An Investigation of Loose Coupling in Evolutionary Swarm Robotics

Jennifer Owen

A thesis submitted for the degree of
Doctor of Philosophy

University of York
Computer Science
January 2013

Abstract

In complex systems, it has been observed that the parts within the system are "loosely coupled". Loose coupling means that the parts of the system interact in some way, and as long as this interaction is maintained the parts can evolve independently. Detrimental evolutionary changes within one part of the system do not negatively affect other parts. Overall system functionality is maintained, leading to faster evolution.

In swarm robotic systems there are multiple robots working together to achieve a shared goal. However it is not always obvious how to program the actions of the robots such that the desired aggregate behaviour emerges. One solution is to use a genetic algorithm to evolve robot controllers, this approach is called "Evolutionary Swarm Robotics".

This thesis makes the case that swarm robotic systems are complex systems, and hypothesises that loose coupling between the robots in a swarm would lead to faster evolution. Robot swarms are investigated where robots describe environmental features to each other as part of a foraging task. Multiple descriptions can be used to describe a feature. The mappings between feature descriptions, and the signals used to express those descriptions, are manipulated. By doing this, the interactions between robots can change over time or stay the same.

Results show that loose coupling leads to higher swarm fitnesses because it makes the communicated information easier to interpret. However there are some subtleties in its working. We also observe that if some of the information is not useful for completing the task, this negatively affects swarm fitness regardless of coupling. This problem can be mitigated by using loose coupling.

This research has implications for the design of communication within robot swarms. Before evolution, it is difficult to know what information is relevant. This research shows that sharing unnecessary information between robots is detrimental to swarm fitness because the cost of interpreting information can be greater than the benefit gained from the information. Loose coupling can reduce, but not eliminate, the evolutionary cost of interpreting multiple pieces of information in exchange for slower message transmission.

Contents

List of Figures

List of Tables

Declaration

The work presented in this thesis is my own, except where explicitly stated.

The hypothesis in chapter 4 and elements of chapters 2 and 3 have been published in:

- Jennifer Owen, Susan Stepney, Jonathan Timmis, and Alan F. T. Winfield. "Exploiting loose horizontal coupling in evolutionary swarm robotics". In *Proceedings of the 7th international conference on Swarm intelligence*, ANTS'10, pages 432–439, Berlin, Heidelberg, 2010. Springer-Verlag.

Work from chapters 7 and 6 has been published in:

- Jennifer Owen "Developing a Simulation and Hardware for a Robot Swarm Using Sound to Communicate". In *Proceedings of the Fourth York Doctoral Symposium on Computer Science*, pages 98–99, York UK, 2011.

Chapter 1

Introduction

In this work we study swarm robots as a complex system. In particular we look at the link between complexity and evolution in evolutionary swarm robotic systems. Our main goal is to investigate whether the robot to robot communication can be predefined and structured to increase the rate of improvement during evolution.

1.1 Background Context

In section 2.2.4 we define complex systems as:

> A system which has a hierarchic structure of subsystems, from which we can observe emergent properties and behaviours which are not apparent from examining its constituent parts.

The phrase "hierarchic structure of subsystems" means that within each subsystem there are more subsystems which interact and have emergent properties, and within each of those there are more subsystems, and within each of those are more subsystems and so on until some atomic unit is reached. An ant colony is an example of a complex system. A colony consists of ants (subsystems) interacting with each other and the environment to produce colony-level behaviour. Within each ant there are cells which interact with each other and their environment to produce ant-level behaviour, and within each cell there are proteins and other structures and so on until some atomic unit is reached.

Complex systems can be studied by making a "model" of them. *"A model is an abstraction that is made to aid understanding or description of something"* [72]. Only an abstraction of the complex system is used because it would be intractable to try and incorporate entire system into the model. The model would become as complicated to understand as the original system and nothing could be gained from building it [6]. Using the example of an ant colony, modelling every cell of every ant in the colony would mean modelling billions of cells, making the model intractably large. The research question the model is used to answer dictates the level of abstraction required. Aspects of the complex system which do not affect the research question do not need to be modelled in detail.

It was observed by Simon [84] that complex systems evolve faster than non-complex ones, and one of the causes of this is "loose coupling" between subsystems. Subsystems are *coupled* if the interactions between them happen in a manner that doesn't change

over time, or change very slowly in comparison to the subsystems. It is possible to have different strengths of coupling [33]. If the coupling between the subsystems is "tight" then they are co-dependent and have a strong influence on each other. Consequently if evolution causes one subsystem to be less effective, other subsystems will be adversely affected, so the whole system may cease to function. If the coupling is "loose" then the subsystems interact and share information, but are largely independent and do not have a strong influence over each other. Detrimental changes in one subsystem worsen the effectiveness of the whole system, but as long as the coupling is maintained it does not cause other subsystems to break down and the system remains functional.

Swarm robotic systems are artificial complex systems. The subsystems are the robots within the swarm and when the robots are together they form a swarm, which displays a collective behaviour not always apparent from the behaviour of the individual robots.

Swarm robotics research is described by Şahin [78] as:

> The study of how large number of relatively simple physically embodied agents can be designed such that a desired collective behaviour emerges from the local interactions among agents and between the agents and the environment.

The advantage of swarm robotics is that the robots are designed to do their task without a centralised controller. No robot has knowledge of the entire swarm or the state of the whole environment, they must each perform actions based entirely on their own robot-level observations. This means that the swarm is scalable, so an arbitrary number of robots can be added and the swarm should still be able to do its task. Similarly, the swarm is robust to robot failures. If a robot breaks, it can be removed from the swarm with no significant loss of swarm functionality [78].

The main challenge of swarm robotics is how to design the robot-level behaviour such that the desired swarm-level behaviour is displayed, particularly when the environment the robots operate in is not fully known. One solution to this problem is to use evolutionary algorithms to artificially evolve the robots' controllers and behaviour [89], this is called Evolutionary Swarm Robotics. The main disadvantage of evolutionary swarm robotics is that it is very slow. It takes time to execute each controller, so that the robot has enough time to attempt its task, and it takes many iterations of scoring old controllers and generating new ones before sufficiently good controllers are created. This process can take many hours or days of experimentation time.

1.2 Aims and Contributions

Loose coupling is theorised in [84] to benefit the evolution of complex systems. In this thesis we aim to address the slowness of evolutionary swarm robotics, by hypothesising that loose coupling can be applied to swarm robots to benefit their evolution. We apply coupling to a swarm robotic system by predefining and structuring the way information is communicated between robots. The strength of the coupling can be changed by altering the structure of that communication. The aim of this work is to test the hypothesis by applying different strengths of coupling to a robot swarm performing a foraging task, and observing the fitness of the resulting robot controllers.

The CoSMoS (**Co**mplex **S**ystem **Mo**delling and **S**imulation) process is a framework to follow for modelling, so that as the model is built assumptions and abstractions in

the model are documented to build trust in the model that its results are reliable [6]. The secondary aim of this work is to provide a case study of the CoSMoS process for modelling an engineered complex system.

1.2.1 Contributions

- It is found, in chapter 10, that loose coupling leads to fitter robot controllers than tight coupling, because some of the information communicated between robots is found not to be useful for completing the task. Loosely coupling the communication between the robots made it easier to extract only the useful information from the messages.

- The results also suggest that making more data available to the robots does not necessarily result in better performance, because there is more information to interpret. This effect was lessened when loose coupling was used.

- In chapters 6 to 8 we develop a model of the robot swarm following the CoSMoS process, providing a case study for the application of CoSMoS to modelling an engineered complex system. In chapter 6 we develop a model of the robots, their environment, the acoustics of the environment and a hardware extension that the robots use to send and receive audio signals. The hardware extension does not exist at the time of the modelling, it is only later in chapter 7 that it is built. In chapter 8 we are then able to take measurements from the extension and use it to calibrate the model.

Additional contributions of this work are given in section 11.2.1.

1.3 Thesis Structure

This document is divided into three parts, relating to the research aims.

Part I: Background and Literature Review In this part we explain in detail the scientific context to the hypothesis, what the hypothesis is and how we test it.

In chapters 2 and 3 we discuss complex systems, swarm intelligence and swarm robotics in detail. Then in chapter 4 we explain our hypothesis, applying observations about complex systems to swarm robotics. In chapter 5 we outline an experiment for testing the hypothesis.

Part II: Co-Development of Simulation and Hardware In this part, we develop a model of the swarm robots and their environment in order to carry out the experiment to test the hypothesis.

In chapter 6 we develop a model of the known aspects of the robots and their environment but note that the robots need additional hardware in order to perform the experiment. In chapter 7 we build a hardware extension for the robots, and in chapter 8 we perform extensive calibration of model, taking measurements from the hardware extension and putting the results into the model.

Part III: Results and Conclusion In the final part we use the model developed in part II to test the hypothesis from part I.

In chapter 9 we perform the experiment from chapter 5, but find it does not adequately test the hypothesis. Building on these results we run a follow-up experiment in chapter 10 and find that loose coupled swarms evolved fitter controllers than tightly coupled swarms. The results also show that if some of the information shared in the environment is not useful, then the communication structure used by the loosely coupled swarm makes it easier for the robots to extract the useful information. Finally in conclusion chapter 11, we summarise the findings and contributions of the thesis and suggest further experiments that follow on from our research.

Part I

Background and Literature Review

Chapter 2

Complexity and Emergence

2.1 Introduction

The core focus of this work is in swarm robotics research. As is discussed in chapter 3, swarm robotics and the swarm intelligent organisms that inspire them, are both types of *complex system*. Swarm intelligent organisms are biological complex systems, and swarm robotic systems are man-made. However, it is important for the understanding of swarm robotics, and our research into swarm robotics, to understand what constitutes a complex system and how they are studied.

In section 2.2 we bring together complex systems literature to form a definition of what the term "complex system" means for our work. In section 2.3 we explain the motivations for studying complex systems and discuss the effects of complexity on the evolution of complex systems. Finally in section 2.4 we describe a way of studying complex systems called "modelling".

2.2 Definitions

Complexity is often described as being a state between order and randomness [19], and complex systems sit on the *"Edge of Chaos"* [53]. This is the boundary between the two extremes where any movement towards one extreme rapidly results in the system devolving into randomness, or order on the other extreme [53]. Any system which dwells on the "Edge of Chaos" will contain both order and randomness, interacting with each other in unpredictable ways on small or large scales. If there are changes in its environment then it must be able to adapt to these changes to maintain the balance. A complex system which achieves this is called a Complex Adaptive System [36]; every living organism falls into this category.

Despite their prevalence in the world around us, what constitutes a complex systems still remains hard to define. The best we can do it try to characterise complex systems by features that they have in common, specifically this is the feature of "complexity".

First we describe different types of complexity, and then we examine the features which complex systems have in common. Finally we define what a complex system is for the purposes of this work.

2.2.1 Types of Complexity

Complexity theory has been an attempt to generalise all complex systems by mathematically describing their common features and behaviours. Manson [56] argues that this has been unsuccessful because systems classed as complex often behave in very different ways. To this end, Manson suggests that there are three categories of complex systems, each of which have slightly different properties and characteristics.

Algorithmic Complexity There are 2 measures of algorithmic complexity as Manson describes it. The first is a measure of the "effort required to solve a problem". For instance the travelling salesman problem is complex because for anything other than the most trivial cases it gets exponentially more difficult to solve. Measuring the algorithmic complexity of a solution allows us to make comparisons between alternative solutions. The second measure, known as Kolmogorov Complexity [44], is to do with finding the complexity of a sequence of behaviours (represented as a string) by finding the length in bits of the shortest computer program able to reproduce the string. For example, the binary sequence 0000 can be reproduced with the program `for(i=0;i<4;i++) print '0';`, so has a low Kolmogorov complexity.

Deterministic Complexity Systems that display deterministic complexity are deterministic in that there are no stochastic elements to their functioning; i.e. if the system is run twice with the same initial conditions, the state of both systems would be the same after n time steps (where n is any positive integer). The complexity in these types of systems comes from their reaction to the initial conditions. Small changes in the initial conditions could cause huge differences in future states, but they could easily cause no change at all. The system has a non-linear, but repeatable reaction to its starting conditions. An example of a deterministic complex system is cellular automata. A cellular automaton is a grid or line of cells which can be in one of many states, state is usually represented by the cell's colour. Their current state and the state of a cell's neighbours dictate what the cell's next state will be. A set of rules is followed to determine this state change [98].

Aggregate Complexity Aggregate complexity is a complex system formed by many elements combining together to create a new entity or to achieve some task; the complexity of this kind of system arises from the elements themselves and how they interact with each other. Without interactions between the elements in the aggregate the system is not complex: A stamp collection is an aggregate of some stamps, but because they do not interact with each other the stamp collection is not a complex system. This type of complexity is the one most relevant to the study of swarm robotics since the field of swarm robotics aims to find ways of getting many robots to collaborate with one another to achieve some common goal. If we can understand the workings of aggregate complex systems and how to manipulate them it would be highly beneficial to swarm robotics research.

The remainder of section 2.2 explores common features of aggregate complex systems. Specific cases and applications of this kind of complex system are described in chapter 3.

2.2.2 Unpredictability

As mentioned at the start of this chapter, complex systems exist on the "edge of chaos". They can also easily devolve into random or ordered states. Both random and ordered systems are predictable, statistically or otherwise, it is on the "edge of chaos" that a system will behave unpredictably.

Beni [12] discusses the nature of what makes a system unpredictable and identifies five key features which would make a system unpredictable.

1. **Statistical unpredictability**. This means that the way the system changes cannot be predicted from one state to the next, but it can be generalised statistically. An example of a system displaying this type of unpredictability would be a stochastic system, such as atoms in a gas or the total of two dice rolls. This form of unpredictability is not as likely to be manifest in complex system as the other forms, since statistically unpredictable systems are potentially entirely random.

2. **Inaccessibility**. The system functions as some sort of "black box" in reaction to its environment. We do not know the contents of the black box so the system is unpredictable. If we were to find out how it functioned the system may become predictable or it may continue to display other forms of unpredictability. Biological systems such as plants are unpredictable because we do not fully know their inner functions so cannot accurately predict how the plant will grow. Research and experimentation may eventually reveal the plant's inner workings (its genetic code or cellular structure for instance), at which point it may be possible to predict how the plant will grow. Or if our predictions prove inaccurate, it would reveal that some other form of unpredictability is present in the system.

3. **Undecidability**. This applies to systems where the *"infinite time behaviour of a system [...] is in general unknowable in any finite time"* [12]. That is, we can not predict how the system will behave after infinity time steps.

4. **Intractability**. This type of unpredictability is linked to undecidability. If a system is intractable this means that it takes longer to make a prediction about the system's future state than it takes for that system to reach that future state. This is the case for any time step in the system's lifetime, not just after infinity steps. In this regard undecidability can be seen as a special case of intractability, in that undecidable, intractable systems will never be predictable whereas non-undecidable, intractable systems may gradually become predictable.

5. **Non-representability**. If a system in unrepresentable it means that it cannot be described by mathematical equations.

In an swarm robotics system the swarm will behave unpredictably. Their inner functions are mostly known because it has been programmed, but there is still some amount of inaccessibility because of inconsistencies with a robot's response to commands. Battery charge, for instance, affects the motor speed on a robot, and the amount of devices it can use at once. The robot's parts will also degrade over time causing changes in its responses to commands. These factors insert unpredictability into the system due to their inaccessibility: they are factors we cannot reliably measure or fully characterise, any attempts we make to do so will become less reliable as the robot degrades. The swarm is also intractable, in part due to the inaccessibility of the robots, but also due to the sheer amount of variables present in a real-life situation.

2.2.3 Emergence and Hierarchy

Simon [83] observes that complex systems have similar "hierarchic structures". He defined a hierarchic structure as the following:

> A system that is composed of interrelated subsystems, each of the latter being, in turn, hierarchic in structure until we reach some lowest level of elementary subsystem.

Take the example of a molecule in a chemical. This molecule is composed of several atoms, each of which interact to give the chemical its particular properties. These properties may be different to the properties of each element in isolation. Water for instance is very different to hydrogen and oxygen from which it is made. Within each of the atoms are yet more "subsystems" of protons, neutrons and electrons, the only difference between the atoms being the number of these "subsystems" it contains. Another example of a system with this structure is the human body. In the body there are organs which interact to keep the body alive, each organ is made cells and cells are made of molecules. In a complex hierarchic structure it can be seen that at each level each subsystem can be said to be a complex hierarchy in its own right. Therefore each hierarchic complex system can be seen to really be a system of systems and from the top down the number of subsystems in the overall complex system grows exponentially. In a single ant colony there may be millions of ants [38] and in each ant there are billions of cells, and so the whole colony contains many trillions of cells.

The idea of emergence and complex systems is a reaction to the idea of *reductionism* [3]. Reductionism is the theory that if you are able to break something down into its most fundamental particles you could gain insight into how it functions [24]. With our ant colony example, analysing trillions of cells is clearly going to be impractical, the sheer quantity of data required and the variables acting upon each cell is too much for us to meaningfully analyse, and even a cell is of far higher complexity than the most fundamental particle. When applied to complex systems the bottom-up approach of reductionism is not appropriate as a means of analysis. From the bottom up, at each successive complexity level, we gain some knowledge of behaviour which was not apparent from observing merely the level below. This is called *emergence* because this knowledge or behaviour is said to *emerge* from the interactions within the lower complexity level but is not actually due to any individual subsystem. Anderson [3] summarises this phenomenon thusly:

> The behaviour of large and complex aggregates of elementary particles, it turns out, is not to be understood in terms of a simple extrapolation of the properties of a few particles. Instead at each level of complexity entirely new properties appear.

From examining a single cell we do not understand how an ant works, by examining an ant we do not understand how an ant swarm works. On each hierarchic complexity level there are interactions between subsystems, cells pass chemicals between each other, ants leave pheromone trials for each other to follow. It is from the aggregate behaviour of subsystems that we get the higher level complex system, and the subsystems themselves are the result of aggregate behaviour of their own subsystems.

Looking at a complex subsystem in isolation from its environment (which includes the other subsystems it interacts with) tells us nothing about the higher level system: there needs to be interactions to put the subsystem in context and provide the data

necessary for this greater understanding. Furthermore, if subsystems can collaborate to produce behaviour or properties that are not directly determinable from the original subsystems, then how can we engineer systems which reproduce this phenomenon? What interactions are necessary to generate an emergent behaviour? Aristotle's summary of emergence is *"the whole is something beside the parts"* [9] often quoted as *"The whole is more than the sum of its parts"*. So how can we put *less* into a system and get out *more*? Solving this problem is a driving force behind swarm robotics research because it should be much easier to make a few simple robots with a useful collaborative behaviour than it is to make a single highly intelligent robot with an equivalent behaviour. Section 3.4 discusses the challenges associated with this goal.

The method that we use in this project to analyse complex systems and their emergent properties and behaviours is called *modelling and simulation* and is discussed in section 2.4.

2.2.4 A Definition

In this work we are primarily interested in aggregate complexity, where large numbers of smaller systems collaborate to form a larger system. From the context of the existing literature we can define these complex systems for the purpose of our project as:

> A system which has a hierarchic structure of subsystems, from which we can observe emergent properties and behaviours which were not apparent from examining its constituent parts. It displays both order and randomness in its behaviour, and consequently will behave unpredictably. Completely accurate predictions are not possible because the system's inner functions are not completely known, and because the act of making predictions about it is intractable.

This definition is general enough to apply to the types of complex systems that this work studies, including swarm intelligent systems and swarm robotics.

2.3 Motivations Behind Complexity Research

Complex systems are a rich and worthwhile area of research because they are present everywhere and in every aspect of our lives. The global economy, for instance, is a very influential complex system and includes everybody in the world [36]. The global climate is also a complex system; a better understanding of how our actions affect the climate would help us to improve the planet and consequently our quality of life. Beneficial complex system research is not just limited to huge global systems. Physics, sociology, biology and other sciences are just some of the research areas which could benefit from this field of research. Understanding complex systems means understanding almost every aspect of our world; in this respect they are worth studying for knowledge alone.

An aspect of this increased understanding is using our knowledge to make predictions about the complex system, perhaps to assess the consequences of a course of action or to test hypotheses about how the system works. In section 2.2.2 we discussed how complex systems are inherently unpredictable, however the definition of unpredictability used allows for the system to be approximated. The question remains as to how accurate the approximation needs to be. It is an intractable problem to try to predict a robot swarm exactly, but if we only need to approximate its behaviour to prove a hypothesis

then such level of detail is unnecessary. Making predictions about complex systems is a useful endeavour but there will always be some degree of error and unreliability in this prediction. For example, predicting the weather is useful so that we can foresee disasters like hurricanes and floods. Predicting the stock market is useful so that we can make informed decisions about which companies to invest in. In neither of these examples will our predictions ever be wholly accurate, but they are usually accurate enough to be of use.

2.3.1 Complexity and Evolution

In [57], [83] and [84] the observation is made that if a system is complex, it will evolve faster than if it is not. It is certainly true that complex systems are everywhere in nature, take any cellular organism as an example, so there must be some reason that these organisms have prospered. This relationship between complexity and evolution is examined in depth in [84], and Simon gives two major reasons as to why complexity speeds up evolution. The first is an idea shared by the other two publications that *"complex systems will evolve from simple systems much more rapidly if there are stable intermediate forms than if there are not"* [83]. Note that stability in this context means that a system is able to maintain some internal state which allows the system to continue functioning, despite external perturbation [70] and that it can do so independently from within its natural environment.

This point is illustrated in [83] and [84] with an analogy of two watchmakers. These watchmakers are both making watches out of 10,000 basic parts and they get interrupted by phone calls, at which point they have to stop what they're working on and it falls apart. On average they can put together 150 parts before a phone call interrupts them. The first watchmaker tries to put all 10,000 pieces together at once and never gets anything done. The second watchmaker is able to build stable components of 100 pieces which do not fall apart when left alone; it is this ability that allows them to eventually finish a watch. Applying this analogy to a real system we could equate watch pieces to atoms; when these are combined together proteins can be formed. These are, for the most part, stable substances which can be used to build a cell. The sheer number of stable, single cell organisms (bacteria for instance) is evidence that a cell on its own can be a stable system. Cells can then be used to form multicellular organisms.

The overall idea is that stable systems can form the building blocks of new, aggregate, complex systems. This idea is connected to that of autopoiesis, defined as self-repair and self-regeneration, with reproduction being a by-product of this process [57]. If stable complex systems can self-repair and reproduce then they are likely to be more prevalent and so are better able to come together to form something new, because there are more of those building blocks available.

The second theory from [84] as to how complexity speeds up evolution is to do with functional equivalence and something Simon calls "Loose Horizontal Coupling". Within a complex system there is coupling between hierarchic layers in that a higher level system will contain things in the lower level; this is a "vertical coupling". Similarly communication and interactions between systems on the same hierarchic level is called "horizontal coupling". In this work, we refer to "loose horizontal coupling" simply as "loose coupling".

If two subsystems on the same level interact with each other in a fixed manner then, as long as they maintain this interaction, each subsystem is able to evolve independently of the other. These changes may affect the higher level system and this is what would

direct the evolution. For example, in the human body there is a digestive system and a circulatory system; the digestive system breaks down food and puts it somewhere where it can be absorbed into the bloodstream. If the circulatory system were to route blood more efficiently then as long as it still absorbs food from the digestive system these changes would not affect the functionality of the digestive system, however the whole body would be an improvement upon its predecessor.

The ideas presented in this section are not limited to biology; take the development of computers. A computer is made of stable subsystems which are used as building blocks to create an entire computer: Motherboard, RAM, hard drives, processors and so on. Within each of these subsystems there are yet more stable subsystems: registers, memory banks, multiplexers; and within each of these are more stable subsystems such as transistors and logic gates. Each of these systems, on all levels, have a fixed way of interacting with each other. This means that is has been possible to develop each part independently from the rest, improving factors like efficiency, speed and cost in one area and improving the computer as a whole. Following the arguments outlined in this section, it is therefore not unreasonable to attribute the improvements in computer hardware at least in part to the fact that modern computers are a complex hierarchic system, constructed of stable subsystems with loose coupling between subsystems on each level of complexity. The use of this structure is probably unintentional but comes about from the simplicity and convenience with which development can be done in this way.

2.4 Modelling

Modelling is an important tool for studying complex systems, *"A model is an abstraction that is made to aid understanding or description of something"* [72]. Replicating an entire complex system is an intractable problem, so it is necessary to abstract only the information that is relevant and create from this a model of the system. This leaves the question of what the relevant information is. If we build a model of an ant colony to understand ant foraging it would be unnecessary to model every atom in every cell of every ant; these things would have no discernible effect on the overall behaviour of the colony. If we were looking at how an ant digests food then modelling the cells in the ant might be useful. The level of detail needed in a model is heavily dependent on the system being modelled and the goals of the model.

As well as modelling the complex system itself we must also incorporate into the model an abstraction of the environment in which it is found, in order to give the model context. A model of white blood cells in the immune system will need to incorporate some abstraction of a blood vessel, a model of a swarm of robots will need to include the environment the swarm will be deployed into. At the very least *"modelling the environment implies identifying its basic features, the resources that can be found in the environment, and the way via which agents can interact with it"* [66]. Again, when modelling the environment, the level of detail necessary is dependent on the complex system and the modelling goals. A complex system without context is useless, and a model of a complex system without its environment is just as useless:

> The behaviour of a complex system depends critically on the way that the (collective) components interact with their environment over time; failure of models to adequately model the environmental context naturally leads to non-realistic models of the complex system. [4]

We now examine two examples of modelling in the scientific literature and look at the level of abstraction used in each case.

Deneubourg et. al. [22] create a model of ants foraging for food. In their model the environment is a grid of cells; each cell has a given probability of containing food. The "ants" all start in the same cell, the nest, and move around the grid collecting food as they go. When an ant moves into a cell looking for food it deposits one unit of "pheromone", as it returns to the nest with food it will deposit ten units of pheromone in each cell it passes through. The ants are more likely to move into a cell if there is more pheromone in it. With this very abstract representation of the environment, the ants, and the ants' interaction with the environment (the pheromone) Deneubourg et. al. are able to realistically demonstrate that changing the environment (the probability of a cell containing food) had a significant effect on the foraging pattern of the ant colony.

Lerman et. al. [54] model a swarm of robots where each member of the swarm autonomously switches between collecting red pucks or green pucks depending on how many pucks it remembers seeing in the environment and how many robots it has seen doing each task. The aim of this research was to compare a differential equation model with the real life swarm of robots developed in [42]. The model was used to predict what proportion of the robots would be assigned to each task as the experiment progressed, this same metric was used by Jones and Matarić to assess their robots [42]. Lerman et. al. modelled the swarm's behaviour by approximating the average swarm robot's memory, and generalising this over every robot in the swarm. This model used as a parameter the number of robots and the proportion of red to green pucks in the environment. By modelling the average memory, the aggregate swarm and the proportion of pucks, they used probabilistic differential equations to approximate the robots' actions over time they were able to accurately replicate the results from [42].

