}
```

The agent calculates f_i^γ based on the coordinates of G_i and the gradient formed by the attractors in the environment:

$$f_i^\gamma = f_i^{motive}(q_i, p_i, q_{G_i}, p_{G_i}) + f_i^{context}(\mu_j, \sigma_j, q_i) \tag{4.7}$$

Each agent multiplies the attractor gradient by a global parameter **ctx_mult** as it senses it, in order to provide a weighting between f_i^{motive} and $f_i^{context}$. See table 4.1 for a summary of each term used in the definition of f_i^γ.

Variable	Description
q_i	Two-dimensional position of agent i
p_i	Velocity of agent i
G_i	Position of virtual goal for agent i
μ_j	Centre-point of attractor j
σ_j	Spread of attractor j

Table 4.1: Descriptions of the variables used in equation 4.7.

Listing 4.4: Pseudo-code for context force calculations

```
calcContextForce()
{
    // get attraction at neighbouring co-ordinates
    S = patchAt(0,1)->attr;
    E = patchAt(1,0)->attr;
    N = patchAt(0,-1)->attr;
    W = patchAt(-1,0)->attr;

    // differences...
    X = E - W;
    Y = N - S;

    // context multiplier (weighting of context vs. motive)
    ctx_mult = getContextMultiplier();
    X *= ctx_mult;
    Y *= ctx_mult;

    fctx[0] = X;
    fctx[1] = Y;

    return fctx;
}
```

Parameter	Description	Typical Values
`ctx_mult`	Multiplier for attractor gradient	185
`d`	Distance from agent to virtual goal	30
`G`$^{\texttt{update}}$	Update period for virtual goal	25
`r`	Radius used to calculate neighbourhood	8

Table 4.2: Typical parameter values and descriptions for the simulation.

4.2.1 *Simulation analysis*

ALEATORY UNCERTAINTY ANALYSIS As with the identity algorithm in chapter 3, the inherent stochasticity within the simulation has the potential to affect the results. Aleatory uncertainty (AU) analysis provides a method to check that any effects measured in the simulation are a result of changes to parameters. The `spartan` package (Alden et al., 2013) provides AU analysis along with a NetLogo wrapper (Alden et al., 2014).

The graph in fig. 4.1 shows that a minimum of 200 replicates is sufficient for the CDM simulation.

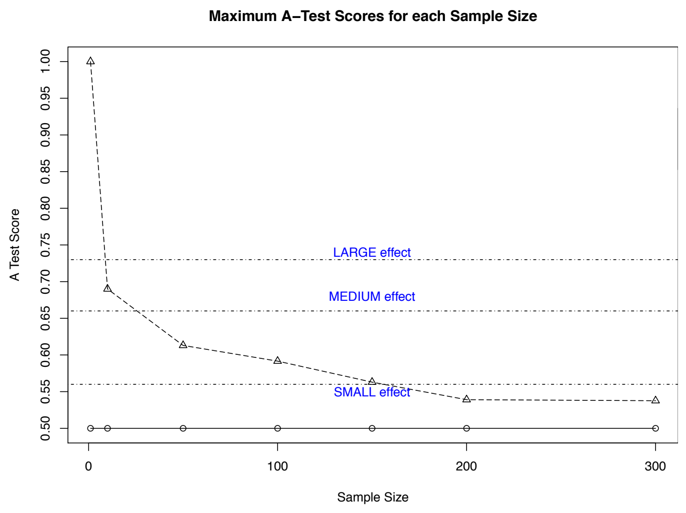
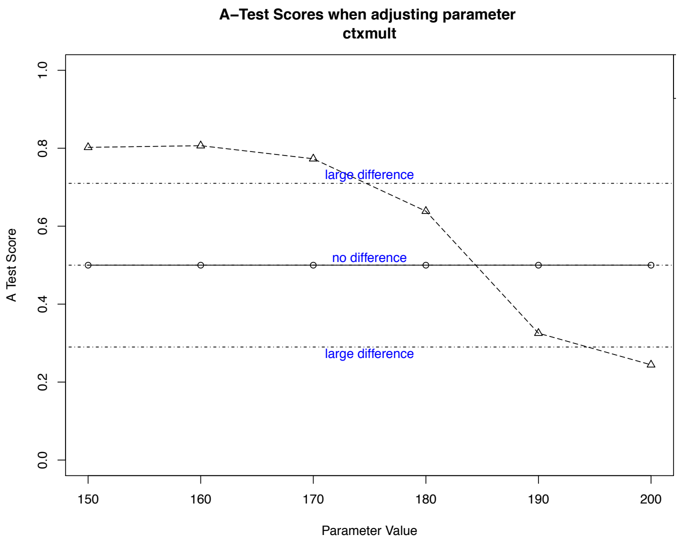


Figure 4.1: Aleatory uncertainty analysis for the CDM simulation. 200 replicates are required for this simulation, as measured through by the number of agents that remain within 2σ of an attractor source in the environment at the end of the run.

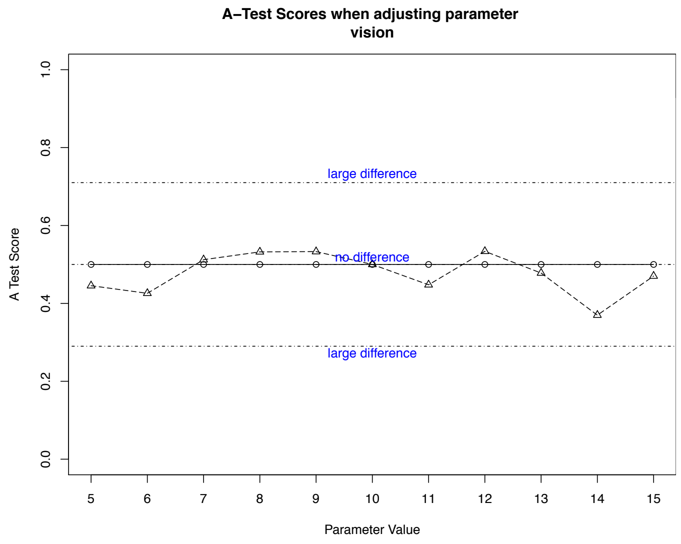
PARAMETER SENSITIVITY ANALYSIS Varying the parameters of a simulation can change the behaviour considerably. The behaviour of any simulation will be more sensitive to certain parameters than to others. In order to quantify the effect that each parameter has on the behaviour of the simulation, `spartan` offers 'one-at-a-time' parameter sensitivity analysis (Read et al., 2012; Alden et al., 2013). This method consists of holding all parameters steady at their baseline levels and varying one of the parameters at a time to see how much of an effect it has on the macroscopic behaviour of the simulation. Running this analysis on the simulation for the parameters listed in table 4.3 gives selected results in figs. 4.2a and 4.2b.

Parameter	Baseline	Min value	Increment	Max value
ctx_mult	185	150	10	200
d (dist)	30	15	5	40
population	35	5	5	75
$\mathbf{G}^{\mathbf{update}}$	25	15	5	40
r (vision)	10	5	1	15

Table 4.3: Parameters varied during the one-at-a-time parameter sensitivity analysis. Parameters are held at their baseline values when not being varied.



(a) Parameter sensitivity analysis for the **`ctx_mult`** parameter.



(b) Parameter sensitivitiy analysis for the **`r`** (vision) parameter.

Figure 4.2: One-at-a-time parameter sensitivity analysis for the (a) **`ctx_mult`** and (b) **`r`** (vision) parameters. The graph in (a) shows that the simulation is highly sensitive to variations in the **`ctx_mult`** parameter, whereas it is robust to even quite large changes in the **`r`** parameter.

4.3 RESULTS

There are a number of different areas to investigate in flock-based cognitive decision-making (CDM), this section focusses on the simplest change possible and varies the number of agents in a flock. Because the motive is a result of the interactions between individuals, more individuals should give rise to a flock that is less susceptible to being drawn into an attractor.

4.3.1 *Increasing flock size increases motive*

This section investigates the effect of the number of agents in the flock on the decision-making behaviour. As the number of agents increases, the number of virtual goals contributing to the swarm motive increases, which should decrease the probability that the swarm will follow the context rather than the motive.

The experimental setup is sketched in fig. 4.3. The flock, F, travels left-to-right across the environment, reacting to the contextual cues provided by the attractor gradient, A. Either the flock will follow the context and stay in the attractor formed by A, or follow its motive and 'escape' the attractor. The percentage of all agents that are within the attractor at the end of the run is measured.

In order to balance computational load with accuracy of results, the population value is incremented by 5 between sets of runs. To analyse these results, a measure is needed that is not biased by the value used for this increment. This allows the same method to be used, and for comparable results to be gathered, regardless of the increment used in each experiment. The method used consists of examining the effects of increasing the population over a specified 'interval'. For example, an interval of 10 would compare the runs with population values 5 and 15, 10 and 20, 15 and 25, etc. This approach shows the effect of increasing the population by a specific amount.

As the population interval increases, we expect to see a larger probability of the swarm escaping the attractor (and so, more runs ending with zero agents in the attractor).

NULL HYPOTHESIS (H_0): Increasing the population interval has no effect on the percentage of agents in the attractor.

The percentage of all agents that are within 2σ of the attractor source is measured. The environment consists of 140x140 simulated patches. The source is centred at $\mu_j = (58, 110)$, with $\sigma_j = 15$. The flock is centred at $(15, 72)$, with a spread that is proportional to the number of agents in the flock.

The population value—i. e. the number of agents in the flock—is varied between 5 and 75, at intervals of 5. Each population value is run for 200 replicates, as calculated using aleatory uncertainty analysis of the simulation (see appendix A.3.1 and (Alden et al., 2013) for details).

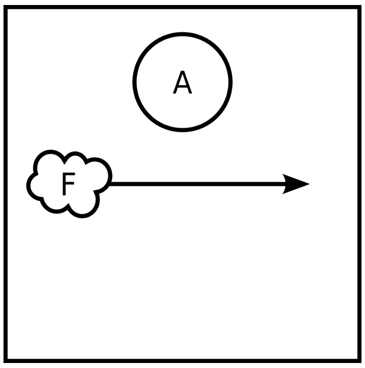


Figure 4.3: Sketch showing the initial layout of the simulation for this experiment. The environment consists of 140x140 simulated patches. The flock, F, is initialised with each agent at a random location within a specified radius of the start point (15, 72). The attractor source, A, is centred at μ_j = (58, 110), with spread defined as $\sigma_j = 15$.

The nature of the experimental setup means that clustering in the data is highly likely. Most runs will result in either all the agents in the attractor, or none at all. Because of this clustering in the data, the median is very sensitive to small changes in the size of the two clusters, meaning a boxplot is not sufficient to represent the data in a form that can be interpreted easily (see fig. 4.4a). By measuring the size of each of these clusters, along with those that are not part of a cluster (as described above), the distribution can be represented through stacked bar charts instead (fig. 4.4b).

Fig. 4.4 shows a clear progression from 100% to 0% (i. e. from context to motive) as the number of agents in the flock is increased. In order to reject H_0, the increase in population interval (the difference in number of agents) needs to result in decreasing the likelihood of the flock being in the attractor.

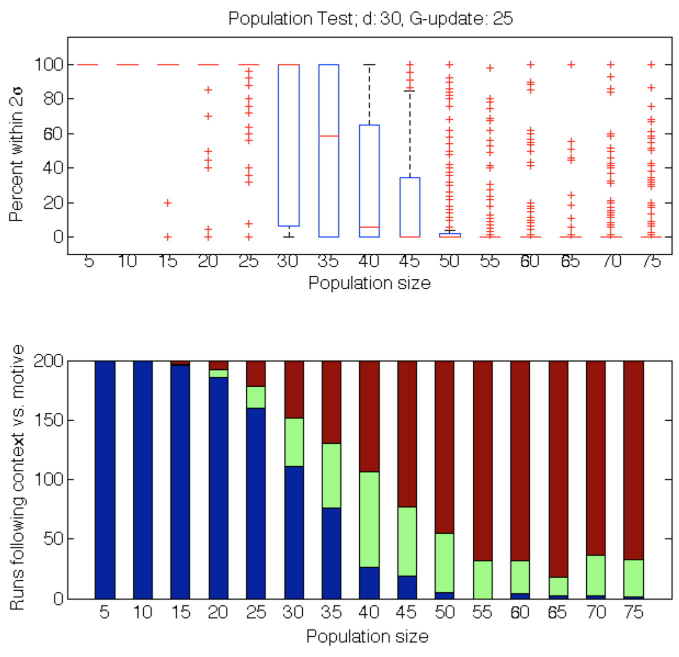
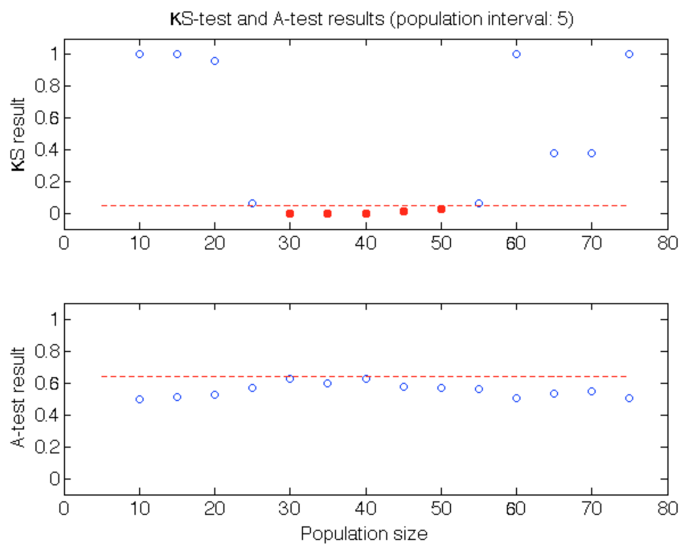


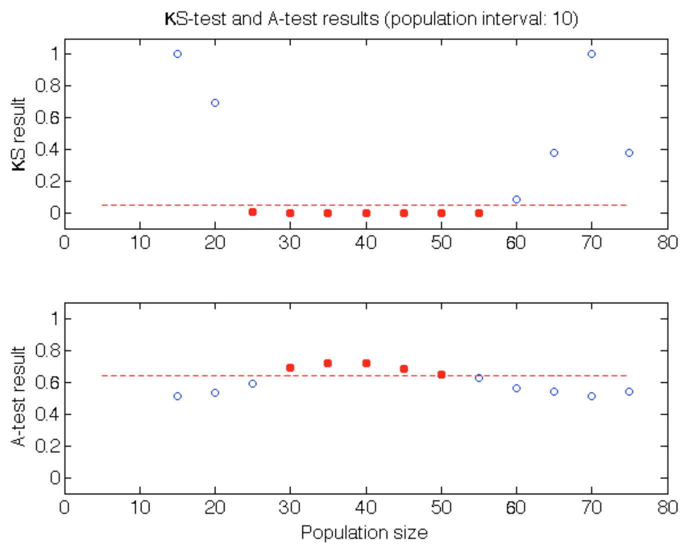
Figure 4.4: Three graphs showing how the behaviour of the system switches from following context to following motive as flock size increases. The top graph shows boxplots for each swarm size between 5 and 75 at 5-agent increments. It is evident that there is a large change between 30 and 40 which is difficult to characterise. The lower graph shows the progression between context (blue) and motive (red) based on the comparative size of the clusters at 100% and 0%, respectively.

The clustered nature of the data invalidates any statistical test that measures a change in median, as the median is very sensitive to minor changes in the size of the clusters. The Kolmogorov–Smirnov test (Massey, 1951), however, is more sensitive to the shape of the distribution. The Kolmogorov–Smirnov test shows that the distributions are significantly different from each other. Those that are different are compared using the Vargha–Delaney effect-magnitude test (Vargha and Delaney, 2000).

Fig. 4.5 shows the output of these tests. When considered alongside fig. 4.4, there is a statistically-significant transition period between population values 30 and 50. Fig. 4.5b shows that a population interval of 10 is sufficient to confirm the magnitude of this effect over the same transition period. From this, it is evident that the null hypothesis H_0 can be rejected with 95% confidence.



(a) Kolmogorov–Smirnov (top) and Vargha–Delaney A-test (bottom) results for population interval 5.



(b) Kolmogorov–Smirnov (top) and Vargha–Delaney A-test (bottom) results for population interval 10.

Figure 4.5: Two sets of graphs showing the effect of increasing the population by 5 (a) and 10 (b). As shown in fig. 4.4 above, there is a clear transition period between population values 30 and 50, with significantly different behaviour when the population value is changed by at least 10. Significant results (with 95% confidence) are shown as red filled circles, other results as blue hollow circles. Significance levels are given by the dashed red lines.

4.3.2 *Does the size of the swarm or number of agents affect the motive?*

The results presented in section 4.3.1 show that an increase in flock population results in the flock following its motive, rather than the environmental context. This section investigates whether it is the increase in the number of agents within a flock, or an increase in the physical size of the flock that has this effect.

The environment is scaled based on the number of agents. By measuring the density of agents within the environment, a multiplier, m, can be calculated for the size of the environment and attractor. By controlling for the physical size of the flock, it can be determined if the behaviour seen in section 4.3.1 results from the change in number of agents, or from the change in physical size of the flock (a natural result of changing the number of agents).

NULL HYPOTHESIS (H_0): An increase in the physical size of a flock will not cause an increase in the probability of following motive.

EXPERIMENTAL SETUP: The agents react to the attractors based on the gradient rather than on single point values. As such, the attractor function needs to be scaled along with the environment. If the distance from an agent to the attractor source is originally $x = (q_i - \mu)$, this can now be scaled to mx, and the same with σ to $m\sigma$.

The derivative of the Gaussian attractor function (eqn. 4.6) is used to assert that the gradient will be the same after scaling by m:

$$f(\mu, m\sigma) = \frac{1}{\sqrt{2\pi}m\sigma} e^{\frac{-m(x-\mu)^2}{2(m\sigma)^2}} \tag{4.8}$$

$$\text{assuming } \mu = 0 \tag{4.9}$$

$$\frac{d}{dx} = -\frac{xe^{\frac{-x^2}{2\sigma^2}}}{m\sigma^3\sqrt{2\pi}} \tag{4.10}$$

When compared to the derivative of the attractor function:

$$\frac{d}{dx} = -\frac{xe^{\frac{-x^2}{2\sigma^2}}}{\sigma^3\sqrt{2\pi}} \tag{4.11}$$

there is a difference of $\frac{1}{m}$. To counter this, the **ctx_mult** parameter is altered for each run. This means that the runs are not comparable with other experiments unless $m = 1.0$, but this does not affect any of the results in this thesis.

As before, the number of agents is increased, and the agents are measured within 2σ of the attractor source. To determine whether the physical size of the flock or the number of agents has an effect on the behaviour, there are two environmental setups: one with $m = 1.0$, one with $m = 1.5$. By pairing runs from the first setup with runs from the second based on the number of agents (e. g. 16 agents in the first setup with 24 in the second), this setup allows the physical size of the flock to remain constant in relation to the size of the environment. By comparing the two setups in this way, it becomes evident whether the physical size of the flock or the number of agents is causing the behaviour presented in section 4.3.1.

As before, the percentage of agents that remain within 2σ of the attractor source is measured. The source is centred at $\mu_j = (58m, 110m)$, with $\sigma_j = 15m$ and flock centre at $(15m, 72m)$, where m is the multiplier calculated as above.

For environmental setup one ($m = 1.0$), the number of agents varies between 8 and 76, at intervals of 4. For environmental setup two ($m = 1.5$), the number of agents varies between 12 and 114, at intervals of 6 (i. e. the same as setup one, but scaled up by m). The resulting datasets are compared using the Kolmogorov–Smirnov test and Vargha–Delaney effect-magnitude test.

To reject the null hypothesis H_0, h_0^i needs to be rejected for each of the paired data sets, where:

$$h_0^i : \ |F_Y - F_X| \geq \Delta \tag{4.12}$$

Using the Kolmogorov–Smirnov test statistic (D^+) to test for equivalence (details in appendix A) gives:

$$h_0^i : \ D^+ \geq \Delta \equiv |max(F_Y - F_X)| \geq \Delta \tag{4.13}$$

where F_X and F_Y are the empirical CDFs for the two paired data sets, D^+ is the maximum difference between F_X and F_Y, and $\Delta = 0.1601$ (follows from theorem 3.5 in (Gibbons and Chakraborti, 2011)).

RESULTS: Values for D^+ are given in fig. 4.6, showing that h_0^i can be rejected at the 95% confidence level for all i, and hence H_0 is rejected with 95% confidence. This implies that that it is the physical size of the swarm relative to the size of the attractor that affects whether the swarm is more likely to follow motive or context.

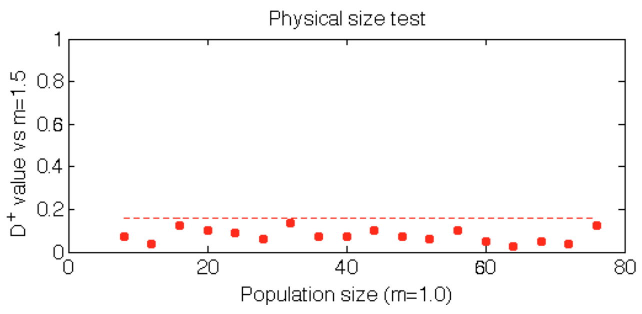


Figure 4.6: Graph showing the effect of increasing the physical size of the flock. Population values are listed for $m = 1.0$, graph shows the result of comparing with corresponding population value at $m = 1.5$ (e. g. comparing population of 12 with 18, with a comparatively larger environment such that the physical size of the swarm remains the same). All comparisons produced significantly similar results at 95% confidence ($D^+ < \Delta$, where $\Delta = 0.1601$).

4.3.3 *How does the presence of multiple attractors affect the decision made?*

Section 3.1 investigated the idea that increasing the number of agents in a swarm would affect whether the swarm followed its motive or the contextual cues provided to it, when faced with a single attractor source. This section looks at whether similar effects are observed when the swarm is presented with two attractors. By positioning the two attractor sources equidistant from the start point of the swarm, the attraction towards both the two sources is equal, and so the effects that motive has on the decision-making process can be observed.

In the context of this experiment a 'decision' is interpreted as the situation where at least 95% of the flock remains in the same attractor, or neither attractor, at the end of the experimental run.

NULL HYPOTHESIS (H_0): Increasing the population interval has no effect on the probability of the flock making a decision.

EXPERIMENTAL SETUP: The number of agents within 2σ of both attractor sources are measured. In this case, there are two attractor sources: $\mu_1 = (58, 110)$, $\mu_2 = (58, 34)$ and $\sigma_1 = \sigma_2 = 15$. The flock is centred at $(15, 72)$, with a spread that is proportional to the number of agents in the flock. We vary the population value (the number of agents in the flock) between 5 and 75, at intervals of 5.

Fig. 4.7 shows the output from the experiment. Each replicate produces two values, measuring the number of agents within 2σ of the respective attractor source. The two datasets are statistically similar (shown using the same method as used in section 4.3.2), meaning they can be reduced down to a single dataset using the following method, simplifying the analysis. The data between the two sources is not independent (if an agent is in one of the sources, it cannot be in the other). Therefore those results with $>= 95\%$ in one source can replace the corresponding $<= 5\%$ in the other source. As the two datasets are statistically similar before this, the rest of the data can be left the same, reducing the data down to one dimension, and making it amenable to statistical analysis.

With this one-dimensional data, the same approach can be used as in section 4.3.1, applying the Kolmogorov–Smirnov test (Massey, 1951) to increasingly large population intervals. Unlike before, however, the data here are not suitable for analysis with the Vargha–Delaney A-test (Vargha and Delaney, 2000), so in order to ensure

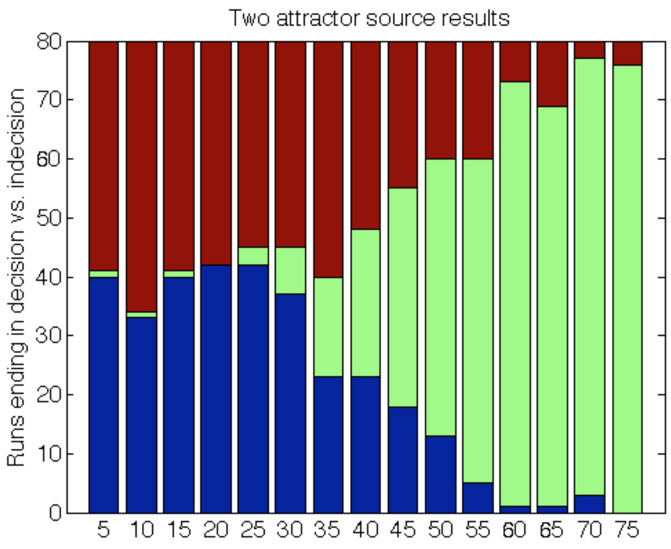


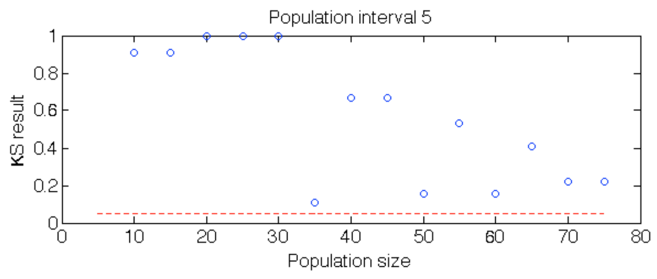
Figure 4.7: Stacked bar chart showing the effect of increasing the population in a dual-attractor setup. As the number of agents is increased, the probability of the swarm being able to make a decision decreases (i. e. it is less likely to pick just one attractor). Blue denotes following context and $>$ 95% of agents in the same attractor; red denotes following motive and $<$ 5% of agents in any attractor; green is all other situations, denoting indecision.

there are no artificially-generated low p-values, the number of replicates is restricted to be below 85 for $\alpha = 0.05$ and statistical power of 90% (Stepney, 2015).

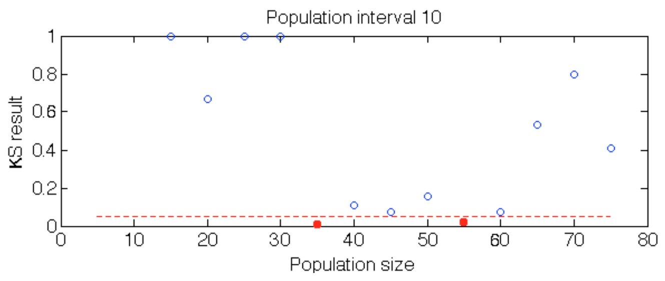
Fig. 4.8 shows that once the population interval reaches 10, significant changes are observed. Once the population interval exceeds 15, there is a very clear transition period between 35 and 60. From this the null hypothesis can be rejected with 95% confidence, concluding that increasing the number of agents in a flock by 15 does reduce the probability of the flock making a decision when faced with two opposing contextual cues.

4.4 SENSOR DESENSITISATION

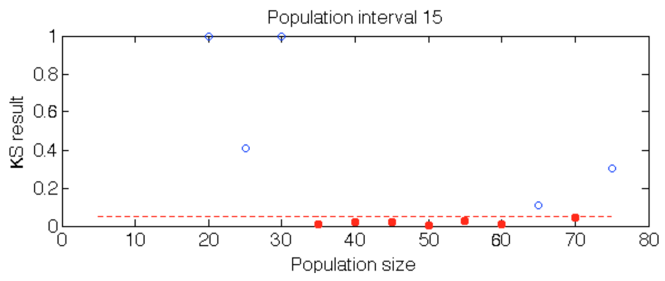
As discussed in section 4.1, real-world applications often demand that the system be able to make multiple, successive decisions. If the swarm in the presented system follows motive and does not stay in the attractor then it should be able to continue making further decisions. If the swarm does not follow motive, however, then it will be stuck in an attractor until the environment changes—in the case



(a) KS results for population interval 5



(b) KS results for population interval 10



(c) KS results for population interval 15

Figure 4.8: Three graphs showing the effect of increasing the population in a dual-attractor setup. Kolmogorov–Smirnov (KS) test results for population intervals between 5 and 15.

of a dynamic environment—or will be stuck in the attractor if the environment is static.

This section looks at one approach to providing the swarm with the ability to escape an attractor after a short period of time, based on the concept of olfactory fatigue (the process through which we 'get used to' a smell). This allows the swarm to make a decision, and then regardless of the decision it made, allows it to make further decisions.

The work presented in this section does not specifically ask questions of the system, but is instead an implementation of a new feature in the system. As such, the work in this section is presented in an engineering fashion (with requirements and tests), rather than the scientific fashion used throughout the rest of the chapter.

AIM: Allow the flock to escape an attractor, without compromising the existing behaviour.

REQUIREMENTS:

Req. 1: The flock must escape the attractor without disintegrating.

Req. 2: The flock must make the same decisions as before.

The addition to the system consists of reducing the effect that the attractor has on an agent if the agent has been close to a source for an extended period of time. Each agent can not only measure the gradient towards an attractor at each time step, but also measures the value itself (referred to as the concentration at that point). While said concentration is sufficiently high, each agent increases a counter ($d_i^{context}$) by a small amount each timestep. Once $d_i^{context}$ is greater than 1, it will be divided into the $f_i^{context}$ term from eqn. 4.7:

$$f_i^{\gamma} = f_i^{motive} + \frac{f_i^{context}}{max(1, d_i^{context})} \tag{4.14}$$

To test whether the system is able to successfully escape the attractor, the environment is set up as sketched in fig. 4.9. By varying the start point of the flock, F, across the x-axis (between values of 25 and 115 at intervals of 5), the setup mimics different approach angles to the attractor source, A. The agents are set up with a high predisposition towards the contextual cues (i. e. high **`ctx_mult`**, see section 4.2 for details) to encourage the flock to get stuck in the attractor, in order to test the ability of the system to escape when the flock is heavily weighted towards remaining in the attractor. The simulation is run for 150 replicates, in line with the aleatory uncertainty analysis performed on the simulation.

Fig. 4.10 shows the effect that sensor desensitisation has on the swarm. Fig. 4.10a shows two sets of stacked bar charts, the top graph is the control case (without desensitisation), where over 90% of runs results in the swarm getting stuck in the attractor. The bottom graph shows the anticipated effect, with approximately two-thirds of the results now escaping the attractor with the help of desensitisation.

Fig. 4.10b shows the field lines pertaining to the attractor formed by the attractor source, and a trace of a single agent starting from x-coordinate 30. It is evident that the swarm is caught in the attractor for an extended period of time, but eventually the desensitization

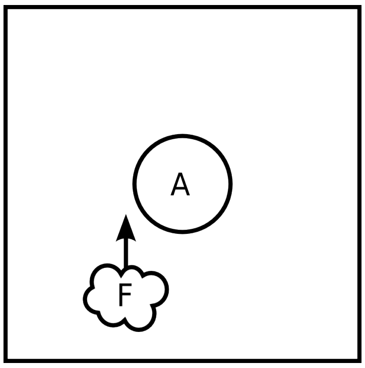
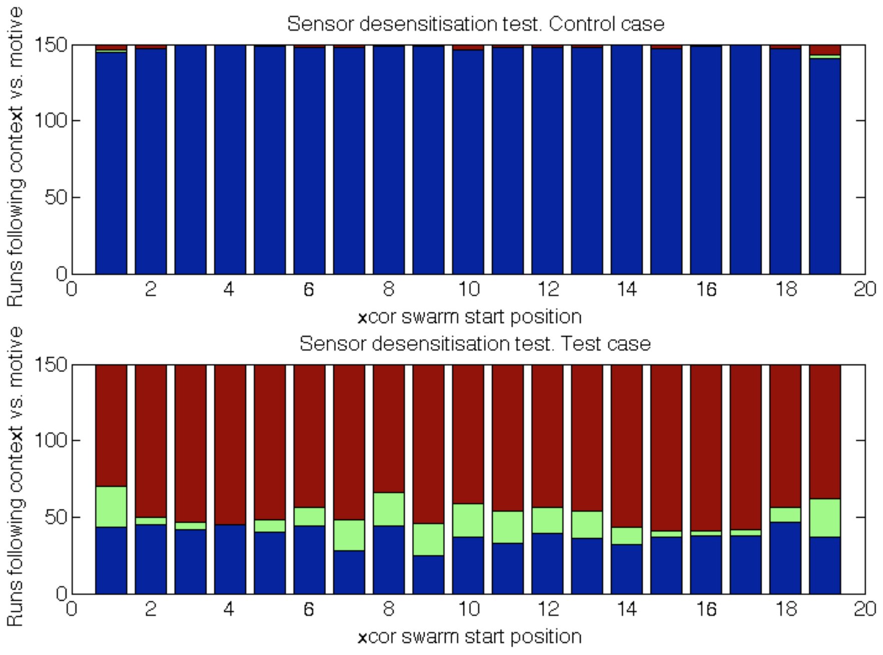


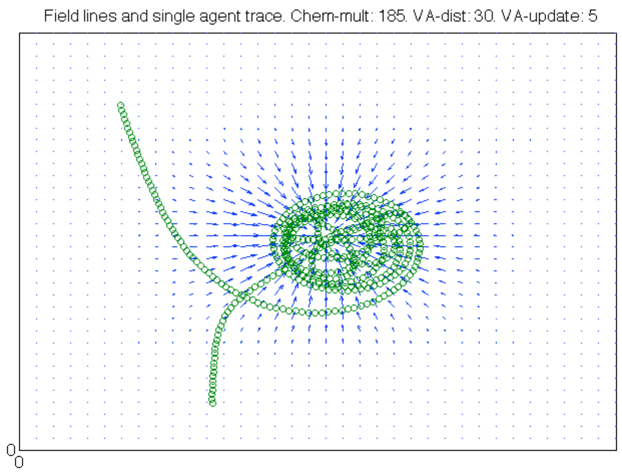
Figure 4.9: Diagram showing the initial layout of the simulation for this experiment. The flock, F, is initialised with each agent at a random location within a specified radius of the start point $(x,\ 15)$, where x is an input parameter, varied between 25 and 115 at intervals of 5. The attractor source, A, is centred at $\mu_j = (70,\ 70)$, with spread defined as $\sigma_j = 15$.

takes hold and the swarm is able to escape and continue traversing the environment.

Requirement 2 states that the flock must make the same decisions as before. To test this, the single- and dual-attractor experiments from section 4.3 are re-run, comparing the original results with the new system. Fig. 4.11 shows that when comparing to the either experimental setup, there is statistically-similar behaviour (using the same equivalence testing method as section 4.3.2).

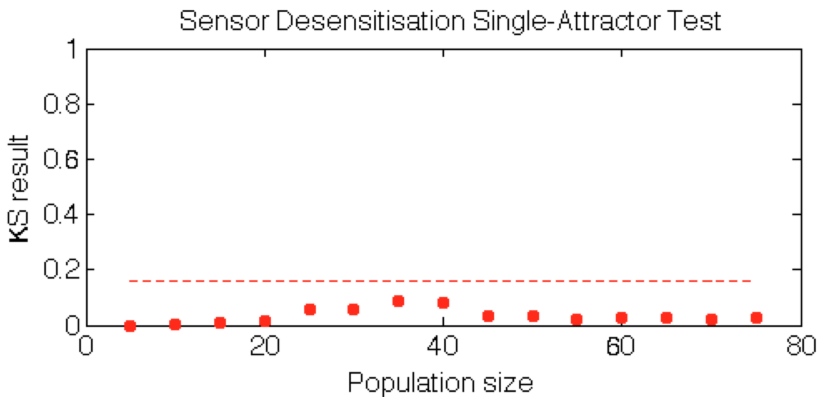


(a) Two graphs showing the effect of including sensor desensitisation in the model. The top graph shows the behaviour of the system without desensitisation (the system is set up to encourage the flock to follow the context), and every run gets trapped in the attractor. The lower graph shows the effect of including sensor desensitisation in the model, with a large number of the runs able to escape the attractor.

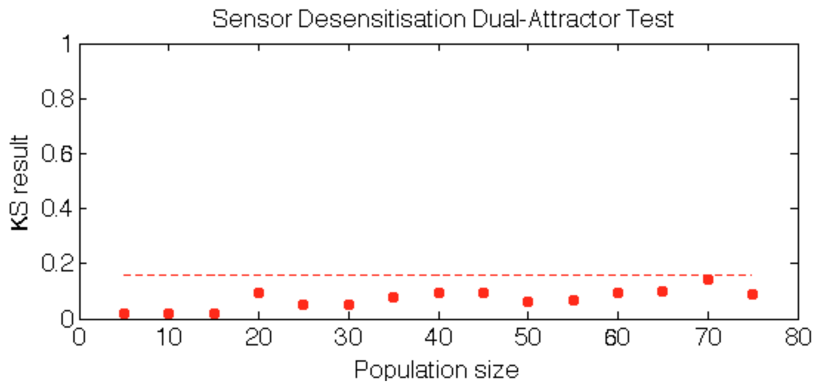


(b) Field lines and trace of a single agent in the flock showing the flock getting trapped in the attractor, then escaping later in the simulation.

Figure 4.10



(a) Single-attractor setup



(b) Dual-attractor setup

Figure 4.11: Two graphs showing the effect of the desensitisation extension on the decision-making abilities of the system. For single (a) and dual (b) attractor setups, the behaviour of the systems is compared to the results presented throughout section 4.3. All behaviour is statistically similar to the behaviour before the extension is implemented.

4.5 DISCUSSION