Both the above examples of modelling show that even very abstract models can be used to make accurate predictions or representations of the original complex system. In each case the model contains only the relevant information, thus keeping it as simple as possible and easier to understand. Deneubourg et. al. used their model to inform later experiments [30], whilst Lerman et. al. compared their results to the real complex simulation to argue the validity of their model. The demands for model accuracy were much greater for Lerman et. al. [54] and there is a corresponding increase in the level of detail. Consequently, it can be seen that there is a trade-off between abstraction and model accuracy. Both of these examples go to an appropriate level of detail for the measurements that are required of the model and so the results of the modelling are accurate enough for the authors' needs without being overly complicated or complex.

2.4.1 The Advantages of Modelling

There are several advantages to modelling a complex system. We have already discussed how models are an abstraction of the real system (the *domain*). This abstraction has the advantage of simplifying the domain down to something we can understand, making it easier to study without irrelevant details obscuring the results. A model can also be transformed into a simulation, allowing us to view the model as it changes over time and thus capturing the dynamic aspects of the domain [4]. It is for this reason that modelling and simulation are a useful and necessary tool in complex systems research.

For this project we use modelling and simulation of robots and robot swarms to test hypotheses and to inform us as to which control rules or methods work best. Running a

simulation of a robot swarm is useful because it is not as costly as using a swarm of real robots, and a simulation can be extended to include vast numbers of robots, whereas with an embodied simulation we are limited to the number of robots on hand. The time taken to set up a real swarm is also considerably longer than the time it takes to run a simulation, but the simulation itself may take weeks to be perfected whilst robots are able to run any code as soon as it is written. Robot simulators do exist though, so any simulation for this project would not need to be written from scratch. For this project we use a simulation platform called Player/Stage. Vaughan [92] gives a comprehensive comparison between Player/Stage and its rival robot simulators. Player/Stage has the advantage that you can run the same code on a simulated robot as you can run on a real robot, with no modifications. The only extra work involved in simulating a robot compared to using a real robot is that the environment must be modelled, although Player/Stage has an API for doing this.

Regardless of which simulation package is used in this project, it must always be remembered that a simulation is never going to be a completely accurate representation of the real-life experiment.

2.4.2 Forms of Modelling

The two most common forms of model are Equation Based Model (EBM) and Agent Based Model (ABM). Correspondingly, there are also two scales of view that a model can take; macroscopic view or a microscopic view. The macroscopic view is when the model views the whole system at once, only the overall behaviours and global trends are modelled. The microscopic view is when the subsystems of the overall complex system are modelled, including rules that govern their interactions and behaviours [99].

The nature of EBMs is fairly self explanatory: they are models which use equations to characterise how variables and parameters within the domain change over time. *"EBMs represent the system as a set of equations that relate observables to one another"* [68]. Typically these are differential equations and take the macroscopic view of the domain. In our previous example of the model built by Lerman et. al. [54], they built a model of a robot swarm using probabilistic differential equations. These were based on approximations and generalisations of the swarm's behaviour and could predict what the proportion of the swarm would be collecting red pucks at the next time step.

To describe ABMs we must first explain what an agent is. [41] identifies two major characteristics of an agent, autonomy and situatedness:

- **Autonomy** An agent is an entity which is distinct from the environment in that there is a well defined boundary between the agent and its environment.
- **Situatedness** They function in a particular environment where they have the ability to monitor and affect aspects of this environment.

Zambonelli [99] also identifies the characteristic of *sociality*, which only applies to systems of multiple agents (Multi-Agent Systems). Agents can directly interact with each other and cooperate to achieve their individual goals, possibly through some medium of communication. An agent may also be proactive or reactive. A proactive agent has its own goals and drives which maybe completely separate from the system and is proactive about achieving these goals. A reactive agent is dormant until it is stimulated in some way, at which point it will have a reaction which may involve activating or being activated by other agents in the system. Zambonelli [99] gives a corporate

marketplace as an example of a system of proactive agents; each corporation may be seen as an agent because it is a distinct, autonomous entity within the marketplace which interacts with other corporations (agents) to achieve the goal of earning itself more money. An example of a system of reactive agents may be a flock of birds. The birds in the flock are reacting to stimuli from other birds and try to maintain distance between each other whilst still remaining close enough to be part of the flock.

With an understanding of what an agent is, we can use this to illustrate the difference between macroscopic and microscopic models. In a microscopic model each subsystem is modelled as an agent; with a macroscopic model the entire system is one single agent incorporating both the complex system and the environment it is situated in [97]. An ABM takes the microscopic view. In these models each agent is programmed with its own behavioural rules. Multiple agents are placed in a simulated environment, and they begin to interact with each other and the environment. The overall behaviour of the swarm is typically measured and evaluated as an output of this kind of model. In the example model by Deneubourg et. al. [22] each ant is an agent. They created one ant agent which decides which location cell to move into by assessing the level of pheromone in the cell, then placed multiple copies of this agent into their environment. The ants interact with their environment by laying pheromone in a location cell and by collecting food and returning it to their nest. The ant agents don't directly interact with each other: their foraging trails are an emergent property of the pheromone laying and following behaviours. In this example the overall behaviour of the ant colony was assessed by looking at pictures of the agent's "foraging patterns".

2.4.3 Validation and Verification

The question of validation and verification is an important one. How should we know when the system is modelled accurately and what do we need to measure to prove this? These questions apply to both equation and agent based models and unfortunately, the answers are extremely task dependent. A simple model to generate ant trails requires much less testing and assurance than, for example, a simulation of a robot swarm for deployment in space. A higher degree of confidence and trust is needed from the latter model and as a consequence the model needs to be more detailed. A greater level of complexity may be required to accommodate this need, and the spatial and temporal scopes of the model may also be increased. However, there is a distinctly non-linear relationship in the value of the model when compared to the cost of building it. When the value of the model is low, small investments in resources yield a large gain in value, but we get diminishing returns from our investments as the value of the model increases [80]. It is therefore important to know *before building the model* how much confidence you wish to place in the model and at what point the model becomes "good enough". Without this precaution resources may well be wasted in giving unnecessary detail to a model which will never be needed. We would further run the risk of making our model as intractable as the original domain, since it is the process of abstraction and simplification which allows models to be studied in preference of the original [4]. In addition to the model's level of detail we must also decide *before* building the model what variables we want to be able to feed into the model and what the outputs that we are going to measure will be. Choosing how to represent this data in the model, and what this data will even be, is the most difficult part of the modelling process [62]. The metrics we choose must be relevant and should capture the information we want from the model. For example, in a model of bird flocking, we can measure the number

of bird-bird collisions or the average bird speed but these things do not fully capture how well the birds are flocking. There are so many variables that can be measured that a great deal of thought needs to be put into which aspects of the model should be measured in order to best characterise the output, and the appropriateness of the chosen measures depends on what the model is to be used for.

To try to address these problems with model building, several model development processes have been proposed. These are aimed at helping the modeller to create pragmatic, measurable and reliable models by ensuring that they consider every aspect of the model before building it so that the final model is fully understood and scientifically valid[1]. An influential modelling and verification process is the one developed by Sargent [80], shown in figure 2.1. This process dictates modelling artefacts that should be produced and the relationships between them. Although it is not specifically designed as a process for modelling complex systems, Sargent's process can certainly be applied to their modelling. There are three major artefacts associated with this process:

1. **Problem Entity**. This is the system to be modelled. It includes every aspect of the original system including any agents that may be present and the relevant aspects of the environment in which the system exists.

2. **Conceptual Model**. This is a representation of what the modeller thinks is happening within the Problem Entity. This is built with the goals of the simulation in mind and should contain information and other aspects which are relevant to the data being measured and put in.

3. **Computerised Model**. This is the Conceptual Model which has been implemented as a simulation on a computer.

The most important aspect of Sargent's process is the validation and verification steps between the creation of each artefact. The conceptual model is created by a process of "analysis and modelling", but it is validated against the problem entity. The conceptual model validity checks for two things: that the conceptual model is a correct representation of the problem entity, and that the model is *"reasonable for the intended purpose of the model"* [80]. Similarly, the computerised model is validated, during its construction, against the conceptual model to ensure that it is a correct implementation and that no errors or differences between the two artefacts have been produced. Finally through experimentation the computerised model is compared to the problem entity. A particular scenario is presented to both the simulation and the problem entity and the reactions of both systems are measured and compared. If the conceptual model and the computerised model have been validated properly then any differences between the simulation and the real system are due to incorrectly abstracting the problem entity into the conceptual model or from some unseen errors introduced in the implementation of the model.

It can be seen that the Sargent process draws heavily from the software engineering lifecycle of specification, design, implementation and then testing. The specification stage is taken from the problem entity, essentially "Create a model that replicates this". The design phase is the building of the conceptual model, as this details the features that the simulation has and what it does. Next the design is implemented into something that can be executed on a computer and this program is tested against the specification

[1] *"Scientific validity, like engineering validity, means that it must be possible to demonstrate, with evidence, how models express the scientific realities"* [4].

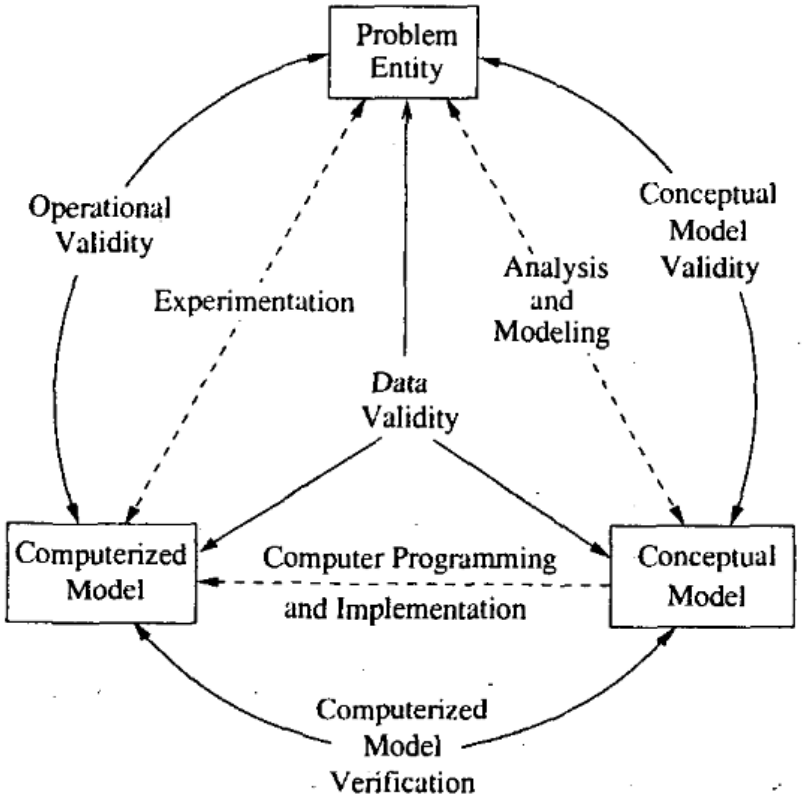


Figure 2.1: The Sargent process for building models [80].

to make sure that it fulfils the requirements i.e. that the computerised model replicates the problem entity. As with software engineering, errors in the implementation means that it is necessary to re-do the implementation or design until errors are removed and the specification is fully met.

DeWolf and Holvoet [21], describe an alternative modelling process specifically for multi-agent systems, which was similarly inspired by the software engineering lifecycle. However, instead of abstracting the lifecycle into a series of inter-related artefacts to produce, they instead leave the lifecycle unchanged, but specify issues that should be addressed at each stage. For example: making sure that a multi-agent system is the best solution for the task and developing suitable metrics of success. They stress that when creating multi-agent systems, the usual goal is to develop a global behaviour, and consequently the modeller must be able to give guarantees about the macroscopic behaviour.

In both modelling processes, the need for verification between tasks is emphasised. Each stage must be thoroughly compared with the last to make sure that all the requirements are fulfilled, which also uncovers any errors in the previous stage. If it is found, for example, that a particular part of the design is unimplementable then the design must be rectified; when the design is altered it then needs to be verified that it still fulfils the requirements of the specification. In both cases the entire process is iterative: the process may need to be repeated and the model revised many times before it is deemed sufficiently accurate. Paunovski et. al. [69] makes the point that with these multiple iterations through the modelling process you gradually build up confidence in the model, iteratively making it more and more correct a representation of the original system because of the repeated verification and validation steps applied to the model. They argue that with complex systems this is the only way to create a true and valued model, because *"it is infeasible to formally verify a complex multi-agent system with stochastic interactions"* [69].

2.5 Conclusion

In this chapter we survey the scientific literature and define complex systems for the purpose of our work. This definition covers swarm intelligent and swarm robotic systems discussed in chapter 3. In section 2.3.1 we describe work by Simon [84] which observes that if a system is complex, then it consists of stable subsystems which are loosely coupled. It is these two features which allow complex systems to evolve more easily than non-complex ones, because the subsystems are stable and independent but can still interact to produce emergent behaviours. In chapter 4 these observations are extended to swarm robotic systems to create the hypothesis of this work.

Complex systems are studied by "modelling": a process of abstracting away any unnecessary parts or detail so that the systems is easier to understand or test. Validation and verification is important part of the modelling process if the results from the model are to be extrapolated to also be true for the real system. In chapter 6 we build a model of a swarm robotic system following the CoSMoS process [6], and in chapter 8 the model is calibrated against robot hardware developed for this work, so that the model is more accurate.

Chapter 3

Swarm Intelligence

3.1 Introduction

In the previous chapter, we established the idea of complex systems, what they are, and why and how they are studied. In this chapter we aim to establish that swarm intelligent systems and swarm robotic systems both fall into the general category of complex systems, and to give background on why these particular complex systems are worth studying. From the scientific literature presented in this chapter and chapter 2 we aim to provide the context necessary to understand the hypothesis of this research.

In section 3.2 we define swarm intelligence for our research and describe the motivations for studying swarm intelligence. Then in section 3.3 we describe swarm robotics, the challenges associated with swarm robotics research, and what benefits can be gained from studying swarm robotics. Finally in section 3.4 we discuss evolutionary swarm robotics as an answer to some of the challenges of swarm robotics.

3.2 What is Swarm Intelligence?

The inspiration for swarm intelligence research comes from biological systems which display collective behaviour [14]. Examples of such swarm intelligent behaviour include ants foraging for food, termites building nests or birds flocking together. In each case a large number of individuals are cooperating to achieve some higher goal; whether it is food, shelter, safety or something else. Each individual benefits from this group behaviour and each individual contributes in some way. If we take our definition of a complex system, from page 11, we can see that swarm intelligent biological systems fit into this category. There is a hierarchic structure to these systems, with many individuals interacting to form a larger system. There is emergent behaviour in that we cannot fully know how the swarm will behave from examining its constituent parts, there is order in the behaviour such as ants forming lines, thermal regulation in termite nests or birds keeping together, but there is randomness in that they may appear undirected.

The phrase "swarm intelligence" was first introduced by Beni [11] to describe Cellular Robotic Systems (CRS) [13] [14] [79]. These systems, are heavily influenced by cellular automata (see section 2.2) but are constructed using embodied robots instead of cells. A CRS is a conceptual robot swarm that has the ability to *"encode information as patterns of its own structural units"* [11], and this leads to *self-organising* behaviour

because the robots are able to arrange themselves into meaningful patterns without any centralised means of coordination, only local interactions. The key features of a CRS are *"decentralised control, lack of synchronicity, simple and (quasi) identical members"* [13]. As mentioned earlier these are also features of natural systems displaying swarm behaviours, and consequently the concept of cellular robotic systems was later renamed to "swarm intelligence" to reflect this similarity [13].

It should be noted that the definition of swarm intelligence derived from CRSs applies only to robotic systems. Bonabeau et. al. later extended this to include *"any attempt to design algorithms or distributed problem-solving devices inspired by the collective behaviour of social insect colonies or other animal societies"* [14]. This definition generalises swarm intelligence to cover any artificial system using decentralised coordination of many simple agents. However, it noticeably rejects the natural systems from which the inspiration was originally drawn, despite the fact that the biological systems fulfil the criteria of decentralisation, asynchrony and near-homogeneity. Furthermore, it would be hard to imagine any definition for intelligence that would apply to artificial swarm systems but not natural ones.

To address this shortcoming Dorigo and Birattari present an alternative definition for swarm intelligence:

> Swarm intelligence is the discipline that deals with natural and artificial systems composed of many individuals that coordinate using decentralised control and self-organisation. In particular, the discipline focuses on the collective behaviours that result from the local interactions of the individuals with each other and with their environment. [25]

It should be noted that this definition now includes heterogeneous systems, as homogeneity (or near-homogeneity) is a common feature of a swarm but not essential to successful group coordination [87]. Another concept introduced in this definition is that of *self-organisation*, that is *"a process in which patterns at the global level of a system emerge solely from numerous interactions among the lower level components of the system"* [18]. Which, put another way, is an emergent property of the system which produces some ordered behaviour. Self-organisation has been shown to be caused by a balance of positive and negative feedback upon the local interactions of the swarm agents [14]. An example of this is a predator-prey cycle: if there are large populations of prey then they will breed to produce more prey, thus there is positive feedback on the prey population because it is self-reinforcing behaviour. However, higher numbers of prey means that there is more food available for the predators, causing an increase in the predator population. This counteracts the positive feedback upon the prey population and acts as negative feedback. Consequently the balance of positive and negative feedback causes cyclical fluctuations in the predator and prey populations.

Additionally, Millonas [60] identifies swarm intelligence systems as having the following characteristics:

- **Proximity** This means that a swarm should be able to perform *"elementary space and time computations"* on local information. This is most often done to measure energy expenditure and the results of future actions may be assessed in these terms. This is similar to Beni's point that a CRS is able "process matter" [11], the swarm observes an object and acts in consequence to these observations.

- **Quality** A swarm should be able to assess the quality of items and locations, for example with respect to safeness or food quantity.

- **Diverse Response** Resources are not just invested in a few potential responses, they are allocated to prepare for many eventualities. Hence, the swarm has a diverse response to inputs.
- **Stability** A swarm should be able to maintain its behaviours despite minor changes in its environment.
- **Adaptability** A swarm can adapt to major changes in the environment and will try to minimise the energy loss when adapting to these changes. The swarm will need to be able to tell when to stay stable and when to adapt.

Millonas states this is not a definitive list and may well be the result of swarm behaviours having evolved [60]. For instance, it is highly plausible that robustness to short-term environmental variations (such as a particularly cold day), and adaption to long-term variations (such as winter) is desirable and would help that particular group of animals to survive in preference of less evolutionarily fit groups.

For this work we use the Dorigo and Birattari definition of swarm intelligence [25], but extend the definition to include the following properties:

- **Multiple Agents** Each member of the swarm is an agent and can be modelled as such. This means that each swarm member has the properties of autonomy, situatedness and sociality, as outlined in section 2.4.2.
- **Asynchrony** This property is a consequence of decentralised control, as there is no global controller and hence no global clock. All agents in the system, consequently, are also asynchronous.
- **Proximity** Each swarm member can observe and potentially analyse items or other agents in its locality. This will usually prompt some decision making process as the member decides what the next course of action should be.
- **Stability** Robustness to minor perturbations, for example losing a small percentage of the swarm members.
- **Adaptability** The ability to withstand large perturbations and to change the swarm behaviour and goals accordingly.

These characteristics have been chosen as defining swarm intelligence for this work because they are displayed by both swarm robotic systems and the natural systems that are used as biological inspiration for swarm robotics.

3.2.1 Motivations Behind Swarm Intelligence Research

There are several aspects that make swarm intelligence a worthwhile subject of research. The most important of these is the idea of "coordination without control" that is, mass coordination and cooperation of multiple agents without a centralised control entity giving commands to each agent. The problem of centrally coordinating a group of agents is not scalable: as more agents are added to a system the demands on such a controller for bandwidth and communication of instructions become increasingly large and cannot be met. Such a controller may easily cope with small numbers of agents, but for a hundred or a thousand the problem is intractable. This is why natural swarming systems are of interest; an ant colony manages to coordinate millions of ants [38] using decentralised methods. Each ant decides its actions based on what it observes and what

it can remember, effectively using local information to create an emergent, collective global behaviour.

One positive side-effect of this decentralised control is that the natural swarm is robust to changes and failures within itself and its operating environment. The death of a single ant does not affect the ant colony, and the depletion of a particular food source does not prevent the colony from finding new ones. Natural swarm systems are highly redundant, and this redundancy is, to some extent, build into the control mechanisms of the swarm. That is, there are many similar (near-homogeneous) agents doing very similar and repetitive tasks. These tasks may often be quite simple, for instance when ants cluster dead bodies they simply move them from areas where there is a low dead body density to where there is a high density [23]. This can be reduced to a set of simple, unintelligent *IF... THEN... ELSE...* rules for the ant to follow. The swarm agent may therefore be seen as being inherently simple, but capable of complex emergent behaviours that have come from this simplicity. Simon offers an explanation of how this is possible:

> Complex behaviour need not necessarily be a product of an extremely complex system. Rather, complex behaviour may simply be the reflection of a complex environment [82].

Simon's idea offers the possibility that natural, swarm intelligent systems have evolved to take advantage of the complexity in their environment to help them achieve tasks. Hence, instead of each ant having to invest time in learning how to do the task, it instead only needs to have a simple set of instructions and the interactions between the environment and the ant cause the task to be completed.

As a subject of research, swarm intelligence can give us clues as to how to achieve robust, decentralised control. We can also learn about how to create unintelligent, simple agents that can exploit their complex, real-world environment to produce useful behaviours.

3.3 Swarm Robotics

The subject of swarm robotics follows closely on from swarm intelligence. It has been defined by Şahin as:

> The study of how large number of relatively simple physically embodied agents can be designed such that a desired collective behaviour emerges from the local interactions among agents and between the agents and the environment [78]

"Relatively simple" means that an individual robot is incapable of performing a particular task, but the same task is achievable through the collaboration of multiple robots. It can be seen from this definition that swarm robotics and swarm intelligence are very closely related subjects; they both concern the mass collaboration of simple agents to achieve some global goal and the investigation of how such emergent behaviour is possible. In actuality, a swarm robotic system is a form of artificial swarm intelligence, as such a swarm robotic system displays the characteristics of a swarm intelligent system: decentralised control, self-organisation, asynchrony, local interactions, stability and adaptability.

In swarm intelligence research the biological system is modelled and simulated to get a better understanding of the mechanisms behind the swarming behaviour. Comparatively, swarm robotics can be seen as the modelling and simulation of swarm intelligence theories using embodied agents, by designing the agents in the system on a microscopic level to try to produce macroscopic emergent behaviours. However, the goal of swarm robotics is not always to accurately copy the inspiration but to achieve the emergent properties of the inspiration for our own benefit. For example, it would be worthwhile for us to be able to make robots that can bridge a gap or collectively push heavy items from one place to another as the Weaver ant does [38, 49]. In both these examples the biological system is used as inspiration for the programming and behaviour of the robots. The methodologies evolved by the ants are heavily abstracted to fit the real-world application which the swarm robots will be used in. The swarm robots may still be seen to be obeying the same rules as the ants and as such are modelling the ants. However, unlike conventional modelling as described in section 2.4.2 what is important is the swarm's ability to perform the task. As long as the task is adequately performed, how closely the robots match the biological system is irrelevant.

> An engineer with a problem to solve does not have to be concerned with the biological plausibility: efficiency, flexibility, robustness and cost are possible criteria that an engineer could use. [14]

In our work, swarm robotics has two purposes: The first is to learn more about emergent swarm behaviours through attempts to replicate them, the second purpose is to engineer a multi-robot system that achieves some beneficial task through swarming behaviour.

3.3.1 Benefits of Swarm Robotics

Swarm Robotics as a Modelling Tool

The advantages of modelling have already been discussed in section 2.4.1. As a simulation tool, swarm robots have the advantage that they are an embodied simulation platform. The advantage of using an embodied simulation over a computerised one is that on a computer it is hard to accurately emulate real-world physics. Simulating something like a ball being knocked across a testing arena would be effortless with an embodied system, but in a computerised simulation factors such as the ball's physical properties, the friction and gradient of the surface and the strength and direction of the push need to either be measured or approximated. Furthermore the resulting path of the computerised ball would not be completely accurate or reliable. We can never fully know all the factors which affect the ball, much like we can never fully know all the factors affecting the agents in a natural swarm (this is properly discussed in section 2.4), but at least with an embodied simulation we do not need to concern ourselves with modelling the physics of the environment. By placing our simulation within the real rather than approximated world, the physics is modelled for us.

Another advantage of embodied simulation is that the real-world is far richer, in terms of information content, than a computer simulation. In a physically homogeneous set of robots there will be different imperfections within each individual robot. For example actuators may not respond in exactly the same way to a command on one robot when compared to another robot. Similarly there may be noise in the sensor readings, a proximity sensor could easily give two different range measurements for the same distance. In computer simulation these differences can be approximated by adding noise, on the sensors and on the motors. An infra-red proximity, for example,

has a minimum distance, any objects that are closer than this distance will result in wildly inaccurate readings. Objects within the minimum and maximum ranges will give more accurate, but still noisy, readings. It is not enough to model these sensors with a constant noise level, and we cannot assume that items will not get closer to the sensor than its minimum range. Using embodied robots we already have noisy sensors and robots, with a computer simulation there needs to be careful consideration and testing to create a realistic and reliable model.

In addition to the richness of the robot there is richness in the environment which we place our robots in. We have already discussed this richness in terms of the world's physics, but there is also a wealth of information in the real world which the robot needs to be able to filter out. If a robot with a camera is tested in a messy room, there is much more information presented to the camera than if the room was empty and painted completely white. The robot must work harder to extract only the relevant information from its camera; this can be both good and bad. The benefit is that the robot is much more robust to sensor noise and consequently can better function in a wider range of environments; the downside is that this extra functionality is considerably more difficult to create.

Brooks summarises the advantages of embodiment over simulation in the phrase:

> The world is its own best model. [15].

Swarm Robotics as an Application

Beyond the realm of modelling, swarm robotics is an attempt to create robotic systems that have the advantages of natural swarm intelligent systems (as outlined in section 3.2.1), and are capable of achieving some collective task in a robust, flexible and scalable way [78].

Robust Robust swarm behaviour is a result of decentralised control: the robot swarm has no global controller to be a single point of failure for the entire swarm. Additionally the robot members of the swarm are simple, both mechanically and functionally, so there is less potential for failure. The robot agents are also performing similar behaviours with high redundancy and so the loss of one swarm member should not affect the functionality of the swarm as a whole. Trianni [89] makes the point that decentralisation and redundancy must both be present for a robust collaborative behaviour. Trianni uses the example of a factory production chain: there is decentralised control in that each robot does the same simple task repeatedly, but if one robot were to fail the entire production chain would fail because their tasks are all different and rely on one another to be completed. There is no redundancy. Winfield and Nembrini [95] give three more desirable features of a robust robotic swarm:

- It should be tolerant to noise and uncertainties in the operational environment.
- It should be tolerant to the failure of one or more robots without compromising the desired overall swarm behaviours.
- It should be tolerant to individual robots who fail in such a way as to thwart the overall desired swarm behaviour.

These criteria link back to the properties of stability and adaptability that are displayed by natural swarm intelligence systems. A truly robust swarm of robots should be

able to maintain its functionality despite failings within the swarm or changes in the environment outside the swarm. However, Winfield and Nembrini [95] have shown that not all robotic swarms are robust to failures. In their paper they show a swarm performing an aggregation and taxis task, where all the members of the swarm come together and move away from a beacon. Aggregation is achieved by each robot using short distance wireless link to broadcast its presence, and trying to maintain the number of robots within wireless range above some threshold β. To achieve taxis the robots have a sensor which detects when a beacon is shining upon them. When a beacon is detected the robot increases β to ∞ so that there is differential movement in the swarm, leading to taxis behaviour. Then Winfield and Nembrini partially failed a small number of robots in the swarm, testing the effects of failing the motors, wireless communication, beacon sensor, proximity sensors, control system and finally the entire robot. They found that the robot swarm was robust to failure of the beacon sensor and to the entire robot failing. Losing the wireless connectivity of a robot caused it to become lost but the swarm was also able to withstand the change. Losing the proximity sensors caused collisions but was also withstood. A control system failure was simulated by the motors becoming stuck either moving forwards or turning on the spot. The former caused the robot to become lost, an effect the swarm was able to adapt to. The robot becoming stuck turning on the spot caused the same effect as failing the motors. The robot would become fixed in one spot but it would continue to wirelessly communicate with other robots and would consequently "anchor" the whole to the spot where the robot was stuck, thereby causing the entire swarm to fail in its task.

What this study ultimately shows is that a robot swarm can be robust to some major failures; the loss of an entire robot agent is easily adapted to, and this is a considerable advantage of swarm robotics. However, for a truly robust swarm it is important to design the swarm behaviour so that it can withstand partial failures and to thoroughly test the swarm's reliability under such circumstances.

Flexible Şahin [78] describes flexibility as another advantage of swarm robotics. He defines this as being a swarm's ability to coordinate its behaviour to perform different tasks [79]. Just as an ant colony is able to distribute agents between the tasks of foraging, cooperative transport, fighting off attackers and so on, so too must a swarm robotic system be able to perform different tasks as is required by its environment. Trianni [89] also describes flexibility as being a characteristic of a swarm robotic system, however, unlike Şahin, Trianni defines flexibility as being adaptive to environmental changes instead of a diversity of tasks performed. Both definitions are similar though, in that they both require an appropriate response to the environment from the robot swarm. The amount of different behaviours required from the swarm is perhaps therefore a reflection of the complicatedness of the environment, a more difficult situation requiring a larger repertoire of responses from the swarm.

Scalable The final major benefit of swarm robotic systems is that they are scalable. This means that it is possible to add or remove robots from the swarm and it will still continue function. Natural swarm systems can coordinate millions of different agents, so it should also be possible for artificial swarm systems to coordinate millions of robots. This is a beneficial characteristic because a greater number of robot agents means that the swarm has more resources available to complete its task and it has a higher redundancy and so is better able to cope with the loss of agents.

A study by Winfield, Liu and Bjerknes [96], which extends the work of Winfield

and Nembrini [95], shows that scalability is not always a guaranteed benefit of swarm robotics. Winfield and Nembrini [95] show that a swarm can be fragile to partial faults. The work by Winfield et. al. not only confirmed this result but also found that increasing the number of robots in the swarm does not fix the problem but, in fact, makes it less robust: a larger swarm was observed to take longer to recover from a partial fault, and an increased number of robots were lost during the recovery. Consequently, scalability may not be as straightforward as adding more robots to the swarm. Careful consideration needs to be given to the process of increasing the swarm size if the aim is to create truly scalable systems.

3.4 Evolutionary Swarm Robotics

The "Design Problem" is the question of how to engineer a behaviour at the agent level that produces an emergent behaviour at the global level.

> The challenge is given by the necessity to decompose the global behaviour that results in the desired organisation in[to] simple mechanisms and interactions among the system components. [89]

In swarm robotics the design problem presents a major hindrance when trying to engineer useful systems. One solution, proposed by Trianni [89], is *Evolutionary Swarm Robotics*. Evolutionary swarm robotics follows on from Nolfi and Floreano's work [65] in the area of evolutionary robotics, which itself is based on Genetic Algorithms. In this section we discuss genetic algorithms, how they are applied to evolving robots and the challenges associated with ESR.

3.4.1 Genetic Algorithms

A genetic algorithm (GA) is an algorithm that uses the principles of evolution, "survival of the fittest" to find an answer to a problem. The problem usually will have many possible solutions, which are scored with a "fitness function". The fitness function is some measurement or function that can assess the quality of possible solutions (genomes), allowing them to be compared against each other. The GA optimises a set of genomes to satisfy this fitness function as well as possible. Consequently, if we do not properly set the fitness function the GA will not give us the most appropriate solution. For simple problems with well defined metrics (such as finding the maximum point in a mathematical function) generating an appropriate fitness function is easy, but in robotics it is much harder to define what a good solution would be and which measures would adequately describe this definition. Often, assessing a robot's performance requires some evaluation of a behaviour and this can be a very qualitative thing. When selecting quantitative metrics for evaluating behaviour we must be careful to ensure that the measures used will adequately capture quality of the behaviour. For example, we might want a group of robots to flock together and we use the amount of cohesion in the group as our measure of fitness. However, if the robot flock was to bifurcate into two separate flocks this may be acceptable behaviour but our measure of fitness (the whole group's cohesion) would be much lower than it should be in this situation.

Assuming an appropriate fitness function has been designed, the GA can begin to generate fit genomes. A basic GA follows these steps:

1. An initial population of genomes are generated randomly.

2. Each genome is passed through the fitness function to measure its fitness.

3. The fittest few genomes are combined (crossover) and mutated to create a new population of genomes. This is called the next generation.

4. The old genomes are discarded and newest generation becomes the current generation.

5. The current generation is measured for fitness and the process repeats.

The search space of possible solutions to the fitness function may be very large. The GA creates new solutions that are similar to the previous best ones, increasing the range of solutions that have been investigated in a directed manner. The hope is that a solution that is similar to one that is known to be good, will also be good or better. So the best place to search for new solutions is around existing good ones. This process is analogous to natural selection or survival of the fittest in evolution. Only the organisms that are best able to survive in their environment are able to reproduce and create a new generation, these organisms pass on their genetic code (the genome) to their predecessors who are then submitted to the same survival process.

3.4.2 Implementing Evolutionary Swarm Robotics

In evolutionary swarm robotics (ESR) the controller for the robot agent is artificially evolved. To achieve this there needs to be a fitness function and some way for the GA to generate robot controllers. To do this, a robot controller needs to be in a form which can be described numerically by a sequence of numbers or a binary string. The sequence of numbers is the genome in the genetic algorithm, and each number in the sequence is a "gene".

At its most fundamental level, all a robot controller does is map between sensor data and actuator movements: a controller could be nothing more than direct links between sensors and motors. The designer could help the evolution process by abstracting the sensor data into a library of environmental cues such as "robot seen" or "item reached location", this way the controller does not need to evolve some means of evaluating its sensors. Similarly an abstraction of the motor movements into a library of different behaviours can be performed, meaning that the evolved robot controller is reduced to a simple mapping between sensory cues and a behaviour reaction. The choice of how to abstract the robot controller is a balance between the level of input the designer can put in and the amount of work required by the artificial evolution. The less the designer does the more work is needed during the artificial evolution, however decisions made by the designer may not be the most appropriate for the task.

The main disadvantage of ESR is the time frame of the evolutionary processes. Measuring the fitness of a single robot controller requires that the robot is controlled for some period of time, and that this should be long enough for the robot to build up a clear picture of how well it is functioning in the world. In a typical genetic algorithm there may be tens of solutions in the search population and the algorithm is run for hundreds of generations [61], consequently thousands of fitness function evaluations will need to be made. If each fitness function takes several minutes to evaluate, a whole GA will take hours or days to compute; this is likely to be longer than the battery life of the robot. Hence, the speed of the fitness function evaluation and of swarm robot evolution in general is another major hurdle for ESR to overcome.

3.5 Conclusion

In this chapter we define what we mean by "swarm intelligence" and argue that swarm intelligent systems are a subset of complex systems. Swarm intelligent systems, particularly biological ones, are of interest because they demonstrate that it is possible to mass coordinate of many thousands of agents without a centralised controller. The decentralisation gives the advantage of robustness and scalability, because adding or removing agents does not destroy the swarm behaviour.

Swarm robotics research aims to create engineered systems which display swarm intelligence, and so have the associated features of robustness and scalability. However, swarm intelligent systems are complex, so the swarm behaviour is *emergent* from the robots' actions and their reactions to the environment (see section 2.2.3). Consequently, it is difficult to design robot-level actions which manifest into the desired swarm-level behaviour.

ESR is one approach to solving this problem, by using a genetic algorithm to evolve robot controllers. The main disadvantage of ESR is that it is slow to evolve usable controllers because the robots need time to evaluate each controller. In the next chapter we propose a hypothesis for helping ESR systems to arrive at viable robot controllers more quickly by exploiting the "coupling" feature of complex systems described in section 2.3.1.

Chapter 4

Proposed Solution

4.1 Hypothesis

In section 2.3.1 we discuss the link between complex systems and evolution, and present Simon's theory that a complex system will evolve much faster than a system that is not complex [84]. Simon hypothesises that this connection is due to the hierarchic structure of a complex system having "stable intermediate forms", and "loose coupling", which is when systems on the same complexity level have a fixed way of interacting with each other and so are free to evolve independently for as long as this interaction is maintained. We then observe that loose coupling has been beneficial to the development of computers.

In chapter 3 we examine natural complex systems in which multiple animals cooperate in order to survive, and their artificial equivalents, swarm robotic systems. In swarm robotic systems the aim is to create robust, flexible and scalable systems that are capable of achieving some goal, however there is the problem of how to design a set of behaviours at the robot level to create the desired emergent behaviours at the global level. Finally in section 3.4 we present a solution to this problem: Evolutionary Swarm Robotics (ESR) [89], but observe that further research still needs to be done in order to make it faster and scalable.

In this chapter, we bring together these elements from previous chapters to propose a way of reducing the time required for ESR to find satisfactory robot controllers by asking:

> It is beneficial to evolve a robot swarm in a structured and loosely coupled way, in order to exploit the speed of evolution in complex systems as a tool for the development of swarm robotics?

4.1.1 Coupling Strength

This work has so far only mentioned loose coupling, but it is possible to have different strengths of coupling between systems [33, 94]. Weick [94] states that *"coupled events are responsive,* but [...] *each event also preserves its own identity and some evidence of its physical or logical separateness"* [94]. Glassman [33] describes coupling strength, or the degree of coupling as:

> The degree of coupling, or interaction, between two systems depends on the

> activity of the variables which they share. To the extent that two systems either have few variables in common or if the common variables are weak compared to other variables which influence the system, they are independent of each other.

In both quotes, the authors describe coupling as a measure of the independence or co-dependence of a subsystem, with a stronger coupling correlating with greater co-dependence of systems. If a system has a strong influence on another then they are in some way co-dependent, and the coupling is strong or "tight". If the systems are independent and have no influence on each other then they are uncoupled. Glassman [33] also argues that *"In the* [tightly coupled] *system, a perturbation in any one variable would require readjustment of all the other variables in the system."* [33], and as such tightly coupled systems are faster to react to changes in the environment, but they are less adaptive to changes because the breakdown of one system will propagate to other systems easily. Conversely, loose coupled systems are more adaptive because of their independence, but also because they are less homogeneous and so have a wider variety of solutions to call upon in response to the perturbation. Loose coupled systems are more varied because of their independence, so they can evolve as an individual within a group, not as an entire group.

We apply coupling to ESR by defining the robot to robot communication within the swarm. If we define the way the robots interact with each other, then as long as the robots are able to maintain that interaction, they can interact with any other robot interchangeably. Although the robots have different internal controllers and evolve independently of each other. This is similar to the idea of interfaces in object oriented programming. As long as a particular class conforms to an interface then the containing class(es) can use it interchangeably with anything else using the same interface without having to change their internal code.

We hypothesise that loose coupling makes evolution faster in ESR. To test this we compare swarm evolution under different coupling strengths. We theorise that without coupling the robots will evolve useful solutions very slowly or not at all, since they will either have to evolve communication or learn to do the task without it. This will lead to slow evolution of solutions with low fitness. With loose coupling the robots will evolve useful solutions quickly because the robots are able to interact whilst being independent of each other, so evolutionary changes will not have a large impact on other robots. In a tightly coupled swarm it may be possible for the robots to evolve better solutions than loose coupled, because the swarm is more responsive to stimuli, but the tight coupled swarm is also more influenced by positive or negative evolutionary changes within each individual robot.

4.1.2 Alphabetisation

Simon introduces the idea of alphabetisation by stating:

> The flexibility of coupling among subsystems can be further enhanced by limiting the variety of different kinds of components that are incorporated into the larger system [84].

Alphabetisation is the idea that a message of information can be broken down into atomic, descriptive primitives. These descriptive primitives *"are more prolific and less specific compared to things that are composed of* [them]*"* [67], meaning that each primitive has a single, generalised meaning but only in combination with other primitives

can a full message expressing some specific information be created. On their own the primitives are not particularly meaningful, but it is the combination of primitives that conveys information. For example, there are 26 letters in the English alphabet, each representing distinct a sound. Individually a letter has little meaning, but by combining letters we can create words, and by combining words we can express ideas.

In section 4.1.1 we establish that we can apply coupling to the swarm by defining the robot-to-robot communication. We do this by giving coupled swarm robots a shared signal to meaning mapping that each robot conforms to. So, a signal from robot A will always have the same meaning as the same signal coming from robot B.

Alphabetisation is the method we use to vary the strength of the coupling between the robots. The messages that the robots want to send are broken down into the smallest number of primitives that, when combined, can express any message that the robots need to share. To use an example from later chapters; the robots want to share information about the size of something and the number of robots near it. The descriptive primitives used could be "small", "medium", "large", "empty", "half full" or "full". To loosely couple the robots, each primitive maps to a signal, and to send a complete message the signals are played sequentially. Several signals are required to express a complete message, but the signal set is small. In the tightly coupled swarm, each possible combination of primitives maps to a signal, so only one signal is required to express a complete message, but the signal set is larger. Therefore in the tightly coupled swarm, if one aspect of the message changes, the whole signal has to change to express the new message. With loose coupling, only the signal corresponding to the changed information would be different. The tight coupling is more susceptible to changes in the environment, and consequently the information expressed, but it is able to express information more quickly because each message only needs one signal.

4.2 Motivations

We believe our proposal to be a worthwhile subject of research because applying Simon's hypothesis about complexity and evolution to ESR will help to advance ESR research. As mentioned in section 3.4 ESR is hindered by the fact that the evolution is very slow, if we can show that Simon's hypothesis is effective in speeding up the evolution process in robotic systems, then our work will help to present evolution as a viable solution to the design problem. Practically, this means that it will be easier to create robust, flexible and scalable swarm robotic systems, and so swarm robotics is more likely to be developed and used in real-life situations; the benefits of which have already been discussed in section 3.2.1.

On a broader scientific level this work will benefit swarm intelligence research because we will be testing whether or not Simon's theory is correct in the context of robotic systems. The findings can then be extended to swarm intelligent systems to aid our understanding of how they came to be, giving us a deeper understanding of how swarm intelligent systems can achieve coordination without control and how to recreate this for ourselves.

Our work can also have practical applications outside complex systems research. If Simon's theory can be verified, then genetic algorithms could be improved to take advantage of stable subsystems and loose coupling to make them more efficient. Work has already been done in the area of using stable subsystems in GAs, such as in [76], [8] or [45]. However, we have not been able to find publications that demonstrate the

effects of loose coupling on evolution in genetic algorithms.

4.3 Conclusion

In this chapter we propose the hypothesis that a system of evolutionary swarm robots will evolve collaborative behaviours faster if the swarm consists of *independent* agents sharing *stable* information. It is the stability of the shared information which causes the evolution to arrive more quickly at a useful solution. Information passed between the robots is *stable* if the mapping between the intended meaning and the actual signal does not change over time, or if the mapping changes slowly compared to the rate of evolution. The robots in the swarm are then *coupled* if they all share the same signal to meaning mapping. We theorise that one way of making information *loosely* coupled is for it to conform to an alphabet of message primitives. Each item of the alphabet is a distinct signal expressing a very basic piece of information, and by combining these primitives a more descriptive message can be created and expressed. We aim to test whether this alphabetised method of communicating allows swarm behaviour to evolve more easily than coupling without alphabetisation, or communication without coupling.

In chapter 5 we outline an experiment to test the hypothesis and compare loose coupling to other strengths of coupling between robots.

Chapter 5

An Experiment to Test the Hypothesis

5.1 Introduction

In this chapter, we describe an experiment to test our hypothesis from chapter 4. In section 5.2 we propose a foraging task where the robots are able to communicate information about a food source using sound signals. In each swarm there is a different mapping between the signal or signals used, and the descriptive primitives the robot is expressing. In some of the swarms this mapping is evolved by the robots over the course of the evolutionary run, in the other swarms the mapping is shared between all robots and never changes throughout the evolutionary run. By using the different mappings we aim to change the coupling between the robots and test the comparative fitnesses of the evolved swarms. This is described in section 5.3. Finally in section 5.4 we review evolutionary swarm robotics in the literature and propose the strategy for the collective evolution that is used in this work.

5.2 The Task

In our experiment, the robots forage for "food". In the environment there is a random distribution of food sources, these can be of several different sizes. The robots must cooperate to collect from a food source, and at the end of the generation a robot's fitness is measured by the amount of "food" it collected.

Each food source has size n, and requires n robots to collect but gives n^2 units of food in return. When collected, these units of food are shared evenly between the robots that collect them, so that each robot receives n food units. Consequently a robot receives more units of food if it collects from a large food source, but this requires a longer investment of time as it will need to wait for help from other robots. The size of a food source slowly degrades over time, so that large sources does not stay in the environment forever due to being too large to collect. When the food source size reaches 0, either through degradation or because the food was collected, it is replenished after some time period.

This task is chosen to test our hypothesis because robot to robot communication is required to indicate that a robot is waiting for help from another; the robot must

indicate that they are waiting at a food item so that other robots will come and assist them. By sharing information about the food the robot is waiting at, the other robots can make a decision about whether or not to assist it. It is also possible for the robots to collect food without communicating with each other, by waiting until enough other robots decide to wait at the same food item or by only collecting small foods. Although it would not be rewarded as well as if the robots were effectively communicating and collaborating to collect the larger foods. Another advantage of this task is that an evolved behaviour can be evaluated using the amount of food the robot was able to collect. This metric is measurable by the robot, so the swarm does not need an external monitor to evaluate each robot's fitness, it can be done by the individual robot.

5.3 Creating an Alphabet

In our experiment, the robots use audio signals of differing frequencies to express information about the food sources they encounter. There are two pieces of information a robot can express about a food source: its size, and the amount of robots currently collecting from it (occupancy). For each piece of information there are a number of descriptive message primitives the robots can use to describe it. For example, the descriptive primitives for size information might be "small", "medium" or "large" and for occupancy information the descriptive primitives might be "unoccupied", "over 33% occupancy" or "over 66% occupancy".

In general, we can express these primitives mathematically as a range, where x is our observed value and D is the number of descriptive primitives about the piece of information:

`if` $d_0 \leq x < d_1$ `express` d_0
`if` $d_1 \leq x < d_2$ `express` d_1
$\vdots$
`if` $d_{D-1} \leq x < d_D$ `express` d_{D-1}

We can calculate our range boundaries from the following equations, where we have D_{size} descriptive primitives to describe food source size, and D_{occ} descriptions for food source occupancy. Both food source size and occupancy are expressed as a percentage between 0 and 100:

$$d_{n,size} = 100\frac{n}{D_{size}}, n \in 0, 1, 2, \ldots D_{size} \tag{5.1}$$

$$d_{n,occ} = 100\frac{n}{D_{occ}}, n \in 0, 1, 2, \ldots D_{occ} \tag{5.2}$$

To vary the coupling in the robot swarm we control how the robots express information. If all the robots use the same signal to meaning mapping when expressing information then the information in the environment is consistent. The interpretation of how to react to any information must still be evolved, so the signal to meaning mapping is not required when interpreting information, only when expressing it.

5.3.1 Varying the Amount of Coupling

In our experiment we compare the rate of evolution and quality of solution in swarms where the communication is tightly coupled, loosely coupled and uncoupled.

In the loosely coupled communication there is one signal for each of the descriptive primitives, these can be combined sequentially to describe a food source's properties. With tightly coupled communication the meaning that is conveyed and the signal are closely tied. Each signal represents a possible combination of descriptive primitives; so for example, there would be one signal to express a "big and 66% occupied" food source, a different signal to express a "big and unoccupied" food source. The difference in the number of signals required is shown in equations 5.3 and 5.4. S is the number of signals required for the communication, I is the number of pieces of information that can be described and D_i is the number of descriptive primitives for information i:

$$S_{loose} = \sum_{i=1}^{I} D_i \tag{5.3}$$

$$S_{tight} = \prod_{i=1}^{I} D_i \tag{5.4}$$

We only use two pieces of information to describe a food source (size and occupancy), so $I = 2$. We also use only 3 primitives to describe each piece of information, so $D_{size} = D_{occ} = 3$. Substituting these values into equations 5.3 and 5.4 gives us $S_{loose} = 6$ and $S_{tight} = 9$. With such small values for I, D_{size} and D_{occ} both cases have a similar number of signals. However, if we slightly increase the expressiveness of the robots, perhaps changing D_{size} and D_{occ} from 3 to 4, S_{loose} increases by just 2 but S_{tight} increases from 9 to 16. Both types of communication are equally expressive but as we increase the expressiveness, by changing I or any value of D, S_{tight} increases far more rapidly than S_{loose}.

In our experiment, S_{loose} and S_{tight} have similar values. This is because the experiment is intended to be able to run on real robots, and there is a limit to the number of audio signals that the robots can play or accurately interpret. In simulation however there are no such limitations, so it is possible to test any values of I, D_{size} or D_{occ}.

Communications Tested

To test our hypothesis we evolve collective foraging behaviour using five different levels of communication couplings. In descending order of coupling strength these are:

1. **Tightly coupled**. S_{tight} signals each expressing a food source's size and occupancy.

2. **Loosely coupled**. S_{loose} signals combined sequentially to express a food source's size and occupancy.

3. **Uncoupled**. S_{loose} signals, sequentially combined as with (2). Within each piece of information the mapping between the D_i primitives and signals is evolved. An example of this type of mapping is shown in figure 5.1.

4. **Uncoupled**. S_{loose} signals, sequentially combined as with (2). The mapping between signals and descriptive primitives is evolved, as demonstrated in figure 5.2.

5. **Uncoupled**. S_{tight} signals, with meanings as with (1). The mapping between signals and signal meaning is evolved.

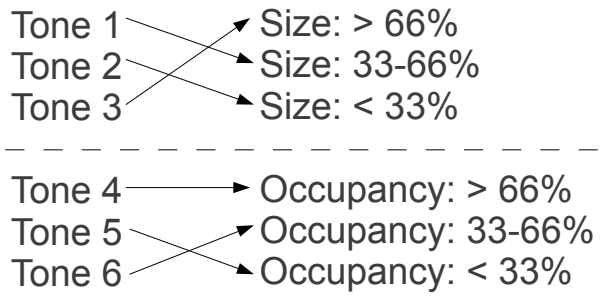


Figure 5.1: Example mapping for uncoupled, alphabetised communication. The mapping is evolved between signals and meanings within each information type.

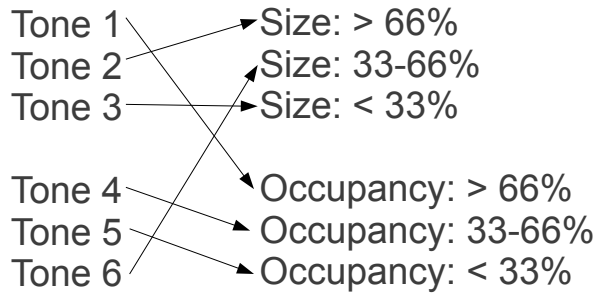


Figure 5.2: Example mapping for uncoupled, alphabetised communication. The mapping is evolved between all signals and all meanings.

In cases where the mapping is evolved, two meanings are prevented from being assigned the same signal, and similarly one meaning cannot have two different signals. In all cases the robot *must* express information when it observes a food source. Consequently there is the implicit mapping of a tone being heard to mean "a robot has detected a food source".

These communication couplings have been chosen because:

- Evaluating (1) and (2) allows us to compare evolution with tight and loose coupling.
- (4) uses alphabetisation without coupling. We can compare the results of case (4) with (5) to see the effects of alphabetisation on the rate of evolution. There are $S_{loose}!$ possible mappings of case (4), which in our experiment gives 720 different mappings.
- (5) uses no alphabetisation and no coupling. There are $S_{tight}!$ possible mappings, which gives 362880 different mappings in this experiment. With so many different mapping possibilities there is very little consistency in the audio information in the environment, it is possible the robots could evolve to ignore audio information altogether.
- (3) is a case where we have a very loose coupling between signals and meanings. The D_i signals that a robot uses to express primitives of information i is specified

prior to the evolution. This is the same for all the robots. Consequently, there is coupling between the robots to some small extent, because certain signals always mean "size" and others always mean "occupancy". However the signal to meaning mapping is still evolved, so the audio information in the environment is not entirely consistent; as it would be in (2) or (1).

Equation 5.5 gives the number of possible mappings for coupling case (3). Using the values from our experiment this gives only 12 different possible mappings, which is considerably less than for (4) or (5).

$$\sum_{i=1}^{I} D_i! \tag{5.5}$$

5.3.2 Benefits of Audio Communication

We use sound as a communication medium because we can carry several pieces of information in a single audio tone. There is information implicit in the tone about where it came from, and this information can be used to signal the location of a food source. By using more than one microphone we can distinguish the direction that a sound came from by analysing the phase difference or comparative volume of the microphone signals. We can estimate the distance from a sound source by how loud the sound is, and this can be compared to other sounds to estimate which sound source is closest to the receiver. There is also information in the broadcast tone which is not implicit. We can control the frequency and duration of a tone, and these can be varied in order to convey different meanings.

One major benefit that sound gives is that when a sound is made, that information is broadcast omni-directionally by one robot, but can be received by many different robots. Furthermore, the receiving robots don't have to be facing the broadcasting robot to receive this information. If we were to use visual messages, such as flashing or coloured LEDs, not only would the receiving robot need to be facing the broadcaster, but complicated visual processing would be needed to locate the light source in the camera image and then determine the rate of the flashing. In comparison, sound is a one dimensional wave, and although it is subject to noise, this is much easier to minimise.

5.4 Implementing ESR

In this section we review the ways that ESR has been implemented by others. We look at how collective evolution has been implemented in ESR systems and to what effect. We then review different architectures that have been used for evolving robot controllers, and give some advantages and disadvantages of each. These findings are used to inform the implementation decisions of this experiment.

5.4.1 Collective Evolution

Watson, Ficici and Pollack [93] use a group of robots, running heterogeneous controllers to evolve a phototaxis behaviour in a distributed, asynchronous way. A lamp is placed in the middle of the arena, when the robot reaches the light it gains energy and moves to a random part of the arena, from there it must repeat the task to gain more energy.