The results presented in this chapter show that a homogenous flock of agents is able to make decisions without relying on distributed sensing through a cognition-inspired algorithm. The use of Olfati-Saber's (2006) flocking mechanism is key to this ability, as it allows information about the environment to spread through the swarm of agents very quickly due to its rigid lattice-based structure (in contrast to the much more fluid structure that features in Reynolds' boids algorithm (Reynolds, 1987)).

Previous efforts at flock-based decision-making consist primarily of (Yu et al., 2010) which requires leaders within the flock to guide it towards a goal in the environment. The system presented is able to react to environmental cues and act according to its emergent internal motive to make a decision. This use of emergent motive allows the agents in the system to remain coherent and make decisions through a leaderless, homogenous flock.

Many notable examples of collective decision-making are based on the idea of aggregation at a certain point in space (Garnier et al., 2005; Schmickl et al., 2007, 2009). This approach makes use of distributed sensing to more effectively sense the environment, then decides on the best option from those sensed by the individuals. This is in direct contrast to the flock-based approach taken here. By using a flock to keep the swarm together, information about the local environment is rapidly distributed amongst the swarm, and so a decision can be made more quickly. This approach has an impact on accuracy, as it is possible that elsewhere in the environment is a 'better' decision, but real-world applications rarely need the global optimum, a heuristic approach is often sufficient (Russell and Norvig, 2003).

Natural collective decision-making systems exhibit a speed–cohesion tradeoff (Franks et al., 2013). The use of flocking behaviour as a control mechanism allows for the reduction in the effect of this tradeoff on the decision-making process. While the disparity between speed and cohesion has been reduced in this approach, the tradeoff emerges in other ways—most notably through the speed–accuracy tradeoff. The system is able to make a decision quickly and cohesively, but by keeping all agents in one area in the environment the swarm becomes more susceptible to local variations in the environment, and so is less able to find the global optimum. By allowing multiple flocks to work in the same environment without interfering with each other, it should be possible to help alleviate this problem. While no results have been presented to confirm this, the work on identity formation presented in chapter 3 provides an appropriate starting point for future work in this area.

In summary, this chapter has reported the development of a decision-making algorithm based on Cohen's (2000) definition of distributed cognition. Section 4.1 takes Cohen's (2000) definition of cognitive decision-making and reduces it to a function of internal motive and environmental context. The environmental context consist, in this case, of cues from attractor sources in the simulated environment. The internal motive emerges from the interactions between the agents in a flock. Section 4.2 details the development of the simulation platform, and is used in section 4.3 to answer three research questions about the behaviour of the cognitive decision-making system. The research questions focus primarily on the effect of varying the number of agents or physical size of the flock on the decision-making process. As the number of agents increases, the strength of the internal motive increases, compared with the context. This is analogous to the idea of how social influence can change the decisions of individuals in a group (Lorenz et al., 2011).

Incorporating cognitive decision-making into a robotic system, without requiring distributed sensing, offers a promising step towards cognitive behaviour. This raises the possibility of having a system that can decide the best course of action based on its current motive, taking into account the mission objectives, and the current environmental situation. Chapter 5 shows how the decision-making system presented in this chapter can be used as a virtual swarm on a simple robotics platform (Hilder et al., 2014) as part of an adaptive homeostatic system.

COGNITIVE ADAPTIVE HOMEOSTASIS IN ROBOTS

The previous two chapters have looked at two different areas of distributed cognition: emergent identity formation and cognitive decision-making. The results presented have been collected using agent-based simulations, which increases the speed that experiments can be run. This chapter, on the other hand, makes use of both simulation and a robotic platform for the experiments. The use of a robotic platform shows that the effects of the 'reality gap' are not so substantial as to invalidate the simulated results (Jakobi et al., 1995). In this chapter, simulation is used only for basic testing of the control architecture, to ensure that the system works as expected in a perfect, noise-free environment—if it fails in a perfect environment it is far less likely to function correctly in a noisy environment.

This chapter reports the development of the CAH architecture (Cognitive Adaptive Homeostasis). The CAH architecture takes the approach of decision-making from chapter 4 and pairs it with an associative memory neural network. This associative memory provides the 'internal image' of the environment that is discussed at length by Cohen (2000) and in section 2.3.2. This combination provides another incremental step towards using cognition in a robot. The results presented in this chapter focus on using cognition as the basis for adaptation and homeostasis in a robot.

The chapter is organised as follows: section 5.1 discusses the problem of artificial homeostasis, presenting alternative approaches taken by other researchers. Section 5.2 describes the CAH architecture that allows the robot to use the cognitive decision-making abilities described in chapter 4 for homeostasis. Section 5.3 presents the results from our experiments with the robotic platform. Finally, section 5.4 concludes the chapter with some discussion on the link between cognition and homeostasis.

5.1 ARTIFICIAL HOMEOSTASIS AND ADAPTIVITY

Biological homeostasis is *"the automatic regulation of physiological functions"* (Vernon, 2015, p. 94). This regulation is provided through the interconnected mechanisms of the nervous, endocrine, and immune systems that react to the internal state of the system (Neal and Timmis, 2003). By altering the high-level behaviour of the organism, the regulatory functions are able to provide an appropriate low-level environment for the various biochemical reactions taking place. A (simplified) example, in humans, is that the acidity of the blood is used to sense the level of carbon dioxide in the body. Carbon dioxide is carried through the bloodstream as carbonic acid. Therefore, as the concentration of carbonic acid in the blood increases, the acidity of the blood will increase. As the acidity increases, our body reacts by increasing our breathing rate, which expels more carbon dioxide, helping to reduce the acidity of the blood. This change in high-level behaviour in response to a small variation in an internal value is a prime example of the process of biological homeostasis.

Artificial homeostasis is the implementation of appropriate control mechanisms to provide homeostatic behaviour to an artificial system (such as a robot). This has been approached in a number of ways previously, most notably by mimicking the biological regulatory mechanisms that give rise to homeostasis. Neal and Timmis (2003, 2005); Vargas et al. (2005) combined artificial neural networks, artificial endocrine systems, and artificial immune systems in a robot control architecture, directly mimicking the homeostatic processes at work in the body.

The architecture proposed by Neal and Timmis (2003) uses an artificial endocrine system to switch between neural networks that are trained to do separate tasks. The artificial endocrine system provides a global 'hormone' that affects the behaviour of the neural networks. This hormone acts as a multiplier for the weights of the networks, allowing the hormone levels to decide which neural network is dominant (and hence, the behaviour of the robot). This approach involves building and training multiple, distinct neural networks that provide the behaviours the robot is expected to perform. The endocrine system can then switch between these trained networks. Neal and Timmis (2005) showed that the architecture could adapt through the use of an artificial immune system that acts on the nodes and connections of the other two systems. By acting on the components of the neural and endocrine systems, the immune system allows the architecture to develop from a simple starting point to a system that has learnt and adapted to the environment it is exposed to. The three systems are able to regulate internal processes, read from sensors and drive actuators, monitor internal state and remove problematic areas of the system. This results in an architecture that has the capacity to provide artificial homeostasis to a robot. The disadvantage of this approach is the speed with which it is able to adapt. In order to provide new behaviour to the robot, the neural and endocrine networks need to be trained to provide the new behaviour (either by an external force, or through the adaptive immune system).

The neuroendocrine approach to regulating the behaviour of a robot is an appropriate one, providing a mechanism through which sensors are able to alter the high-level behaviour of a neural network (section 2.3.1 provides a more thorough description of the approach). Stradner et al. (2009); Schmickl et al. (2011) follow a similar approach but without the neural network. The sensors on the robot secrete hormones that interact with each other and diffuse around the robot. The actuators react to the concentrations of each hormone in the system around them, providing movement in the environment. The resulting network has no capacity to adapt to new environments, but is able to avoid collisions.

This approach to homeostasis—directly mimicking mechanisms in the body—has its advantages and disadvantages. The advantages are that, because this system is already implemented and working in almost all living systems, there is substantial evidence that it works. All that is required is to work out what is happening and to copy it, abstracting away any unnecessary biochemical details. The disadvantage is that simulating biological systems, even after abstraction, is complicated. Biology is complex, abstracting away seemingly minor details can alter the behaviour of the system in unexpected ways (Andrews et al., 2010).

The approach taken in this chapter is to look at homeostasis from the perspective of cognition and autonomy (Vernon, 2015). The high-level behaviour of homeostasis has many similarities to cognition, such as making decisions about the high-level behaviour of the system, adapting to variations in the low-level environment. Much in the same way as Neal and Timmis (2005) proposed using an artificial immune system to provide adaptation, this chapter looks at how distributed cognition—often referred to in terms of the cognitive immune system (Cohen, 2000)—can provide homeostatic behaviour to a robot.

The CAH architecture (Cognitive Adaptive Homeostasis) presented in this chapter provides adaptive homeostasis to a robot. Making use of the cognitive decision-making (CDM) system from chapter 4 to provide action selection based on the internal state. By pairing the CDM simulation with an associative memory neural network, the architecture is able to associate and recall previous experiences from the environment. This pairing of CDM with associative memory endows the robot with the ability to alter its high-level behaviour, based on its internal state and previous experience. It is able to associate contextual cues with corresponding effects on the system, and then recall these based on the current needs of the system. This allows the robot to make cognitive decisions, both internally (through the CDM simulation), and externally (through the associative memory). These cognitive decisions can be used to alter the high-level behaviour of the robot to help it to survive for an extended period of time.

This architecture is compared with a robot controller that reacts to its internal state directly, instead of relying on the CDM simulation. Comparing with this system, termed the 'purely needs-based system', will help to provide evidence in support of the following research questions:

> Can the CAH architecture...
>
> **RQ1**: ...learn from previous experiences and use them to influence future behaviour?
>
> **RQ2**: ...provide homeostatic behaviour to a robot?
>
> **RQ3**: ...balance two conflicting requirements to provide homeostatic behaviour to a robot?
>
> Does the CDM component...
>
> **RQ4**: ...change decisions more slowly at the start of the simulation?

5.2 IMPLEMENTATION AND EXPERIMENTAL SETUP

This section details the development of the adaptive homeostatic architecture, including simulations of the architecture in a simple environment. This section also presents the experimental setup and robot platform used in experiments, along with the 'needs-based' system that will be used for comparison.

5.2.1 *CAH architecture*

Cohen (2000) proposed three components of distributed cognition: self-organising behaviour, decision making, and internal images. The similarities between cognition and homeostasis have been discussed above, in section 5.1. The architecture presented in this section provides adaptive homeostasis to a robot, based on these components of cognition.

Fig. 5.1 shows the proposed CAH (Cognitive Adaptive Homeostasis) architecture. The cognitive decision-making (CDM) simulation from chapter 4 is used to decide which of the low-level signals should affect the high-level behaviour. By considering the CDM component as a black box, it can be replaced by a different component to provide different behaviour to the robot. The 'needs-based' system that is used as a control case in the experiments uses a simple threshold component in place of the CDM. This threshold component provides output signals based on, for example, the charge in the battery dropping below a certain, fixed threshold.

The CDM component is linked to the robot sensor inputs through the attractor sources in its virtual environment. By increasing the attraction towards the attractor source based on the sensor inputs, the agents can react to the sensor inputs collectively as a flock. In the test cases for this chapter, the attractor sources are linked to the charge level and temperature sensors on the robot.

Having already discussed the CDM component at length in the previous chapter (section 4.2), the rest of this section describes the behaviour of the CMM and proof-of-principle simulation. This is followed by a description of the robot platform and experimental setup.

correlation matrix memories The outputs from the CDM component are fed into a CMM (Correlation Matrix Memory). As described in section 2.3.2, a CMM is a form of memory that associates stimuli with stored responses. In the CAH architecture the CMM stores an 'internal image'—or imprint—of the environment for the robot to use. As such, the CMM offers the robot the capacity to make

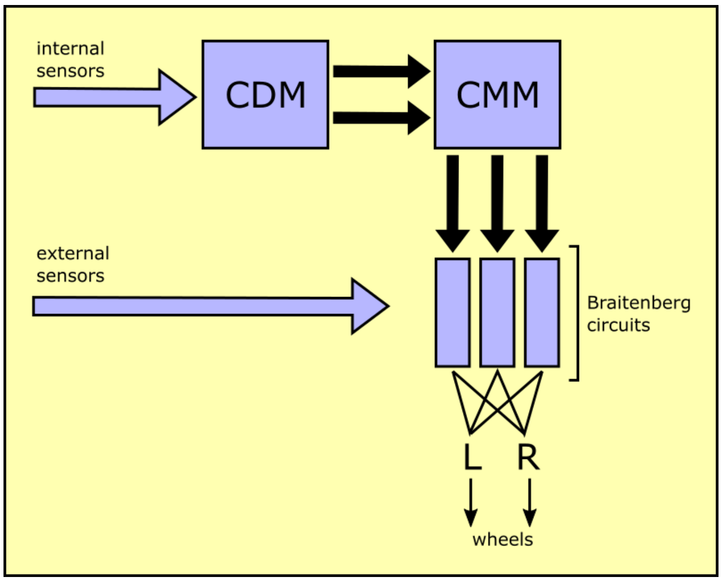


Figure 5.1: Diagram depicting the high-level architecture of the CAH system. The internal sensors (battery charge, temperature) drive the behaviour of the CDM simulation. As the agents within the CDM react to the internal sensors, we count the proportion of agents in each attractor source within the CDM, and use this to drive the binary signals (black arrow) to the CMM input. The CMM (trained to recall which colour lamp is mounted above charging stations and high-temperature areas) then recalls the colour to search for according to which input is triggered by the CDM. The output from the CMM is then fed into the light-seeking circuits that perform basic chromotaxis (algorithm described in Appendix A.5) and seek out or flee from the appropriate colour in the environment.

associations between changes in internal and external sensor values that occur at the same time.

For example, by marking a charging station with a blue light, the robot can sense the increase in blue light at the same time as an increase in battery charge. By associating these two signals, the CMM offers the ability to recall which colour to search for in order to find a charging station.

Fig. 5.2 shows how the CMM is constructed for two internal sensors (charge, c, and temperature, t) and for the three colour components from the external light sensor (red, green, and blue). When one of the sensors passes a threshold, the corresponding binary value switches from 0 to 1. If this occurs on one of the internal sensors, at the same time as one of the external sensors, then the corresponding value in the association matrix is set to 1 as well, storing this association between the two sensors. To recall the association, we can re-present one of the internal sensor inputs (e. g. 'charge'), by setting it to 1 again. This will result in the stored associations to be recalled, and the appropriate values set in the output vector. This allows the system to test which colours have been associated with the 'charge' input.

$$\begin{array}{cc|cc} \begin{pmatrix} c \\ t \end{pmatrix} & \begin{pmatrix} cr & cg & cb \\ tr & tg & tb \end{pmatrix} & \begin{pmatrix} 1 \\ 0 \end{pmatrix} & \begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix} \\ & \begin{pmatrix} r & g & b \end{pmatrix} & & \begin{pmatrix} 0 & 0 & 1 \end{pmatrix} \end{array}$$

Figure 5.2: Basic CMM architecture (left), where c and t correspond to the internal sensors for charge and temperature, and r, g, and b correspond to the red, green, and blue colour components from the light sensor, respectively. The vectors $\left(\begin{smallmatrix} c \\ t \end{smallmatrix}\right)$ and $(r\,g\,b)$ are provided as stimuli to the association matrix. Any points in the association matrix that has a 1 on both input vectors is set to 1, storing the association between the two. An example association is shown on the right, as would occur after the robot finds a charging station under a blue light.

5.2.2 *Experimental setup*

In order to answer the research questions laid out in section 5.1, a laboratory-based arena is required. This section details the experimental setup, including the robot platform that the architecture controls.

The arena is 224x108cm with 15cm high boundaries (see fig. 5.3). It has coloured LED lamps that can be moved so they point at different parts of the arena.

The robot platform is the Pi-Swarm (Hilder et al., 2014). This platform provides basic IR distance sensing and wheeled movement, along with colour LEDs on the top edge of the robot, and a simple speaker built-in. The robot is controlled using an ARM mbed LPC1768 chip (mbed, 2016) which runs custom C++ code. It has very limited flash and RAM storage, along with limited computational capacity. In order to provide information to the robot about the

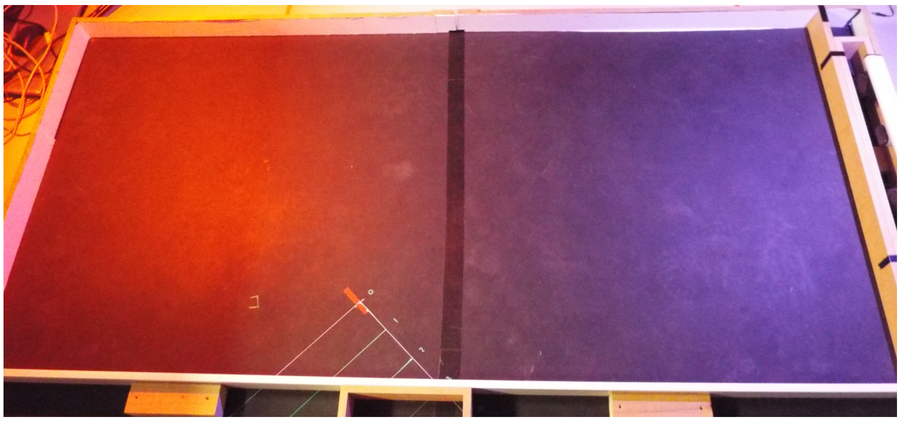


Figure 5.3: Photo showing the laboratory arena. The arena is 224x108cm with 15cm high boundaries and coloured LED lamps (out of view) that provide environmental cues to the robot.

environment, the Pi-Swarm platform has a single[1] RGB colour sensor (TCS3472) mounted on top.

LED lamps, mounted on tripods, provide a gradient of colour for the robot, using Perspex colour filters over the front in order to separate the different gradients. After initial analysis (see appendix A.4), the best colours for this laboratory setup (in terms of other lights in the surrounding area and the contrast in the sensor) is a combination of the orange and yellow-green filters, or the blue filter.

For the purposes of this work, the robot makes use of two internal sensors: battery charge and temperature. These sensors are simulated, exploiting the fact that they are always beneath a lamp. As such, by setting the threshold for 'detecting' a charging station higher than the threshold for detecting the corresponding colour, the charging station is placed at the centre of the lamp's gradient. The colour sensor returns values that are scaled according to the integration time and gain of the sensor (see (AMS, 2012) for full details). The integration time and gain used for this experimental setup are 14.4ms and 60x, respectively. The colour sensor returns values based on the irradiance of its sensor, which is dependent on a wide range of environmental factors. The values used are proportional to lux, but it makes more sense to report values in terms of the relative responsivity of the sensor. For this reason, we define the 'arbitrary colour unit' (acu) as the value returned from the TCS3472 given an integration time of 14.4ms and gain of 60.

1 Appendix A.5 gives details of how the single colour sensor detects gradients in the environment.

The high-level behaviour of the CAH architecture is given in code listing 5.1.

5.2.3 *Proof-of-principle simulation*

Before the architecture is implemented on the robot platform, the idealised behaviour of the system is tested using a simulation. This proof-of-principle simulation provides key information about how the architecture *should* behave, before the 'reality gap' is crossed (see section 2.2 and (Jakobi et al., 1995)).

The proof-of-principle simulation is shown in fig. 5.4. This simulation, constructed in NetLogo (Wilensky, 1999), has the entire CAH architecture, other than the CDM component, implemented. The signals provided by the CDM to the rest of the architecture are simulated using switches on the simulator interface. The simulation consists of a single, zero-mass agent that roams around a simple environment of different light gradients. The agent performs a random walk around the environment, until one of the 'motives' is switched on. This is a simple switch telling the agent that it has a certain need, and that the architecture should find the corresponding part of the environment. For example, 'need charge' recalls the appropriate colour from the CMM and searches for that colour in order to recharge.

Fig. 5.4 shows a trace of the agent heading towards the green light source in the environment when the 'need charge' switch is enabled (green arrow). Once this motive is removed (blue arrow), the agent leaves the light source. The simulation shows that the architecture functions as it should when appropriate signals are provided from the CDM component (which is missing in this case). This implies that any variation we see by including a CDM component (or a control component) can only be from the changes within the CDM, or noise in the environment, rather than as a result of the architecture.

5.2.4 *Training phase*

The CAH architecture adapts to the environment in which it is placed, through the use of a CMM. The CMM associates large changes in the internal sensors with large changes in external sensors. For example, if the robot discovers a charging station under a blue lamp, the CMM will associate blue light with charging. While the CAH architecture is

Listing 5.1: Pseudo-code for high-level CAH robot control code