When a robot has enough energy it can reproduce, it does this by broadcasting a slightly mutated version of its genome over a small local area. Other robots may take this broadcast genome as their own, completely replacing their existing genome, with a probability dependent on the receiving robot's energy level. If it has low energy there is a higher probability that it will accept the new genome than if the robot has a high energy. This method of sharing and evolving genomes is a swarm behaviour and does not require any outside assistance or centralised controller to monitor, evolve or distribute a robot's genome; it is all performed collectively by the swarm. It is open-ended and adaptive, so if the environment changes over time the swarm will also change in reaction.

Floreano *et. al.* [29] evolved simple communication in a swarm of 10 foraging robots. They demonstrated that swarms where the robots all ran the same controller, were more likely to signal to each other in the presence of food and so in general had higher fitnesses. The experiment was performed using a high fidelity simulation of the robots and the winning genome was transferred to the real, embodied robots and were found to display the same behaviour as in simulation.

Pugh and Martinoli [74] investigated the effects of genome diversity on the fitness of a simulated robot swarm running heterogeneous genomes. They compare two optimisation algorithms. One, a genetic algorithm, used all the genomes in the population to generate the next population. In the other algorithm[1] each genome generates one offspring for the next generation based on its own fitness and the fitnesses of its nearest two neighbours in the population array. In the second algorithm there is more genetic diversity in the population because all genomes are drawn from a different set of "parents". The authors found that the more diverse algorithm gave higher fitnesses, even in the presence of sensor and actuator noise. When they investigated *why* this was the case they found that *"by maintaining high diversity throughout evolution, [the more diverse algorithm] is able to continuously discover new and better solutions and continue improving throughout the entire evolution"* [74].

Heterogeneous swarms have the advantage to ESR research that the genomes can be evaluated in parallel with each other rather than sequentially, but this means that the evolutionary algorithm's genome population must be the same as the number of robots in the swarm; with a homogeneous swarm there is no such limit. Consequently heterogeneous swarms are faster to advance from generation to generation but because population size is limited there is less genetic diversity within the genome population. Additionally, genomes are assessed by a robot's ability to perform a collaborative task, therefore using a metric that is partly dependent on the performance of other genomes. For example, if in this experiment the GA were to somehow generate a genome which encapsulates the perfect solution but it is one of a heterogeneous population of otherwise poorly performing genomes. The perfect genome would not collect as much food as it would in a better performing population, because it still relies on the others to help it collect. An "average" solution in an "average" population may well collect more food and so be awarded a higher fitness. In a homogeneous swarm this wouldn't be a problem because all the robots would be using the same genome.

The biological counterparts to swarm robotic systems, swarm insects, are not genetically identical but they also have little genetic diversity between insects in a colony.

[1] This algorithm is called "Particle Swarm Optimisation". A review of the particle swarm optimisation algorithm is outside the scope of this work, however, readers are referred to [43] for more information. A description of the specific particle swarm optimisation algorithm used in [74] is given in [73].

With ants for example, all the ants in a colony are offspring from a very small number of queens [37], if two ants share the same parents they will, on average, share 75% of their genes [50]. Worker ants are sterile and contribute to the evolution of the colony by foraging for food and defending the ant nest from invaders, so that the colony is better able to produce the queen and male ants. Evolution and diversification of the ants occurs through new queens and males leaving the nest to breed and start new colonies. Thus, it is the *ant colonies* that evolve, not individual ants.

Swarm Diversity in the Experiment

In our experiment we would ideally like to use the collective evolution strategy from Watson *et. al.* [93]. This strategy resembles evolution in natural systems because each robot has a different genome and measures the its fitness by the robot's ability to gather a resource, energy, whilst following the behaviour specified by its genome. Robots decide between themselves whether to reproduce with each other and create a new genome based on their comparative fitnesses. Consequently, the genetic functions of selection, crossover and mutation are all performed locally by the robots, so the evolution is diverse, asynchronous, decentralised and scalable. These are some of the properties of swarm intelligence given in section 3.2, making the evolution a swarm intelligent behaviour. With evolution that is analogous to the evolution of biological systems, the robot swarm is closer to the complex systems that Simon describes in [84].

Implementing a *decentralised* evolutionary strategy depends on the robots being able to pass data to each other without help from an external controller, but unfortunately the robots we have for this experiment (described later in section 6.3.4) are not able to reliably pass more than one byte of data between each other. In order to transfer genomes to and from the robots we must therefore use a computer to manage the population of genomes by collecting the fitnesses of genomes running on the robots, generating the next generation, and being responsible for distributing the new genomes to the robots. Consequently, we are not able to implement decentralised evolution because updates to the robots' genomes must be controlled through the computer. In addition to a being single centralised controller, this also removes localisation because if the robots control the genome reproduction then a robot can only share its genome with other robots that are nearby. The external computer does not know which robots are spatially close so it cannot limit genome mutation to be between robots that are neighbours. Robots that are near to each other are just as likely to reproduce as ones which are far apart.

We approximate the evolution of Watson *et. al.* by using a heterogeneous swarm with a centralised controller for running the GA. This approach is much quicker to evaluate a generation than a homogeneous swarm, but if the heterogeneous swarm is unsuccessful at testing the hypothesis we can easily run the experiment with a homogeneous swarm by changing the GA on the computer.

5.4.2 Evolvable Robot Architectures

Neural Networks

The most commonly used evolvable robot controller architecture is a neural network; for example [10, 29, 65, 74, 89, 93]. Neural networks are a way of mapping a set of inputs to a set of outputs which is inspired by the way neurons in the brain learn and process information [58]. To do this a neural network is made of artificial neurons,

each of these takes a weighted sum of its inputs and thresholds it to give a high or low output. A neural network is made of any number of artificial neurons which may take the weighted sum of the neural network's inputs or the outputs from other artificial neurons. It is possible to modify a neural network's input to output mapping by keeping the connections between artificial neurons constant but changing the weighting of each neuron's inputs. It is the weights on the inputs to each artificial neuron which contain the input to output mapping in a neural network and not the way the network is connected. Consequently, it is difficult to analyse neural networks to understand how they work.

For several examples of neural network controllers in ESR, the reader is referred to [89], for examples of evolved neural network controllers in individual robots see [65]. With few exceptions [34], the most common approach to evolving a neural network is to keep the structure and connectivity of the neural network constant across all generations and robots, and evolve the weights on each connection. It is this method of evolving neural network controllers that is reviewed in this section.

Advantages There is a simple one-to-one mapping between the genome and the controller. One gene maps directly onto one weight in the neural network without any need for scaling, so small changes in a gene's value cause small changes in the weight and large changes in the gene cause large changes. This is good because similar genomes will produce similar neural network controllers, making it easier for the GA to explore possible solutions. If a genome with a high fitness is found then the GA can generate and test similar genomes because they will convert into similar controllers.

Disadvantages Neural networks, inspired by brain cells, are predominantly learning systems. The weightings in a neural network are normally modified over time so that the correct mapping between inputs and outputs is learnt over time with training. With training techniques, every weight can be updated simultaneously upon receiving feedback about the neural network's performance, and this update can be directed by the feedback. However, when using evolution only some of the weights are updated (due to crossover and random mutations) and the update is not necessarily directed by the feedback. Another major disadvantage is that, with exception of very small networks, there are a large number of weights in a neural network. Consequently large genomes are required to contain all the weightings necessary to describe the controller. This makes the GA's task of searching for fit solutions difficult because there are so many variables to evolve.

Genetic Programming

Genetic programming (GP) is a way of using a genetic algorithm to evolve procedural, executable code such as an equation or a computer program [46]. The genome represents the program in a syntax tree structure, each internal node on the tree is a function in the program, and each terminal is a variable or constant. To execute the program, each node visited and evaluated in depth-first traversal order. Crossover, the merging of two genomes to create two new genomes, is performed by swapping sub-trees in a genome as shown in figure 5.3. Mutation is performed by swapping a terminal for a random new terminal or randomly generated sub-tree.

To evolve a robot controller the terminal variables are sensor readings, constants or actuator actions, and the functions are processes, comparisons or decisions to perform

on the data from the child nodes [47]. Figure 5.4 shows an example of a robot controller evolvable with GP.

Advantages GP evolves an executable program that can immediately be run on robot, and because it is a program its behaviour and functionality can be more easily understood than a neural network. Another advantage of directly evolving a program is that we can have functions that incorporate control flow statements, such as `if`, `for` or `while`. This would be difficult to achieve with a neural network, and is a useful feature for a robot controller, where actions may need to be performed continuously or repeatedly.

Disadvantages GP has no restrictions on the size of genome that can be produced, so large and expressive programs can be evolved. Although this itself is not a bad thing, it means that the evolved programs often contain redundant code, known as bloat, which is never executed [52].

The function and terminals used to generate the programs must be predefined before evolution by the designer, so the function and terminal sets available to the GP algorithm may not be the most appropriate for the problem being solved. When the GP algorithm generates a genome syntax tree, all entries in the function set or terminal set are equally likely to be put into the tree at any point. Therefore all nodes must satisfy the property of "closure" [20, 46]:

> Closure requires that every function and terminal return a value of compatible type, and that every function accepts arguments of this same type. This constraint guarantees that any combination of functions and terminals that comprise a complete tree structure *[...]* will be a syntactically valid computer program that can be evaluated. [20]

Consequently all sensor terminal outputs must be in the same range, and all actuator terminals must output some value even if there is no meaningful data to output. Additionally, if all terminals are equally likely to be placed at any point in a syntax tree, terminals could be placed inappropriately. Using the example program from figure 5.4 of `if(right proximity reading` $<$ `20)` {`turn left 90°; move forwards;`}, the GP algorithm is just as likely to have evolved the less meaningful program `if(move forwards` $<$ `turn left 90°)` {`right proximity reading; 20;`}.

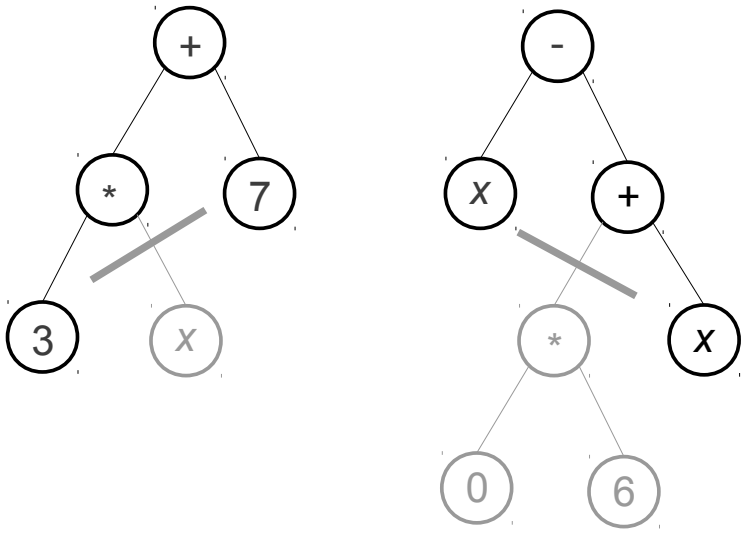
Grammatical Evolution

Grammatical Evolution (GE) is a way of evolving something which conforms to some predefined structure, known as a "grammar" [77]. The grammar is constructed by the user, so GE is theoretically just as capable of evolving an equation to fit a curve as evolving an executable series of instructions for a robot.

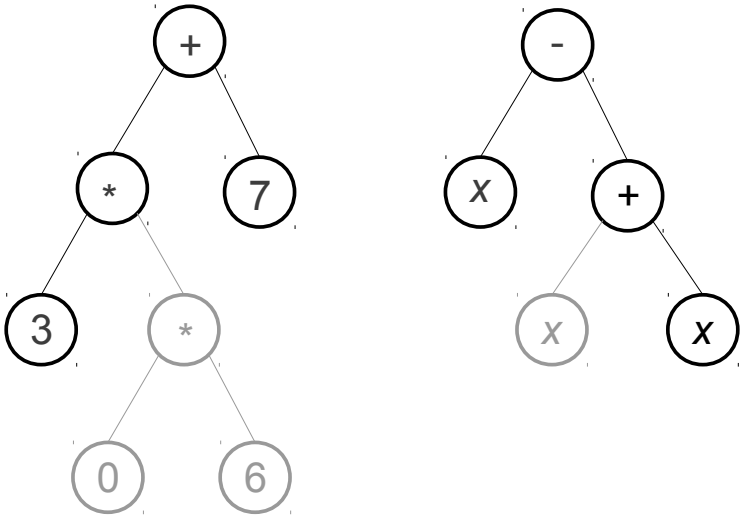
To illustrate how GE works we use a slightly adapted an example from [77] to develop an equation. Grammatical evolution starts with a predefined grammar and a starting point to initialise the grammar, written in Backus-Naur form:

```
<expr> ::= <expr> <op> <expr>   (0)
         | <var>                (1)

<op>   ::= +                    (0)
```



(a) Before crossover



(b) After crossover

Figure 5.3: Example of crossover in GP. A random sub-tree is selected from each genome (a), these sub-trees are then placed into the other genome at the vacated position (b).

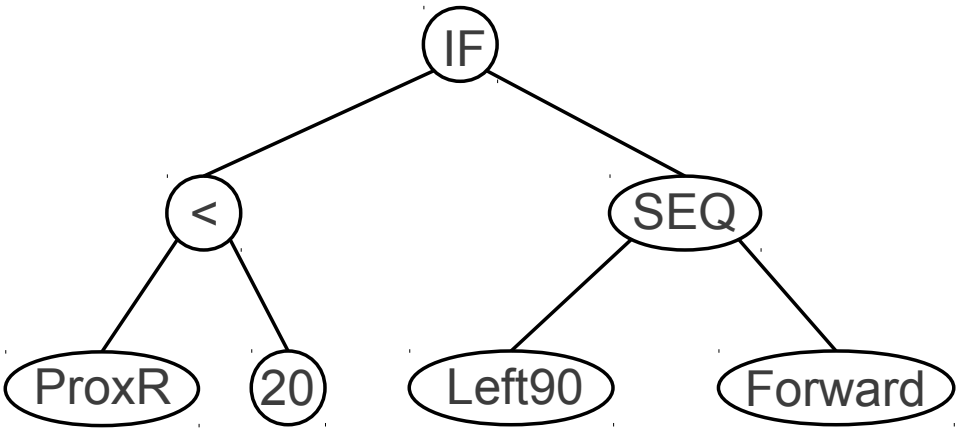


Figure 5.4: Example robot controller using syntax tree structure. The `SEQ` function evaluates its two child functions in sequence. The example code converts to the program: `if(right proximity reading` $<$ `20)` {`turn left 90°; move forwards;`}

```
        | -                 (1)
        | /                 (2)
        | *                 (3)

<var>  ::= X                (0)

START = <expr>
```

In GE, the genome is a string of integers which are used to select expressions, operations and variables from the grammar. For example:

```
[68, 829, 123, 499, 1024, 1]
```

The starting point is `START = <expr>`, so our first gene, 68, must select an option from the `<expr>` rule. The `<expr>` rule contains two options: `<expr> <op> <expr>` or `<var>`. To convert the gene to an option, we modulo the gene by the number of options available for the rule. The resulting number gives the index of the option to use. With the gene 68 and 2 possible options, 68 mod 2 is 0, so selecting expression (0) from the `<expr>` rule gives us an equation `<expr> <op> <expr>`. Next we fill in the first part of the expression: the first `<expr>`. Again there are 2 options, taking the next number from the genome 829 (829 mod 2 = 1) gives us a `<var>` the only option for which is X. Now our generated equation is `X <op> <expr>`. The next thing to fill in is an `<op>` with four different possibilities, 123 mod 4 = 3, so `<op>` (3) is $*$ and our equation is now `X * <expr>`. Carrying on in the same way we can fill in the equation to get `X * X`. There are no more `< >` clauses to evaluate so the equation is complete, any remaining numbers in the genome are ignored.

Examples of grammatical evolution for robot control are rare in the scientific literature. Burbidge, Walker and Wilson [16] successfully use GE to evolve a robot controller that moves towards a light and avoids collisions with obstacles. GE is used to generate C code that uses readings from proximity and light sensors to assign speeds to two differential motors on the robot. This experiment is done in simulation, and only for an individual robot rather than a swarm.

Advantages GE, like GP, can be used to evolve an executable robot controller program. This means that, like GP, evolved GE programs are easier to understand than a neural network controller, and the evolved program can include control flow statements. By using grammar to control how functions and variables should fit together, GE also eliminates some of the problems associated with GP. There is no need for closure because the inputs to functions can be specified by the grammar. Furthermore, the grammar specifies when to use sensors readings, actuator control or constants, so it is not possible to generate controller programs where these are placed inappropriately.

Disadvantages As with GP, the grammar must be defined before evolving a robot controller, so the functions in the grammar may not be the the most appropriate for the problem. GE can also cause program bloat, although because of the restrictions imposed by the grammar this can be less extensive than the bloat with GP.

The main disadvantage of GE is that because of the indirect mapping between the genome and resultant program, genomes which are similar could represent very different robot controllers. Although small changes can be made to a genome, there is no fine control on how much the resulting controller could be mutated. Even a single changed gene near the beginning of a genome could potentially cause the controller to be entirely different, but if the changed gene is in an unused part of the genome the controllers would be identical [17]. This makes it very difficult to improve good solutions, because if a good controller is found it is hard to generate similar controllers to try and optimise any part. In this regard, neural network and GP controllers have an advantage over GE controllers.

Evolvable Architecture in the Experiment

Given the advantages and disadvantages of several different robot architectures we have decided to use grammatical evolution in this experiment. What we want to evolve in this experiment is an executable program, which GE or GP allows us to do. Compared to GP, GE gives us greater control over how functions, variables and constants in the program should fit together, meaning that there is a greater probability of evolving meaningful, working programs than with GP.

The problem of GE mutations causing unknowable amounts of controller change can be reduced slightly by performing a crossover similar to the sub-tree crossover in GP (figure 5.3). Code within a $< >$ bracketed grammar clause could be swapped for code under a similar clause in the other genome. If no other genes are changed then this crossover would not affect the structure or functionality of the rest of the program.

5.5 Conclusion

In this chapter we outline an experiment for testing the hypothesis from chapter 4. We describe a foraging task where the robots must collect food from different sized food sources. Larger food sources need more robots present before they dispense food, but give each collector a larger amount of food as a result, therefore encouraging cooperation through greater reward. The robots are able to signal to each other to share information about a food source's size, occupancy and also it's location in the arena, which is implicit information in the sound signal. By signalling to each other, the robots should be able to recruit help with collecting food and decide which of the food sources to go to for the largest reward.

Section 5.4 contains a review some of the ESR research to date. We look at how collective evolution has been implemented in the literature and found that homogeneity in a swarm encourages collaborative behaviour, whereas heterogeneity allows faster evaluation of the GA population and more diversity within a swarm. From these findings, and the constraints of the robots available, we decide to use a heterogeneous swarm of robots in our experiment. We then review different evolvable robot controller architectures and decide that grammatical evolution is the most appropriate for our research.

In part II of this thesis we develop a model of the robots using sound signals to communicate. In chapter 9 we use the model from part II to simulate this experiment, and present the results.

Part II

Co-Development of Simulation and Hardware

Chapter 6

Basic Simulation

6.1 Introduction

To carry out the experimentation described in chapter 5, we develop a model of the swarm robots using audio signals to communicate and achieve a collective goal. With this model we can run experiments on a simulated robot swarm, requiring no human intervention and consuming less power than experiments with embodied robots. Importantly, experiments using the simulated robot swarm would be much faster than with an embodied swarm, significantly reducing experimental time.

The experiment from chapter 5 is intended to be run using embodied robots, adn to do this we need to design and build a hardware extension board for the robots capable of sending and receiving audio messages. We call the hardware extension board the "soundboard", and it is developed alongside the model. Through co-development we hope to build confidence in the simulation. In creating the simulation *before* the soundboard we can begin to make decisions about what is required of the soundboard before beginning hardware development. As the soundboard is developed we make design decisions affecting its abilities and constraints, which can later be fed back into the model in order to calibrate it and make the model more representative of the real system.

The co-development of the simulation and robotic hardware is broken down into three phases:

1. **Initial Modelling**. Here we develop a model of the experiment from chapter 5, with multiple foraging robots using audio signals to communicate. The model includes an approximation of the soundboard based on our requirements for it.

2. **Developing the Hardware**. In this phase (chapter 7) we build the soundboard for the robots, capable of detecting audio signals, their direction and their frequency.

3. **Feedback of Hardware into the Model**. Once the soundboard is completed we can measure its properties and use it to calibrate our model (chapter 8).

We would ideally also like to validate the model by comparing it to the embodied robots performing the experiment, but unfortunately time constraints prevent this from being possible. Repeating our experiment using an embodied robot swarm is therefore

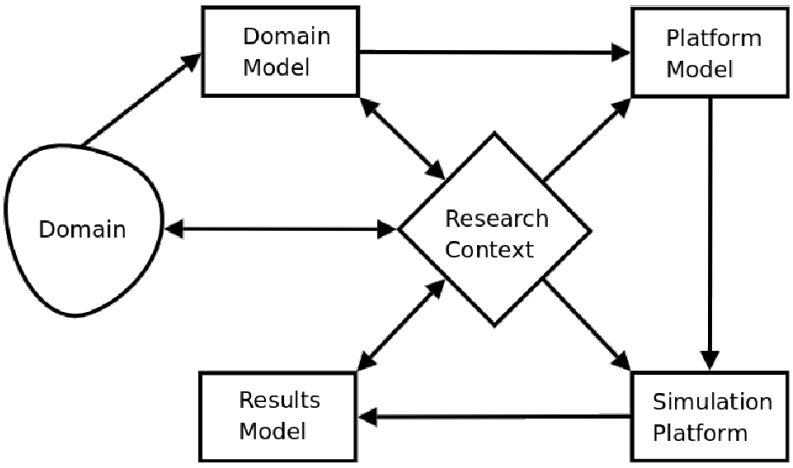


Figure 6.1: The basic CoSMoS Process Diagram [5]

left as future work. Despite this, the hardware development phase of the modelling is still necessary.

We use the CoSMoS (**Co**mplex **S**ystem **Mo**delling and **S**imulation) process, described in section 6.2, to develop the model, so that each stage of development is documented and any assumptions made during the modelling process are explicitly stated. One of the aims of this thesis is to provide a case study of the CoSMoS process for modelling swarm robots. By building some of the hardware required for the experiment on embodied robots, we can use it to realistically model sound transmission and reception. The modelling, hardware development and model calibration phases allow us to do a full cycle of the CoSMoS process for engineered systems (section 6.2.2), iteratively improving the model as more is known about the modelled domain.

In this chapter we perform the initial modelling phase, following the CoSMoS process. In section 6.2 we describe the CoSMoS process and the motivations for using it. The rest of the chapter is devoted to each part of the CoSMoS process. Where possible, measured or known values are used for any model parameters. For aspects of the model that are unknown, such as the soundboard, we use estimated values but make a note to calibrate these areas later when more is known, or measurements can be taken.

6.2 The CoSMoS Process

The CoSMoS process (figure 6.1) is a framework for modelling complex systems, and can be applied to many different types of complex system. Case studies it is being developed upon include robots, plants and immune systems. It is not limited to modelling these kinds of system, but aims instead to be a generalised process which can be followed to model *any* complex system [31].

The following is an overview of the CoSMoS process shown in figure 6.1 [6]:

Domain This is the thing we wish to make a model of; not just the aspects of it which are known but also the things we wish to know more about through modelling. With the CoSMoS process the emphasis is on domains which are complex systems. As

an example, the paper [75] models Experimental Autoimmune Encephalomyelitis (an autoimmune disease in mice) using the CoSMoS process. The domain in this case is the immune system of a mouse, which the authors wish to learn about to further their understanding of how a mouse recovers from the disease.

Research Context This part of the process establishes how and why the model is being created, including the broader scientific context of the work: *"The Research Context defines the fundamental scope and purpose of a CoSMoS project"* [5]. The model is produced in close collaboration with a "domain expert". A domain expert is an expert in the domain we wish to model. For the example of [75] the domain expert was Vipin Kumar, an immunologist studying Experimental Autoimmune Encephalomyelitis. Through the domain expert we are always considering the research context of the model. Polack [71] emphasises that collaboration with a domain expert builds trust in the model for both the developer and the domain expert, and that *"the simulation is only useful to a scientist if the scientist trusts the simulation"* [71].

When considering the research context of the model, the following points should be considered [71]:

- **What questions the model will address**. What do we want to find out from the model?
- **Motivations for the research**. How is the building of the model grounded in scientific research?
- **How will the results of the modelling and simulation be assessed for validity**. How will we know that the information our model gives is accurate?

Domain Model This deliverable is a top-down summary of our current knowledge of the domain, including any emergent behaviours present in the domain of interest. The domain model includes things that are not relevant to the final simulation, as we do not make decisions as to what is and is not relevant until later stages.

Platform Model This is the "design" model. This is an abstraction of the domain model so that it can be implemented on the platform of choice. Parts of the domain model which do not relate to our research context are removed, including emergent properties so that they do not get explicitly implemented. This leads to the modeller making assumptions about what is and is not relevant. Theses assumptions must be documented so that we can later decide whether they were erroneous or not.

Simulation Platform This is the implementation of the platform model using the platform of choice. The simulation is assessed according to the criteria set out in the research context. First we must *"calibrate the simulator by adjusting parameters to match our knowledge of the real world system"* [7], this allows comparisons between the simulation and the domain of interest. The simulation can then be used for experimentation and to make predictions about the domain. Metrics from the research context are measured to analyse whether emergent behaviour is achieved and if the model is valid. From this experimentation a *Results Model* is made.

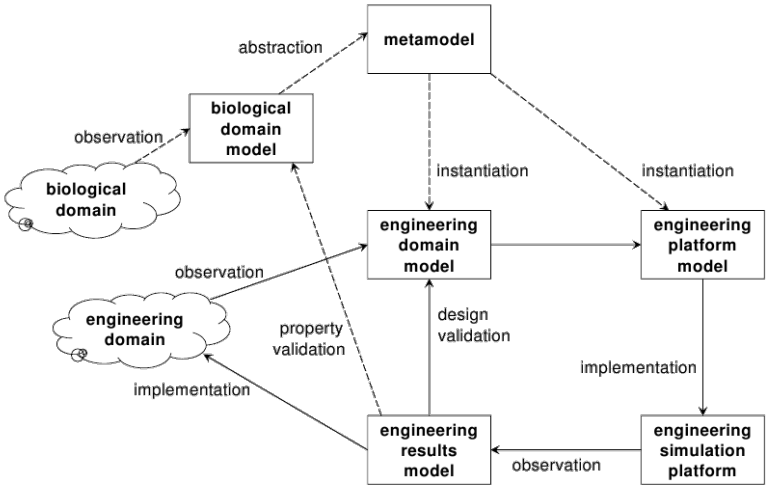


Figure 6.2: The CoSMoS process for bio-inspired engineered systems [6].

Results Model This is a model of our simulation. Just as the domain model captures what we know of the domain, the results model captures what we know of the simulation. In effect, the simulation is the domain for the results model. The results model is assessed according to the success criteria set out in the research context, the outcome of this assessment *"highlights deficiencies in the earlier modelling stages"* [7].

6.2.1 Benefits of Following the CoSMoS Process

The CoSMoS process draws quite heavily upon the software engineering life cycle. In constructing the domain model we are building a specification of what needs to be captured within the simulation. The platform model is a design of how to build the simulation, which is then implemented to build the simulation. Finally, to build an analysis model we test the simulation against our specification and the metrics set out in the research context. Errors uncovered at this stage require previous phases to be revisited and re-done until the simulation passes all tests.

The strength of the CoSMoS process is that at each stage there is transparency in the construction of the simulation. As we progress through the process, the domain of interest is continually modelled and refined so that each phase is more specific than the last. Any assumptions about the domain incorporated into the model are recorded along with the reasoning for such assumptions. This ensures that the modeller has a record of every place that errors could have entered into the model so that it is easier to diagnose problems later. It also provides anyone who might wish to use the model with a list of the model's weaknesses so they can decide whether the model is accurate enough for their needs.

6.2.2 The CoSMoS Process for Engineered Systems

When engineering a bio-inspired system, we aim to solve some real-world problem by copying something found in nature. The CoSMoS process can be followed to ensure that the final product is good by simulating and testing ideas and algorithms before

committing to construct the system. This puts the modeller in the situation where they are modelling something which does not yet exist. For this situation, there is an extension of the CoSMoS process, shown in figure 6.2. Although this process does not specifically state that the research context should be considered as with figure 6.1, it still influences all the decisions taken whilst developing the model. Unlike the original CoSMoS process however, our research context isn't "can we learn more about our domain?" but "does this engineered system adequately solve the problem we have?". This modified process acknowledges that there is more than one domain in a bio-inspired engineered system: the engineered system, and any biological systems that inspired it. The "engineering domain" from the extended CoSMoS process is the system we're trying to model, but the biological domains influences how it will work and the processes it uses.

The CoSMoS process for engineered systems dictates that we need a domain model for each domain. The domain model for the biological domains is developed first and then abstracted into a meta-model. Andrews et. al. [6] describe this meta-model as *"[capturing] the relevant concepts and relationships of the biological model. At this meta-level, we can abstract away contingent details of the biology that are of no relevance to an algorithm"*. Essentially then, the meta-model encapsulates the core concepts of the biological domain. These core concepts are what is copied in order to replicate the inspirational behaviour in the engineered system. The engineering domain model is initially a specification for the engineered domain. This model is an instantiation of the meta-model from the biological inspiration, since the meta-models are an abstract outline of features of the final model. Similarly the engineering platform model is also an instantiation of the meta-models because the platform model describes how to apply the contents of the meta-model in the simulation platform.

Once the platform model has been simulated and provided results, the results are validated against the engineered domain model, because this is the specification for the final engineered system. The results are also validated against the biological domain model to ensure that the simulation displays the desired properties of the inspiration. If the simulation is shown to be valid, then it can be implemented by creating the engineered system, thereby creating the "engineering domain" section of the process model. Finally the engineered system is tested and validated. If it is shown to not adequately solve the problem it is designed for then the whole process iterates again, this time with a real engineered system to inform the 'engineering domain model'. In this way, the model and engineered system are refined until a final system can be engineered and tested which does fulfil its requirements.

6.3 Following the CoSMoS Process

We follow the CoSMoS process for engineered systems to develop our simulation. This process gives us the flexibility to bring together multiple systems in order to build a working, engineered complex system. The process diagram for the CoSMoS process for engineered systems is given in figure 6.2 on page 54.

6.3.1 Research Context

As stated in sections 6.2 and 6.2.2, the main points the research context must answer are:

- What questions the model will address
- Motivations for the research
- How will the results of the modelling and simulation be assessed for validity
- Does this engineered system adequately solve the problem we have?

The purpose of creating this model is to answer our experiment's hypothesis, which is: Does loose coupling between subsystems of a complex system affect the rate of evolution as claimed in [84], and can this be applied to evolutionary swarm robotics by "alphabetising" the information communicated between robots. The motivations and the scientific context of this hypothesis have already been given in part I. If our model can be shown to test this hypothesis, then our engineered system *does* adequately solve the problem we have. However, it remains to be shown that the model is a valid representation of the final engineered system, and that the model tests the hypothesis.

Validating the Model Against the Engineered System

Unlike using the CoSMoS process for developing a model of a complex system, we are trying to model something that does not yet exist. Consequently there is no domain expert to collaborate with to guide the modelling process. We, as the engineers, must guide ourselves through the modelling based on what we want the final system to be able to do and the limitations imposed on it. The process of co-developing the model and engineered system helps with this, because as we develop the system we can feed back our knowledge into the model. This lack of a pre-existing domain can be a disadvantage, as the modelling process is unguided by a domain expert. However, it does award us the freedom to change the domain to be more like our model if we so choose. For example, we can control the environment the robot swarm operates in: the amount of light, background noise, number of obstacles in the environment or the size and shape of the robot arena. We could also impose constraints on what the robots are able to do, so that they are easier to model. Whether or not this is appropriate depends on whether the engineered system is still able to perform its task under these new conditions. In our case this purpose is to test our experiment's hypothesis, but in other cases there may be less flexibility on the restrictions that can be imposed on the final system.

In the introduction to this chapter we stated that the experiment will not be run on embodied robots, so the model will not be validated against real robots. In this work, our validation is limited to creating the soundboard and validating the model of the soundboard against reality. The complete model is not validated in this work, but nevertheless we can still provide some answers to the question of "How will the results of the modelling and simulation be assessed for validity".

Assessing Validity The following is a non-exhaustive list of aspects of the model that should be validated. For all aspects of the model listed, the results in the model and the embodied robots should be measured. These results should then be compared using a Wilcoxon rank-sum test ([64] page 45) to see if there is a statistically significant difference between the two sets of results. If the data from the real robot and the data from the simulated robot are not different to a 95% significance level then they are deemed to be "similar enough" for the model to be valid.

- **Within the robot**. The robots have the ability to detect food sources, and they have the soundboard to detect sound. The robots also use motors to move itself. For each kind of sensor or actuator, measure:
  - sensor readings at different ranges.
  - sensor noise at different ranges.
  - Motor speed. When the motors are set to a particular speed, do they move the same distance. How long does it take the robot to move some fixed distance.
  - variation in these measures when using different robots.
- **Robot behaviour**. Do robots with the same genome behave in similar ways when given the same environmental conditions?
  - Within a fixed length of time do the robots move, on average, similar distances and collect similar amounts of food.
  - If two embodied robots were given the same genome and environmental conditions would they behave in the same way.
- **Swarm behaviour**. Do we observe similar results when running the experiment on embodied robots as we do with the model?

Validating the Model Against our Hypothesis

Assuming that the experiment adequately tests our hypothesis, we would need to show that the simulation performs the experiment described in chapter 5. To then be able to claim that we would get similar results using the engineered robotic system, we would need to show that:

- The simulation adequately models the hardware.
- The code running the simulation is the same or equivalent to the code running the experiment on the real robots.

We have already discussed how to validate the simulation against the hardware in this section. To demonstrate that the simulation code and the code for the real robots are equivalent we use a robot simulator called Player/Stage.

Player/Stage [92] is a combination of a robot controller Player, and a robot simulator Stage. Player is a *Hardware Abstraction Layer* between controlling code and robot drivers which control sensors and actuators. Controlling code is written to use the Player API. The API makes sensor readings available and allows the code to move actuators without the person writing the code needing to know anything about how the drivers work. The Stage simulator is a plugin for Player which acts as a driver and creates a simulation of the robot and the environment. Interaction with the Stage simulation is done through the Player API. This means that we can write simulation code using Player and Stage, together called Player/Stage, and the *same code* can be run on a robot using just Player without Stage. Figure 6.3 illustrates the flow of control from user code to either the simulation or robot hardware.

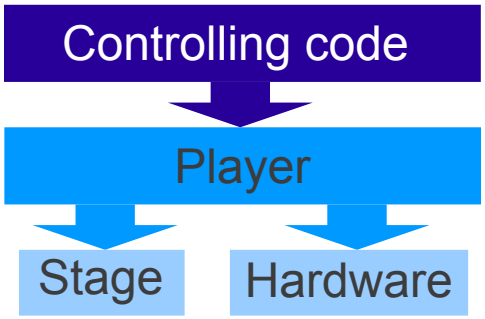


Figure 6.3: Diagram of control flow from user code through Player to the robot hardware or to Stage. The controlling code interacts with the Player API. Player can be set up to control a Stage simulation or interact directly with robot drivers and hardware. In either setting, the controlling code remains unchanged.

6.3.2 Biological Domain, Domain Model and Meta-Model

In the CoSMoS engineering process the system we create aims to mimic the desirable behaviour or features of the biological domain of inspiration. In this work we are creating a robot swarm that can be said to forage for food in the same manner as swarming insects. However, we do not seek to mimic the behaviour of these swarms or to recreate their emergent properties to complete our task. Just as ants evolved to collectively forage for food we aim to evolve a robot swarm that collectively forages for food. As such our biological domain is evolution, rather than swarm insects.

In section 3.4.1 we describe genetic algorithms, and in section 5.4 we described how collective evolution is implemented in our experiment. In this model we use Grammatical Evolution (GE), which has already been reduced from the domain of evolution down to a meta-model suitable for evolving a computer model by Ryan, Collins and Neill [77].

6.3.3 Engineering Domain

With our model we aim to simulate the experiment described in chapter 5. This involves e-puck robots using custom-built audio hardware, a "soundboard", to communicate with one another to achieve their shared goal. Consequently, there are several aspects to our domain which are modelled:

- The e-puck
- The soundboard
- The arena and environment
- Acoustics

6.3.4 Domain Model

For each aspect of the engineering domain we create a domain model, containing what we currently know about that domain.

e-puck

Figure 6.4 shows an e-puck robot. The technical specification for the e-pucks can be found on the e-puck website `www.e-puck.org`. We have access to 13 e-pucks in order to run our experiment.

Properties of the e-puck:

- Physical Dimensions
  - Height: 55 mm
  - Diameter: 75 mm.
  - Weight: 150 g
- Maximum Speed: 12.8 cm/s (0.128 m/s) either forwards or reverse.
- Processor:
  - Processes 14 million instructions per second
  - 8 kB RAM
- Battery:
  - 3.3V
  - 5Wh suitable for *"2 to 3 hours of intensive use"* [1].
- Sensors:
  - Colour camera. Image resolution 640 x 480 pixels. However, the RAM is not large enough to hold an image that size so the image must be sub-sampled, or only a portion of the image can be used. Frame rate is dependent on the image size, larger images causing a smaller frame rate than smaller ones.
  - Infra-red proximity sensors: range approximately 9 cm. The e-pucks are able to use their infra-red sensors to communicate with each other, although they can only send one byte at a time. The success of this communication is dependent on whether the robots are facing in the correct direction and can be broken if they moved out of infra-red range.
  - Bluetooth communication to a computer.
  - Sound. These are the properties of the audio hardware that is built into the e-pucks as standard, rather than the soundboard extension:
    - There are 3 microphones on the e-puck with a maximum sampling rate of 33 kHz.
    - These microphones are on the left and right outer edge of the e-puck and the third is central. Each microphone is pointing upwards.
    - The maximum distance that a microphone can reliably detect a tone is 13 cm.
    - There is a speaker on the e-puck with a sample rate of 7.2 kHz. This is also pointing directly upwards.

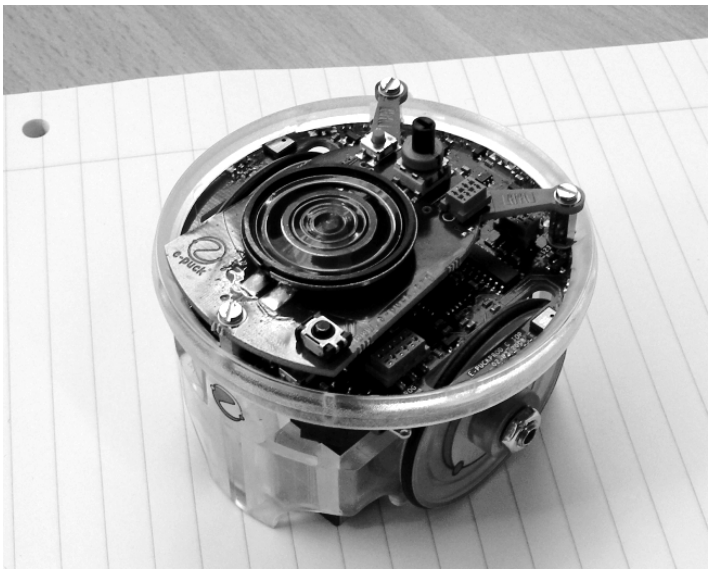


Figure 6.4: A photograph of an e-puck robot.

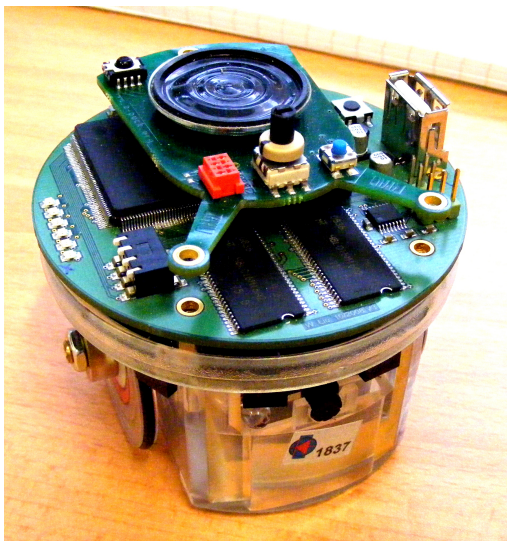


Figure 6.5: A photograph of an e-puck robot with the Linux extension board.

In addition to the e-puck, we use a Linux extension board [55] which allows the e-puck to run Linux, and gives them the ability to communicate over a wireless network using a wifi adaptor which plugs into the board's USB port. Figure 6.5 shows an e-puck with the Linux extension board attached. This extension completely covers all of the e-puck's microphones, rendering them unable to detect sound. The advantage of using the Linux extension board is that it allows the e-puck is able to process and store much more information. Crucially, it also allows the e-puck to run the Player/Stage robot simulator and controller locally without requiring separate processing elsewhere.

Properties of the Linux extension board:

- Runs an ARM9 AT91SAM9260 processor.
- 210 MHz clock
- 32MB RAM
- Runs emDebian[1] off a 2GB microSD card.
- Has a USB port for a wireless network adaptor [2].

The Soundboard

At this stage in development, the soundboard has not been created so we do not yet know what its properties are and cannot model them. We do however know what the general requirements are of the soundboard. Consequently we can create a model knowing what we expect the soundboard to be able to do, and build into our model the capacity to add noise and other faults later in chapter 8.

This is what we know about the soundboard so far:

- The soundboard is an extension board for the e-puck which sits on top of the robot.
- It must be capable of sending and receiving at least nine different tones.
- It must be able to monitor the environment for sounds. If a tone is heard it must be able to calculate its frequency and the direction the sound came from, relative to the forward direction of the robot. This is known as the "direction of arrival" (DOA), it is measured relative to the robot as shown in figure 6.6).
- To measure the frequency of a tone, the soundboard should perform a fast Fourier transform (FFT). This quantises the frequency spectrum into several bands. The number of FFT bands that will be used is currently unknown, and where these bands appear in the frequency spectrum is dependent on the microphone sample rate and the number of FFT bands.

[1] `http://www.emdebian.org`

[2] In section 5.4.1 of the experiment chapter we stated that the robots cannot send data between each other, and this is why we used centralised evolution. Although the Linux extension boards do allow the e-pucks to communicate wirelessly this causes a significant drain on the battery, reducing operating time from three hours to only half an hour. This reduction in battery longevity compromises the length of experiment we can run, so it is not used in this research.

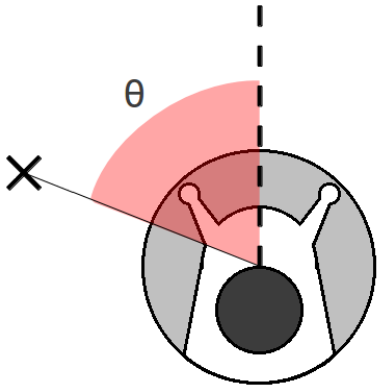


Figure 6.6: Illustration of how the e-puck with soundboard measures the direction a tone came from. θ is the angle between the robot's heading and the sound source.

The Arena and Environment

The arena is rectangular and reconfigurable to any size up to 9×9 metres. The arena walls are approximately 15cm high and the ground is flat, grey painted concrete.

In the experiment there are food sources. In the embodied robot experiment these can be implemented using stationary e-pucks. The e-pucks are capable of sending small 1 byte messages to each other using their infra-red proximity sensors. This messaging system can be used to share a food source's size and occupancy information and to signal to a food source that a robot is collecting from it.

In the model we model the food sources as e-pucks. Details about the e-pucks and the infra-red proximity sensors are given earlier in the domain model.

Acoustics

For our domain model we use chapter 1 of the "Acoustics and Psychoacoustics" textbook by Howard and Angus [39]. This book chapter contains explanations and equations describing sound waves, sound intensity and pressure, the combination of sound waves, sound reflection and frequency filtering.

Although [39] is not as in-depth as other acoustics texts, it does provide an overview of all the acoustics information we may need for our model. Since the focus of our model is the study of evolving a swarm of robots and not the study of acoustics, this simplification is appropriate at this stage to prevent later stages of the modelling process from becoming intractable. If we later find that the model does not accurately represent the acoustics of the real experiment this decision may need to be reassessed.

Dissipation Sound gets quieter the further away the source is. If a sound is louder then the robots may reason that it is closer, if a robot is far enough away from a sound source it may be too quiet to detect.

Sound energy is the *"amount of energy transferred per unit of time [...] the number of joules per second (watts) that propagate"* [39]. Sound intensity is *"the flow of energy through a unit area"* [39], which is measured joules per second (watts) per unit area.

When sound spreads from a source it travels in all three dimensions, so the sound energy is spread across the surface of a sphere. At r metres from the sound source the

sound intensity is:

$$I_{measured} = \frac{W_{\text{source}}}{4\pi r^2} \tag{6.1}$$

where $I_{measured}$ is the sound intensity at the receiver, W_{source} is the number of watts the speaker emits and $4\pi r^2$ is the surface area of a sphere, of radius r metres.

The sound intensity is usually expressed as a level in decibels:

$$\text{Sound Intensity Level} = 10 \log_{10} \frac{I_{measured}}{I_{ref}} \tag{6.2}$$

where I_{ref} is a reference intensity of $10^{-12} Wm^{-2}$.

Sound Combination There could be several sound sources present in the environment at the same time. Consequently, we must find out how they combine so it can be modelled.

Sound waves can be correlated or uncorrelated. Correlated sound waves have the same sound source but have somehow become slightly separated. For example if one wave is a reflection off a surface or if the same sound source is using multiple loudspeakers.

Uncorrelated sounds waves have different sources, for example multiple e-pucks making a noise. With uncorrelated sources, the sound intensities measured from each source (in Wm^{-2}) are summed to give their cumulative intensity.

Reflection Sound waves bounce off boundaries such as the floor and walls and cause the sound to be louder. If there is a boundary where we want to measure the sound intensity, the sound wave reflects off it and sound is even louder. If several boundaries meet where we want to measure, then the sound wave reflects even more and causes the it to be even louder.

Equation 6.3 is a modification of equation 6.1 taking this effect into account.

$$I_{\text{with reflections}} = \frac{QW_{\text{source}}}{4\pi r^2} \tag{6.3}$$

Q is a factor of the number of boundaries meeting. With only one boundary, such as a floor $Q = 2$, with two boundaries such as where the floor meets a wall $Q = 4$ and if there are 3 boundaries like in the corner of a room $Q = 8$. However, this equation only holds if the boundaries are orthogonal.

6.3.5 Platform Model

The platform model has been constructed with the knowledge that much of it may need to be calibrated in later phases of the simulation and hardware co-development. For each aspect of the platform model we explicitly state what will be calibrated in later phases and any assumptions that have been made in developing the model.

Several assumptions have already been made whilst constructing the domain model. These are:

> **Assumption 1: [39] describes acoustic science in enough depth for our model.**
>
> **Assumption 2: Player/Stage adequately models robot movement physics.**

e-puck

The e-puck is modelled as a dark green octagonal prism with a diameter of 7cm and a height of 5.5cm. The model is given proximity sensors in the same locations as they would be on a real robot, and a forward facing colour camera with a resolution of 80 by 30 pixels. A picture of the Player/Stage simulated e-puck and soundboard is shown in chapter 8 on page 91.

Assumption 3: The e-pucks all have the same physical characteristics.

Assumption 4: No e-pucks have faulty motors or sensors.

Acoustics

The "loudness" of the tones is described in the model by the sound intensity, measured in Wm^{-2}, as the readings for the loudness of a tone.

To approximate the dissipation of tone loudness over distance from the source we use equation 6.3, only taking into account the reflections from the floor of the arena so the Q value from equation 6.3 is 2. We can find W_{source} from the domain model of the e-puck: The battery gives 3.3V, and from measuring the speaker on an e-puck we know its resistance is 8Ω, giving a maximum W_{source} of 1.36W. However, the speaker on the soundboard may have a different resistance to the speaker on the e-puck.

Consequently, the formula used to calculate sound dissipation is:

$$I_{\text{at robot}} = \frac{W_{\text{source}}}{2\pi r^2} \tag{6.4}$$

where r is the distance, in metres, from the sound source. If $r = 0$ the intensity is set to W_{source}.

Assumption 5: The microphones in the soundboard have perfect gain across all frequencies of interest.

Assumption 6: The change in sound intensity caused by sound waves reflecting off arena walls and corners is negligible.

Assumption 7: Obstacles in the environment (including other e-pucks) do not affect sound propagation.

Calibration 1: The resistance of the speaker on the soundboard.

The Arena

The arena is initially modelled as a 2 metre by 2 metre square, with the walls 15 cm tall. There are no obstacles in the environment so that our assumption about obstacle acoustics can be correct.

Calibration 2: The arena size, shape and height.

Soundboard

The soundboard is a means for the e-pucks to pass sound information between each other. We model the soundboard by passing sound information directly between the robots and approximating how the data will appear after being passed through a noise medium and the soundboard.

Calibration 3: The shape and size of the soundboard.

Calibration 4: The soundboards all have the same characteristics.

Playing a Tone When a robot makes a sound we record the frequency, volume of the tone and the robot's position. This information is made available in a global data pool. When the robot stops playing the tone, that data is deleted from the pool. The frequency of the tone played is limited to be between half the microphone sample rate and some minimum frequency.

Assumption 8: when an e-puck plays a tone it does not move.

Calibration 5: The microphone sample rate.

Calibration 6: The minimum tone frequency.

FFT effects Performing a fast Fourier transform (FFT) on time domain data quantises it into frequency bands. To replicate this, when information is put into the data pool we perform quantisation so that the frequencies are sorted into the bands as shown in figure 6.7. The lower bounds of the quantisation bands are given by equation 6.5. To minimise the risk of the soundboard in the engineered system incorrectly banding a frequency, the signal tones all fall at the midpoint of a frequency quantisation band.

$$FFT_{\text{lower}} = \text{band index} * \frac{\text{microphone sample rate}}{\text{number bands}} \tag{6.5}$$

Where $0 \leq band\_index < number\_bands$.

Calibration 7: The number of FFT bands.

Listening to a Single Tone When a robot listens for sounds it reads all the information from the global data pool. For each tone in the data pool, the distance of the listening robot to the playing robot is calculated and if the playing robot is further away than the e-puck's sound sensing range then the tone is ignored. For each of the remaining tones, the sound intensity is calculated using equation 6.4, and the direction of the sound with respect to the robot's current heading is calculated (see figure 6.6 on page 62). This is the same information that the soundboard measures, but because we have access to the exact coordinates of the sound playing robot and the listening robot our information is perfect. To approximate the noise and inaccuracy that will exist in the real hardware, noise is added to both measurements. The sound intensity is randomly changed by up to $\pm 10\%$ and the direction is rounded to the nearest $10°$. These are arbitrary amounts of noise and must be calibrated to the true amount of noise when the hardware is built.

Assumption 9: The e-puck is stationary whilst it listens for tones.

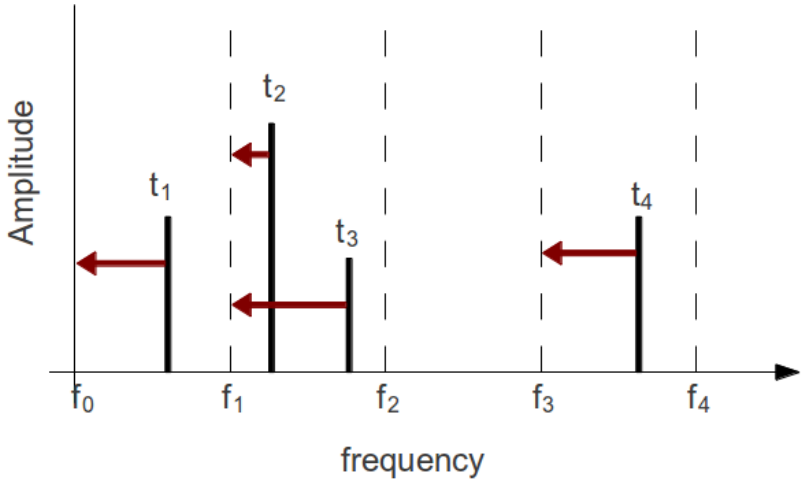


Figure 6.7: The quantisation of frequencies. Tone t_1 is quantised to f_0, t_2 and t_3 are quantised to f_1 and t_4 to f_3.

Calibration 8: The range the e-puck can detect a tone.

Calibration 9: The amount of noise on the sound intensity measure.

Calibration 10: The amount of noise on the direction of arrival measure.

Within each frequency band there might be more than one tone. If this is the case then all the tones within the frequency band are combined. Tones are represented as a polar coordinate where the radius is the sound intensity measure, and the coordinate angle is the direction the sound came from after noise has been added to the measurements. This is illustrated in figure 6.8.

To combine the multiple tones, the mean polar coordinate is calculated. The polar coordinates are converted to Cartesian coordinates, and the mean x and y difference between the robot and the sound source coordinates is calculated using equations 6.6 and 6.7, where N is the number of tones we are combining. These are then converted back into polar coordinates, using equations 6.8 and 6.9, to get the combined sound intensity r_T and direction θ_T.

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} r_i \cos(\theta_i) \tag{6.6}$$

$$\bar{y} = \frac{1}{N}\sum_{i=1}^{N} r_i \sin(\theta_i) \tag{6.7}$$

$$r_T = \sqrt{\bar{x}^2 + \bar{y}^2} \tag{6.8}$$

$$\theta_T = \arctan\frac{\bar{y}}{\bar{x}} \tag{6.9}$$

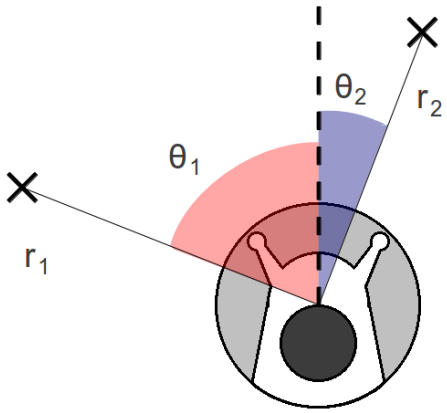


Figure 6.8: Tones modelled as a polar coordinate with the e-puck at the origin.