```
// ask CMM how to behave...
behaviourVec[3] = {false,false,false};
inputs[2]       = {fleeTemp, wantCharge};
output[3]       = {0,0,0};

// perform CMM recall and threshold
output          = cmm.recall(inputs);
behaviourVec    = cmm.thresholdResults(output, false);

// translate CMM response to real-world...
seekRed         = behaviourVec[0];
seekGreen       = behaviourVec[1];
seekBlue        = behaviourVec[2];

// and again for flee behaviour
behaviourVec[0] = false;
behaviourVec[1] = false;
behaviourVec[2] = false;
behaviourVec    = cmm.thresholdResults(output, true);

fleeRed         = behaviourVec[0];
fleeGreen       = behaviourVec[1];
fleeBlue        = behaviourVec[2];

// seek/flee colour
if (seekRed || seekGreen || seekBlue ||
    fleeRed || fleeGreen || fleeBlue)
{
    if (seekRed)
        seek_colour(RED);

    if (seekGreen)
        seek_colour(GREEN);

    if (seekBlue)
        seek_colour(BLUE);

    if (fleeRed)
        flee_colour(RED);

    if (fleeGreen)
        flee_colour(GREEN);

    if (fleeBlue)
        flee_colour(BLUE);

}
else
{
    // default behaviour: random walk...
    piswarm.forward(0.1);
}

// if we have reached where we want to be, pause...
while ( (charging && wantCharge) || (cooling && fleeTemp) )
{
    piswarm.stop();
    wait(1.0f);
}
```

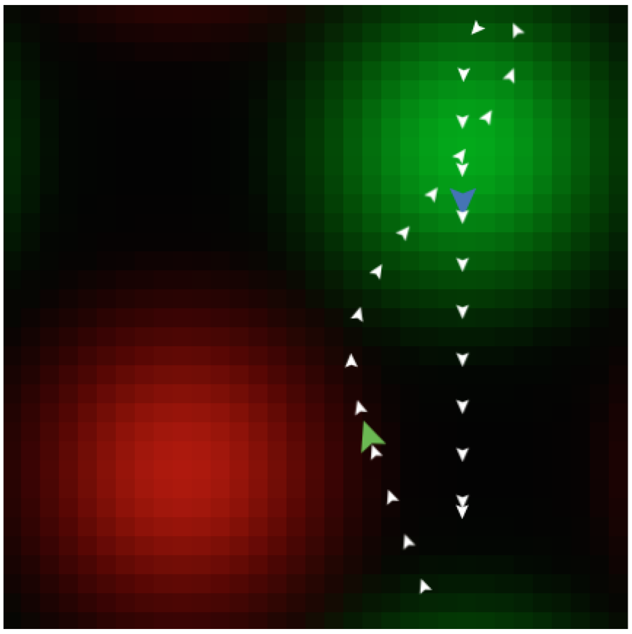


Figure 5.4: Screenshot of the proof-of-principle simulation, along with trace of the agent searching for the green light source. The white arrows are the trace of the agent position, the green and blue arrows signify when the 'needs charge' motive switch is enabled and disabled.

able to adapt on-line, this functionality is disabled in all experiments presented in this chapter. This ensures that any variation between test and control cases are only due to swapping out the CDM component. This section details how the CAH architecture is trained before the experiments described in section 5.3 below. While section 5.2.3 made use of a simulation, this section, and the results presented below in section 5.3 are gathered using a real robot.

In the training phase, the robot is trained using a random walk around the environment from an arbitrary start point. As the robot wanders, the CMM associates high values from the internal sensors with high values from the external sensors. The threshold for external (colour) sensor values is 600 acu; for charging stations, 500 acu; for high-temperature areas, 300 acu.

Due to the small size of the CMM used in this chapter (3x2 matrix), the training phase can be tested with relative ease. In the following matrices, the following key can be used to determine whether the training has been successful, where r, g, b correspond to the red, green, and blue external sensors, and c, t correspond to the charge and temperature internal sensors:

$$\begin{pmatrix} cr & cg & cb \\ tr & tg & tb \end{pmatrix}$$

After exposing the robot to an environment with one blue lamp over a charging station, the CMM is trained correctly as:

$$\begin{pmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \end{pmatrix}$$

After exposing the robot to an environment with one orange-green lamp over a high-temperature area, the CMM is trained correctly as:

$$\begin{pmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

After exposing the robot to an environment with both an orange-green lamp over a high-temperature area and a blue lamp over a charging station, the CMM is trained correctly as:

$$\begin{pmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

Finally, testing the combination of two stations in the same area, after exposing the robot to an environment with a single orange-green lamp over a charging station and high-temperature area, the CMM is trained correctly as:

$$\begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

At this stage, it is evident that the robot is able to train the CAH architecture based on its experiences in the environment. As such, the results presented in the rest of this chapter will have the training phase omitted, and the corresponding CMM matrix from the above results inserted in its place. This will reduce any possibility of variation between replicates throughout the remaining experiments.

5.3 RESULTS

This section aims to answer to research questions RQ2 and RQ3, as stated in section 5.1. The work in this section also contributes (jointly with section 5.2.4) towards RQ1.

5.3.1 *CAH architecture provides homeostatic behaviour to a robot*

This section looks at how well the CAH architecture is able to alter the high-level behaviour of a robot according to its low-level needs. As the internal state of the robot varies, the CDM signals to the rest of the CAH architecture what it needs in order to keep the state within its limits.

The internal state for the robot platform is markedly different to that of biological systems, but providing even limited information to the CDM is sufficient for it to guide the behaviour of the robot in the environment. The CDM will work to keep the internal sensors (charge and temperature, see section 5.2.2) within certain limits. If the temperature gets too high or the charge too low then the robot will fail. The values used for these variables are given in table 5.1.

	Temperature	**Charge**
`Start value`	10.0	5.0
`+ve Delta`	1.8	0.8
`-ve Delta`	0.5	0.1
`Fail Point`	55.0	0.1
`Limits`	10.0, 60.0	0.01, 15.0
`Control threshold`	N/A	2.5

Table 5.1: Parameter values used in the basic homeostasis experiment. These values have been picked to result in a challenging, but possible scenario.

This section looks at a basic scenario: asking whether the robot is able to keep itself charged while attending to another task. This other task is to warm itself under a lamp. By arranging the experimental arena (depicted in fig. 5.5) so that the charging station (a blue lamp) is far away from the warming lamp (a red lamp), the robot needs to leave the warming lamp in order to recharge, and vice-versa. This should result in the robot switching from sitting under one lamp to sitting under the other repeatedly and indefinitely. The parameters are chosen such that it is not able to warm itself sufficiently to complete its task before needing to charge again (thus cooling the robot back down again).

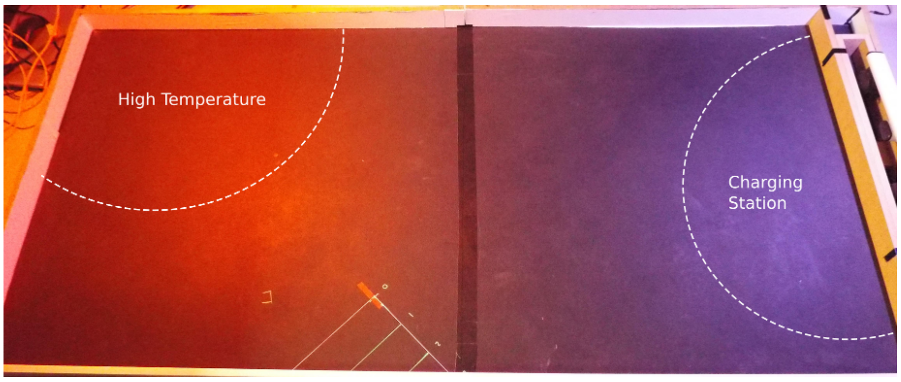


Figure 5.5: Photo of arena with charging and warm areas superimposed, arena size: 224x108cm

NULL HYPOTHESIS (H_0): A robot running the CAH architecture will survive no longer with the CDM component than with the threshold component.

The design of the architecture described in section 5.2 was such that the CDM component could be replaced by another component that provides different signals, based on the values of the internal sensors. The threshold component (described in section 5.2.2) is used in this way, and has a simple threshold mechanism. The output switches from 0 to 1 once the charge value drops below the control threshold of 2.5 (see table 5.1).

The threshold value for the control system was calculated from the amount of time it takes for the robot to get to the light source it is searching for. This was determined empirically as 45 seconds (the upper quartile of the data represented by fig. 5.6). Using this value for traversing the environment, the threshold can be calculated as 2.0. In order to take into account the variability of working in a noisy environment, the threshold was increased slightly to 2.5, allowing 54 seconds in the worst case for the robot to find its way to the light source. This also helps to reduce the chance of a false positive result by making it easier for the control setup to survive for longer.

Fig. 5.7 shows the path of one run (a) with the CDM in place. The graph in (b) shows the internal sensor data provided to the CDM component for the duration of the experimental run. The robot was

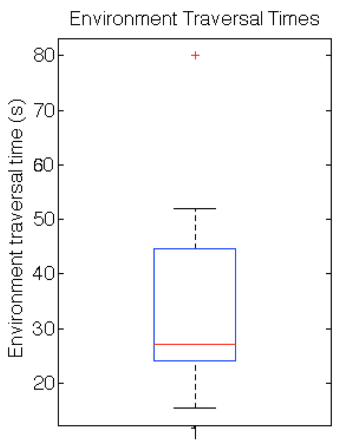


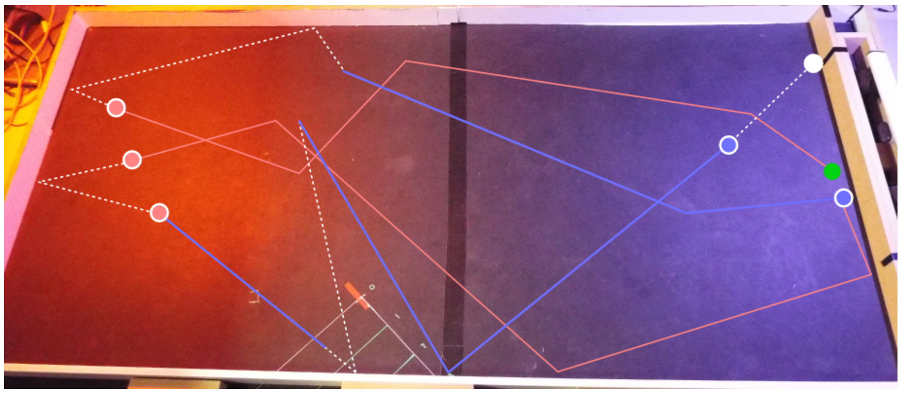
Figure 5.6: Boxplot showing how long it takes to cross the arena between light sources.

allowed to continue running until it reached 29 minutes, at which point it would be stopped.[2] The traces show that the robot is able to perform a task while preventing itself from running low on charge.

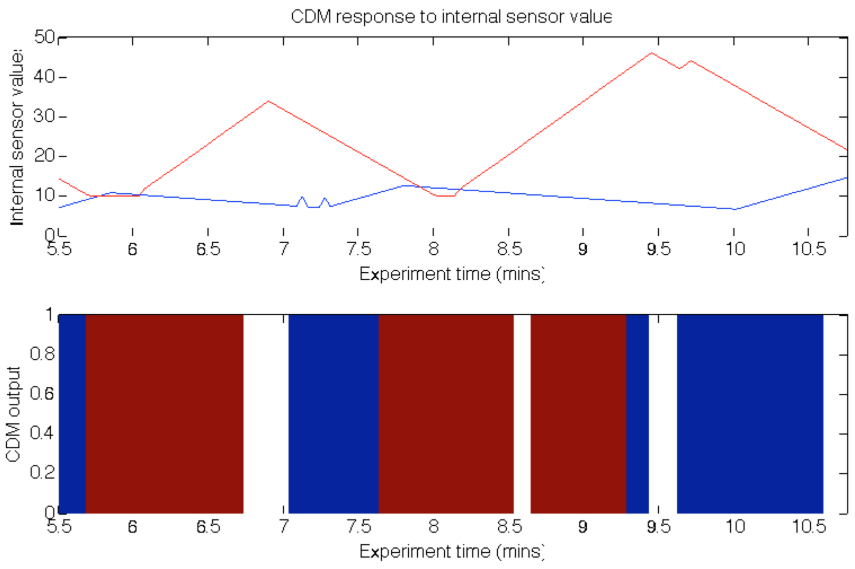
Table 5.2 shows the length of time that the robot survived when set up with the two components described above (up to the maximum of 29 minutes). This data is presented in a table format, rather than a boxplot due to the relatively few replicates in this experiment. The high variance in the CDM data is unexpected and is the basis for further experimentation in section 5.3.3. Even with only 6 replicates, it is clear that the CDM code can survive for longer than the control code. This is confirmed through the Mann–Whitney–Wilcoxon test and Vargha–Delaney A-test, giving $p < 0.05$ and $A < 0.16$ respectively. From this, H_0 can be rejected at the 95% confidence level.

While performing a relatively simple task, the results presented in this section show that the CAH architecture is a viable option for providing simple artificial homeostasis to a robot. This provides the evidence required to answer the research question RQ2. The next section discusses how well the architecture survives when faced with more complicated scenarios that involve conflicting needs.

2 29 minutes was used because this was the maximum length of video that could be recorded with the used camera, and was a sufficiently long period of time that any transient behaviour occurring at the beginning of the run had settled and was not the behaviour observed.



(a) Trace of robot path between 5.5 and 10.5 mins superimposed on the environment. The blue and red circles indicate points where the robot stopped to charge or warm itself respectively. The green and white circles indicate the start and end points. The red and blue solid lines indicate that the robot is searching for that colour. The white dotted line is when the robot wanders (not actively searching for anything).



(b) Graphs showing internal sensor values (top) and CDM outputs (bottom) for the robot between 5.5 and 10.5 mins. The red and blue line show the internal values for temperature and charge as presented to the CDM component. The red and blue blocks of colour in the bottom graph show the output of the CDM component (red indicating to the robot to search for red light, and blue indicating to search for blue light).

Figure 5.7: A short (5 min) excerpt from one of the 29-minute replicates. The trace (a) shows how the robot moves between the red and blue areas in the environment as it needs to charge. The graphs in (b) show that as the internal sensor values for charge get too low, the robot actively searches for the charging station, and once it is charged sufficiently, it returns to warming itself under the red lamp.

Replicate	CDM (mins)	Threshold (mins)
1	29	3.5
2	2.5	4
3	29	2
4	6	2.5
5	29	10
6	29	3

Table 5.2: Table showing how long the robot survived in the environment when using the CDM (test) component and the threshold (control) component. Mann–Whitney–Wilcoxon test and Vargha–Delaney A-test give $p < 0.05$ and $A < 0.16$ respectively, rejecting H_0 with 95% confidence.

5.3.2 *Conflicting decisions*

While the work above showed how the CDM system could be used to control a robot in a simple environment (two distinct light sources), with a simple task (survive as long as possible, while performing a basic task), this section addresses research question RQ3: how long the CAH-controlled robot is able to survive when faced with conflicting decisions about what to do.

In order to achieve this, the experimental setup is altered such that the charging station and the higher temperature area are provided by the same light source (see fig. 5.8). The behaviour of the CDM is inverted with respect to temperature, so that the robot now tries to avoid getting warm. Therefore, in order to charge its battery, the robot will have to endure warmer temperatures. Once the battery runs out, the robot will fail, and once the temperature reaches a set maximum, the robot will fail. This experiment once again makes use of the same architecture between test and control cases, except for the CDM component that will be replaced by the threshold component for the control case.

NULL HYPOTHESIS (H_0): A robot running CAH architecture will survive no longer than the threshold component in the presence of conflicting requirements.

The experimental setup is as shown in fig. 5.8. This is the same arena as above, except for the removal of the blue light from the right-hand side. In this more difficult scenario, the robot is expected to fail more often than it did in the previous setup, as before it was possible for the robot to sit and charge indefinitely without failing, whereas here this would result in a failure. After preliminary testing, it was evident that the parameter values used previously (as given in table. 5.1) provided a scenario that was too difficult for the robot to complete with either CDM or threshold-based results (all survival times were under 2.5 minutes). As such, the parameter values are adjusted to those shown in table 5.3, to make the robot less likely to overheat straight away, but balancing this by making it slower to charge.

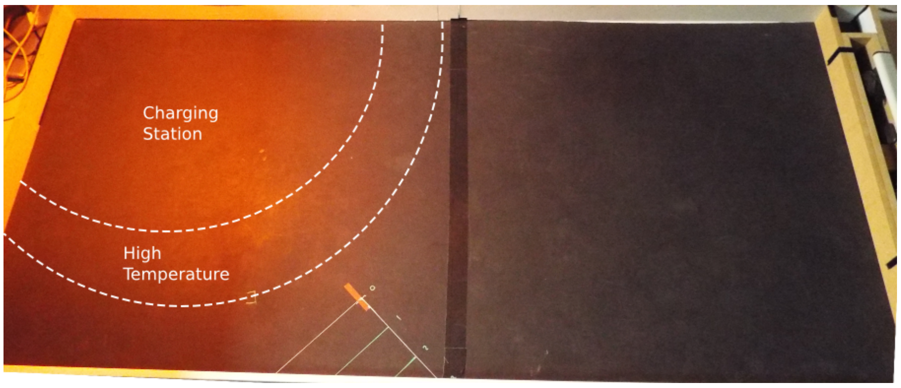
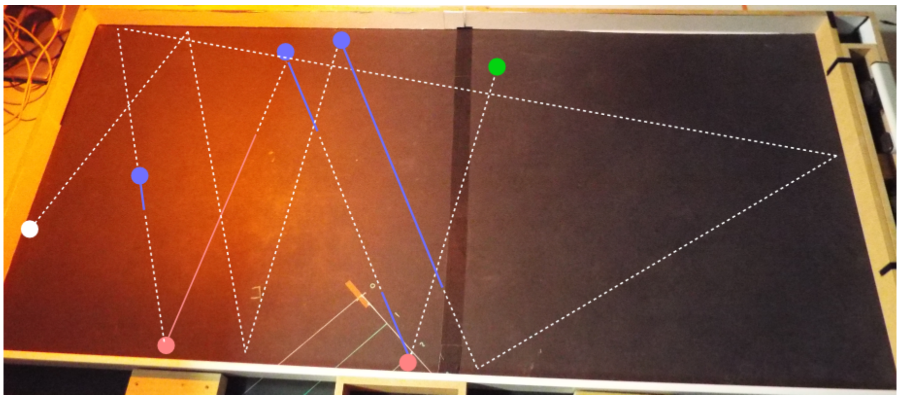


Figure 5.8: Photo showing the new arena for conflicting decisions experiment, arena size: 224x108cm

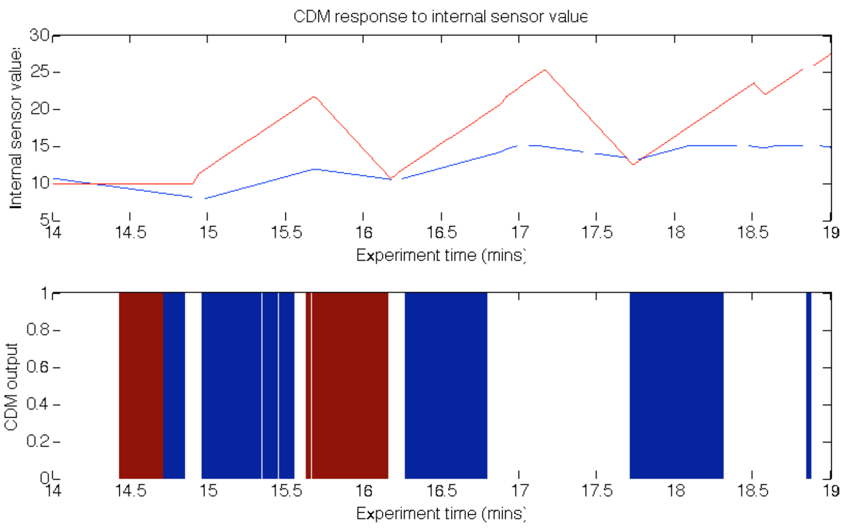
	Temperature	Charge
Start value	10.0	5.0
+ve Delta	1.3	0.3
-ve Delta	0.8	0.1
Fail Point	55.0	0.1
Limits	10.0, 60.0	0.01, 15.0
Control threshold	25.0	2.5

Table 5.3: Table showing the parameters used in the 'conflicting decisions' experiment. These values have been picked to result in a challenging, but possible scenario.

Fig. 5.9 shows a trace of the sensor values as provided to the CDM component. The graphs show how the robot balances the need to charge its battery while avoiding getting too warm. The robot with CDM component was able to balance the conflicting decisions well, failing only once in 13 replicates, whereas the control case failed 6 times out of 13. These results are presented in the boxplots in fig. 5.10.



(a) Trace of robot path between 14 and 19 mins superimposed on the environment. The blue and red circles indicate points where the robot stopped to charge or cool itself respectively. The green and white circles indicate the start and end points. The blue and red solid lines indicate that the robot is searching for, or fleeing from, the red light, respectively. The white dotted line is when the robot wanders (not actively searching for anything).



(b) Graphs showing internal sensor values (top) and CDM outputs (bottom) for the robot between 14 and 19 mins. The red and blue lines show the internal values for temperature and charge, as presented to the CDM component. The red and blue blocks of colour in the bottom graph show the output of the CDM component (red indicating to the robot to flee red light, and blue indicating to search for red light).

Figure 5.9: A short (5 min) excerpt from one of the 29-minute replicates. The trace (a) shows how the robot moves in and out of the red area in the environment as it needs to charge and cool down. The graphs in (b) show that as the internal sensor values for charge get too low, the robot actively searches for the charging station, and vice-versa for when the temperature gets too high. Note that, contrary to the behaviour described in fig. 5.7, when the temperature gets too high at around 17 mins, the CDM does not actively push the robot away from the red light, as it is already out of the area by chance. This has the effect of starting to reduce the temperature, reducing the likelihood that the CDM component would push the robot to seek out a cooler area.

The boxplots in fig. 5.10 show the amount of time that the two setups survived (up to the maximum of 29 minutes). There are 13 replicates for each. Results from the control code are presented on the left, and from the CDM (test) code on the right.

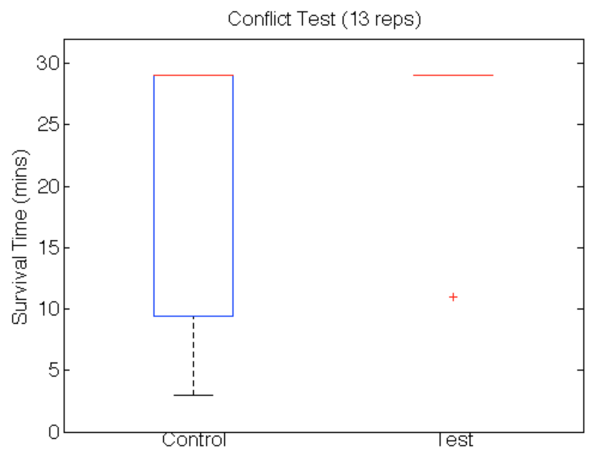


Figure 5.10: Boxplots showing the distribution of survival times from the conflicting decisions experiment. The two distributions are significantly different, with $p = 0.03$ and $A = 0.30$.

While these results were expected to be less clear-cut than the previous experiment above, analysis of the data using the same techniques as before gives similar results. The Mann–Whitney–Wilcoxon test gives $p = 0.03$, and the Vargha–Delaney effect-magnitude test gives $A = 0.30$. From this, H_0 can be rejected with 95% confidence, resulting in the conclusion that the CDM component is better at balancing conflicting decisions than the threshold component, and providing sufficient evidence to answer research question RQ3.

5.3.3 *Burn-in period*

The plot in fig. 5.11 shows a compilation of the test data from the previous two experiments. 100% of the failed replicates result in a value that is below 50% of the maximum. This suggests that, once the algorithm 'settles', it is able to run for much longer, as though there is a threshold past which the system is more stable, and so can run for an extended period of time without trouble.

Analogous to the effects seen in the Ising model (Ising, 1925), it appears that the CDM component might require a 'burn-in' period before it can reliably make decisions, possibly due to the time taken for the agents to move across the environment at the start. The lack

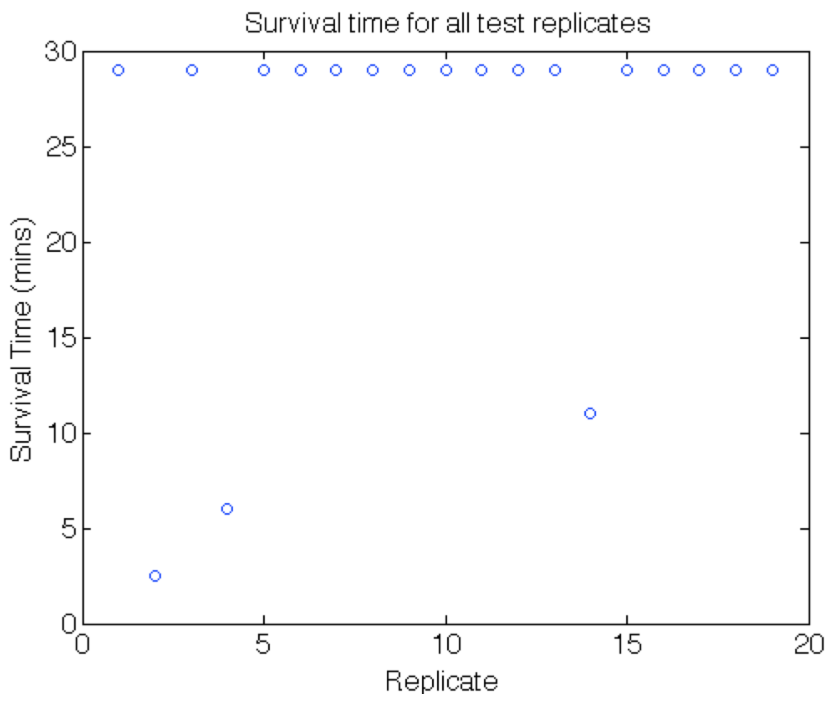


Figure 5.11: Test-case data from two previous experiments. Of those failed replicates, all are below 50%, indicating that initially 'burning-in' the simulation might help the decision-making process.

of a burn-in period can give rise to failed runs (as shown in fig. 5.11) that result from the inability of the CDM component to respond to changes in the internal environment quickly enough. The burn-in period provides the opportunity for the simulation to settle down, allowing the agents to flock towards the attractors, reducing the time required for the flock to switch between attractors when the environment changes.

NULL HYPOTHESIS (H_0): Burning-in the CDM has no effect on the speed it can change decisions.

This hypothesis is tested using simulation. A rapidly-changing signal is provided to the CDM component, and the CDM output is recorded. In order to test its ability to change decision quickly, the simulation measures how long the CDM takes to switch decision after the signal changes from increasing to decreasing, and vice-versa.

The CDM component is inserted into the architecture shown in fig. 5.12. This provides the CDM component with the number of positive and negative gradients between the preceding 5 samples from the signal. The CDM then tries to decide whether the signal will increase or decrease next. The signal is produced using a sine wave centred around 1. Two experiments are run, the first with the clean signal as described here, the second with a noisy signal that includes a small amount of pseudo-random noise ($SNR = 22$).

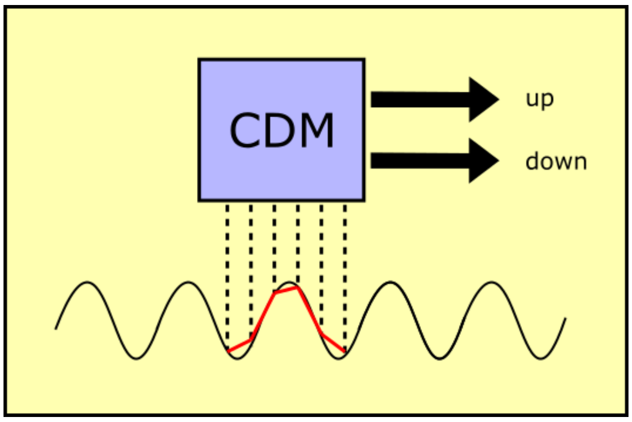


Figure 5.12: The experimental setup for testing whether burning-in the CDM component enables it to make decisions more quickly. The CDM component samples the input signal, and calculates whether there is a positive or negative gradient between each sample. These samples are then used to alter the strength of two attractors. The number of agents in each attractor is then measured and the higher value is considered the decision made ('up' or 'down', relating to which way the signal is moving). As the signal varies, the difference between the signal and the output of the CDM is measured, and is used to determine the 'speed' of the decision, presented in fig. 5.15.

As in previous chapters, aleatory uncertainty analysis provides the minimum number of replicates needed for the simulation. The results—shown in figs. 5.13 and 5.14—show that a minimum of 1000 replicates are required. This number of replicates is high, indicating that the simulation is particularly sensitive to the input signal.

Fig. 5.15 shows the distribution of results from the simulation. The first set of boxplots in fig. 5.15 show the first set of runs, with the clean signal. It is evident that there is a change in the behaviour between the two sets of runs, confirmed with the Mann–Whitney–Wilcoxon ($p = 2.8e^{-20}$) and Vargha–Delaney ($A = 0.3808$) tests. This indicates a 'medium' change between the two distributions (as per Vargha and Delaney's (2000) definition of the A test, summarised in appendix A.1.1). The second set of boxplots in fig. 5.15 show the second set of runs, with the noisy signal. It is less clear whether these two distributions are different, with analysis showing $p = 0.014$ and

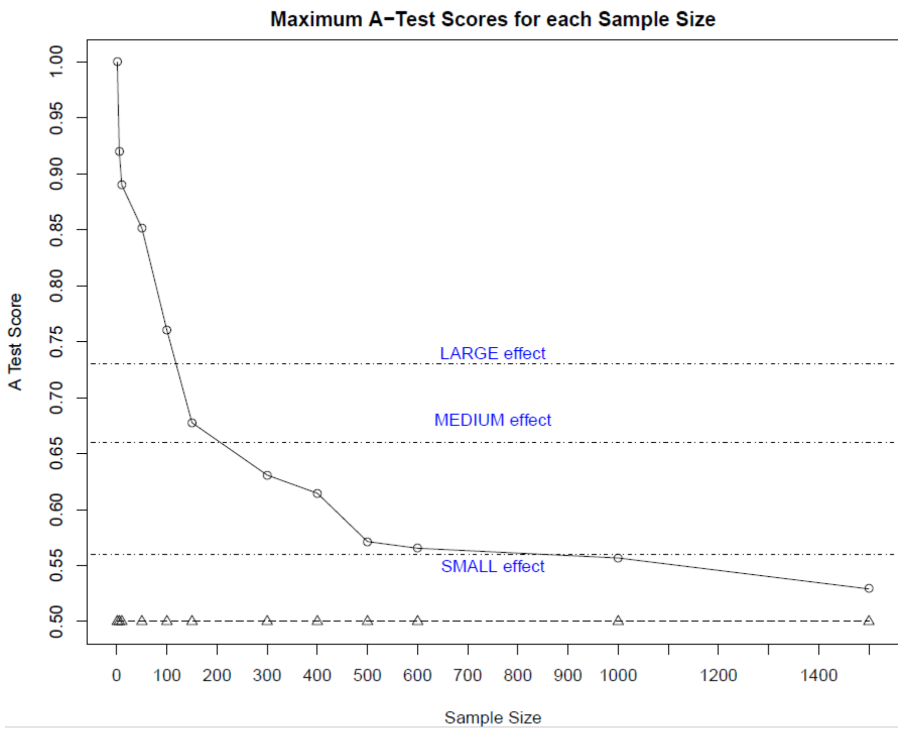


Figure 5.13: Aleatory uncertainty analysis for the burn-in simulation, with the clean signal.

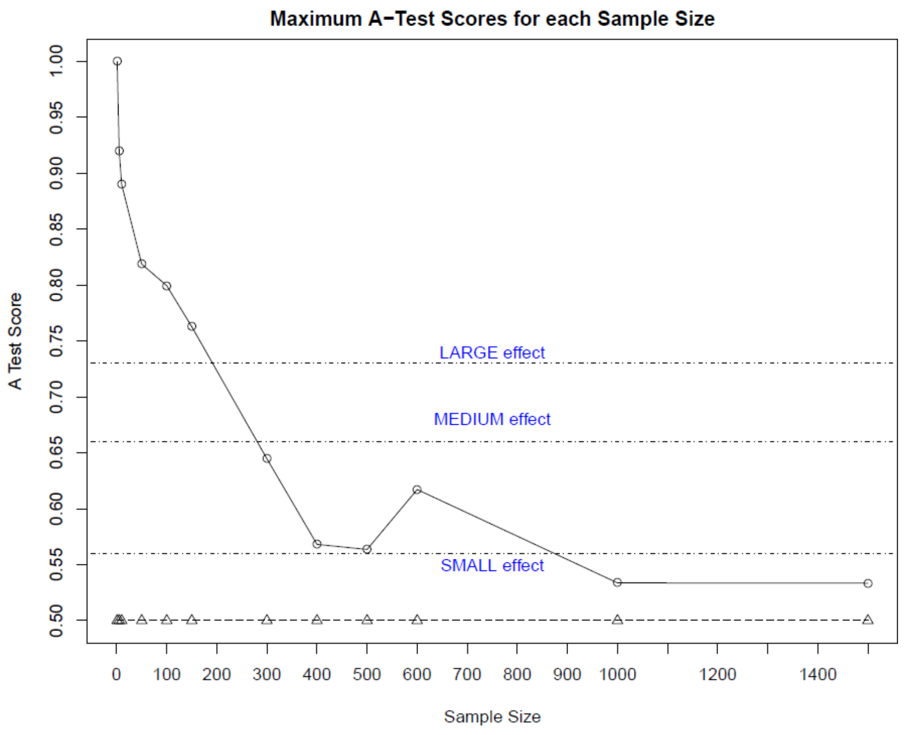


Figure 5.14: Aleatory uncertainty analysis for the burn-in simulation, with the noisy signal.

$A = 0.53$. This shows that the two distributions are not significantly different. Taking the results presented here, H_0 can be rejected with 95% confidence when the signal is clean, but not when it is noisy.

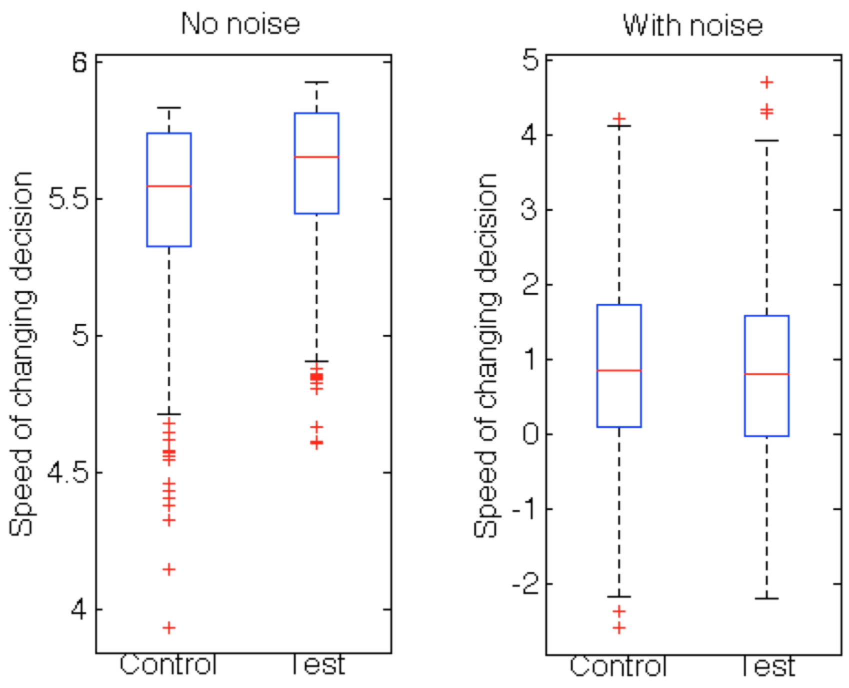