Assumption 10: Multiple tones of the same frequency combine as the mean of their polar coordinates.

Assumption 11: Tones from different frequency bands will not affect each other.

6.3.6 Simulation Platform

In this part of the CoSMoS process the model is calibrated to match real world measurements. This is done in phase 3 of the co-development of the simulation and hardware in chapter 8.

6.3.7 Results Model

The results of using the model to test the hypothesis are presented in chapters 9 and 10.

The next phase of the CoSMoS process would be to validate the results model against the engineering domain of the experiment running on an embodied robot swarm, so that the domain model and platform model can be improved.

6.4 Conclusion

In this chapter we build a model of an embodied robot swarm for simulating the experiment from chapter 5, following the CoSMoS process described in section 6.2. In section 6.3.4 we note down everything we know about the embodied robot swarm. In section 6.3.5 we describe how this knowledge is put in the model, noting down any assumptions and areas to be calibrated later.

In the next chapter we build the soundboard e-puck extension, and in chapter 8 we calibrate the model developed in this chapter to have a more accurate representation of the soundboard.

6.4.1 List of Assumptions

Assumption 1: [39] describes acoustic science in enough depth for our model.

Assumption 2: Player/Stage adequately models robot movement physics.

Assumption 3: The e-pucks all have the same physical characteristics.

Assumption 4: No e-pucks have faulty motors or sensors.

Assumption 5: The microphones in the soundboard have perfect gain across all frequencies of interest.

Assumption 6: The change in sound intensity caused by sound waves reflecting off arena walls and corners is negligible.

Assumption 7: Obstacles in the environment (including other e-pucks) do not affect sound propagation.

Assumption 8: when an e-puck plays a tone it does not move.

Assumption 9: The e-puck is stationary whilst it listens for tones.

Assumption 10: Multiple tones of the same frequency combine as the mean of their polar coordinates.

Assumption 11: Tones from different frequency bands will not affect each other.

6.4.2 List of Calibration Points

Calibration 1: The resistance of the speaker on the soundboard.

Calibration 2: The arena size, shape and height.

Calibration 3: The shape and size of the soundboard.

Calibration 4: The soundboards all have the same characteristics.

Calibration 5: The microphone sample rate.

Calibration 6: The minimum tone frequency.

Calibration 7: The number of FFT bands.

Calibration 8: The range the e-puck can detect a tone.

Calibration 9: The amount of noise on the sound intensity measure.

Calibration 10: The amount of noise on the direction of arrival measure.

Chapter 7

Developing The Hardware

7.1 Introduction

In chapter 5 we describe the experiment we want to perform, and in chapter 6 we describe how it is modelled. In this chapter we describe the development of the "soundboard" audio hardware extension that enables the e-puck robots to communicate using audio tones.

Although our experiment will be carried out in simulation rather than on a real robot swarm, we develop robot hardware so that we can use it to calibrate the simulation. This is important because the physics of real world has a very strong influence on the performance of the audio communication. If we were to simulate how the hardware extension would work in theory, the simulation would not have a grounding in reality because the data used to create the simulation hasn't been subjected to the noise of the real world. By having a real artefact to simulate, it allows us to measure how the sensors and hardware react to the environment and gives more strength and confidence to the simulation as a realistic model of reality.

The e-pucks come with inbuilt microphones and a speaker. The microphones resemble small surface mounted integrated circuits, and they are soldered to the body of the e-puck pointing directly upwards, as shown in figure 7.1. Similarly, the e-puck's speaker is small and points directly upwards. The effect of the microphone positioning is that an e-puck is well equipped for detecting sounds from above it, but not with things that are at the same height as it, such as other e-pucks. Consequently the range for e-puck to e-puck audio communication is very small, at less than 10 cm [35]. The communication range is further reduced when we add a Linux extension board which allows us to run Player/Stage locally on the e-pucks [55]. This extension is vital for running the experiment on the e-pucks, but completely covers the e-puck's inbuilt microphones rendering them unable to reliably detect sound, as shown in figure 7.2.

We try two different approaches to developing the soundboard. In the first approach, in section 7.2, we try phased array beamforming. However, it was found to be unreliable, and the measurements of a sound's direction of arrival (DOA) to the nearest 45° were worse than random guessing. In the second approach, in section 7.3, we compare left and right microphone readings from soundboard to measure the DOA. Test results show that it correctly estimates the frequency of an audio tone 84% of the time, but it only correctly estimate the DOA (to the nearest 45 degrees) with 30% success. This does increase to 93% and 39% respectively if signalling frequencies are chosen to give the

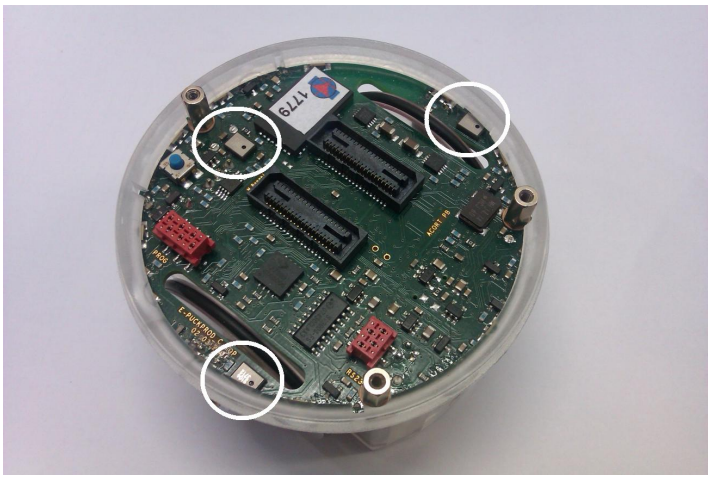


Figure 7.1: The positioning and size of the microphones built into the e-puck

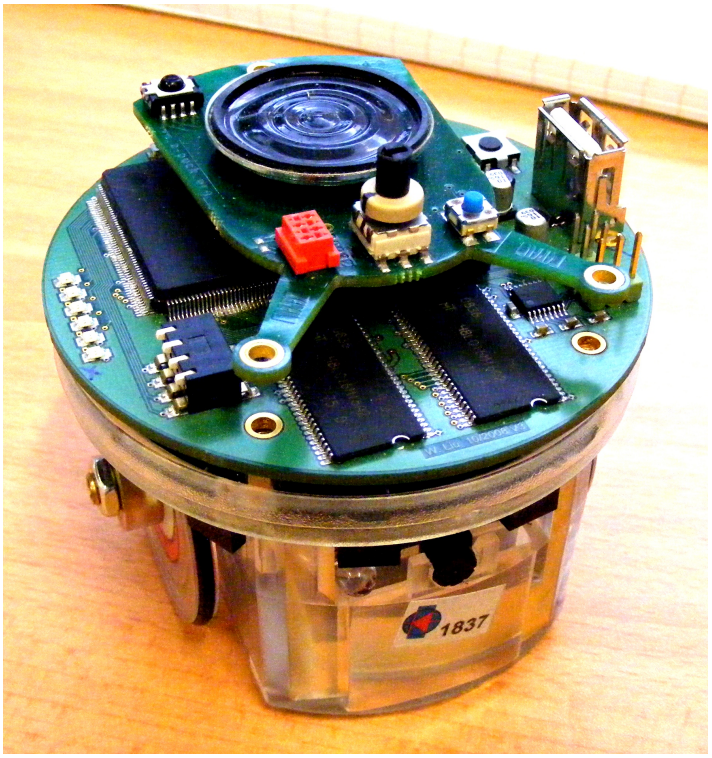


Figure 7.2: An e-puck with the Linux board extension [55]

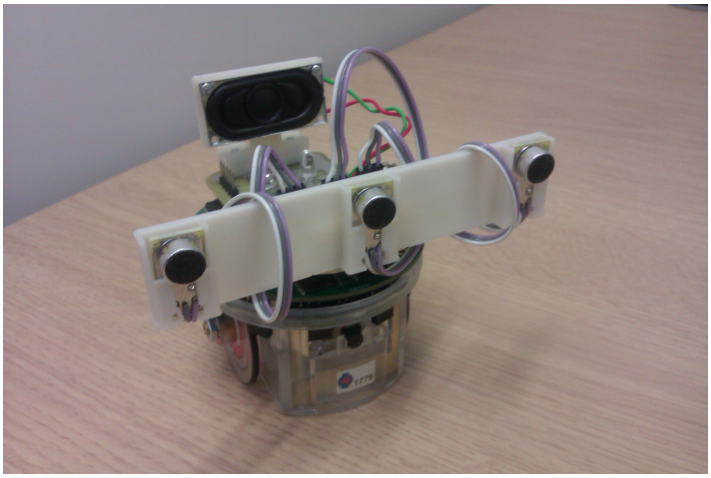


Figure 7.3: The completed soundboard.

best soundboard accuracy.

The final version of the soundboard is shown in figure 7.3.

7.1.1 Requirements

In section 6.3.4 we set out the basic requirements of the soundboard based on what we need the e-pucks to be able to do for our experiment. To summarise these are:

- The soundboard is an extension board for the e-puck which sits on top of the robot.
- It must be capable of sending and receiving at least nine different tones.
- It must be able to monitor the environment for sounds. If a tone is heard it must be able to calculate its frequency and the direction the sound came from.

There are also restrictions imposed by the e-puck itself:

- The battery powering the e-pucks is 3.3 volts, so microcontrollers and other components must all be able to run off this voltage.
- The soundboard must be small enough to fit on top of the e-puck.
- It must use as little power as possible to reduce battery consumption.
- The soundboard will be prototyped and fabricated by hand. It is possible to fabricate printed circuit boards but these must be assembled by hand, so all components must be large enough to be soldered manually.
- It must send data to and from the e-puck along an I^2C bus. The design should try to minimise the amount of data sent this way.

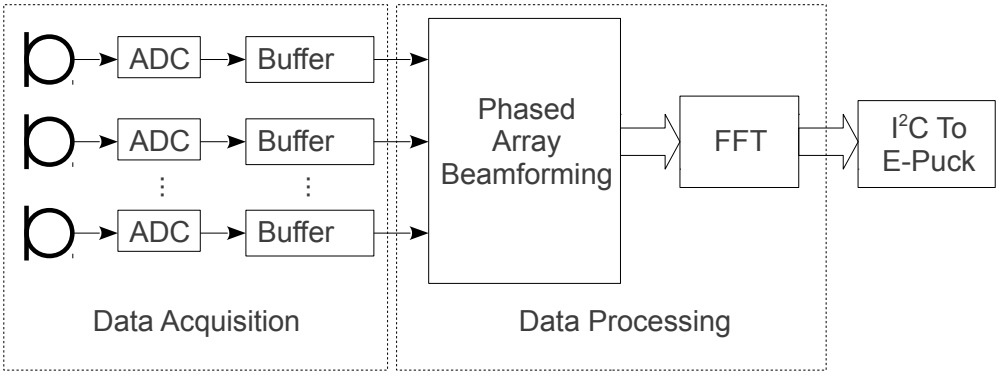


Figure 7.4: The process of measuring the sound information in the environment and sending the information to the e-puck.

7.2 Version 1: Phased Array Beamforming

In this section we describe our attempt to create a soundboard which uses phased array beamforming to detect the DOA of an audio tone.

We first give an overview of the process of acquiring audio information and preparing it for sending to e-puck. Then we describe phased array beamforming, and give some examples of it being used to measure the direction of a sound source. In section 7.2.3 we show the effect of changing the number of microphones and their positioning on the idealised frequency response of the beamforming, and use this information to decide on a microphone layout on the soundboard. Finally in this section we describe some tests for measuring the microphone sample rate and the soundboard frequency response and present the results of these tests.

7.2.1 Soundboard Process

Figure 7.4 shows the process that is followed to detect sounds. The data acquisition and processing in the soundboard will be performed by an 18F26K22 PIC microcontroller [59]. To acquire sound we use electret microphones with pre-amplifiers using a suitable circuit found online from Sparkfun Electronics [28]. The circuit diagram is given in appendix A.1. The output of the amplifiers goes into the PIC, which performs analogue to digital conversion and stores each microphone's reading into a memory buffer.

To detect the DOA of a sound and its frequency we will be analysing the recordings from the microphones using phased array beamforming. This part of the processing is discussed in section 7.2.2. The output of the phased array beamforming is then put into a 128 band Fast Fourier Transform algorithm. To perform the FFT we used the picFFT18 library from Alciom [51]. This library is capable of performing a fast Fourier transform on real valued data. The power spectral density of the FFT is found by squaring the real and imaginary FFT components and summing them for each FFT band. Finally the power FFT data is sent to the e-puck along an I^2C bus.

Once the e-puck receives all the FFT data, it will find the peaks in the power spectrum and compare their relative amplitudes to estimate the DOA. The circuit diagram for the extension is given in appendix A.2.

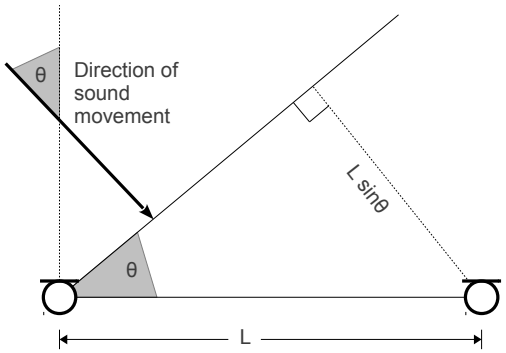


Figure 7.5: Calculating the distance a sound wave must travel before reaching successive microphones. L is the distance in metres between microphones and θ is the DOA of the wave. The source is assumed to be far enough away that the sound can be modelled as a planar wave.

7.2.2 Phased Array Beamforming

Phased array beamforming is a technique for measuring waves and determining their DOA. A simple microphone phased array uses several microphones placed along a flat plane. When sounds arrive at the microphones, depending on the DOA, the phase on the signal at each microphone is slightly different. This is because the microphones are in different places and so the sound will reach each microphone at a different time. We can calculate the time delay between the sound reaching each successive microphone using equation 7.1.

$$\text{delay} = \frac{L \sin \theta}{c} \tag{7.1}$$

Figure 7.5 shows how this equation relates to the physical microphone array. The distance between the microphones is L and measured in metres, c is the speed of sound, which is 343 m/s, and θ is the DOA of the sound. $L \sin \theta$ is therefore the distance, in metres, that the sound wave must travel before it reaches the next microphone, and dividing this by the speed of sound gives the time delay.

We can effectively steer the direction in which the microphone array "listens" by applying a delay to the signal from each microphone. These delays are calculated such that the sound will reach each microphone at the same time only if the sound source is in the direction we steered the array. The resulting delayed signals are then summed. If the sound source is in the direction in which the array was steered then the summation will give a peak, and the summation will be noise if the DOA was from a different angle. This process is called "delay and sum beamforming" and is illustrated in figure 7.6.

Benefits of Phased Array Beamforming

The principle of phased arrays can be applied to measuring any kind of waveform that propagates through a medium; including electromagnetic and acoustic waves. Phased array DOA estimation techniques are a useful area of research because of their potential applications: for example RADAR, SONAR or ultrasound scanning [48]. By using delay and sum to "point" the soundboard in any direction we can theoretically achieve

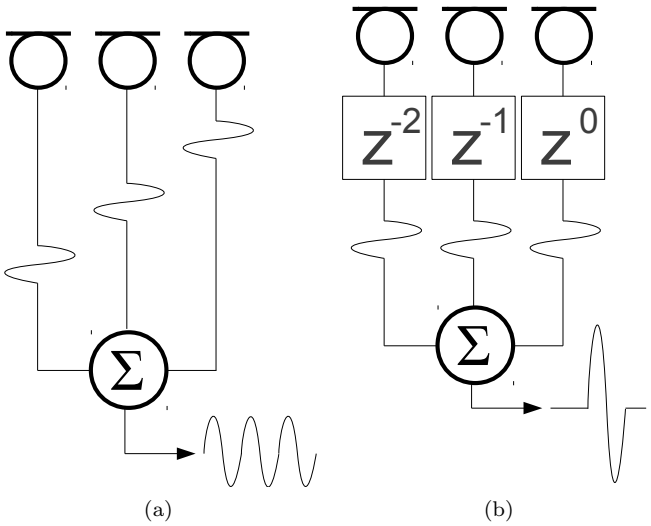


Figure 7.6: As a sound wave moves from left to right, it reaches each microphone in turn. Without delay (a), the sum of the signals is noisy. By delaying the signal from the microphones (b), the signals can reach the summation module at the same time, resulting in a peak in the summation. The amount to delay the signals by varies depending on the DOA of the sound wave.

a high degree of accuracy in DOA estimation, as has been demonstrated in the scientific literature.

In Valin et. al. [90] the authors create an array of 8 microphones on the corners of 50 x 40 x 36 cm cuboid. This is mounted onto a robot and is used for navigating the robot towards a sound source. The robot was tested by playing tones at it with a speaker, and the results are presented as the mean azimuth prediction error as it varies with distance from the source. The system gave an error of up to 3° over a distance of 3 metres. Dumbacher et. al. [26] compare several DOA estimation techniques, including delay and sum phased array beamforming, by using them to detect the sound source in a car wheel moving on a rolling road dynamometer. The results were not statistically analysed but presented as a image of the car tyre with the measured sound intensities super-imposed on top. The results correctly located the sound source as the lower front edge of the tyre. In [100] a four microphone phased array used to locate noise sources in a car. The equipment is capable of measuring the DOA in a three dimensional space, so gives not just the DOA azimuth, but its elevation too. The authors claim that *"in general, detection can achieve more than 90% correct rate"* [100].

The main disadvantage of delay and sum beamforming is that it requires a high sample rate on the microphones. The discretisation caused by the sampling means that we can only apply time delays that are some factor of the sample rate. With a higher sample rate, the duration of a single sample is shorter, so delays are more able to closely match the delay needed to steer the phased array. According to [63] *"frequently the inputs sample rate is five to ten times that required for waveform reconstruction"*. If our maximum expected frequency is 10 kHz then we need to sample at at least 20 kHz to be able to reconstruct the waveform, so for delay and sum beamforming our actual

sample rate must be at least 100 kHz to 200 kHz.

Our intended application of phased array beamforming is smaller than any we have found in the literature. To fit on top of an e-puck the microphones need to be close together, requiring very short beamforming delays and hence a very high sample rate. As we are using a PIC to do all our processing, this presents an upper limit on the capabilities of the hardware.

Delay and sum beamforming is the most basic form of phased array DOA estimation [63]. Considerable work has gone into developing more advanced signal processing techniques. Krim and Viberg [48] give a thorough review of these up until 1996, the most notable of which is the Multiple Signal Classification (MUSIC) algorithm [81]. MUSIC is capable of detecting multiple signals in the environment, and performs significantly better at DOA estimation than other popular algorithms [48, 81].

Despite MUSIC being a superior DOA estimator to delay and sum beamforming, we have chosen to use the simpler algorithm in our work. This is because the focus of our work is not on DOA estimation but the evolution of swarm robotic behaviour. The aim of this chapter is to develop hardware that is adequate for measuring a sound's frequency and DOA, without investing too much time on its development. Advanced signal processing is outside the scope of this work and investigation into this field is left as future research.

7.2.3 Design Considerations

Array Size

The frequency response of a phased array of microphones is dependent on both the number of microphones it contains and the distances between them. At certain combinations of sound frequency and DOA the extra distance the sound wave must travel to reach each successive microphone is a multiple of the wavelength. When this happens the signals from the microphones are in phase. The delay and sum beamformer will interpret this as if the DOA of the wave is the same as the steered angle, even when that is not the case.

Equation 7.2 gives the normalised summation of the microphone signals as would be received by a delay and sum beamformer with N microphones equally spaced at intervals of L metres, as a function of the input wave frequency f and DOA θ. The derivation of this equation is given in appendix A.3.

$$\frac{1}{N}\sum_{m=0}^{N-1}\exp(\frac{j2\pi fmL\sin\theta}{c}) \tag{7.2}$$

where c is the speed of sound, which is 343m/s at 20°C [39].

Figure 7.7 shows an example where there are 10 microphones at 20cm spacing. Ideally, the response will be at maximum for $\pm 0°$ over all frequencies, and at minimum at all other angles and frequencies. Meaning that, ideally, the response will be at maximum *only* when the steering angle matches the wave's DOA. Waves from other directions are completely ignored. What figure 7.7 shows is that the phased array is not ideal. Even with a large value of N and L there are certain combinations of frequency and DOA at which the phased array cannot be accurate. Furthermore, at low frequencies, regardless of the DOA, the response is near maximum. Meaning that the directionality of a wave cannot be measured this way if it is low frequency.

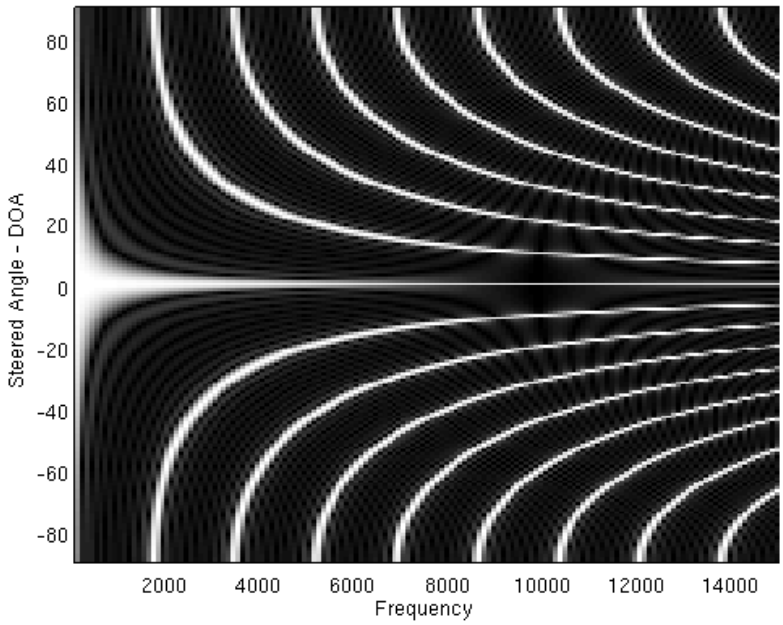


Figure 7.7: Example theoretical frequency response of a microphone phased array of 10 microphones spaced 20cm apart. The y axis is the DOA (in degrees) with respect to the angle in which the array is steered, the x axis is frequency of the sound wave. Areas of white indicate that the normalised summation is 1 and the signals are in phase. Areas of black are where the summation is 0 and the signals are 180° out of phase.

This lack of directionality at low frequencies is caused by the incoming wave's wavelength being larger than the array. The array size in metres is given by $(N-1)L$, which corresponds to a frequency of:

$$f = \frac{c}{(N-1)L} \tag{7.3}$$

At this wavelength, the microphone signals will cancel each other at $\pm 90°$ to the steering angle. Frequencies below this will not begin to properly cancel, and consequently the beamformer lacks directionality.

Figure 7.8 shows how changing the number of microphones and the spacing between them affects the frequency response of the phased array beamformer. It can be seen that increasing the number of microphones means that directionality can be measured at lower frequencies. Increasing the spacing between microphones makes the beamforming more precise, so DOAs just above or below the steered angle are more likely to be minimised.

The best design would have a very large number of microphones which are well spaced apart. However, we are constrained by the requirements of the hardware (given in section 7.1.1) that it must fit onto an e-puck, so we must pick values for N and L such that $(N-1)L \approx 0.07$, since the e-pucks have a diameter of 0.07m.

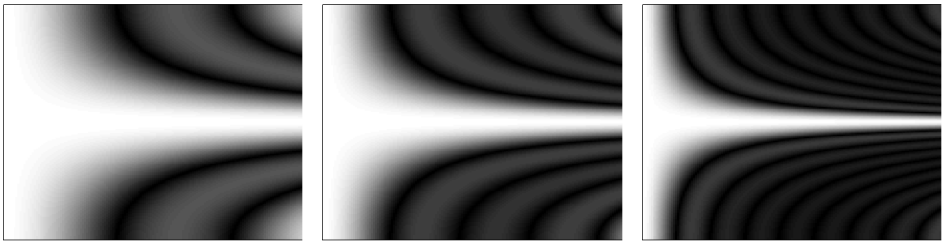
In section 7.2.2 we observed that a high sample rate is needed for the beamforming delays to be accurate. This imposes a limit on the number of microphones we can use because the PIC cannot process multiple analogue inputs in parallel. The PIC has only one analogue to digital converter, which must be multiplexed between different inputs in order to get readings from different sensors. Consequently the microphones can't be sampled at exactly the same time, but must be sampled consecutively so that corresponding points in the microphone memory buffers are as close together in time as possible. Therefore using additional microphones reduces the overall sample rate and this could negatively impact on the performance of the hardware.

The final phased array design uses 3 microphones spaced at 5cm intervals. The frequency response for a phased array with these parameters is shown in figure 7.9. These parameters are not ideal, but present the best performance within the given constraints. The array size is 10 cm, which gives a minimum frequency to get directionality of 3430 Hz. To measure a this frequency the array will need a sample rate of 6860 Hz. However, as stated earlier in section 7.2.2, we need to sample at a rate five to ten times this. Consequently the minimum sample rate of the microphones must be 34 kHz.

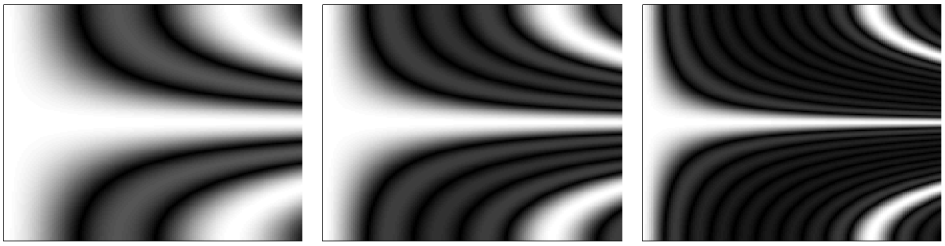
I^2C Data Transfer Timings

To reduce power consumption, beamforming will not be performed constantly but only on request from the e-puck. In order to present requested data to the e-puck in a timely manner and to limit the amount of data sent to the e-puck along the I^2C bus, we cannot perform delay and sum beamforming for every possible DOA but only for a small subset of angles. Consequently, we will only be testing angles $90°$, $45°$, $0°$, $-45°$ and $-90°$, as illustrated by figure 7.10. By only doing beamforming for five angles we also reduce the amount of time spent computing Fourier transforms, and there is much less data to send to the e-puck.

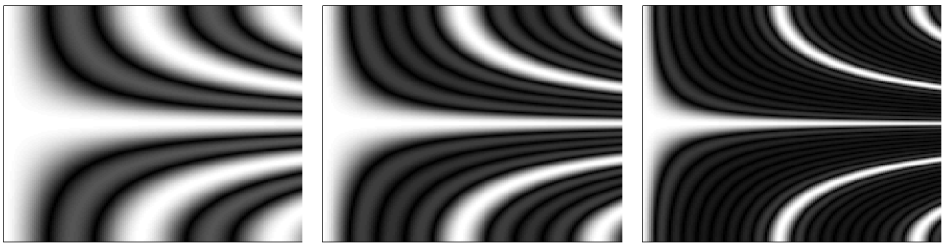
The FFT algorithm uses 128 points, so will return 128 data values. The first 64 are the power spectral density for each frequency, the last 64 are a complex reflection of the first 64 and can be ignored. To transfer the Fourier transforms of 5 different



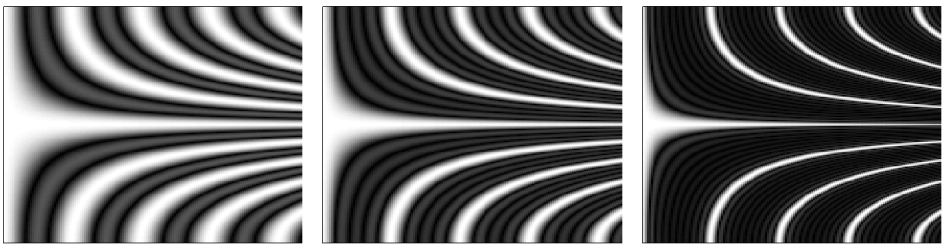
(a) 3 mics 2cm separation (b) 5 mics 2cm separation (c) 10 mics 2cm separation



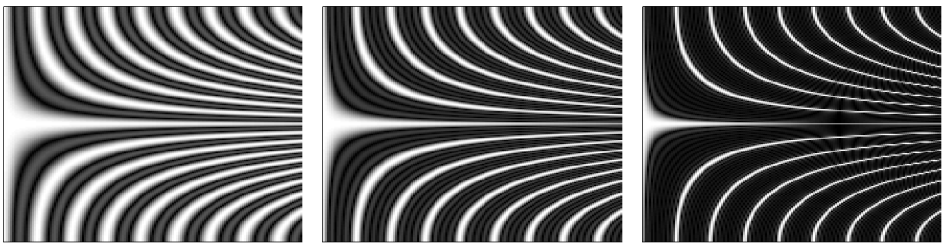
(d) 3 mics 3cm separation (e) 5 mics 3cm separation (f) 10 mics 3cm separation



(g) 3 mics 5cm separation (h) 5 mics 5cm separation (i) 10 mics 5cm separation



(j) 3 mics 10cm separation (k) 5 mics 10cm separation (l) 10 mics 10cm separation



(m) 3 mics 20cm separation (n) 5 mics 20cm separation (o) 10 mics 20cm separation

Figure 7.8: The effects of the number of microphones and the distance between them on the theoretical frequency response of the phased array.

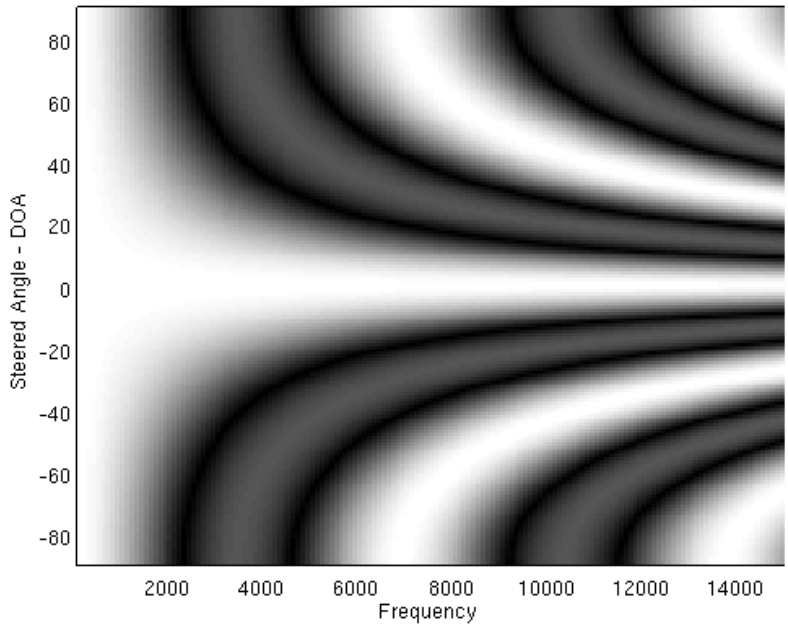


Figure 7.9: The frequency response of the microphone phased array. There are 3 microphones spaced 5cm apart.

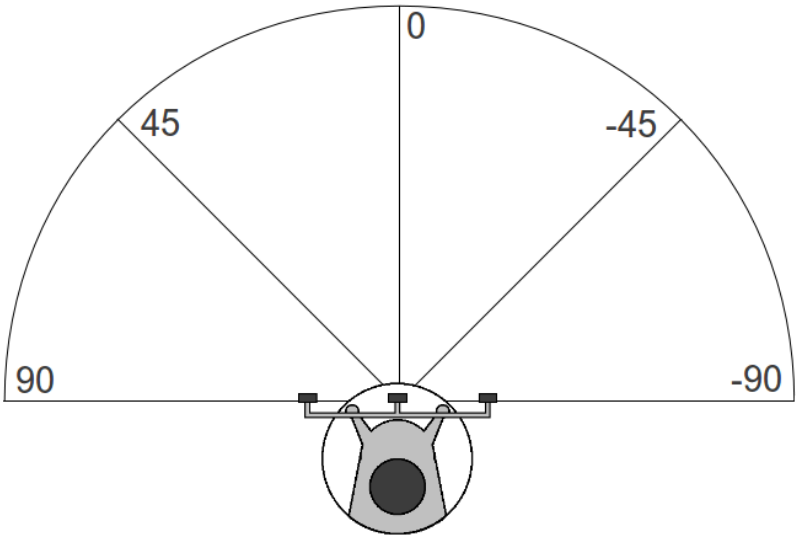


Figure 7.10: The angles for which delay and sum beamforming will be performed.

beamforming values along the I^2C bus we must send 640 bytes [1]. I^2C buses are clocked at 100 kHz, so it will take 51.2 ms to transfer the data.

This is quite slow, but not unworkable. It would certainly be faster to perform FFT analysis on the board and send the resulting tone frequency and DOA to the e-puck. However, there are two factors which influence our decision to not do this. The first is simply that the time between the e-puck requesting data, to the board having done 5 delay and sum and Fourier transforms, is long enough for the original I^2C request to time out. Secondly, it is far easier to re-program the e-puck than the soundboard. The analysis code may need to be calibrated to cope with different environments, so it is much more flexible if we can change parameters within the e-puck code and leave the board unchanged, rather than reprogramming the soundboard's PIC each time the swarm is moved to a new environment.

7.2.4 Verification Tests

There are two different tests we use gather data about the soundboard, so that we can verify that it measures frequency and DOA. We must measure the sample rate of the microphones so that we can perform delay and sum beamforming. We must also collect the FFT data from each of the beamformed angles so that we can analyse the data in MATLAB to find the best way of reducing it to a set of frequency and DOA estimates.

Sample Rate

To measure the microphone sample rate we played a 1 kHz tone at the soundboard and recorded the contents of the microphone buffers.

Figure 7.11 shows how the number of samples in the measured 1kHz wave, relates to the sample rate. If there are x samples per wavelength of the 1 kHz tone and f_s is the sample rate we want to find, then:

$$x\frac{1}{f_s} = \frac{1}{1000} \tag{7.4}$$

$$f_s = 1000x \tag{7.5}$$

To find x, the number of samples in 30 different wavelengths are counted. The mean value is then taken as a reading for x.

Measurement Accuracy

The aim of this test is to gather examples of FFT data that the e-puck driver must interpret into tone frequency and DOA information.

Figure 7.12 shows the setup for this test. A speaker connected to a signal generator is placed 20cm away from an e-puck equipped with the audio extension. The speaker is placed facing the robot, and a number of test tones are then played at a peak to peak voltage of 16V. This is repeated with the speaker at 90°, 45°, 0°, −45° and −90° DOA, so that for each combination of DOA and test tone frequency we have at least 30 FFT readings for every beamformed angle.

[1] each FFT point is stored as an integer, using 2 bytes. So 5 angles × 64 FFT points × 2 bytes per FFT point = 640 bytes.

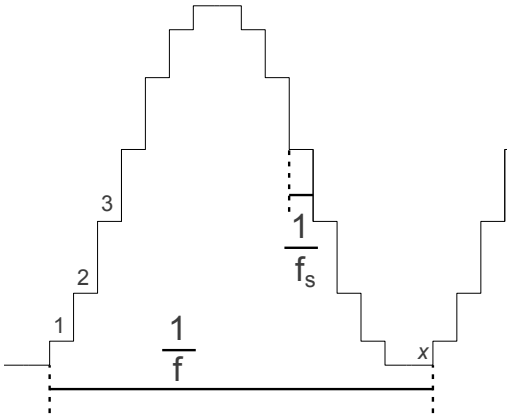


Figure 7.11: The relation between the known frequency played f and the unknown sample rate f_s. The wave shown is the data stored in the microphone buffer.

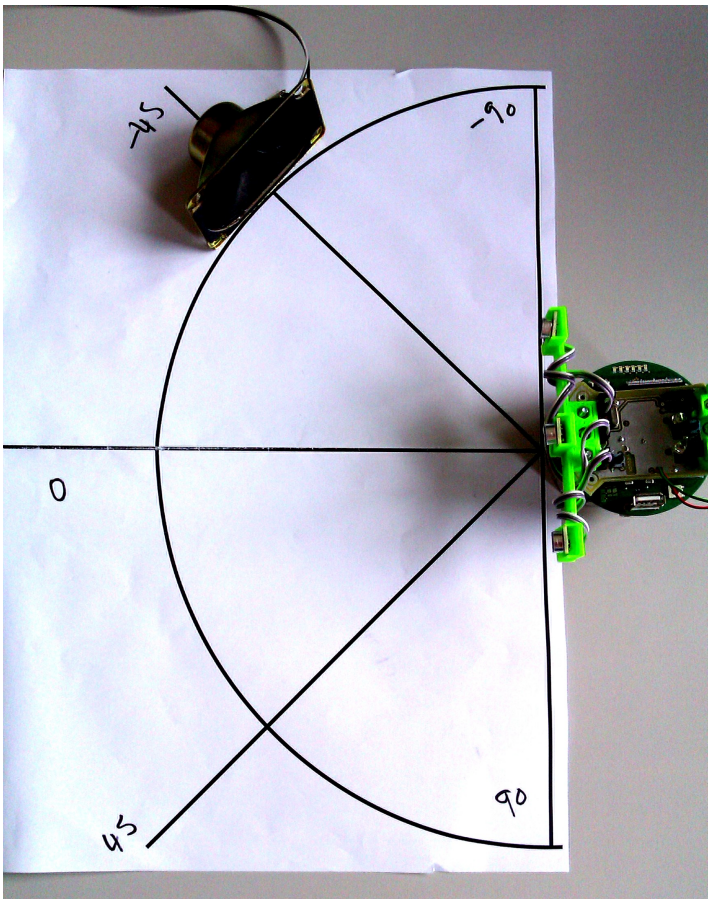


Figure 7.12: Photograph of the measurement accuracy test set up.

7.2.5 Test Results

The sample rate was found to be 38 kHz. At this rate, the delays for the beamforming algorithm are:

- 0°: No delay.
- ±45°: 103μs, which is 3.917 samples. This is rounded to a 4 sample delay.
- ±90°: 146μs, which is 5.539 samples. This is rounded to a 6 sample delay.

To measure the accuracy of the beamforming algorithm we used test signals of 500, 1000, 2000, 3000, 4000, 5000, 6000, 7000 and 10000 Hz. The resulting FFT data was then loaded into MATLAB for analysis.

By simply searching the FFT results for the highest peak and using its corresponding frequency and beamformed angle as the correct answer, we correctly estimated 83 frequencies and 128 DOAs out of a total of 495 FFT sets. This corresponds to a success rate of 16.7% and 25.9% respectively. By only checking the FFT bands of the test frequencies, and by looking for the band where the *sum* of the FFT results was largest, we were able to increase the frequency estimation success to 327 out of 495 which is 66.0%. However, using the beamformed angle with the highest value at that frequency gave a success rate of only 17.2% (85 correct out of 495), which is worse than random. Neither of these methods are reliable, and they make the assumption that there is always exactly one tone being played, which is not always the case.

The unreliably of the beamforming in accurately estimating the correct audio information is most likely due to the compromises made in its design. The array size of 3 microphones spaced 5cm apart is the largest that can be implemented within the size and processing speed limitations, but it still has poor directionality. Another factor may be that we only perform delay and sum beamforming for a small number of DOAs, but there is a non-linear relationship between the beamforming result and the DOA. Re-examining the frequency response in figure 7.9 we can see that DOAs that are close to the steered angle tend to have a smaller response, especially at higher frequencies. This non-linearity makes it difficult to correctly analyse the results from the extension. However, in our test we placed the sound source along each of the measured DOAs, so this effect should have been mitigated. If we were to repeat the test using DOAs different to the ones specifically measured (for example ±30° or ±60°) this problem would become more pronounced.

7.3 Version 2: Microphone Amplitude Comparison

To try and get better results, the beamforming was removed entirely from the soundboard. The revised process is shown in figure 7.13. Instead of beamforming, the FFT of each microphone buffer is sent to the e-puck and no other processing occurs on the extension. Since we are only removing signal processing, no changes need to be made to the board's circuitry or the microphone positioning.

The e-puck determines the frequency and DOA by comparing the power density FFT measures for each of the microphones. The microphone where the FFT power density is largest is assumed to be closest to the sound source. This is essentially the same as comparing the amplitude of the signals measured at each microphone to decide a sound source's DOA.

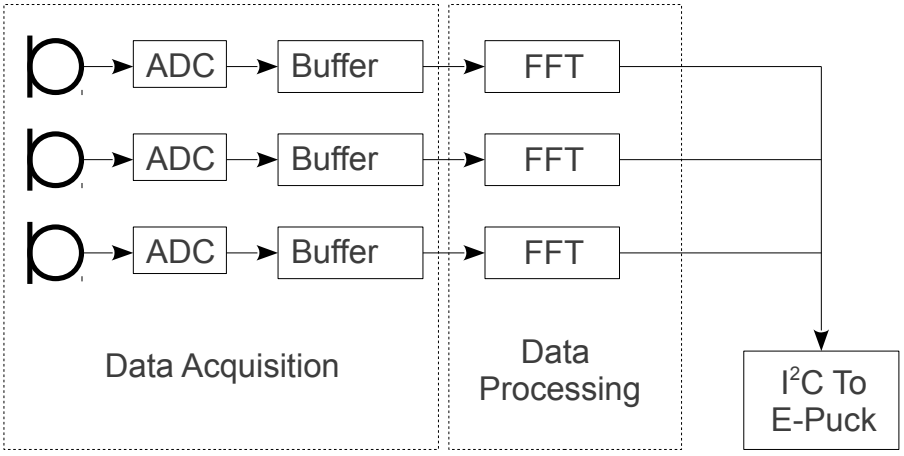


Figure 7.13: The revised process for measuring the sound information in the environment. Instead of sending the FFT of delay and sum beamforming data, as in figure 7.4, we now send the FFT of the microphone readings.

This process has the advantage that it does not require a high sample rate to work and has no lower limit on the frequencies it can detect. Consequently, we can use the e-puck's sound synthesis module to play signalling tones. We can also lower the microphone's sample rate so that the FFT bands are narrower and the soundboard can distinguish between tones that are closer together in frequency. However, at short range the signals may be equally loud and at long range equally quiet; such that the accuracy of comparing amplitudes to estimate the DOA is dependent on the distance of the soundboard from the source.

7.3.1 Creating a Sound

The e-pucks already have the hardware for playing a sound that has been pre-recorded and stored in program memory [1]. The Glowbot project [40] extends this, with drivers capable of wavetable synthesis. In wavetable synthesis, a single waveform is stored in a block of memory, N samples wide. When a tone is played, the synthesiser increments through the wavetable at sample rate f_s. However, the synthesiser does not play every sample from the wavetable but skips several. Depending on the number of stored samples skipped each time the wavetable is accessed, the frequency that is being played can be varied. Equation 7.6 gives the formula for the number of samples to skip each time as a function of the sample rate f_s, the wavetable size N, and frequency that the synthesiser must play f:

$$\text{skip} = N\frac{f}{f_s} \tag{7.6}$$

The speaker sample rate is limited to 7200 Hz by the e-puck hardware (section 6.3.4), this means that the maximum frequency that can be produced by the e-puck is 3600 Hz.

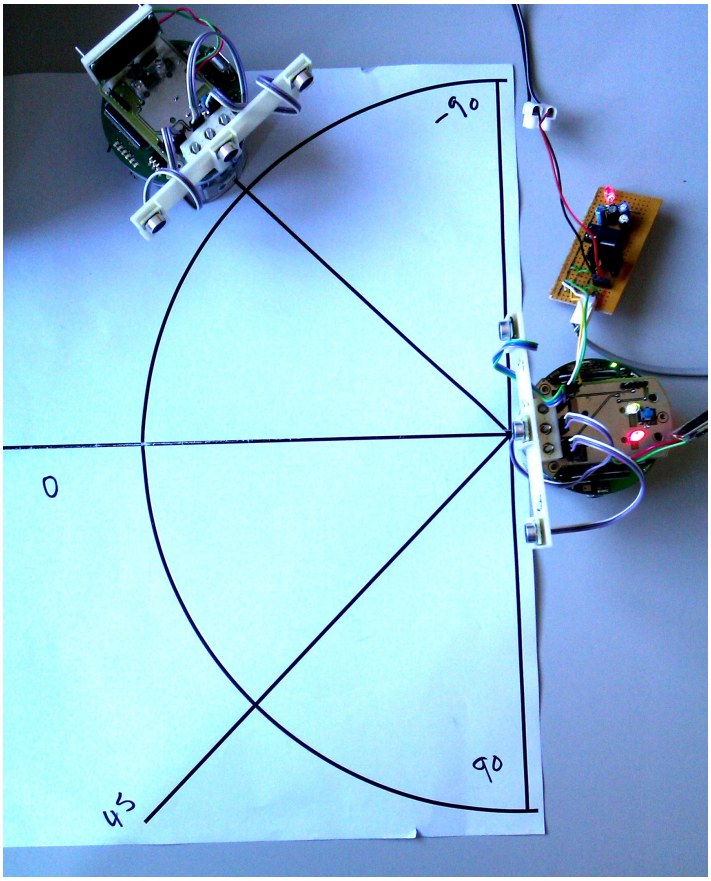


Figure 7.14: Photograph of the measurement accuracy test set up. Unlike figure 7.12, an e-puck with a soundboard can be used as the sound source to give results that are more representative of the measurements that would be taken in am embodied, multi-robot experiment.

7.3.2 Test Results

The microphone sample rate needs to slightly larger than the speaker sample rate so that it can detect all the possible playable tones, without being so large that there is unnecessary processing. Following the procedure described in section 7.2.4 the microphone sample rate was adjusted to be 8 kHz.

To measure the accuracy of the extension we performed the test described in section 7.2.4. However, instead of using a speaker and signal generator all tones were generated by another e-puck, so that the data recorded is a realistic example of the data the extension might receive when it is used in our experiment. Figure 7.14 shows a photograph of the experimental set up using an e-puck as the sound source.

We used test signals of 530, 720, 900, 1100, 1280, 1470, 1650, 1840, 2030, 2220, 2400, 2600, 2780, 2970 and 3150 Hz. These values were chosen because they are the mid-points of the FFT bands, and between each band with a test signal there are two bands with no expected signals. This is to reduce interference between adjacent FFT bands. Figure 7.15 shows an example of the data sent to the e-puck using the new

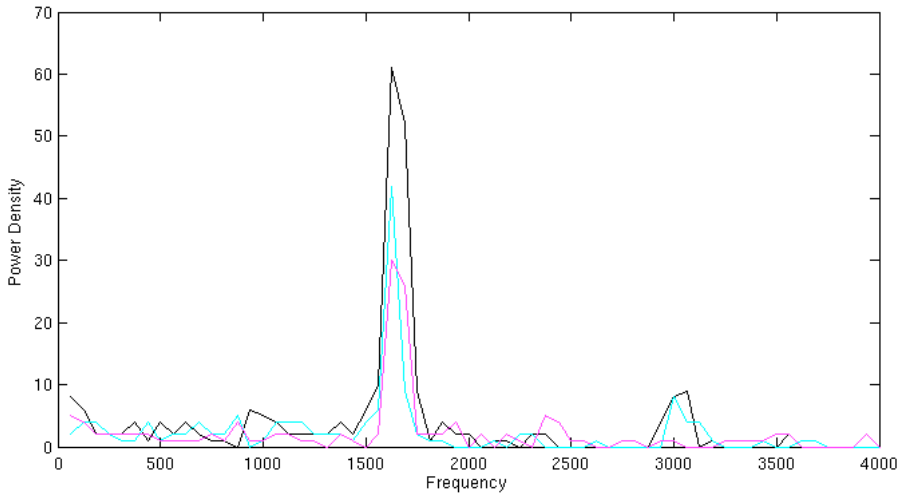


Figure 7.15: Example FFT of the microphone buffers. Results shown here are for a frequency of 1650 Hz and DOA 90°. The line with the highest peak is the FFT of the microphone closest to the source, and the line with the lowest peak is for the microphone furthest away.

version of the hardware. It can be seen that the peaks in each FFT response occur in the same place for each microphone, the size of the peak is greater the closer the microphone is to the sound source.

To determine whether any tones were heard, and the frequencies of any tones the following pseudo-code was used:

1. Find the mean of each microphone's FFT data
2. For all bands in FFT where a test tone would be detected:
    - Mark mic1 tone as heard if: mic1 FFT result for this band > freq threshold factor × mic1 mean.
    - Mark mic2 tone as heard if: mic2 FFT result for this band > freq threshold factor × mic2 mean.
    - Mark mic3 tone as heard if: mic3 FFT result for this band > freq threshold factord × mic3 mean.
    - If two of the three microphones heard a tone, then a tone was heard in this FFT band.

The freq threshold factor parameter controls how sensitive the soundboard is to the loudness of the tones played. The advantage of this method of analysing the data is that it doesn't assume that exactly one tone is played at all times, so it can also detect if there were none or multiple signals. It also uses a threshold that is relative to the noise level, so is somewhat robust to changes in the ambient noise environment.

To estimate a tone's DOA we compared the peaks in the left and right microphone FFT data at the bands where a tone was detected. If there is a small difference then the source is estimated to be 45° in the direction of the louder microphone reading. If there is a large difference then the source is assumed to be at 90° to the robot. If there

is no significant difference then the source is estimated to be in front of the robot. The DOA estimator is calibrated by finding a "microphone difference factor" (MicDifference) which is a multiple of the reading from the microphone's FFT and determines the range around which the microphone readings can be compared. This process is illustrated in figure 7.16.

As with the frequency estimation, the DOA estimation uses thresholding which is relative to the FFT result, so if the FFT reading for all the microphones in that band were low or high, the algorithm would scale accordingly. The MicDifference parameter controls how sensitive the estimation algorithm is to differences in the microphone FFT results. It is the percentage difference that microphone readings must be in order to qualify as different.

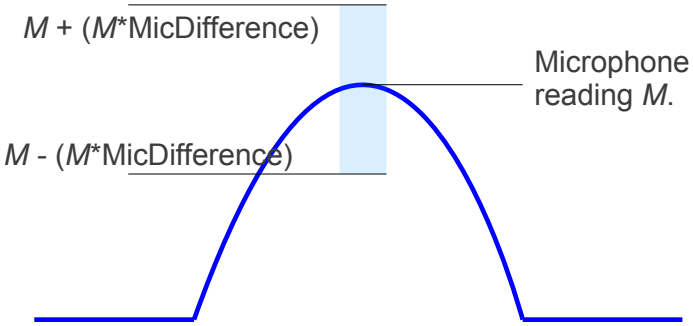
The FreqThreshold and MicDifference parameters were 2.4 and 0.32 respectively, and these were chosen to give the largest possible correct score for the data set we tested. When this estimation algorithm was applied to the data from our accuracy test, it was able to correctly guess the frequency 696 times out of 825 data sets (84%), and correctly guessed the DOA 249 times out of 825 (30%). These values, particularly the DOA, are not ideal but are at least better than random. However, the success rate is not consistent over all test frequencies and DOAs, as shown by figure 7.17. Figure 7.17a shows that the $\pm 45°$ angles were measured with the least accuracy. This is an artefact of how we determine the DOA from the FFT data. For $\pm 45°$ to be guessed, the FFT readings must be within a certain range of each other. For $0°$ the range of differences can be larger, and for $\pm 90°$ it can be larger still. The lack of accuracy in the $\pm 45°$ tests is consequently due to noise on the FFT readings pushing the differences outside of the range where $\pm 45°$ would be guessed, causing an incorrect estimation.

The mid-range frequencies of around 1 kHz to 2 kHz, gave the most accurate estimates for both frequency and DOA. This is most likely because the speaker is not capable of playing frequencies below 200 Hz, and resonates at 400 Hz [2]. Additionally, the low sample rate causes the wavetable synthesiser to generate the higher frequency tones with a squarer waveform than lower frequencies. This introduces unwanted harmonics into the tone which may interfere with the FFT, causing erroneous readings. At the tone mid-range frequencies, both these effects are lessened and there is less unwanted noise in the microphone readings. In chapter 5, we establish that we need 9 different signal frequencies in order to perform our tightly coupled communication. From the results shown in figure 7.17b, the frequencies 720, 900, 1100, 1280, 1470, 1650, 1840, 2030, and 2220 Hz are chosen for the experiment. These signals give the best combination of frequency and DOA success rate, for an overall frequency successful estimation rate of 93% and DOA successful estimation rate of 39%.

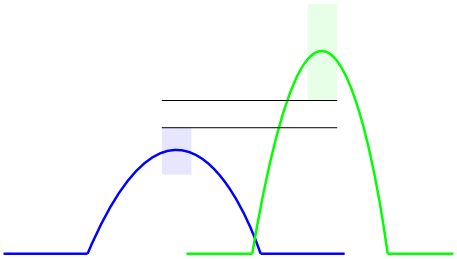
7.4 Conclusion

In this chapter we describe the process of developing the soundboard extension to the e-puck robots that must be functional within the constraints on the extension's physical size, processing speed and memory size. This soundboard is capable of monitoring the audio environment and interpreting it into frequency and DOA information.