Figure 5.15: Boxplots showing the distributions of data for noiseless and noisy experiments in the burn-in experiment. The noiseless case shows a significant difference between control and test ($p = 2.8e^{-20}$, $A = 0.38$). The noisy case shows no significant difference between control and test ($p = 0.014$, $A = 0.53$). The 'speed' of a decision is calculated as the difference between the point where the signal gradient switches from positive to negative, and the point at which the CDM output switches.

The results presented show that—as expected—the burn-in period can help the CDM component make decisions at the start of the runs. This was, however, only the case when no noise is involved in the input signal. A small amount of noise is sufficient to remove any effect of burn-in. It is likely that the effect of the noise indicates why burn-in was not required to gather usable results in the previous sections. The failed runs seen in the compilation in fig. 5.11 are not described by the burn-in theory after all, and could just be a result of noise in the environment, or an artefact in the laboratory arena affecting the movement of the robot.

5.4 DISCUSSION

This chapter has presented an architecture for cognitive adaptive homeostasis, based on the cognitive decision-making algorithm presented in chapter 4. Evidence towards answering four research questions is provided in sections 5.2.2 and 5.3. These research questions are summarised below.

RQ1 (*Can the CAH architecture learn from previous experiences and use them to influence future behaviour?*) Section 5.2.2 presents the testing of the CMM training phase. This process consists of the robot performing a random walk in the environment and associating simultaneous changes in the internal and external sensors. These associated changes represent the previous experiences that the robot has learnt. The behaviour of the robot in sections 5.3.1 and 5.3.2 show that the robot is able to successfully recall these experiences and use them to influence its high-level behaviour.

RQ2 (*Can the CAH architecture provide homeostatic behaviour to a robot?*) Section 5.3.1 presents results showing that the robot is able to alter its high-level behaviour according to the value of the two internal sensors (temperature and charge). The behaviour of the robot is compared between two versions of the CAH architecture: one with the CDM component, one with a simple threshold function (see section 5.2.2). The CDM component consistently outperforms the threshold function at providing homeostatic behaviour to a robot, as shown by the survival task in fig. 5.7.

RQ3 (*Can the CAH architecture balance two conflicting requirements to provide homeostatic behaviour to a robot?*) Section 5.3.2 presents results showing that the robot is able to balance two conflicting requirements. The robot was required to withstand a high-temperature region in order to gain access to a charging station. The behaviour of the robot is again compared between two versions of the CAH architecture as with RQ2, above. The CDM component once again consistently outperforms the threshold function at providing homeostatic behaviour to a robot, while balancing two conflicting requirements.

RQ4 (*Does the CDM component change decisions more slowly at the start of the simulation?*) Section 5.3.3 presents an experiment to test whether the CDM component requires a ‘burn-in’ period before it can make successive decisions. The results show that the CDM component does change decisions slower at the start of the simulation, but that the effect is only significant if there is no noise on the input signal. Given the inherently noisy environment in which the robot operates, the results in this chapter show that the CDM component in the CAH architecture does not need the burn-in period.

While other approaches to homeostasis have directly mimicked the natural systems of the body (Neal and Timmis, 2003, 2005; Vargas et al., 2005; Stradner et al., 2009; Schmickl et al., 2011), the CAH architecture presented in this chapter is based on the ideas of distributed cognition (Cohen, 2000; Mitchell, 2005). Distributed cognition has many similarities to adaptive homeostasis, combining self-organisation, decision-making and an internal image of the environment to alter the behaviour of the system (Cohen, 2000). Combining the CDM with a CMM associative memory provides the CAH architecture with the ability to alter its high-level behaviour according to the low-level state of the robot, while adapting to previous experiences.

The CMM is trained by coincident ‘spikes’ in the internal and external sensors (e. g. when the ‘charge’ internal sensor rises at the same time as the ‘blue’ external sensor—this relates to a charging station under a blue light in the real-world environment). This provides adaptivity to the robot, as it is able to learn about new parts of an environment and adapt its behaviour accordingly (for example, if the robot were to find another charging station, it would store the association in the CMM and recall it as before). The robot is able to make decisions about its current internal state through the CDM, and transfer this to high-level behavioural changes in the environment, showing that the CAH architecture provides the capacity for homeostasis to the robot. Furthermore, the use of the CMM to recall previous experiences allows the robot to proactively search for, or flee from, specific areas in the environment. This proactivity allows for behaviour that is more realistic for real-world applications.

DISCUSSION

To conclude the work in this thesis, this chapter discusses the contribution each chapter has made, reflects on the use of terms such as cognition and autonomy, and puts the results in the context of the literature. The results presented in chapters 3, 4, and 5 have each contributed towards the aim of using cognition to provide adaptivity and homeostasis to a robotics system.

Section 6.1.1 discusses the use of the terms cognition and autonomy in the literature and how this might affect the interpretation of the work presented. Section 6.1.2 addresses the effect of using different flocking algorithms in chapters 3 and 4. Section 6.1.3 ties the cognitive decision-making formula derived in section 4.1 to the formal mathematical models of decisions from the cognitive psychology literature. Section 6.2 addresses the results in turn, placing them in the context of the thesis aim, and detailing how each chapter meets its objectives along with the contributions made. Finally, section 6.3 presents potential avenues of future research, and section 6.4 concludes the thesis.

6.1 REFLECTIONS ON THE USE OF COGNITION IN AUTONOMOUS ROBOTS

The work in this thesis is motivated by a drive to understand the fundamental basis of collective intelligence and cognition, with a view to move swarm robotics from laboratory test cases towards real-world applications. The work presented provides an incremental step towards this goal, helping to provide the foundations for swarms of robots to form identities, enabling them to work in environments containing other swarms, as well as providing the ability to learn about and adapt to placement of resources in the environment.

The evolution of animals in complex, hazardous environments gave rise to complex behaviours. The ability to make decisions, to adapt, and to learn from experience, gave some animals an advantage over other animals, allowing them to survive for longer. The work in this thesis takes inspiration from this as the basis for a system that is able to continually adapt and survive in a complex, real-world environment, unaided by human operators—an autonomous system.

If swarm robotics is to move from laboratory test cases to real-world applications, then new methods for providing autonomous behaviour to robots will need to be developed. Cognition is a key part of this, closely associated with autonomy in a number of ways. The relationship between the concepts of autonomy and cognition are discussed below, in section 6.1.1.

This thesis has focussed on two areas of collective intelligence. First, by considering the problem of how multiple swarms are able to work in the same environment, the identity algorithm was developed, allowing for the emergent formation of identity in swarms of robots through the use of a synchronisation algorithm. Second, the problem of collective decision-making without distributed sensing was addressed. The ability to make a decision repeatedly, in a timely fashion, is essential to the long-running of real robotics platforms, and distributed sensing can cause problems with this. By combining cognitive decision-making with sensor desensitisation, a robot is able to repeatedly make decisions, without leaders or distributed sensing.

The remainder of this section discusses the relationship between autonomy and cognition (section 6.1.1), addresses the compatibility of the two algorithms presented in this thesis (section 6.1.2), and finally, puts the cognitive decision-making algorithm derived in section 4.1 in the context of the literature on cognitive psychology (section 6.1.3).

6.1.1 *Autonomy and cognition*

Many researchers have previously used a top-down, reductionist approach to developing artificial homeostasis (Neal and Timmis, 2003, 2005; Vargas et al., 2005; Timmis et al., 2009). This approach, as described in section 5.1, consists of building artificial systems that mimic their biological counterparts—for example, artificial nervous systems for controlling the robot, artificial immune systems for adapting to changing environments, artificial endocrine systems to help switch between behaviours. While this approach is effective, top-down approaches can result in large architectures that are difficult to scale. The *complicated* systems used by Neal and Timmis; Vargas et al. are limited in how they can approach the *complex* problem of autonomous behaviour.

Cognition could provide the complex approach needed to approach the complex problem of autonomy. As discussed in detail in (Vernon, 2015), autonomy and cognition are two closely-linked concepts, stating that *"cognition may be an important attribute of an autonomous system."* (p. 103). Vernon continues to discuss the links between autonomy and cognition, specifically arguing that without cognition, an autonomous system would be less robust in precarious or changeable environments. The descriptions of autonomous and cognitive systems are so close that it can be difficult to distinguish the two. Why is one system described in terms of autonomy, one in terms of cognition? It seems plausible that this problem comes from the way these terms are defined. Autonomy and cognition both have notoriously nebulous definitions (Boden, 2008). Furthermore, the definitions that do exist are often inconsistent with each other. This has resulted in a large number of 'sub-forms' of autonomy, such as 'behavioural autonomy' (Froese et al., 2007) or 'energy autonomy' (Melhuish et al., 2006), which describe the areas of the system that are considered autonomous.

Cognition, as with autonomy, has many different definitions. The large number of definitions for sub-types of autonomy or cognition (in excess of 20 separate types and definitions (Schillo and Fischer, 2004; Vernon, 2015)) gives rise to the problem of how to develop an 'autonomous' or 'cognitive' system. While it might be possible to construct a system that meets the requirements for one definition, there are a large number of other definitions that contradict it. Because of this, the work presented in this thesis has been based on one specific definition. While this will inevitably not meet the requirements for all cognitive systems, this definition also has evidence in biological systems to support it (Cohen, 1992a,b; Mitchell, 2005).

Given the focus on swarm robotics, the work presented in this thesis made use of Cohen's (2000) definition of 'distributed cognition', shown below. This aligns well with the distributed nature of swarm robotics, and so was deemed to be the most appropriate definition of cognition to use: (Cohen, 2000)

> "Cognitive systems, I propose, differ strategically from other systems in the way they combine three properties:
>
> 1. They can exercise options; *decisions*.
> 2. They contain within them images of their environments; *internal images*.

3. They use experience to build and update their internal structures and images; *self-organization*." (p. 64, emphasis original)

The thesis aim focusses on the role cognition has in "adaptivity and homeostasis in autonomous robots". The system presented provides a specific form of autonomy, described by Bickhard (2000). This form of autonomous system is called a *recursive self-maintenant* system, and is based on the idea that it can employ different processes or behaviours (related to self-maintenance) depending on the environment it is in. The behaviour of the Cognitive Adaptive Homeostasis (CAH) architecture described in chapter 5 is such that it is specifically aimed at altering its behaviour to survive as long as possible in the environment. As its internal and external environments change, the behaviour it employs varies accordingly. The associative memory in the CAH architecture is able to learn about the environment through previous experiences in order to adjust its behaviour as necessary.

The process of homeostasis consists of keeping a specific variable from varying too far from a setpoint (Ashby, 1960). As such, homeostatic behaviour works very well for simple systems like thermostats. The problem with homeostatic systems is that the setpoint is either fixed, or is only varied by an external agent (e. g. a human changing the temperature on the thermostat). The mammalian body works differently, however. While there are processes that exhibit homeostatic behaviour, Sterling and Eyer (1988) suggest that the human body is more complex. The idea of our body blindly maintaining a constant internal environment is replaced by a predictive, adaptive system that tries to balance the needs of our body with what is available to it. This idea of *allostasis* (Sterling and Eyer, 1981, 1988; Sterling, 2004, 2012) can be thought of as a cognitive system overriding the behaviour of the reactive homeostatic system based on environmental pressures (Vernon, 2015).

The system presented in this thesis appears to be providing something more than just homeostasis by combining it with cognitive decision-making, and looks to be an appropriate starting point for developing artificial allostasis. Further work is needed in order to determine whether this system could meet the comprehensive definition of allostasis provided by Sterling (2004), and section 6.3.3 considers how allostasis could be used to provide more efficient control systems for robots.

6.1.2 *Flocking algorithms*

Chapters 3 and 4 both present work based on the flocking of agents. The identity algorithm presented in chapter 3 consists of linking a synchronisation algorithm with the flocking algorithm controlling the agents. In this case, the synchronisation algorithm is based on the flashing of fireflies (Tyrrell et al., 2006), and the flocking algorithm is Reynolds' boids algorithm (Reynolds, 1987). Chapter 4 presents a method of using a flock to make cognitive decisions, based on the Olfati-Saber (2006) flocking algorithm. This section looks at how the algorithms in each chapter are based on different flocking algorithms, yet the results presented are still compatible with each other.

The identity algorithm is focussed on the effect of synchronisation on the control algorithm. Provided that the synchronisation is used to influence the alignment of agents, then the specific flocking algorithm is less important. The decision-making algorithm, on the other hand, does require a specific form of flocking algorithm. The algorithm described by Olfati-Saber (2006) is based on pushing the agents towards a lattice formation, helping to prevent agents from overlapping each other. Reynolds' (1987) boids algorithm, on the other hand, has very little to prevent the agents from passing over the top of one another, or from the flock compressing in response to external pressures. Without this rigidity, the CDM algorithm would not function in the way it does. The more rigid structure provided by the quasi-lattice formation in Olfati-Saber's algorithm means that when the flock is presented with an environmental cue, that information is transmitted across the flock quickly. For example, if the flock is pushed or pulled from one side (such as by one of the attractors), the other side responds very quickly, unlike in Reynolds' (1987) boids where the flock is more likely to split or disintegrate. Without Olfati-Saber's algorithm, the CDM algorithm would not function correctly: it would not be able to decide between contextual cues and internal motive, the motive would be overridden by context every time, as the information from the contextual cues would not be incorporated into the motive.

There is a specific link between the Olfati-Saber's (2006) flocking algorithm and Reynolds' (1987) boids algorithm (see (Olfati-Saber, 2006), section 6, pp. 16–17), where the algorithm provided by Olfati-Saber is shown to "*[embody] extended forms of all three rules of Reynolds in a single equation*". This link between the two flocking algorithms shows that while all the results for the identity algorithm are presented using Reynolds' boids algorithm, the results are still applicable for the Olfati-Saber flocking algorithm used for the decision-making results in chapter 4.

If the CDM algorithm is extended to comprise of multiple flocks, then the identity algorithm could be introduced to prevent the flocks from interfering with each other during the decision-making process. For example, consider an extension to the CDM algorithm that makes use of multiple flocks and multiple attractors in a much larger environment. As one attractor diminishes (and those agents in the attractor leave), the rest of the flocks are already forming close to the other attractors. By giving these flocks the ability to form distinct identities, the agents leaving the first attractor will not interfere with or influence other decisions until later. The process of maintaining multiple swarms in the same CDM environment is analogous to the flexibility exhibited by the ant colony optimisation (ACO) algorithm (Dorigo et al., 1996), where sub-optimal paths through a graph are maintained in case the optimal route becomes blocked. This also provides a mechanism through which the problems associated with not using distributed sensing can be alleviated by distributing flocks, rather than individuals, across an environment.

6.1.3 *Cognitive decision-making*

This section discusses the question of how the cognitive decision-making algorithm described in this thesis is different from conventional decision-making algorithms in the literature. Returning to the definition of distributed cognition from (Cohen, 2000):

> "Cognitive systems, I propose, differ strategically from other systems in the way they combine three properties:
>
> 1. They can exercise options; *decisions*.
> 2. They contain within them images of their environments; *internal images*.
> 3. They use experience to build and update their internal structures and images; *self-organization*." (p. 64, emphasis original)

The first point (as is the case with a large number of definitions of cognition) is that the cognitive system must be able to make decisions.

Cohen describes the process of decision-making in a cognitive system as:

> "[A] decision emerges... from a match between an environmental case and an internal motive. Decisions are associations." (p. 69)

Section 4.1 discusses how an agent can make a cognitive decision. Specifically, it must be able to do more than simply react to the environment: it must be able to react differently to the same environmental context, based on the internal state of the agent. This is reduced to a formula for cognitive decision-making:

$$Decision = f(motive, context)$$

An alternative (but similar) perspective comes from Wang and Ruhe (2007). Wang and Ruhe define a mathematical model for the cognitive process of making decisions:

> "A *decision*, d, is a selected alternative $a \in \mathcal{A}$ from a nonempty set of alternatives $\mathcal{A}$, $\mathcal{A} \subseteq U$, based on a given set of criteria C, i. e. :
>
> $$\begin{aligned} d &= f(\mathcal{A}, C) \\ &= f : \mathcal{A} \times C \to \mathcal{A},\ \mathcal{A} \subseteq U,\ \mathcal{A} \neq \emptyset \end{aligned}$$
>
> where $\times$ represents a Cartesian product [, and U is the universal set.]" (p. 75, emphasis original)

This definition depicts how the cognitive process of making a decision is very similar to that presented in section 4.1. By considering the definition of Wang and Ruhe in the terms used throughout this thesis then the similarity becomes more evident ('context' to represent the perceived environment, and 'motive' to represent the likelihood of selecting an action in a context).

As 'context' is the perceived environment, this can be considered as the set of alternatives, $\mathcal{A}$, and motive—as the likelihood of selecting one of the alternatives—can be considered as the set of criteria, C. With these terms linked, the similarity of the two definitions becomes evident (one derived from the definition of distributed cognition in (Cohen, 2000), and one from a strict mathematical model of the cognitive processes involved in decision-making. This is an interesting congruence between two different approaches to defining decision-making, and is something that warrants further investigation.

6.2 THESIS SUMMARY AND CONTRIBUTIONS

This section summarises the contribution of each chapter in turn, considering the work in terms of the thesis aim and objectives, as provided in the introduction.

THESIS AIM

> *To investigate whether distributed cognition can be used as the basis for adaptivity and homeostasis in autonomous robots.*

OBJECTIVES

1. Determine a method of providing a distributed system with the ability to form a boundary.
2. Derive a form of collective decision-making that does not require distributed sensing.
3. Build an architecture that uses distributed cognition to provide adaptivity and homeostasis to an autonomous robot.
4. Test the ability of the autonomous robot to adapt to, and sustain itself in, previously-unseen environments.

CHAPTER 3: EMERGENT IDENTITY FORMATION This chapter details the development of an algorithm that allows a distributed system to form a boundary. Due to the nature of swarm systems, if a system is scaled up such that there are multiple swarms of agents in the same environment, those swarms will require some method of distinguishing themselves from each other. Without this self–not-self distinction, the swarms will combine into one large swarm, rather than remaining as distinct swarms.

The work presented in this chapter specifically addresses objective 1 (*Determine a method of providing a distributed system with the ability to form a boundary*). The algorithm presented allows swarms of agents to form identities, and to distinguish themselves from other swarms. By considering the effects that the identity algorithm has on local and global polarisation, along with the relative survival time for swarms, the results presented show that swarms are able to form identities, and maintain those identities over time. As multiple sub-swarms encounter each other in space, the identity that a sub-swarm has formed helps to decouple its interactions with other swarms, enabling the swarms to overlap and pass through the same areas of space without disintegrating.

The results presented in this chapter surrounding the formation of identities, and the general ideas of how distributed systems form boundaries provides the first insights into the fundamental basis of collective intelligence presented throughout this thesis. Furthermore, there are certain similarities between the development of a collective identity and the theories of autopoiesis from Varela et al. (1974), where the boundary of a 'unity' is defined from the interactions of the components within it. This is outside the scope of this thesis, but raises a number of interesting avenues of future research regarding collective intelligence and its relationship to the behaviour of living cells.

In terms of impactful applications in the real world, swarm robotics is the natural outlet. When considering real-world applications of swarm robotics, search-and-rescue applications are one of the most promising routes, due to the potential for using divide-and-conquer methods. This approach is only possible, however, if multiple swarms are able to function effectively in the same environment. The work presented in this chapter provides an incremental step towards making this possible.

CHAPTER 4: COLLECTIVE DECISION-MAKING WITHOUT DISTRIBUTED SENSING This chapter presents the algorithm for, and results from, a simulation of flock-based cognitive decision-making (CDM). One of the key aspects of the CDM algorithm is the development of leaderless directionality, based on emergent flock motive. This attribute of the CDM algorithm is key to distinguishing it from other flock-based methods in the literature. The use of a flock in our CDM algorithm has the benefit of removing the need for distributed sensing in the decision-making process. This distinguishes it from a wide range of other collective decision-making algorithms that rely on the aggregation of individuals to make a decision (Schmickl and Hamann, 2011; Garnier et al., 2005). The algorithm presented makes decisions in such a way that the system is able to repeat the process without the swarm disintegrating, allowing the system to respond to changing environments, and to make successive decisions.

The work in this chapter specifically contributes towards objective 2 (*Derive a form of collective decision-making that does not require distributed sensing*). The work presented takes the definition of a decision from (Cohen, 2000) and derives a formula ($decision = f(context, motive)$) from which the CDM algorithm is developed. Analysis of the CDM is then undertaken in a simulated environment, specifically focussing on the effect of varying the number of individuals, when presented with either one or two attractors in the environment. This analysis characterises the behaviour of the algorithm over a range of population values (between 5 and 75), providing the information required to effectively use it in a robotic control architecture.

The ability to make a decision is one of the few points that is consistent across the definitions of cognition, and it is essential to consider in any discussion of intelligence (Pfeifer and Scheier, 2001). Comparing the work in chapter 4 with existing decision-making algorithms, such as BEECLUST (Schmickl and Hamann, 2011), shows that decision-making in distributed systems can occur in many different forms. While distributed sensing is used by ant colonies, it is highly likely that flocks of birds make decisions in a similar way to the CDM. Reducing decision-making down to a formula consisting of just environmental context and internal motive allows for different methods of decision-making to be considered through the same mechanisms. The congruence between the CDM and the work of Wang and Ruhe (2007) shows that, even though the perspective of decision-making might differ, the processes are often very similar.

CHAPTER 5: COGNITIVE ADAPTIVE HOMEOSTASIS IN ROBOTS

The final results chapter details the development of an architecture for artificial homeostasis in robots, based on the cognitive decision-making algorithm described in chapter 4. The CAH (Cognitive Adaptive Homeostasis) architecture described in this chapter combines the CDM algorithm from chapter 4 with an associative memory to provide homeostasis and adaptivity to a robot. The architecture developed is able to adapt its high-level behaviour based on the decisions made by the CDM component. The CDM is set up to provide internal motive, based on the state of internal sensors in the robot. Furthermore, by providing the ability to associate the internal motive of the robot (the output of the CDM component) with the environmental context (provided by the light sources in the arena), the associative memory allows the robot to make cognitive decisions at a higher level (robot level) as well as within the CDM component (component level). These multiple layers of cognitive behaviour are characteristic of collective and group cognition (Goldstone and Gureckis, 2009; Trianni et al., 2011). This results in a system that is able to make cognitive decisions about how to survive for extended periods of time. This is indicative of something more advanced than homeostasis, as discussed in section 6.1.1.

The work presented in this chapter specifically contributes towards objectives 3 and 4. In targeting objective 3 (*Build an architecture that uses distributed cognition to provide adaptivity and homeostasis to an autonomous robot*), the architecture presented makes use of the CDM system from chapter 4 by coupling the (virtual) environment of the CDM to the internal sensors on the robot. This allows the attractors to vary as the sensor values vary. The output of this CDM is passed to the associative memory (CMM) that recalls the colour in the environment to search for. This CMM acts as the 'internal image' to the robot (Cohen, 2000), maintaining an 'imprint' of the world it has experienced over the course of the lifetime of the robot. The CMM is trained by coincident 'spikes' in the internal and external sensors (e. g. when the 'battery' internal sensor rises at the same time as the 'blue' external sensor—this relates to a charging station under a blue light in the real-world environment). This provides adaptivity to the robot, as it is able to learn about new parts of an environment and adapt its behaviour accordingly (for example, if the robot were to find another charging station, it would store the new association in the CMM and recall it as before). The robot is able to make decisions about its current internal state through the CDM, and transfer this to high-level behavioural changes in the environment. This process is indicative of the basic process of homeostasis, and so shows that this architecture provides the capacity for homeostasis in a robot.

In targeting objective 4 (*Test the ability of the autonomous robot to adapt to, and sustain itself in, previously-unseen environments*), the robot is presented with a laboratory environment, set up with either one or two light sources (filtered to only present one colour each, in order for the colour sensor to reliably distinguish them). The system is able to simulate charging stations and high-temperature areas in the environment. This is achieved by programming a small controller that increases and decreases the internal sensors based on the brightness of specific colours in the environment. This environment is then used to test how well the architecture is able to adapt to, and sustain itself in an environment, having not seen it before. The CMM provides the ability to learn about, and adapt to, the unseen environment, and the homeostatic behaviour provides the ability to sustain itself over extended periods of time, making use of the internal 'imprint' of the environment, as learnt by the CMM.

6.2.1 *Summary of contributions*

- A synchronisation algorithm can be used to form emergent identities in swarms, allowing multiple swarms to operate in the same area.
- Cognitive decision-making provides the ability to make multiple successive decisions without distributed sensing or leaders within the swarm.
- Distributed cognition can be used to provide homeostatic and adaptive behaviour to a robot.

6.3 FUTURE WORK

Over the course of this thesis, a number of avenues for potential future work have been identified. This section discusses these in relation to the work already presented. Three areas of future work are presented, relating to: cognition within a swarm, in section 6.3.1; swarm-level distributed memory, in section 6.3.2; and, allostatic load and stress hormones, in section 6.3.3.

6.3.1 *Swarm cognition*

The first area of future work consists of extending the cognitive architecture from single robots to a swarm of robots. Developing a 'cognitive swarm' is the next logical step in considering collective intelligence, and how it relates to moving swarm robotics towards real-world applications. Providing the swarm with the ability to react to each other, and make decisions as a swarm as well as individually, opens up new possibilities to the swarm system.

The CAH architecture presented in chapter 5 provides a robot with the ability to make decisions based on the internal motive of the robot, and the environmental context in which it is situated. By connecting robots together through the CAH architecture, we are able to provide the motive of each robot to its neighbours' CMM. Connecting these robots together in this way provides each robot with the ability to make decisions based on the motives of all the robots around it, along with the environmental context. This opens up the possibility of one group of robots deciding to act in one way, and another group (within the same swarm) deciding to act in a different way. When faced with this conflicting decision at a swarm-level, the result could be the swarm splitting into two sub-swarms. Using the identity algorithm alongside the CAH architecture would then help to keep these two sub-swarms separate once they move apart.

Real-world applications, such as search-and-rescue operations and environment mapping, can be approached in a more efficient way if a 'divide-and-conquer' approach is taken. By reacting to environmental information during the course of a search-and-rescue operation, a swarm of robots is able to make a decision about when to split to cover more area. This could be as simple as detecting something through a chemical sensor, or from finding a natural split in the path (such as a Y-junction). If there is no obvious consensus among the robots on which direction to take, then splitting into two groups makes the most sense. Due to the nature of swarm robots (they react to the presence of other robots), getting a swarm to split reliably is surprisingly difficult. The CAH architecture allows the robots to make associations between high-level behaviour and the motives of the individual and the swarm. This provides the basic building blocks to develop a system that can make decisions about when to split. The identity algorithm is able to help the two groups of robots ignore signals from those robots in the other sub-swarm, helping it to split cleanly and reliably.

6.3.2 *Endogenous fault detection through a swarm-level distributed memory*

Section 6.3.1 presented a way of using the motive from other robots to influence the decisions of the swarm, essentially connecting the CMMs in each robot together. This section proposes an approach to the problem of fault detection in a swarm by implementing a swarm-level CMM, distributed across each robot. While these two architectures are distinct, the ideas are related because the swarm-level memory could be built on top of the previous CAH architecture, rather than replacing it.

Fault detection is a vital aspect of swarm robotics research. Contrary to early ideas that a large number of robots would provide inherent robustness, swarms and swarm algorithms are surprisingly fragile (Bjerknes et al., 2007; Bjerknes and Winfield, 2013). Recent research in fault detection has aimed to circumvent this by making use of an external robot tracking system (Millard et al., 2014), or by helping to recharge the failed robot (Timmis et al., 2016). The tracking system provides individual robots with information about whether the swarm is behaving correctly, but would require a robot tracking system in place for all applications—not something that is possible in real-world applications. By using a distributed swarm-level memory in place of this tracking system, the robots are able to determine what is happening at the higher level, provided they can communicate with other members of the swarm. Further to this, through the ability of the CMM to generalise inputs (see section 2.3.2), if a robot fails, the rest of the swarm is still able to recall the information that was on the failed robot. The CMM allows this missing information in the input pattern to be generalised, just like noise in a signal, and the stored pattern is still retrievable.

Due to the hazardous nature of real-world applications, fault detection is likely to be one of the most important topics in the move from laboratory test cases to real-world applications. The distributed, swarm-level memory proposed here provides the basis for endogenous fault detection, that ultimately could be used for a wide range of real-world problems, while providing insight into the nature of organisational memory (Walsh and Ungson, 1991).

6.3.3 *Allostatic load and stress hormones*

The third avenue for future work addresses the idea of allostasis, as discussed above in section 6.1.1. Allostasis is commonly associated with 'allostatic load', a buzzword for the pathophysiology of chronic stress. The mammalian body has methods for effectively handling acute threats, such as the presence of a predator, in the form of the 'fight-or-flight' mechanism. This is implemented using a wide range of different internal mechanisms, largely centred around stress hormones such as adrenaline, noradrenaline, and cortisol. By implementing a similar system in a robotic platform, it might be possible to use stress hormones to alter the behaviour of a robot to survive for longer. For example, implementing a fight-or-flight mechanism into the robot would allow it to alter its internal state rapidly, according to the presence of certain environmental cues. This could include switching from passive to active sensing or increasing its maximum speed and turning speed—essentially switching from a power-saving mode to a high-power mode, in order to effectively handle the situation presented to it.

6.4 concluding remarks

The distinction between biomimesis and bioinspiration is subtle, but important. Both biomimesis and bioinspiration consist of exploiting a natural solution to the problem we are trying to solve. The difference comes from how the solution provided by nature is used. Biomimesis consists of imitating (mimicking) the solution directly, such as the tiny hooks on Velcro tape that mimic the structure of hooks found on burrs (De Mestral, 1961). Bioinspiration, however, consists of taking the essence of the natural solution and engineering an alternative that works better for our purposes. Rajeshwar (2012) discusses this in terms of early attempts at flight: mimicking the flight mechanisms of birds (i. e. flapping wings) produces too little lift for a human to reliably fly; by reducing this to the essence of what is needed (more lift), we are able to construct planes with fixed wings that produce sufficient lift and allow us to fly.

Vullev (2011) shows the difference between biomimesis and bioinspiration with the image reproduced in fig. 6.1. The two molecular structures provide similar solutions to the same problem, but one has been engineered, the other is mimicking nature directly. The biomimetic structure has evolved over millions of years, whereas the bioinspired structure has been engineered based on how the structure of the biological system works. As such, without biomimesis the bioinspired solution would not exist, as it took the initial mimicry in order to determine how the system worked, to subsequently improve upon it. This is the approach taken in this thesis. While many previous researchers have mimicked natural systems (neural, endocrine, immune, etc.) to achieve the behaviour required, the work presented in this thesis attempts to take a step back, and use the essence of what is going on (basic cognitive behaviour) in order to achieve something similar with a simpler system. This follows the path described in (Vullev, 2011), which suggests that the move to bioinspiration from biomimesis *"represents the... nature of technological development"* (p. 507).

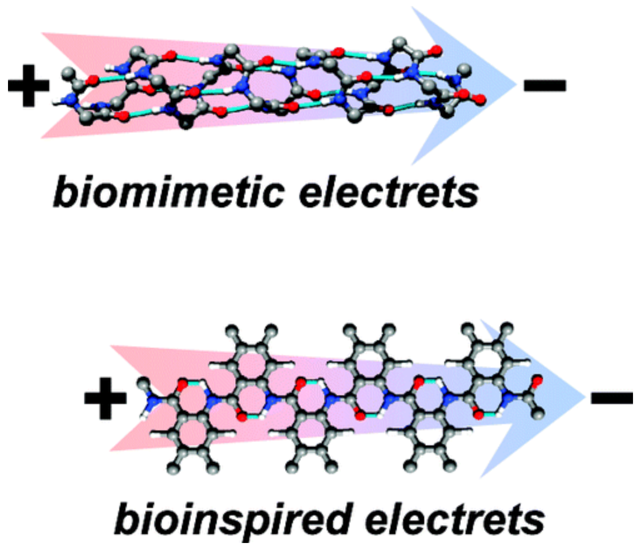


Figure 6.1: A simple example of the difference between biomimetic and bioinspired approaches to a problem. The biomimetic electrets are a result of natural selection over millions of years, whereas the bioinspired electrets are engineered for a specific problem, using inspiration from the natural system. Figure from (Vullev, 2011).

In conclusion, the bioinspired system presented in this thesis provides insight into the fundamental basis of collective intelligence, and represents an incremental step in the move from laboratory test cases to real-world applications of swarm robotics. By making use of distributed cognition, compared with previous reductionist (biomimetic) systems, adaptive homeostatic behaviour can be provided to individual robots, and the use of CMMs in the architecture provide the basis for moving to swarm-based systems (see section 6.3). While only an incremental step, the work presented shows that the use of cognition in robotic systems can be effective. Its close relationship with autonomy makes it a topic that will provide a rich source of interesting problems in the future.

MATERIALS AND METHODS

This appendix details the statistical techniques, simulation platforms, simulation analysis, laboratory setup, and light sensor readings made use of throughout the work in this thesis. Section A.1 describes the Vargha–Delaney A-test and the process of testing equivalence through the Kolmogorov–Smirnov test. Section A.2 describes the simulation platform, both the NetLogo and C++ versions, and describes the process of ensuring the two simulations produce consistent behaviour. Section A.3 describes the simulation robustness analysis and parameter sensitivity analysis used throughout the thesis to ensure the simulated results are being interpreted correctly. Section A.4 describes the laboratory setup and filters required for the LED lamps, along with light sensor readings from the robot sensor. Finally, section A.5 describes the chromotaxis algorithm used by the robot in chapter 5 to detect a gradient from a single sensor.

A.1 STATISTICAL METHODS

This section describes the statistical methods used throughout this thesis. Some statistical analysis is performed using conventional methods such as the Mann–Whitney–Wilcoxon test (Mann and Whitney, 1947), but the less well-known Vargha–Delaney A-test, and equivalence testing in the Kolmogorov–Smirnov test are explained here in sections A.1.1 and A.1.2 respectively.

A.1.1 *Vargha–Delaney A-test*

The nature of computer simulations, especially the ease of rapidly increasing the number of replicates, means that artificially producing small p-values is a possibility. As such, conventional statistical tests such as the Mann–Whitney–Wilcoxon U-test and Student's *t*-test are less useful in comparing simulation outputs. This problem comes down to the nature of these tests, as described below.

The Student's t-test tests the null hypothesis that the means of two populations are equal (Student, 1908). In doing so, the test acts on a population of samples from the real distribution. As the number of replicates increases, the amount of evidence the test has increases. Increasing the amount of evidence allows much smaller variations in the data to be detected. Without a way of measuring the effect this has (i. e. how big is the change in means between the two populations), a small p-value has little use. Pairing the p-value with an effect-magnitude test provides a mechanism for achieving this.

The Vargha–Delaney A-test is such an effect-magnitude test (Vargha and Delaney, 2000). Running the A-test on two distributions, $\mathcal{A}$ and $\mathcal{B}$, returns the probability that a randomly-selected sample $a \in \mathcal{A}$ is larger than a randomly-selected sample $b \in \mathcal{B}$. As such, the A-test can determine how dissimilar two distributions are and, unlike the t-test, this value cannot be manipulated by increasing the number of replicates. Typical values for the A-test are provided in table A.1.

Effect size	**A-test**
No effect ($\mathcal{A} = \mathcal{B}$)	0.5
'Small' effect	≥ 0.56
'Medium' effect	≥ 0.66
'Large' effect	≥ 0.73

Table A.1: The effect size and typical values for the Vargha–Delaney A-test.

A.1.2 *Equivalence testing in KS test*

The Kolmogorov–Smirnov (KS) test (Massey, 1951) is used in chapter 4 to compare distributions of highly-clustered data. The nature of the experiments in sections 4.3.1 and 4.3.2 is such that the distribution of data is clustered at the minimum and maximum measured values (0% and 100%). Due to this highly-clustered nature, switching comparatively few data points from one cluster to the other has a disproportionately-large effect on the median. As a consequence, conventional statistical methods such as the Mann–Whitney–Wilcoxon (Mann and Whitney, 1947) that operate on the differences of medians will not work with this data.

The Kolmogorov–Smirnov (KS) test operates on the cumulative distribution function (CDF), rather than the probability distribution function (PDF). Consequently, it is less susceptible to the problems associated with highly-clustered data. The test consists of measuring the maximum difference between two empirical CDFs (eCDF)—see fig. A.1 for an example.

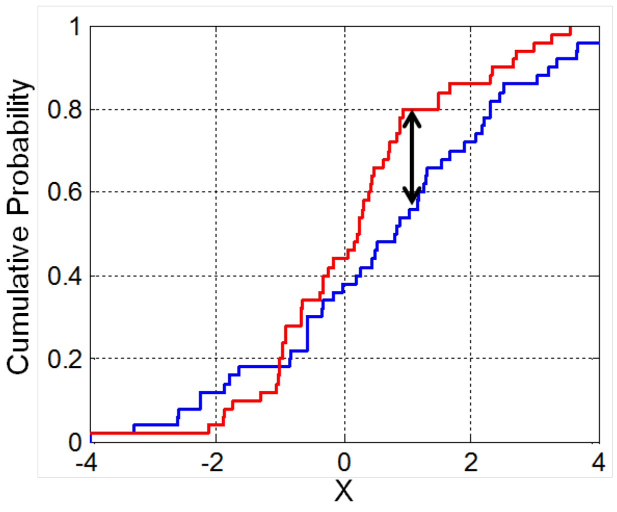


Figure A.1: Two-sample Kolmogorov–Smirnov example. The KS statistic (D^+) is shown by the black arrow between two eCDFs shown in blue and red. Image courtesy of Wikipedia user 'Bscan', distributed under CC0 1.0 license.

Testing for equivalence using the KS test statistic is performed as follows. The null hypothesis, H_0, is given that at some point t, the eCDFs F_X and F_Y differ by some amount greater than or equal to Δ, hence:

$$H_0 \ : \ |F_X(t) - F_Y(t)| \geq \Delta \tag{A.1}$$

As such, we can use the KS test to calculate D^+, hence:

$$H_0 \ : \ D^+ \geq \Delta \equiv |max(F_Y - F_X)| \geq \Delta \tag{A.2}$$

where $\Delta = 0.1601$ (follows from Theorem 3.5 in (Gibbons and Chakraborti, 2011), with $\alpha = 0.05$). Once we have D^+ it is trivial to determine whether to reject H_0.

A.2 SIMULATION DETAILS

The work presented in this thesis makes use of simulation more often than real-world robots. The use of simulation drastically increases the speed with which algorithms can be developed, even after the reality gap is accounted for (Jakobi et al., 1995).

The simulations described in sections 3.2, 4.2, and 5.2.3 are all based on the NetLogo simulation platform (Wilensky, 1999). NetLogo is too large and CPU-intensive to run on the robot directly, so the CDM component (described in section 4.2) is implemented in C++ for the CAH control architecture described in section 5.2.1.

This section describes the two simulation platforms, and details the analysis undertaken to ensure the behaviour between the C++ simulation and NetLogo simulation were consistent.

NETLOGO The NetLogo simulation platform (Wilensky, 1999) consists of a customisable environment, and one or more massless agents. The agents are able to interact with the environment around them, and with other agents. The environment is divided up according to a coordinate system. Each cell is termed a 'patch' of the environment. Each patch stores information about its area in the environment (for example, the concentration of chemicals, the temperature, or how bright the light is, at that point in the environment). This information can be queried (and if necessary, edited) by the agents. NetLogo provides functions for pseudo-random placing of agents across the environment, and the order with which the agents execute the control code is also stochastic.

C++ The basic NetLogo simulation platform is too large and CPU-intensive to be able to run on the Pi-Swarm robot platform (Hilder et al., 2014). In order to implement the CDM simulation on the robot for the CAH architecture, a stripped-down version of the NetLogo simulation environment is implemented in C++. In order to reduce the memory footprint, the static environment is removed, and all environmental variables are calculated when they are required, rather than storing each variable for each patch.

SIMULATION CONSISTENCY After developing the C++ simulation, it needs to be tested against the original NetLogo simulation. This process is essential because of the nature of swarm algorithms, where a minute change at a low level can have drastically-large effects at a higher level. For example, during the process of testing the new C++ system, the behaviour of the agents was qualitatively different from that of the NetLogo system. This was a result of a minor implementation difference in the way the agents interact with the environment. In the NetLogo system, when an agent looks ahead by one patch, it will round the coordinates of the patch to the nearest value. In the C++ system, this same operation always rounded down to an integer. While both are correct, this minor change caused large changes in the behaviour of the swarm.

To ensure the C++ simulation used on the robot correctly produces behaviour consistent with the NetLogo simulation, the equivalence testing method described in section A.1.2 is used. Fig. A.2 shows the result of running the experiment from section 4.3.1 on both simulation platforms. The third plot in this figure shows the KS test statistic, along with the A-test values when comparing the two sets of distributions. For equivalence, the KS test statistic needs to be below 0.1601, and the A-test to be below 0.56, as described in section A.1.2 above. Both of these conditions are met.

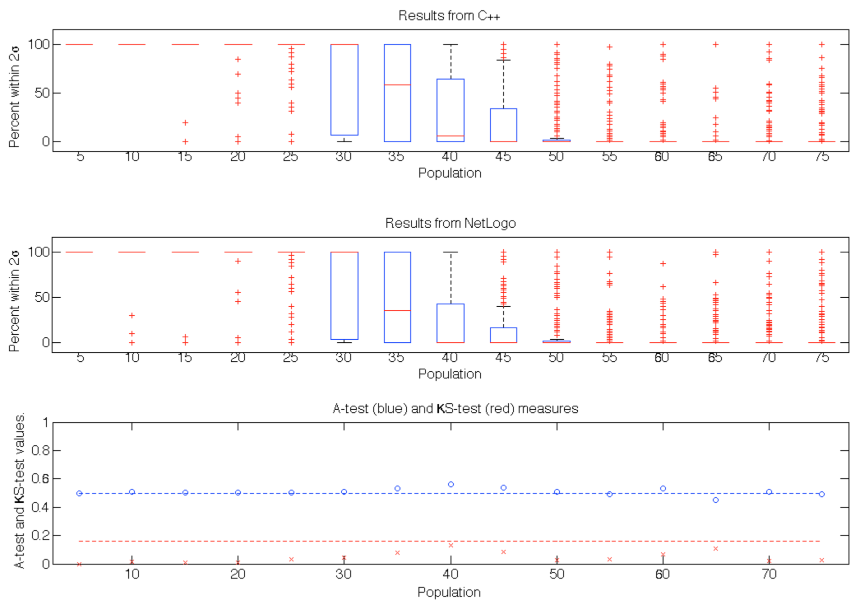


Figure A.2: Results from the simulation consistency experiment. By running the experiment from section 4.3.1 on both C++ and NetLogo simulation platforms, we are able to see any variation between the two platforms. Each population value is run for 200 replicates, and for equivalence, we require $KS < 0.1601$ and $A < 0.56$, as described in section A.1.2, above. The results show that the two simulations exhibit equivalent behaviour.

A.3 SIMULATION ANALYSIS

Stochastic agent-based simulations, like those used in chapters 3 and 4, require a minimum number of replicates to be sure that any effects observed are a result of varying the parameters, and not an artefact of the inherent stochasticity within the simulation (Read et al., 2012). Furthermore, when varying the parameters of a simulation, it is important to understand the amplitude of the effect that change will make to the simulation, as shown by the minor change between simulations in section A.2 above.

This section describes aleatory uncertainty and parameter sensitivity analysis, including any limitations that the analysis might have.

A.3.1 *Aleatory uncertainty analysis*

Aleatory uncertainty (AU) analysis provides a mechanism through which the minimum number of replicates for stochastic, agent-based simulations can be ascertained. While section A.1.1 describes the problems associated with too many replicates, the natural complement to this is what the minimum number of replicates is for any given simulation. AU analysis tests the simulation with increasing numbers of replicates until the variation between replicates from the inherent stochasticity is too small to have an effect on the results of the experiments. At this stage, any changes observed in the simulation must be a consequence of varying a parameter, rather than the stochastic nature of the simulation.

Example `spartan` output for AU analysis is shown in fig. A.3. As the number of replicates increases, the variation reduces until the variation is low enough to not affect the results of an experiment (i. e. when the line drops beneath the 'small effect' line, see table A.1).

A.3.2 *Parameter sensitivity analysis*

Parameter sensitivity analysis provides a method of quantifying the effect each parameter has on the behaviour of the simulation. The behaviour of a simulation will be more sensitive to certain parameters than to others. By holding each parameter at their baseline values, and varying each in turn, the analysis provides an accurate measure of how quickly the behaviour changes, and hence how sensitive the simulation is to that parameter. This method is called 'one-at-a-time' analysis (Read et al., 2012; Alden et al., 2013).

Example `spartan` output for parameter sensitivity analysis is shown in fig. A.4. This is an example graph taken from the CDM simulation analysis in chapter 4. In this particular example, the

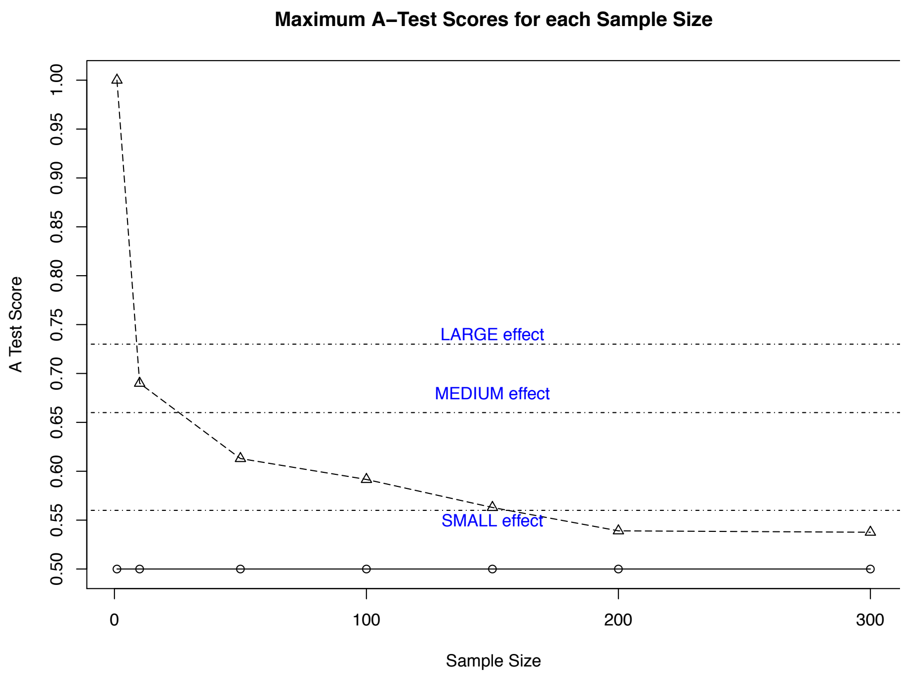


Figure A.3: Example AU analysis output from `spartan`. As the number of replicates is increased (Sample Size, x-axis), the variation of behaviour decreases, as measured by the A-Test. This particular graph shows that the simulation requires a minimum of 200 replicates to be sure that any effects witnessed are a result of changing a parameter.

`population` parameter is varied, and the effect on the number of agents that remain in the attractor after a set period of time is measured. `spartan` produces a graph like this to help visualise the effect of changing this parameter. It is clear from this example that varying the **`population`** parameter by 10–15 agents is enough for a 'large' difference in the A-test score, which is sufficient to quantify the sensitivity of this particular metric to the **`population`** parameter. It is worth noting that this will vary according to the metric used.

A.4 LIGHT SENSOR READINGS

This section details how the colour filters were chosen for the experimental setup, including finding the distances that the robot needs to move in order to detect the light gradient for the filters.

FILTER COMPARISONS Fig. A.5 shows the readings from the RGB sensor when stationary under a series of different filters. The

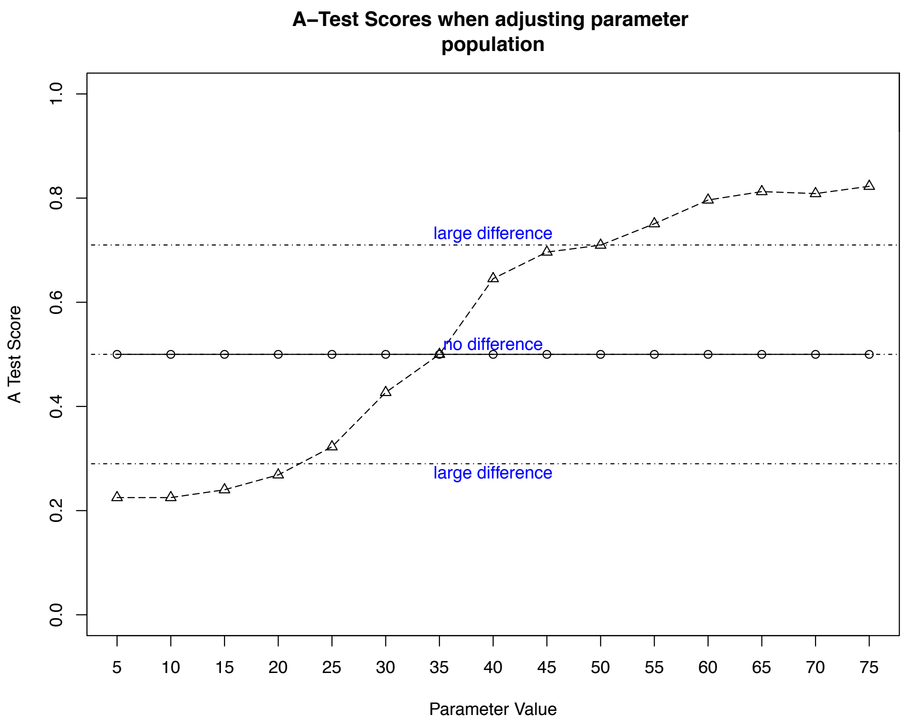


Figure A.4: Example one-at-a-time analysis output from `spartan`. This particular example is varying the **`population`** parameter (i. e. the number of agents in the simulation). As the number of agents increases, the effect on the metric varies. In this case, the metric is the number of agents in the attractor from the CDM simulation in chapter 4. (N. B. While the A-test score increases as the population value increases, the number of agents in the attractor actually decreases. This is a result of the A-test score being symmetrical around 0.5, and has no effect on the analysis.)

combination of two filters (yellow-green and orange) was best when the robot has to detect the gradient.

While fig. A.5 shows the RGB readings for a series of filters when the robot is stationary, fig. A.6 shows the readings for different filters when the robot is moving. Fig. A.6a shows the light values for a red filter. Almost all the blue and green has been filtered out (a few values drop below zero after removing the background light). Comparing this with fig. A.6b, which shows the light values when using a combination of yellow-green and orange filters, we see that the absolute values for the combination of filters is much higher (red values in the region of 500–1500, compared with 100–500 in the pure red filter). This combination of filters was used for this reason, as it lets a much larger amount of light through. For those experiments that require a second colour in the environment, the

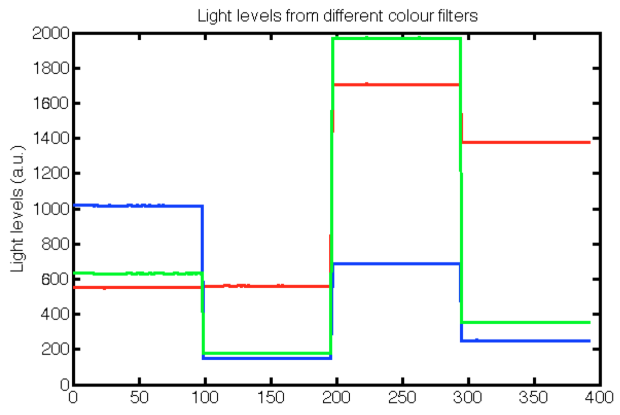


Figure A.5: Red, green, and blue light levels as recorded by a stationary robot under a series of different colour filters. Filter colours (l-r): blue, red, yellow-green, orange.

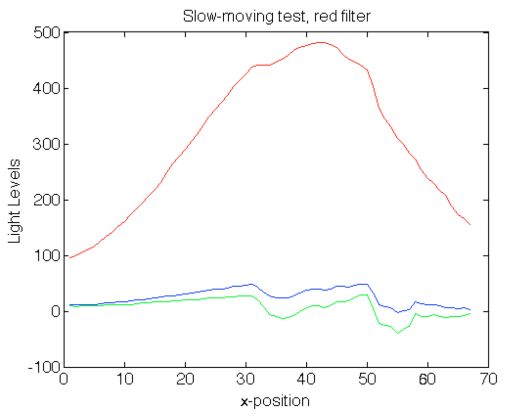
blue filter (fig. A.6c) lets through as much red light as the red filter, indicating that the robot would struggle to distinguish between the two light sources.

While these graphs show how easily the sensor can distinguish between colours while under bright light, we are interested in how easily it can perform this task when further away from the light source. Without this information, the limits of the system are not certain, in terms of locating light sources in the environment. Fig. A.7b shows the measurements taken during similar tests at 0cm (as above), 50cm, and 100cm perpendicular from the source (see fig. A.7a for a diagram depicting the experimental setup).

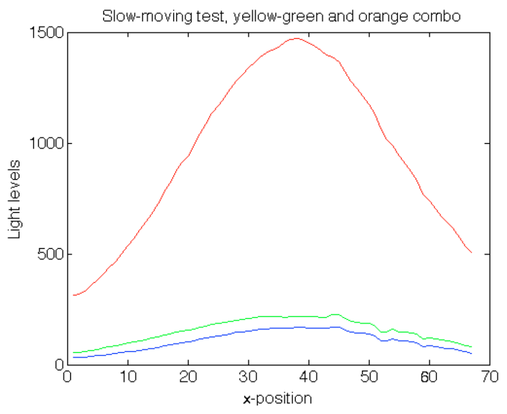
Fig. A.8 shows a heatmap of the sensor readings over a $1m^2$ arena. This was built up by taking between 55 and 60 measurements[1] across a 1m strip at 10cm intervals from the light source (see fig. A.8b for a diagram showing the experimental setup).

Fig. A.8a shows the heatmap corresponding to the baseline measurements of red light across the environment. The baseline measurements are taken with no light sources switched on, to characterise the background light in the environment. This also serves as a method for validating the method for taking measurements, by comparing the heatmap (fig. A.8a) with a photo of the setup (fig. A.8b), and observing that the heatmap correctly characterises the shadow in the top-left corner, along with the progressive decrease in light towards

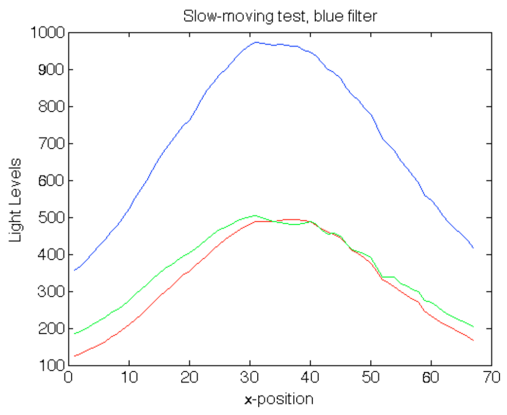
1 we set the robot up to take measurements every 0.25sec, but the robot would occasionally take slightly longer to traverse the environment, resulting in the variation in measurements we see here. The number of measurements were truncated in order to form a matrix containing all the results.



(a) Light sensor responses for red filter.



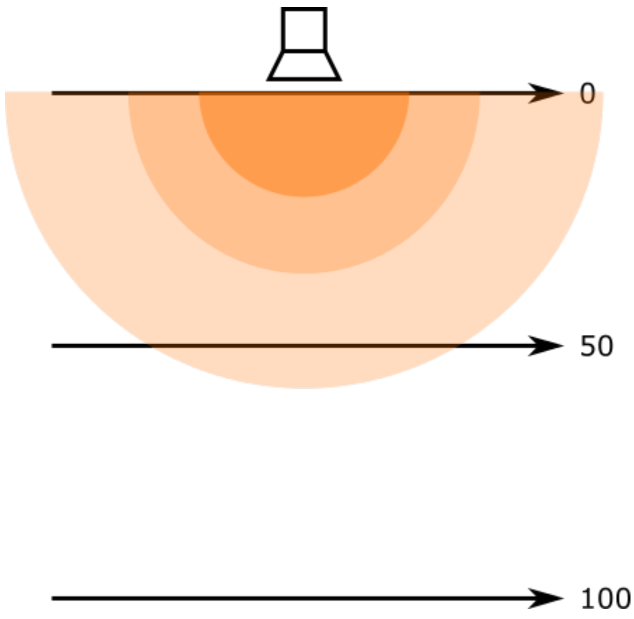
(b) Light sensor responses for yellow-green and orange filters.



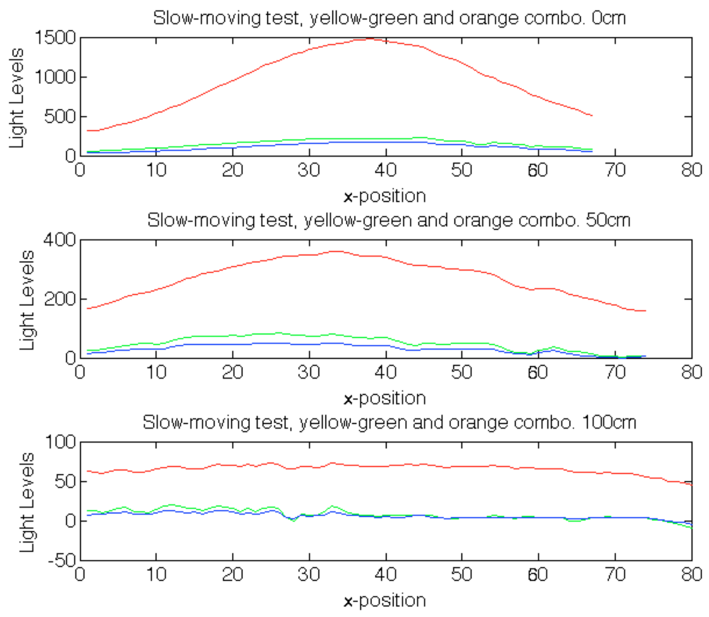
(c) Light sensor responses for blue filter.

Figure A.6: Light sensor responses for a slow-moving robot under different-coloured filters. The robot traverses the environment in a straight line across the 0cm line shown in fig. A.7a.

the bottom-right corner, resulting from the increase in distance from the workshop lights in the lab.



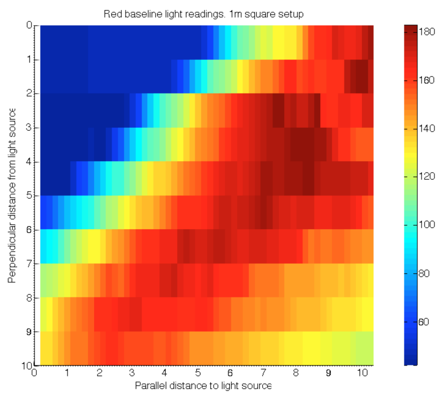
(a) Diagram showing experimental setup for distance tests.



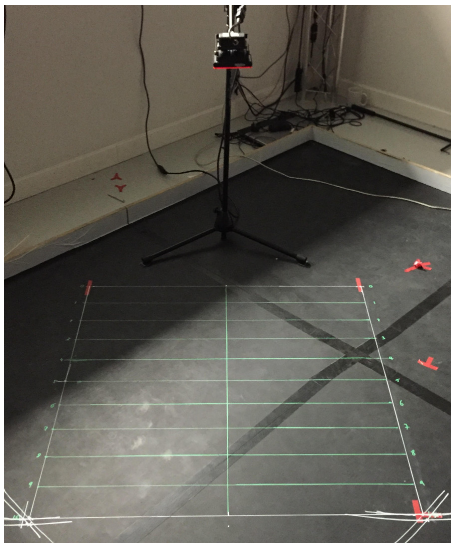
(b) Slow-moving tests at 0cm, 50cm, and 100cm. As the robot traverses the environment at increasing perpendicular distances, the prominence of the red light decreases compared with the blue and green.

Figure A.7

Following the same process as for fig. A.8a, fig. A.9 shows how the robot perceives the $1m^2$ arena, and fig. A.10 shows the effect that the light source has on the environment (i. e. the difference between the raw sensor data in fig. A.9 and the baseline data in fig. A.8a).



(a) Heatmap showing the concentration of red light as measured by the robotic platform in a series of baseline measurements.



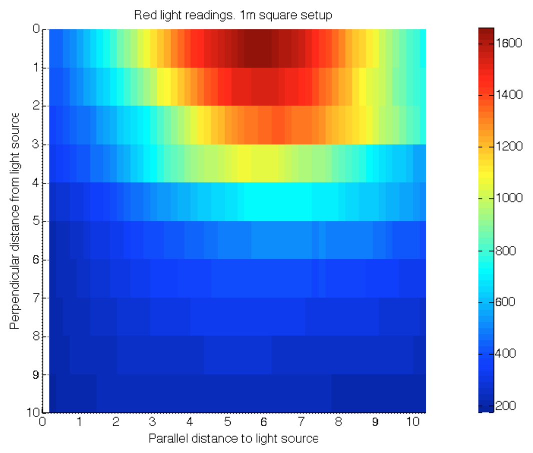
(b) Photo showing the experimental setup, confirming the decrease in light in (a) due to shadows in the environment.

Figure A.8

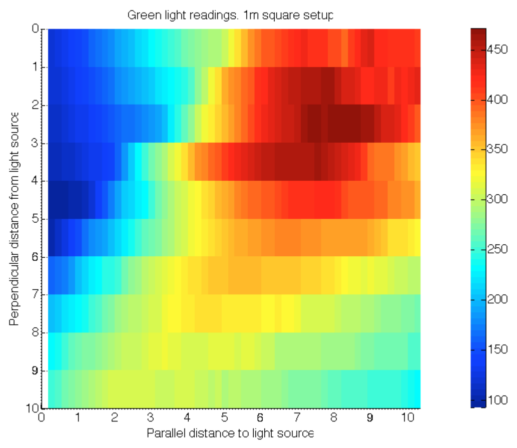
GRADIENT DETECTION The gradient test shows us how easily the robot can detect changes in the gradient of light. By setting the robot up a set distance from the light source and having it steadily move adjacent to the light source at a constant speed, the resolution of the system can be determined, in terms of detecting small changes in light. This information can then be used to determine the minimum distance that the robot would have to travel before resampling the environment. These values directly influence the behaviour of the algorithm described in appendix A.5.

The robot follows the 50cm grid-line shown in fig. A.8b, which is 50cm from the light source at its closest. By picking this line, the ability of the robot to detect changes in light when further from the light source can be calculated.

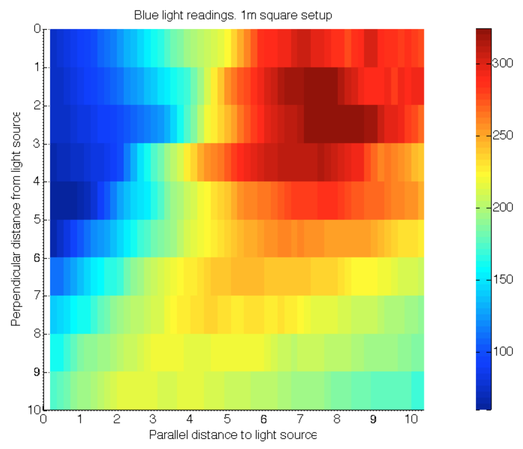
Based on the values from the blue filter in fig. A.11, the following is calculated: the robot moved $50cm$ in 33 timesteps, giving $1.5152cm$ per timestep. The change in light per cm can therefore be calculated as $3.68acu\,cm^{-1}$, meaning that to detect a colour change of 50, the robot needs to move approximately $14cm$. Similar results follow for the yellow-green filter.



(a) Heatmap showing the concentration of red light as measured by the robotic platform when the light source is enabled with yellow-green and orange filters.

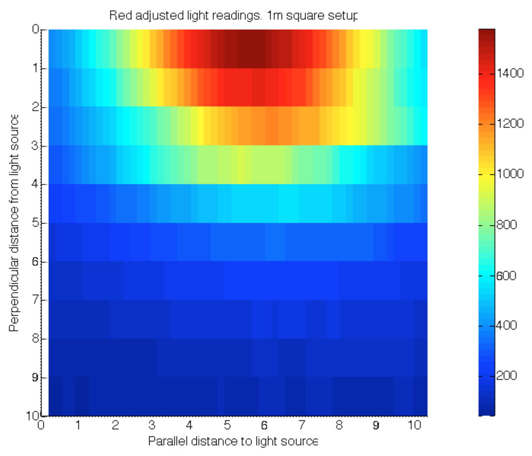


(b) Heatmap showing the concentration of green light as measured by the robotic platform when the light source is enabled with yellow-green and orange filters.

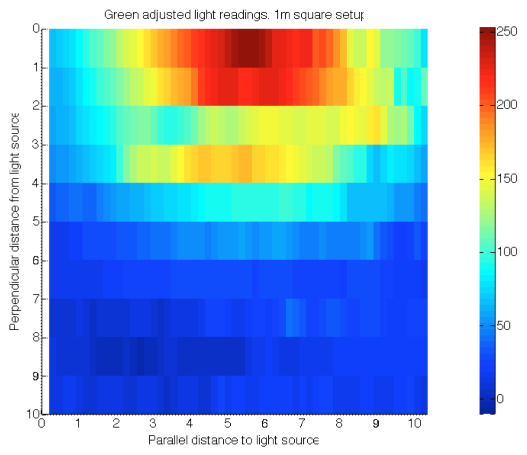


(c) Heatmap showing the concentration of blue light as measured by the robotic platform when the light source is enabled with yellow-green and orange filters.

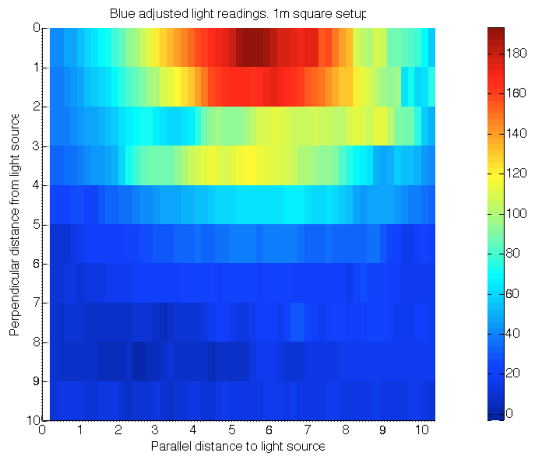
Figure A.9: These values are those as actually measured by the robot (including any background light), and so are an accurate representation of what the robot will be reacting to.



(a) Heatmap showing the concentration of red light as provided by the light source with yellow-green and orange filters.

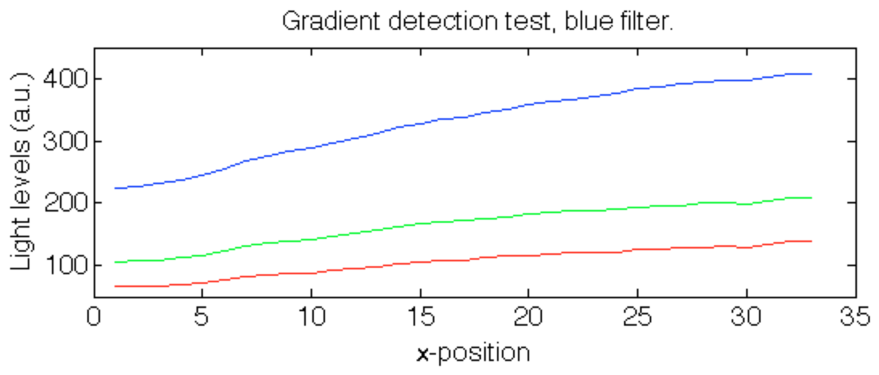


(b) Heatmap showing the concentration of green light as provided by the light source with yellow-green and orange filters.

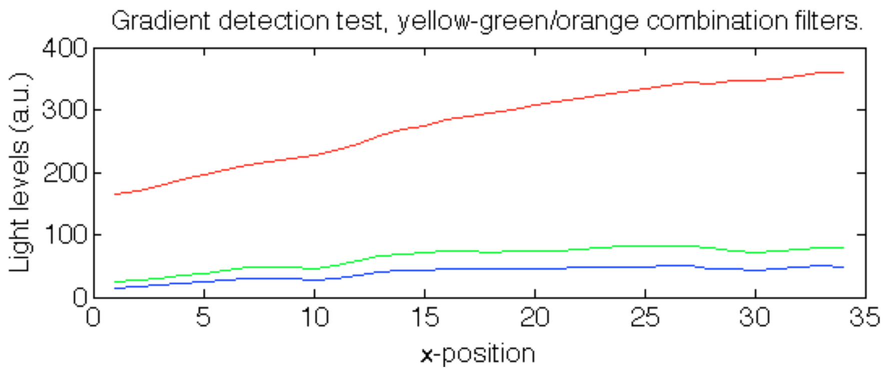


(c) Heatmap showing the concentration of blue light as provided by the light source with yellow-green and orange filters.

Figure A.10: These values are those as measured by the robot, after being adjusted for the background light, to show how the light sources affect the environment.



(a) Blue filter.



(b) Yellow-green and orange combination filters.

Figure A.11: Colour sensor readings from a robot travelling along the 50cm line (see fig. A.7a). Data used as the basis for calculating the minimum distance to travel in order to detect a given change in light.

A.5 CHROMOTAXIS FROM A SINGLE SENSOR

The architecture described and demonstrated in simulation in section 5.2 relies on a control architecture that uses a light-seeking circuit similar to Braitenberg's (1986) vehicles 2a and 2b ("fear" and "aggression"). This very simple circuit drives the robot towards the source of light that it is currently trying to find (or drives it away when it is trying to avoid light).

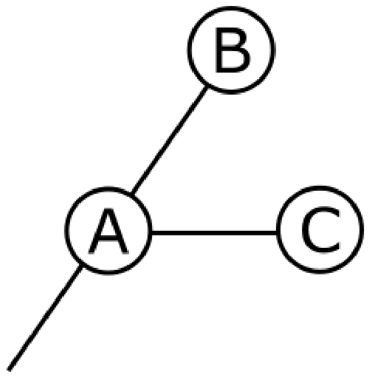
The robotic platform—as described above in section 5.2.2—only has one colour sensor on it. This poses a problem, as the Braitenberg vehicles require two sensors to be mounted on them in order to detect a gradient in the light. To solve this problem, an algorithm is developed for the robot to search its way up (respectively, down) the light gradient. Fig. A.12 and code listing A.1 show how this algorithm works. The robot moves to the three points, $\mathcal{A}$, $\mathcal{B}$, and $\mathcal{C}$, to take measurements. It then calculates the highest (lowest) of the measurements taken, and continues in the appropriate direction. If the highest (lowest) of the measurements is $\mathcal{A}$, it returns to $\mathcal{A}$ and performs a 180° turn. The effect this algorithm has when repeatedly applied is shown in fig. A.12b. In order to prevent the robot getting stuck, the algorithm alternates whether the robot moves to the left or to the right for $\mathcal{C}$, and keeps $\mathcal{B}$ in line with its current direction.

While this algorithm is used to seek out a light source in the environment (i. e. phototaxis), for the purposes of the experiments in chapter 5, the ability to seek out a specific-coloured lamp in the environment is requird (i. e. chromotaxis). The algorithm described is sufficient to perform chromotaxis in theory (and in simulation), but in reality there are imperfect filters on the lamps, which will, for example, let some red light through a blue filter. Fig. A.13 shows the colour that is detected by the colour sensor beneath each lamp in the environment.

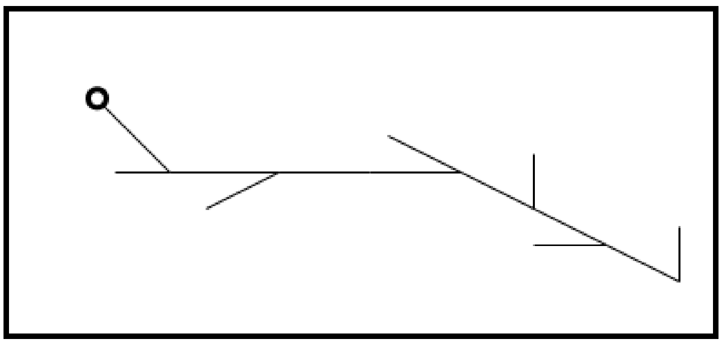
The problem with imperfect filters is that when the robot is sat under the red light, and wants to go to the blue light, every direction it heads in will have less blue light than where it currently is, as it will have less light overall. As such, the robot will be stuck under the red lamp because it has no way of telling the difference between the two lamps. To deal with this, in multi-lamp setups the colour-measurement code is altered to return the difference between the colour it is searching for and the other colour in the environment.

Listing A.1: Pseudo-code algorithm for light-seeking behaviour

```
// take measurements
move_to(B);
b = measure();

move_to(A);
a = measure();

move_to(C);
c = measure();

// compare measurements to find highest value
if (c > a && c > b)
{
    // already at C, just return
    return;
}
else
{
    // currently at C, move back to A
    move_to(A);
    if (a > b && a > c)
    {
        // at A, facing wrong way
        turn(180);
        return;
    }
    else
    {
        // currently at A, move to B
        move_to(B);
        return;
    }
}
```



(a) Light-seeking algorithm.

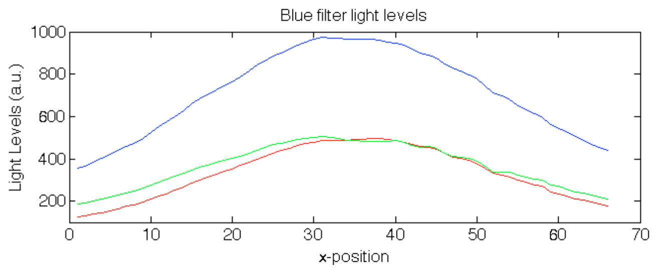


(b) Expected path of robot climbing a light gradient.

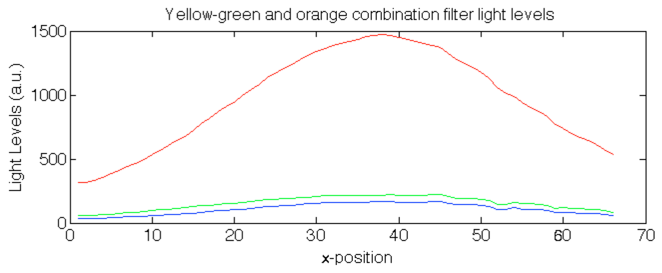
Figure A.12: (a) The basic chromotaxis algorithm, and (b) the longer-term behaviour of the algorithm as it climbs a gradient.

This will return a weaker signal for the colour it is searching for, but in return the robot is less likely to get stuck under the wrong lamp.

This change should have no effect of the validity of the reported results. The experiments described are testing the idea that the CAH architecture can perform certain tasks. The robot in these experiments is making use of coloured lamps but in any real-world situation the robot would have a more advanced platform, capable of localisation and mapping (Smith et al., 1988; Durrant-Whyte and Bailey, 2006) and could make use of an on-board camera that is able to store more information about the environment, rather than just relying on a simple colour gradient.



(a) Light levels provided by the light source with blue filter fitted.



(b) Light levels provided by the light source with yellow-green and orange filters fitted.

Figure A.13: Graphs showing the difference in light levels between the two colour filters used in the experiments. The blue filter (a) has a prominent blue component (as expected), but also has a significant portion of red and green. The combination of filters in (b) reduces this problem for the red component.

BIBLIOGRAPHY

Alden, K; Read, M; Timmis, J; Andrews, P. S; Veiga-Fernandes, H, and Coles, M, 2013. `spartan`: A comprehensive tool for understanding uncertainty in simulations of biological systems. *PLoS Comput Biol*, 9(2):e1002916.

Alden, K; Timmis, J, and Coles, M. Easing parameter sensitivity analysis of NetLogo simulations using `spartan`. In *ALIFE 14: The Fourteenth Conference on the Synthesis and Simulation of Living Systems*, volume 14, pages 3–5, 2014.

AMS, . *TCS3472 'Color Light To Digital Converter with IR Filter'*. ams, August 2012.

Anderson, P. W, 1972. More is different. *Science*, 177(4047):393–396.

Andrews, P. S; Polack, F. A. C; Sampson, A. T; Stepney, S, and Timmis, J. The CoSMoS process version 0.1: A process for the modelling and simulation of complex systems. Technical Report YCS-2010-453, Department of Computer Science, University of York, March 2010. URL `http://www.cs.york.ac.uk/ftpdir/reports/2010/YCS/453/YCS-2010-453.pdf`.

Ashby, W. *Design for a brain: The origin of adaptive behaviour*. Chapman and Hall, London, 1960.

Austin, J and Stonham, T. J, November 1987. Distributed associative memory for use in scene analysis. *Image Vision Comput.*, 5(4):251–260.

Beni, G. From swarm intelligence to swarm robotics. In Şahin, E and Spears, W. M, editors, *Swarm Robotics: SAB 2004 International Workshop, Santa Monica, CA, USA, July 17, 2004, Revised Selected Papers*, pages 1–9. Springer Berlin Heidelberg, 2005.

Beni, G and Wang, J. Swarm intelligence in cellular robotic systems. In *Robots and Biological Systems: Towards a New Bionics?*, volume 102 of *NATO ASI Series*, pages 703–712. Springer Berlin Heidelberg, 1993.

Bickhard, M. H, 2000. Autonomy, function, and representation. *Artificial Intelligence, Special Issue on Communication and Cognition*, 17(3-4):111–131.

Bjerknes, J and Winfield, A. On fault tolerance and scalability of swarm robotic systems. In Martinoli, A; Mondada, F; Correll, N; Mermoud, G; Egerstedt, M; Hsieh, M. A; Parker, L. E, and Støy, K, editors, *Distributed Autonomous Robotic Systems*, volume 83 of *Springer Tracts in Advanced Robotics*, pages 431–444. Springer Berlin Heidelberg, 2013.

Bjerknes, J; Winfield, A, and Melhuish, C. An analysis of emergent taxis in a wireless connected swarm of mobile robots. In *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*, pages 45–52, 2007.

Boden, M. A, 2008. Autonomy: what is it? *BioSystems*, 91(2):305–308.

Braitenberg, V. *Vehicles: experiments in synthetic psychology*. Bradford Books. University Press Group Limited, 1986.

Burton, J. L and Franks, N. R, 1985. The foraging ecology of the army ant *eciton rapax*: an ergonomic enigma? *Ecological Entomology*, 10(2): 131–141.

Camazine, S; Visscher, P. K; Finley, J, and Vetter, R. S, 1999. House-hunting by honey bee swarms: collective decisions and individual behaviors. *Insectes Sociaux*, 46:348–360.

Camazine, S; Deneubourg, J. L; Franks, N; Sneyd, J; Theraulaz, G, and Bonabeau, E. *Self-organization in biological systems*. Princeton University Press, 2002.

Chisholm, R. *Person and object: a metaphysical study*, volume 5 of *Metaphysics : in 17 volumes*. 2004.

Cohen, I. R, 1992a. The cognitive paradigm and the immunological homunculus. *Immunology Today*, 13(12):490–494.

Cohen, I. R, 1992b. The cognitive principle challenges clonal selection. *Immunology Today*, 13(11):441–444.

Cohen, I. R. *Tending Adam's garden: evolving the cognitive immune self*. Academic Press, 2000.

Couzin, I. D, 2009. Collective cognition in animal groups. *Trends in Cognitive Sciences*, 13(1):36–43.

Couzin, I. D; Krause, J; James, R; Ruxton, G. D, and Franks, N. R, 2002. Collective memory and spatial sorting in animal groups. *Journal of Theoretical Biology*, 218(1):1–11.

Couzin, I. D; Krause, J; Franks, N. R, and Levin, S. A, Feb 2005. Effective leadership and decision-making in animal groups on the move. *Nature*, 433(7025):513–516.

De Mestral, G. Separable fastening device, Nov 1961. US Patent 3,009,235.

Deneubourg, J. L; Goss, S; Franks, N, and Pasteels, J. M, 1989. The blind leading the blind: modeling chemically mediated army ant raid patterns. *Journal of Insect Behavior*, 2:719–725.

Deneubourg, J. L; Aron, S; Goss, S, and Pasteels, J. M, 1990. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3(2):159–168.

Diwold, K; Schaerf, T. M; Myerscough, M. R; Middendorf, M, and Beekman, M, 2011. Deciding on the wing: in-flight decision making and search space sampling in the red dwarf honeybee *Apis florea*. *Swarm Intelligence*, 5(2):121–141.

Dorigo, M and Gambardella, L, 1997. Ant Colony System: a cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on*, 1(1):53–66.

Dorigo, M; Maniezzo, V, and Colorni, A, Feb 1996. Ant system: optimization by a colony of cooperating agents. *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, 26(1):29–41.

Dorigo, M; Floreano, D; Gambardella, L. M; Mondada, F; Nolfi, S; Baaboura, T; Birattari, M; Bonani, M; Brambilla, M; Brutschy, A; Burnier, D; Campo, A; Christensen, A. L; Decugniere, A; Caro, G. D; Ducatelle, F; Ferrante, E; Forster, A; Gonzales, J. M; Guzzi, J; Longchamp, V; Magnenat, S; Mathews, N; de Oca, M. M; O'Grady, R; Pinciroli, C; Pini, G; Retornaz, P; Roberts, J; Sperati, V; Stirling, T; Stranieri, A; Stutzle, T; Trianni, V; Tuci, E; Turgut, A. E, and Vaussard, F, Dec 2013. Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics Automation Magazine*, 20(4):60–71.

Dorigo, M and Şahin, E, 2004. Guest editorial. *Autonomous Robots*, 17 (2):111–113.

Durrant-Whyte, H and Bailey, T, 2006. Simultaneous localization and mapping: part I. *Robotics Automation Magazine, IEEE*, 13(2):99–110.

Edelman, G. M. *Neural Darwinism: the theory of neuronal group selection*. Basic Books, New York, 1987.

Edelman, G. M and Mountcastle, V. B. *The mindful brain: cortical organization and the group-selective theory of higher brain function.* MIT Press, 1978.

Evison, S. E. F; Webster, K. A, and Hughes, W. O. H, 2012. Better the nest site you know: decision-making during nest migrations by the pharaoh's ant. *Behavioral Ecology and Sociobiology*, 66(5):711–720.

Floreano, D and Mattiussi, C. *Bio-inspired artificial intelligence: theories, methods, and technologies*. Intelligent robotics and autonomous agents. MIT Press, 2008.

Floreano, D and Mondada, F, 1998. Evolutionary neurocontrollers for autonomous mobile robots. *Neural Networks*, 11(7–8):1461–1478.

Flushing, E. F; Gambardella, L, and Di Caro, G, 2012. Search and rescue using mixed swarms of heterogeneous agents: modeling, simulation, and planning. *IDSIA, Lugano, Switzerland, Tech. Rep*, pages 05–12.

Franks, N. R, March 1989. Army ants: a collective intelligence. *American Scientist*, 77:138–145.

Franks, N. R; Pratt, S. C; Mallon, E. B; Britton, N. F, and Sumpter, D. J. T, 2002. Information flow, opinion polling and collective intelligence in house–hunting social insects. *Philosophical Transactions of the Royal Society of London B: Biological Sciences*, 357 (1427):1567–1583.

Franks, N; Richardson, T; Stroeymeyt, N; Kirby, R; Amos, W; Hogan, P; Marshall, J, and Schlegel, T, 2013. Speed–cohesion trade-offs in collective decision making in ants and the concept of precision in animal behaviour. *Animal Behaviour*, 85(6):1233–1244.

Froese, T; Virgo, N, and Izquierdo, E. Autonomy: A review and a reappraisal. In Almeida e Costa, F; Rocha, L. M; Costa, E; Harvey, I, and Coutinho, A, editors, *Advances in Artificial Life: 9th European Conference, ECAL 2007, Lisbon, Portugal, September 10-14, 2007. Proceedings*, pages 455–464, 2007.

Gamma, E; Helm, R; Johnson, R, and Vlissides, J. *Design patterns: elements of reusable object-oriented software*. Addison–Wesley, 1994.

Garnier, S; Jost, C; Jeanson, R; Gautrais, J; Asadpour, M; Caprari, G, and Theraulaz, G. Collective decision-making by a group of cockroach-like robots. In *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, pages 233–240, June 2005.

Gauci, M; Chen, J; Li, W; Dodd, T. J, and Groß, R. Clustering objects with robots that do not compute. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, pages 421–428, Richland, SC, 2014a.

Gauci, M; Chen, J; Li, W; Dodd, T. J, and Groß, R, 2014b. Self-organized aggregation without computation. *The International Journal of Robotics Research*, 33(8):1145–1161.

Gibbons, J. D and Chakraborti, S. *Nonparametric statistical inference, 4th ed.* Springer, 2011.

Goldstone, R. L and Gureckis, T. M, 2009. Collective behavior. *Topics in Cognitive Science*, 1(3):412–438.

Gordon, D. M, 2002. The regulation of foraging activity in red harvester ant colonies. *The American Naturalist*, 159(5):509–518.

Hauert, S; Leven, S; Zufferey, J.-C, and Floreano, D. Beat-based synchronization and steering for groups of fixed-wing flying robots. In Martinoli, A; Mondada, F; Correll, N; Mermoud, G; Egerstedt, M; Hsieh, M. A; Parker, L. E, and Støy, K, editors, *Distributed Autonomous Robotic Systems*, volume 83 of *Springer Tracts in Advanced Robotics*, pages 281–293. Springer Berlin Heidelberg, 2013.

Haykin, S. *Neural networks: a comprehensive foundation*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 1994.

Hilder, J; Naylor, R; Rizihs, A; Franks, D, and Timmis, J. The Pi Swarm: A low-cost platform for swarm robotics research and education. In Mistry, M; Leonardis, A; Witkowski, M, and Melhuish, C, editors, *Advances in Autonomous Robotics Systems*, volume 8717 of *Lecture Notes in Computer Science*, pages 151–162. Springer International Publishing, 2014.

Hofstadter, D. R. *Gödel, Escher, Bach: an eternal golden braid*. Random House, New York, 1979.

Holland, O and Melhuish, C, April 1999. Stigmergy, self-organization, and sorting in collective robotics. *Artificial Life*, 5(2):173–202.

Hornik, K; Stinchcombe, M, and White, H, 1989. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366.

Ijspeert, A. J; Martinoli, A; Billard, A, and Gambardella, L. M, 2001. Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Autonomous Robots*, 11(2):149–171.

Ising, E, 1925. Beitrag zur theorie des ferromagnetismus. *Zeitschrift für Physik*, 31(1):253–258.

Jakobi, N; Husbands, P, and Harvey, I. Noise and the reality gap: The use of simulation in evolutionary robotics. In Morán, F; Moreno, A; Merelo, J, and Chacón, P, editors, *Advances in Artificial Life*, volume 929 of *Lecture Notes in Computer Science*, pages 704–720. Springer Berlin Heidelberg, 1995.

Jeanson, R; Deneubourg, J.-L; Grimal, A, and Theraulaz, G, 2004. Modulation of individual behavior and collective decision-making during aggregation site selection by the ant *Messor barbarus*. *Behavioral Ecology and Sociobiology*, 55(4):388–394.

Kaur, R; Anoop, K, and Sumana, A, 2012. Leaders follow leaders to reunite the colony: relocation dynamics of an indian queenless ant in its natural habitat. *Animal Behaviour*, 83(6):1345–1353.

Kengyel, D; Schmickl, T; Hamann, H; Thenius, R, and Crailsheim, K. Embodiment of honeybee's thermotaxis in a mobile robot swarm. In Kampis, G; Karsai, I, and Szathmáry, E, editors, *Advances in Artificial Life*, volume 5778 of *Lecture Notes in Computer Science*, pages 69–76. Springer Berlin Heidelberg, 2011.

Kohonen, T, April 1972. Correlation matrix memories. *Computers, IEEE Transactions on*, C-21(4):353 –359.

Lindauer, M, 1957. Communication in swarm-bees searching for a new home. *Nature*, 179(4550):63–66.

List, C; Elsholtz, C, and Seeley, T. D, 2009. Independence and interdependence in collective decision making: an agent-based model of nest-site choice by honeybee swarms. *Philosophical*

Transactions of the Royal Society of London B: Biological Sciences, 364 (1518):755–762.

Lorenz, J; Rauhut, H; Schweitzer, F, and Helbing, D, 2011. How social influence can undermine the wisdom of crowd effect. *Proceedings of the National Academy of Sciences*, 108(22):9020–9025.

Louchard, G and Bruss, F. T. Finding the κ best out of n rankable objects. A consecutive thresholds algorithm. Working paper or preprint, October 2015. URL `https://hal.archives-ouvertes.fr/hal-01208327`.

Makinson, J. C; Oldroyd, B. P; Schaerf, T. M; Wattanachaiyingcharoen, W, and Beekman, M, 2010. Moving home: nest-site selection in the red dwarf honeybee (*Apis florea*). *Behavioral Ecology and Sociobiology*, 65(5):945–958.

Mann, H. B and Whitney, D. R, 1947. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics*, 18(1):50–60.

Marino, S; Hogue, I. B; Ray, C. J, and Kirschner, D. E, 2008. A methodology for performing global uncertainty and sensitivity analysis in systems biology. *Journal of Theoretical Biology*, 254(1): 178–196.

Marshall, J. A. R and Franks, N. R, 5 2009. Colony-level cognition. *Current Biology*, 19(10):R395–R396.

Marshall, J. A; Bogacz, R; Dornhaus, A; Planqué, R; Kovacs, T, and Franks, N. R, 2009. On optimal decision-making in brains and social insect colonies. *Journal of The Royal Society Interface*.

Marshall, J. A; Favreau-Peigne, A; Fromhage, L; Mcnamara, J. M; Meah, L. F, and Houston, A. I, 2015. Cross inhibition improves activity selection when switching incurs time costs. *Current zoology*, 61(2):242–250.

Massey, F. J, 1951. The Kolmogorov-Smirnov test for goodness of fit. *Journal of the American Statistical Association*, 46(253):68–78.

mbed, . mbed LPC1768. Online: `https://developer.mbed.org/platforms/mbed-LPC1768/`, April 2016.

Melhuish, C; Ieropoulos, I; Greenman, J, and Horsfield, I, 2006. Energetically autonomous robots: Food for thought. *Autonomous Robots*, 21(3):187–198.

Melucci, A. The process of collective identity. In Johnston, H and Klandermans, B, editors, *Social Movements and Culture*, chapter 3. University of Minnesota Press, 1995.

Millard, A; Timmis, J, and Winfield, A. Towards exogenous fault detection in swarm robotic systems. In Natraj, A; Cameron, S; Melhuish, C, and Witkowski, M, editors, *Towards Autonomous Robotic Systems*, volume 8069 of *Lecture Notes in Computer Science*, pages 429–430. Springer Berlin Heidelberg, 2014.

Mitchell, M, 2005. Self-awareness and control in decentralized systems. *Metacognition in Computation*, pages 80–85.

Moioli, R. C; Vargas, P. A, and Husbands, P. A multiple hormone approach to the homeostatic control of conflicting behaviours in an autonomous mobile robot. In *2009 IEEE Congress on Evolutionary Computation*, pages 47–54, May 2009.

Neal, M and Timmis, J, 2003. Timidity: A useful emotional mechanism for robot control? *Informatica*, 27(2):197–204.

Neal, M and Timmis, J. Once more unto the breach... towards artificial homeostasis? In De Castro, L. N and Von Zuben, F. J, editors, *Recent Developments in Biologically Inspired Computing*. Idea Group Pub., 2005.

Nieh, J. C, 1993. The stop signal of honey bees: reconsidering its message. *Behavioral Ecology and Sociobiology*, 33(1):51–56.

Nieh, J. C, 2010. A negative feedback signal that is triggered by peril curbs honey bee recruitment. *Current Biology*, 20(4):310–315.

Nolfi, S; Floreano, D, and Floreano, D. *Evolutionary robotics: The biology, intelligence, and technology of self-organizing machines*. MIT Press, Cambridge, MA, USA, 2000.

Olfati-Saber, R, 2006. Flocking for multi-agent dynamic systems: algorithms and theory. *Automatic Control, IEEE Transactions on*, 51 (3):401–420.

Pais, D; Hogan, P. M; Schlegel, T; Franks, N. R; Leonard, N. E, and Marshall, J. A. R, 2013. A mechanism for value-sensitive decision-making. *PLoS ONE*, 8(9):e73216.

Palm, G, 1980. On associative memory. *Biological Cybernetics*, 36(1): 19–31.

Palm, G, 1981. Towards a theory of cell assemblies. *Biological Cybernetics*, 39(3):181–194.

Parker, C. A. C and Zhang, H, April 2009. Cooperative decision-making in decentralized multiple-robot systems: The best-of-n problem. *IEEE/ASME Transactions on Mechatronics*, 14(2):240–251.

Parker, C. A. C and Zhang, H, 2011. Biologically inspired collective comparisons by robotic swarms. *The International Journal of Robotics Research*, 30(5):524–535.

Passino, K; Seeley, T, and Visscher, P, 2008. Swarm cognition in honey bees. *Behavioral Ecology and Sociobiology*, 62:401–414.

Passino, K. M; Seeley, T. D, and Visscher, P. K, 2010. Honey bee swarm cognition: Decision-making performance and adaptation. *International Journal of Swarm Intelligence Research*, 1(2):80–97.

Petersen, K; Nagpal, R, and Werfel, J. TERMES: an autonomous robotic system for three-dimensional collective construction. In Durrant-Whyte, H. F; Roy, N, and Abbeel, P, editors, *Robotics: Science and Systems VII, University of Southern California, Los Angeles, CA, USA, June 27-30, 2011*, 2011.

Pfeifer, R and Scheier, C. *Understanding intelligence*. MIT Press, Cambridge, MA, USA, 2001.

Putnam, H, 1960. Minds and machines. *Dimensions of mind*, pages 148–180.

Rajeshwar, K, 2012. Biomimetic or bioinspired? (editorial). *Electrochemical Society Interface*, 21(3–4):3.

Read, M; Andrews, P. S; Timmis, J, and Kumar, V, 2012. Techniques for grounding agent-based simulations in the real domain: a case study in experimental autoimmune encephalomyelitis. *Mathematical and Computer Modelling of Dynamical Systems*, 18(1):67–86.

Reina, A; Miletitch, R; Dorigo, M, and Trianni, V, 2015a. A quantitative micro–macro link for collective decisions: the shortest path discovery/selection example. *Swarm Intelligence*, pages 1–28.

Reina, A; Valentini, G; Fernández-Oto, C; Dorigo, M, and Trianni, V, 2015b. A design pattern for decentralised decision making. *PLoS ONE*, 10(10):e0140950.

Reynolds, C. W. Flocks, herds and schools: A distributed behavioral model. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, SIGGRAPH '87, pages 25–34, 1987.

Rosenblatt, F, 1958. The Perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

Rubenstein, M; Cornejo, A, and Nagpal, R, 2014. Programmable self-assembly in a thousand-robot swarm. *Science*, 345(6198):795–799.

Rumelhart, D. E; Hinton, G. E, and Williams, R. J, 10 1986. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536.

Russell, S. J and Norvig, P. *Artificial intelligence: a modern approach*. Pearson Education, 2 edition, 2003.

Ryan, A. J, 2007. Emergence is coupled to scope, not level. *Complexity*, 13(2):67–77.

Şahin, E. Swarm robotics: from sources of inspiration to domains of application. In Şahin, E and Spears, W, editors, *Swarm Robotics*, volume 3342 of *Lecture Notes in Computer Science*, pages 10–20. Springer Berlin / Heidelberg, 2005.

Schaerf, T. M; Makinson, J. C; Myerscough, M. R, and Beekman, M, 2013. Do small swarms have an advantage when house hunting? The effect of swarm size on nest-site selection by *Apis mellifera*. *Journal of The Royal Society Interface*, 10(87).

Schillo, M and Fischer, K. A taxonomy of autonomy in multiagent organisation. In Nickles, M; Rovatsos, M, and Weiss, G, editors, *Agents and Computational Autonomy: Potential, Risks, and Solutions*, pages 68–82. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

Schmickl, T and Hamann, H, 2011. BEECLUST: A swarm algorithm derived from honeybees. *Bio-inspired Computing and Communication Networks. CRC Press (March 2011)*.

Schmickl, T; Möslinger, C; Crailsheim, K; Sahin, E; Spears, W, and Winfield, A. Collective perception in a robot swarm. In *LNCS*, volume 4433, pages 144–157. Springer Berlin / Heidelberg, 2007.

Schmickl, T; Thenius, R; Moeslinger, C; Radspieler, G; Kernbach, S; Szymanski, M, and Crailsheim, K, 2009. Get in touch: cooperative decision making based on robot-to-robot collisions. *Autonomous Agents and Multi-Agent Systems*, 18(1):133–155.

Schmickl, T; Hamann, H, and Crailsheim, K, 2011. Modelling a hormone-inspired controller for individual- and multi-modular robotic systems. *Mathematical and Computer Modelling of Dynamical Systems*, 17(3):221–242.

Schoonderwoerd, R; Holland, O. E; Bruten, J. L, and Rothkrantz, L. J. M, 1997. Ant-based load balancing in telecommunications networks. *Adaptive Behavior*, 5(2):169–207.

Seeley, T. D and Buhrman, S. C, 1999. Group decision making in swarms of honey bees. *Behavioral Ecology and Sociobiology*, 45(1): 19–31.

Seeley, T. D and Visscher, P. K, 2004. Group decision making in nest-site selection by honey bees. *Apidologie*, 35(2):101–116.

Seeley, T. D; Camazine, S, and Sneyd, J, 1991. Collective decision-making in honey bees: how colonies choose among nectar sources. *Behavioral Ecology and Sociobiology*, 28(4):277–290.

Seeley, T. D; Visscher, P. K; Schlegel, T; Hogan, P. M; Franks, N. R, and Marshall, J. A. R, 2012. Stop signals provide cross inhibition in collective decision-making by honeybee swarms. *Science*, 335(6064): 108–111.

Smith, R; Self, M, and Cheeseman, P. A stochastic map for uncertain spatial relationships. In *Proceedings of the 4th International Symposium on Robotics Research*, pages 467–474, 1988.

Stepney, S. How many times should you run your experiment? Private Communication, July 2015.

Stepney, S; Polack, F, and Turner, H. Engineering emergence. In *ICECCS 2006: 11th IEEE International Conference on Engineering of Complex Computer Systems, Stanford, CA, USA, August 2006*, pages 89–97, 2006.

Sterling, P. Principles of allostasis: optimal design, predictive regulation, pathophysiology and rational therapeutics. In Schulkin, J, editor, *Allostasis, Homeostasis, and the Costs of Physiological Adaptation*, pages 17–64. Unviersity of Cambridge Press, 2004.

Sterling, P, 2012. Allostasis: A model of predictive regulation. *Physiology & Behavior*, 106(1):5–15.

Sterling, P and Eyer, J, 1981. Biological basis of stress-related mortality. *Social Science & Medicine. Part E: Medical Psychology*, 15(1):3–42.

Sterling, P and Eyer, J. *Handbook of life stress, cognition and health*, chapter Allostasis: A new paradigm to explain arousal pathology, pages 629–649. John Wiley & Sons, Oxford, England, 1988.

Stovold, J; O'Keefe, S, and Timmis, J. Preserving swarm identity over time. In Sayama, H; Rieffel, J; Risi, S; Doursat, R, and Lipson, H, editors, *Proceedings of the 14th international conference on the synthesis and simulation of living systems (ALIFE '14)*, pages 728–735, 2014.

Stovold, J; O'Keefe, S, and Timmis, J, 2016. Flock-based cognitive decision-making as an abstract model of action selection (in preparation). *Artificial Life Journal*.

Stradner, J; Hamann, H; Schmickl, T, and Crailsheim, K. Analysis and implementation of an artificial homeostatic hormone system: A first case study in robotic hardware. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 595–600, Oct 2009.

Student, , 1908. The probable error of a mean. *Biometrika*, 6(1):1–25.

Timmis, J; Neal, M, and Thorniley, J. An adaptive neuro-endocrine system for robotic systems. In *Robotic Intelligence in Informationally Structured Space, 2009. RIISS '09. IEEE Workshop on*, pages 129–136, March 2009.

Timmis, J; Ismail, A; Bjerknes, J, and Winfield, A, 2016. An immune-inspired swarm aggregation algorithm for self-healing swarm robotic systems. *Biosystems (in press)*.

Trianni, V and Dorigo, M. Emergent collective decisions in a swarm of robots. In *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, pages 241–248, 2005.

Trianni, V; Tuci, E; Passino, K, and Marshall, J, 2011. Swarm cognition: an interdisciplinary approach to the study of self-organising biological collectives. *Swarm Intelligence*, 5:3–18.

Tyrrell, A; Auer, G, and Bettstetter, C. Firefly synchronization in ad hoc networks. In *Proceedings of the MiNEMA Workshop*, 2006.

Valentini, G; Hamann, H, and Dorigo, M. Self-organized collective decision-making in a 100-robot swarm. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2015a.

Valentini, G; Hamann, H, and Dorigo, M. Efficient decision-making in a self-organizing robot swarm: On the speed versus accuracy trade-off. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '15, pages 1305–1314, 2015b.

Varela, F. G; Maturana, H. R, and Uribe, R, 5 1974. Autopoiesis: The organization of living systems, its characterization and a model. *Biosystems*, 5(4):187–196.

Vargas, P; Moioli, R; Castro, L. N; Timmis, J; Neal, M, and Zuben, F. J. Artificial homeostatic system: A novel approach. In Capcarrère, M. S; Freitas, A. A; Bentley, P. J; Johnson, C. G, and Timmis, J, editors, *Advances in Artificial Life: 8th European Conference, ECAL 2005, Canterbury, UK, September 5-9, 2005 Proceedings*, pages 754–764. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

Vargha, A and Delaney, H. D, 2000. A critique and improvement of the "CL" common language effect size statistics of McGraw and Wong. *Journal of Educational and Behavioral Statistics*, 25(2):101–132.

Vernon, D. *Artificial Cognitive Systems*. MIT Press, 2015.

Visscher, P. K and Camazine, S, Feb 1999. Collective decisions and cognition in bees. *Nature*, 397(6718):400–400.

von Frisch, K. *The dance language and orientation of bees.* Harvard University Press, Cambridge, MA, 1967.

Vullev, V. I, 2011. From biomimesis to bioinspiration: What's the benefit for solar energy conversion applications? *The Journal of Physical Chemistry Letters*, 2(5):503–508.

Walsh, J. P and Ungson, G. R, 1991. Organizational memory. *The Academy of Management Review*, 16(1):57–91.

Wang, Y and Ruhe, G, 2007. The cognitive process of decision making. *International Journal of Cognitive Informatics and Natural Intelligence (IJCINI)*, 1(2):73–85.

Watkins, C. J. C. H and Dayan, P, 1992. Q-learning. *Machine Learning*, 8(3):279–292.

Werfel, J; Petersen, K, and Nagpal, R, 2014. Designing collective behavior in a termite-inspired robot construction team. *Science*, 343 (6172):754–758.

Wilensky, U. NetLogo fireflies model. `http://ccl.northwestern.edu/netlogo/models/Fireflies` Center for Connected Learning and Computer-Based Modelling, Northwestern University, Evanston, IL., 1997.

Wilensky, U. NetLogo flocking model. `http://ccl.northwestern.edu/netlogo/models/Flocking` Center for Connected Learning and Computer-Based Modelling, Northwestern University, Evanston, IL., 1998.

Wilensky, U. NetLogo. `http://ccl.northwestern.edu/netlogo/` Center for Connected Learning and Computer-Based Modelling, Northwestern University, Evanston, IL., 1999.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. In *Machine Learning*, pages 229–256, 1992.

Willshaw, D. J; Buneman, O. P, and Longuet-Higgins, H. C, June 1969. Non-holographic associative memory. *Nature*, 222(5197):960–962.

Yu, C.-H; Werfel, J, and Nagpal, R. Collective decision-making in multi-agent systems by implicit leadership. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 3*, AAMAS '10, pages 1189–1196. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

Zavlanos, M. M and Pappas, G. J, 2008. Distributed connectivity control of mobile networks. *Robotics, IEEE Transactions on*, 24(6): 1416–1428.