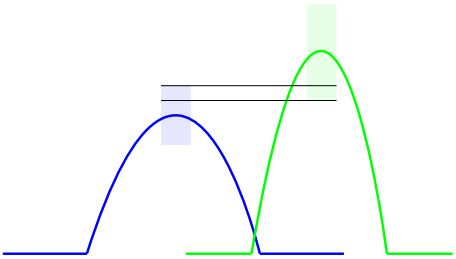
Our initial attempt at building the soundboard involves delay and sum beamforming. Although this method is theoretically promising, the implementation is too limited by the physical constraints of the hardware to be effective. It may be that more advanced DOA estimation algorithms would produce better results, but these would still need



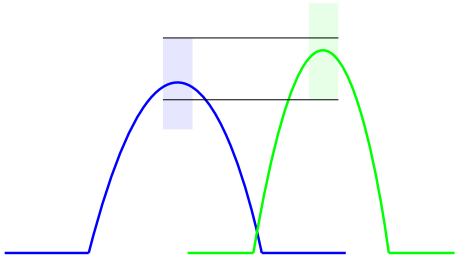
(a) The microphone difference factor (MicDifference) is used to generate a boundary around the microphone reading M.



(b) When there is no overlap between boundaries the DOA is estimated as 90° in the direction of the larger microphone reading

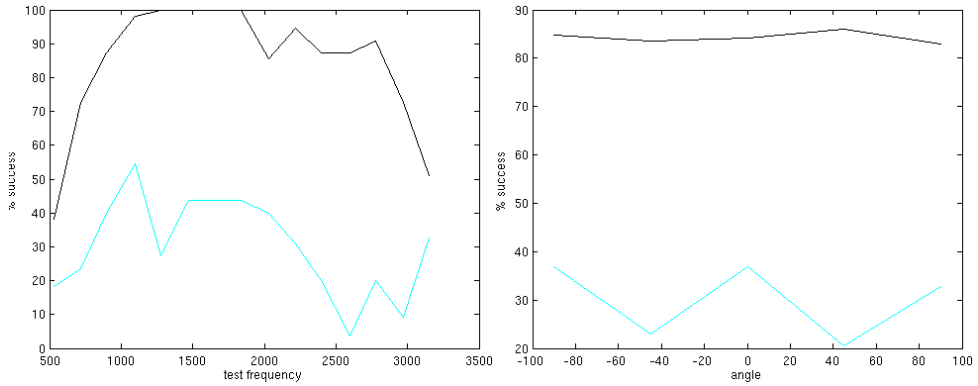


(c) When there is less than 50% overlap the DOA is estimated at 45° in the direction of the larger microphone reading



(d) With more than 50% overlap the DOA is estimated at 0°

Figure 7.16: Using the microphone difference factor (MicDifference) to estimate the DOA of a signal.



(a) Estimation success with respect to test tone frequency.

(b) Estimation success with respect to test tone DOA.

Figure 7.17: The success rate of the extension board as it changes with test tone frequency and DOA. The black lines indicate the frequency estimation success, and the lighter, cyan lines the angle estimation success.

to function within the size and processing constraints described in section 7.1.1. The second attempt uses a more simple system of comparing the signal amplitudes of three microphones to estimate which is the closer to the sound source. Tests show this to be a more reliable solution, and the 9 signalling frequencies are chosen to give a success rate of 93% for frequency and 39% for DOA.

The problems with the initial solution shows the value of co-developing the hardware and simulation alongside each other. If we had simulated the extension as being as reliable as beamforming would be under better conditions then the simulation would be wrong. By developing the hardware and then calibrating the simulation to model it, we prevent the simulation from becoming too idealised. The simulation stays grounded in reality and can always be compared to a real artefact.

In chapter 8 the model from chapter 6 is calibrated to better incorporate our new knowledge of the soundboard. Now that it is possible to take readings and measurements with the soundboard we can get a much better understanding of the acoustic and audio environment we are modelling. In this chapter we have already measured the soundboard's accuracy with different combinations of signal frequency and DOA. To calibrate the model we should also measure:

Calibration 11: The soundboard accuracy as it gets further from the sound source.

Calibration 12: The soundboard accuracy in the presence of two or more tones.

Chapter 8

Calibrating the Model

8.1 Introduction

In this chapter we address the calibration points from chapters 6 and 7, so that we can build a more realistic model of the soundboard for our simulation, which better represents the measurable characteristics of the hardware.

To achieve this we perform several tests on the soundboard to measure its audio characteristics, the results of which are fed back into the model to improve its realism. Through these tests it emerges that the soundboards developed in chapter 7 are very unreliable and erratic, so two different soundboard models are developed; one that is idealised and not subject to noise, and one that is noisy and more realistic. Both models are much simplified versions of the real soundboards, but the simplifications allow the hypothesis of this work to be tested.

We begin this chapter by addressing the calibration points answered during the development of the soundboards. In section 8.3 we compare two soundboards to see if they produce equivalent readings under the same test conditions. In section 8.4 we measure a soundboard's response to changing the direction of arrival (DOA) of a test tone. In section 8.5 we test the soundboard using two simultaneous tones, then in section 8.6 the response of the soundboard to the test tone distance. Finally in section 8.7 the results of these tests are fed back into the simulation to produce both realistic and idealised models of the soundboard.

8.1.1 List of Calibration Points

This is a summary of all the calibration points raised in chapters 6 and 7:

Calibration 1: The resistance of the speaker on the soundboard.

Calibration 2: The arena size, shape and height.

Calibration 3: The shape and size of the soundboard.

Calibration 4: The soundboards all have the same characteristics.

Calibration 5: The microphone sample rate.

Calibration 6: The minimum tone frequency.

Calibration 7: The number of FFT bands.

Calibration 8: The range the e-puck can detect a tone.

Calibration 9: The amount of noise on the sound intensity measure.

Calibration 10: The amount of noise on the direction of arrival measure.

Calibration 11: The soundboard accuracy as it gets further from the sound source.

Calibration 12: The soundboard accuracy in the presence of two or more tones.

8.2 Calibration Points Already Answered

Several of the calibration points raised in chapter 6 have already been answered in chapter 7 as the soundboard hardware is developed. In this section we briefly repeat the answers to these calibration points.

Calibration 1: The resistance of the speaker on the audio hardware.

The speaker has a resistance of 8Ω.

Calibration 2: The arena size, shape and height.

The dimensions of the arena remain variable, as the dimensions can easily be changed in simulation and in reality. In our initial simulation we use an arena of 2 metres by 2 metres, but if this is found to be too large or small for the number of e-pucks in the arena it can still be changed.

Calibration 3: The shape and size of the soundboard.

The soundboard had three microphones on it, spaced 5 cm apart, giving it a total length of 10 cm. Figure 8.1 shows the simulated e-pucks with a soundboard.

Calibration 5: The microphone sample rate.

The microphones are all sampled at 8000 Hz.

Calibration 6: The minimum tone frequency.

The minimum tone frequency used is 720 Hz.

Calibration 7: The number of FFT bands.

There are 128 bands in the soundboard's FFT. Combined with the sample rate of 8000 Hz, this means the soundboard is capable of detecting frequencies between 0 and 4000 Hz, and each FFT band has a width of 62.5 Hz.

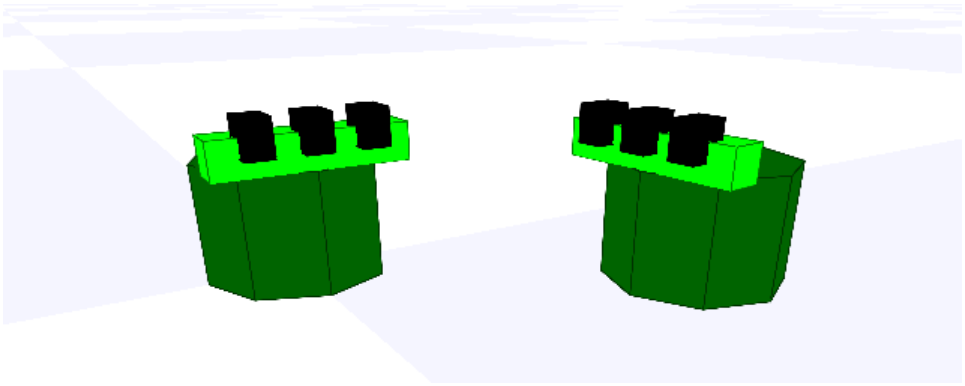


Figure 8.1: Image of the simulated e-pucks with soundboards.

8.3 Comparing Soundboards

Calibration 4: The soundboards all have the same characteristics.

The purpose of this test is to learn whether physical copies of the soundboard give equivalent readings to one another. This is so that in our simulation we will know whether the simulated soundboards can be identical copies of each other, or whether each one must be given different noise characteristics. If different noise characteristics are required, this test should indicate areas in which they should differ. For example, whether there are significant variations between soundboards at any particular frequency or direction of arrival (DOA).

Two soundboards have been constructed following the work presented in chapter 7. Ideally we would use more for this test, but the number of boards we are able to make for this experiment is limited by the resources and time they take to build. For this work, it would take too much time to construct the additional soundboards when the focus of our work is to test the hypothesis from chapter 4 rather than the construction of an e-puck extension. With results from only two soundboards we will have to assume that both boards are typical, and that our results can be extrapolated to apply to all other potential soundboards. So, if the two boards are found to be similar we assume all soundboards are similar and can be simulated in the same way. If the soundboards are different we assume that all soundboards have such variations and the simulation is adjusted accordingly.

This test is imperfect but a useful measure of the soundboard capabilities. It may be that the assumption about both boards being typical is true, so that the simulation of the soundboards will be good enough to test the hypothesis. The testing and comparison of additional soundboards is left as future work.

Assumption 12: The two soundboards used to calibrate the model are representative of all soundboards.

8.3.1 Experimental Method

To test each soundboard, we play several different test frequencies, at varying DOAs to the soundboard under test, and record the FFT data sent from the soundboard to the e-puck.

Soundboard	Parameter	Value	Success Rate (%)
board 1	Frequency threshold factor	2.7	90.4
	Microphone difference factor	0.18	32.9
board 2	Frequency threshold factor	2.4	86.9
	Microphone difference factor	0.34	16.1

Table 8.1: The frequency threshold and microphone difference factors for each soundboard, to give the best success rate.

The frequencies used to test the soundboards are the signalling frequencies decided upon in chapter 7, of 720, 900, 1100, 1280, 1470, 1650, 1840, 2030, and 2220 Hz. The sound source is an e-puck equipped with its own soundboard at a distance of 20cm. DOAs of $-90°, -45°, 0°, 45°$ and $90°$ are measured. For each combination of test frequency and DOA, 28 data readings are recorded, each reading consisting of the FFT results from each of the three microphones on the tested soundboard.

8.3.2 Results

Before analysing the measured results, the algorithm which determines frequency and DOA information from the raw data needs to be calibrated. This algorithm is described in section 7.3.2 and consists of:

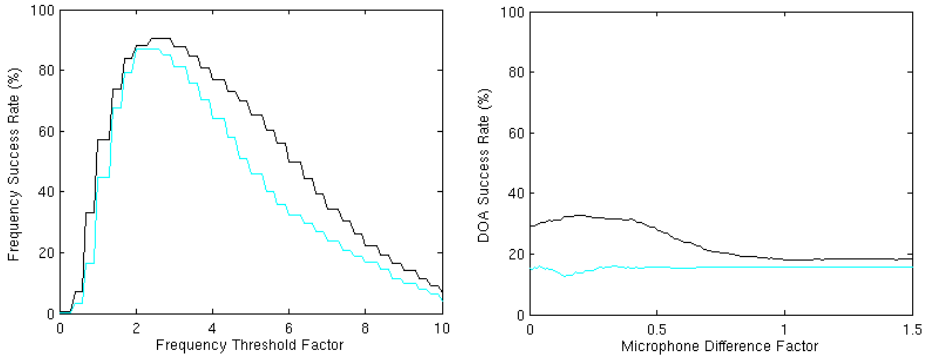
1. **Frequency threshold factor**. An amount to multiply the mean FFT reading for each microphone by to get a threshold. If, two or more microphones have a result over this threshold at any point along the frequency spectrum where we know a test frequency may be, then a tone is detected.

2. **Microphone difference factor**. A factor controlling how similar the left and right microphone readings must be to estimate DOA.

For calibrating the soundboards, we measured 10 runs of each combination of signal frequency and DOA and recorded the FFT readings from each microphone. This data is separate from the data used in this test so that the soundboards are not calibrated on the data being used to test them. We tested the calibration data with frequency threshold factors of 0 to 10, and microphone difference factors of 0 to 1.5. The frequency and DOA estimation success rates, as a function of these parameters, are shown in figure 8.2. The best parameters and the corresponding success rate for each soundboard are shown in table 8.1:

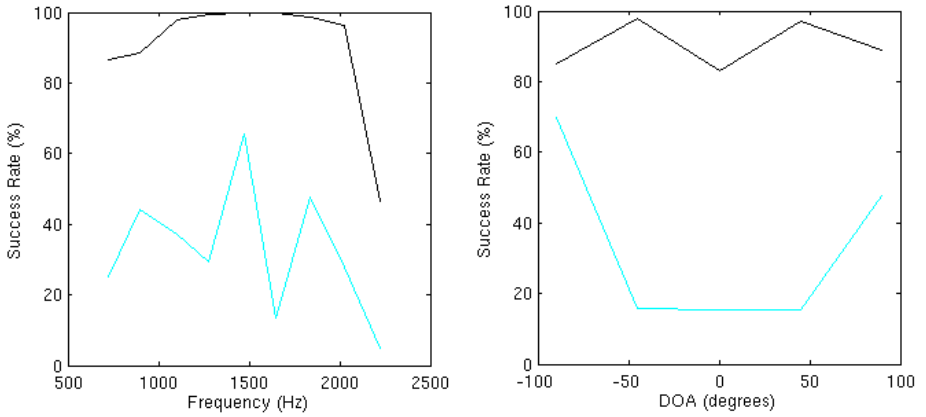
Figure 8.2b indicates that the soundboards may be similar when predicting frequency, but will have different performances when predicting DOA. Using the best performing estimation algorithm parameters, given in table 8.1, figures 8.3 and 8.4 show the performance of both soundboards as a function of test frequency and DOA. It can be seen that:

- The frequency estimation accuracy of both soundboards appears similar.
- Soundboard 1 performs better than soundboard 2 at estimating the DOA.
- Soundboard 1 and board 2 have different responses when estimating DOA. While board 1 is more accurate when the DOA is $\pm 90°$, board 2 is more accurate with DOAs of $-45°$ and $0°$.



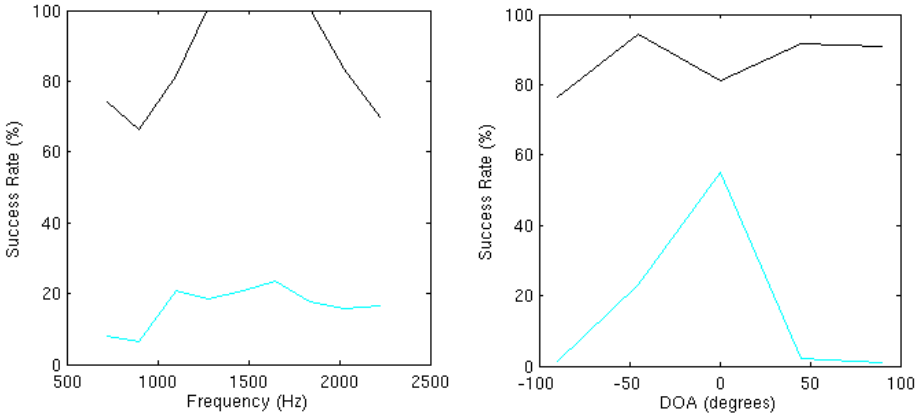
(a) (b)

Figure 8.2: The effects of changing the frequency threshold factor on the frequency estimation success rate (a) and the microphone difference factor on the DOA estimation success rate (b). The black line is the success rate for soundboard 1 and the lighter line for soundboard 2.



(a) (b)

Figure 8.3: Success rate of soundboard 1 estimation algorithm with respect to test tone frequency (a) and DOA (b). The black lines indicate frequency success rate and the lighter lines the DOA success rate.



(a) (b)

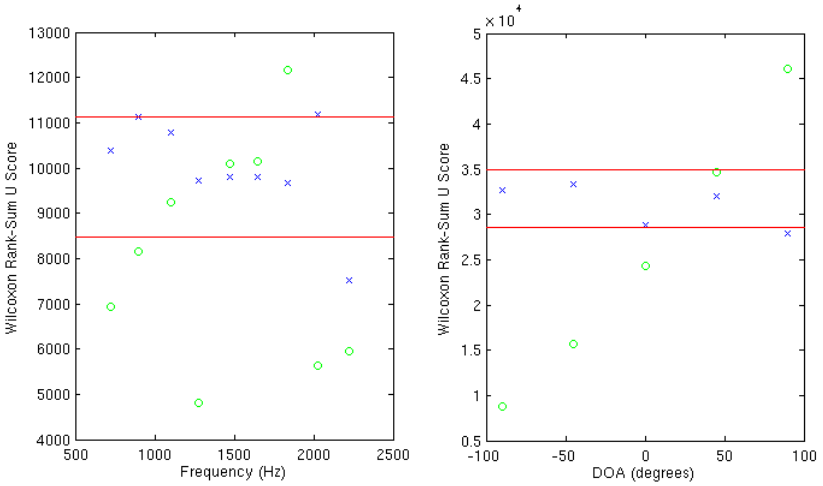
Figure 8.4: Success rate of soundboard 2 estimation algorithm with respect to test tone frequency (a) and DOA (b). The black lines indicate frequency success rate and the lighter lines the DOA success rate.

The frequency and DOA estimations from each soundboard were compared using a Wilcoxon rank-sum test [64]. The Wilcoxon rank-sum test is a non-parametric statistical variance test to determine whether two sample sets of data are could have been taken from the same overall set. It tests the null hypothesis that the two samples are from the same set. A score U of the comparison of the sample sets is generated, as well as a mean and standard deviation for the expected U that would be seen if the two sample sets are the same. The Wilcoxon rank-sum tests the null hypothesis that the sample sets are from the same source. If U falls within acceptable confidence limits then this null hypothesis holds. [64]

For the comparison of the two soundboards, we compare the boards' frequency and DOA estimations with respect to the test frequency and test DOA. These results are shown in figures 8.5a and 8.5b respectively. We use 95% confidence limits to judge the Wilcoxon rank-sum results. Where the U score for frequency or DOA estimations fall outside these limits indicates where the soundboards have statistically significant differences in characteristics.

Examining the frequency estimations first, figure 8.5b shows that both soundboards give similar results as the test tone's DOA is varied; only at 90° do they significantly differ. As test tone frequency is changed (figure 8.5a), the soundboard characteristics show more variation. From figures 8.3 and 8.4 we can see that for mid-range frequencies of 1280 to 1840 Hz both boards had near 100% accuracy, hence Wilcoxon rank-sum values of $U \approx \mu$ at these frequencies. However, at higher and lower frequencies, the soundboards were both less accurate, causing more variation in their results and pushing the U comparison further from the mean. Only the readings for 2030 and 2220 Hz differed enough to fall outside the Wilcoxon rank-sum 95% confidence limits, so for input tones of 720 to 1100 Hz both boards were inaccurate, but at least consistent with each other.

In figure 8.5b the DOA estimation shows an interesting linear trend. This is due to the DOA estimations of board 2 not changing as a results of input tone DOA or frequency. Figure 8.6 shows the probability density function graph of all 1260 of sound-



(a) Comparison of tone frequency estimations. Sample size is 140 for each board, the Wilcoxon rank-sum $\mu = 9800$ and $\sigma = 677.5$.

(b) Comparison of tone DOA estimations. Sample size is 252 for each board, the Wilcoxon rank-sum $\mu = 31752$ and $\sigma = 1635$.

Figure 8.5: Wilcoxon rank-sum results between board 1 and board 2 results for frequency estimation (marked with $\times$) and DOA estimation (marked with $\circ$). The horizontal lines indicate the 95% confidence limits.

board 2's DOA estimations. Ideally the bars should be equal, but 70% of all DOA guesses were 0° and 26% were −45°. This tendency to estimate either 0° or −45° did not change, regardless of the correct DOA. Consequently, soundboard 2 is useless for correctly estimating a tone's DOA. The high accuracy at inputs of 0° shown in figure 8.4b is just a result of this tendency to guess 0° 70% of the time.

A new approach to DOA estimation is required to try and improve the soundboards' accuracy.

8.3.3 Frequency and DOA Estimation From FFT Amplitude and Differential

In our analysis of the soundboards so far, we have been examining the FFT data only after applying our own processing techniques to it. Information contained in the FFT readings for each microphone has been filtered and reduced to give an estimate of the tone frequency and DOA. It may instead be that the work of processing the raw FFT information can be left for the evolutionary algorithm.

Instead of our approach so far, we now try converting the information to a measure of amplitude, and a left-right differential. The amplitude is the sum of the three microphones' FFT readings, and the differential is the left microphone's FFT reading minus the right microphone's. The amplitude should give an indication of which frequencies are currently detected, whilst the differential gives an indication of the DOA. The differential reading for each FFT band will *ideally* be positive when the DOA is to left of the robot, and negative when the DOA is to the right, getting a larger magnitude with DOAs closer to ±90°. Figure 8.7 shows an example set of FFT reading and how

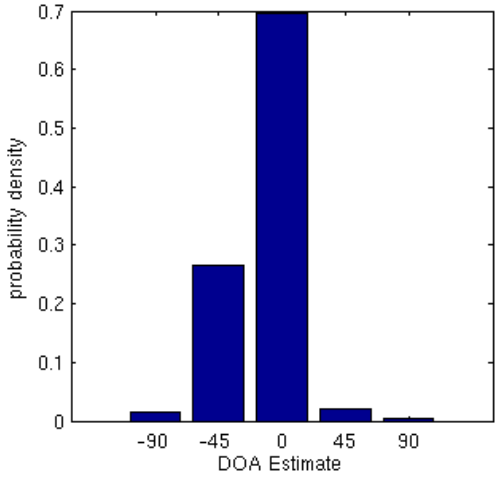


Figure 8.6: The DOA guesses of soundboard 2 expressed as a probability density function.

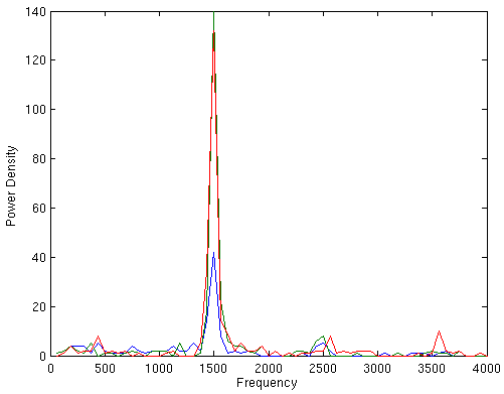
it reduces to an amplitude and differential.

Using the same test data as with section 8.3.2 we generate amplitude and differential readings, across all test frequencies and DOAs for both soundboards. Figure 8.8 shows the median readings from both soundboards. It can be seen that in both boards there is a considerable difference between the amplitude reading for frequencies where no tone is played, and the amplitude reading when a tone is present. Figure 8.9 shows a comparison of the distribution of differential readings as DOA changes. The differential readings from soundboard 1 show a slight trend towards the differential increasing as the test DOA moves from $-90°$ to $90°$. However, each distribution also has a large spread of results, so from the differential alone it would be difficult to determine what the actual DOA of a tone would be. Conversely, the results for soundboard 2 in figure 8.9b cover a much smaller range of values, with no significant variation between distributions.

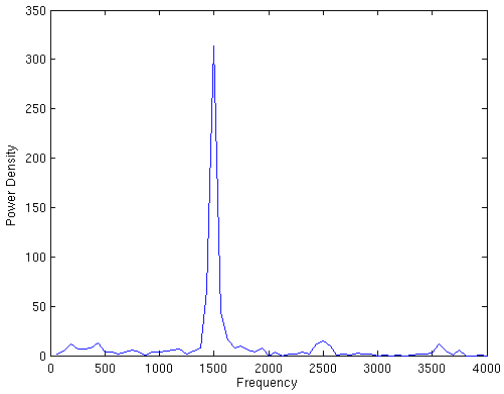
The results from the Wilcoxon rank-sum test are shown in figure 8.10. These results corroborate our analysis of the data from figures 8.9 and 8.8. At DOAs of $\pm 45°$ and $0°$, U was within the 95% confidence limits since there was little to no difference in the differential measurements. At $\pm 90°$, the differential reading from soundboard 1 was more responsive to changes in test tone DOA, so the U at these angles diverges from the Wilcoxon rank-sum expected mean. For the Wilcoxon rank-sum by frequency in figure 8.10a, the comparison results are mixed. The mid-range frequencies have a U value within the 95% confidence limits, with the exception of 1650 Hz. Both soundboards are quite sensitive to this frequency, as can be seen in figure 8.8, but board 1 has a much larger median reading, causing U to fall outside the confidence limits. At the high and low end of the tested range, soundboard 2 is less responsive, and so U falls outside the confidence limits at these frequencies too.

8.3.4 Discussion

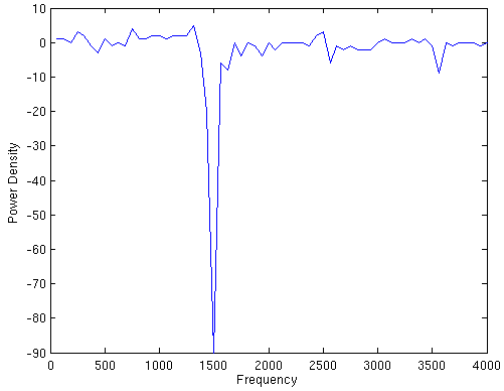
In this section we have tested the soundboards' response to tones of different frequency and DOA, and how these responses differ between the two soundboards. The frequency



(a) Example FFT readings from the soundboard.

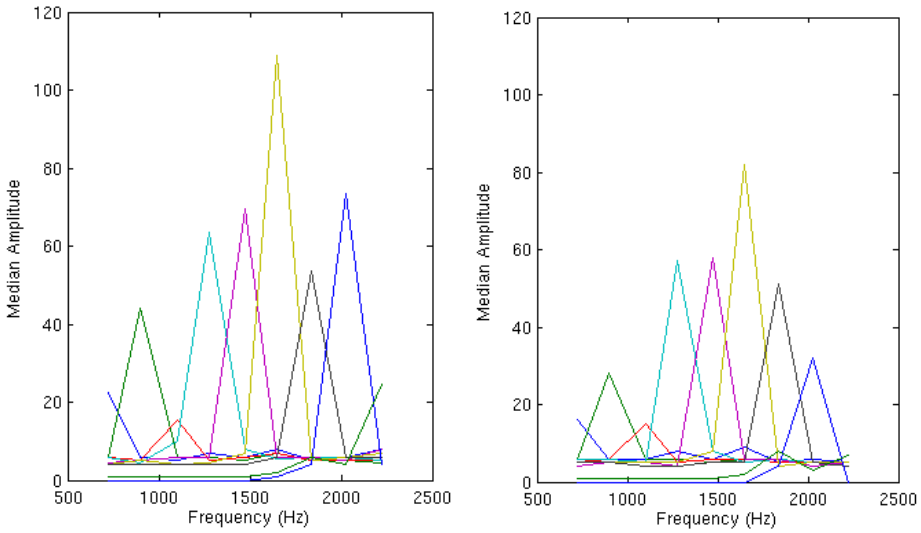


(b) Amplitude reading of the data from a)



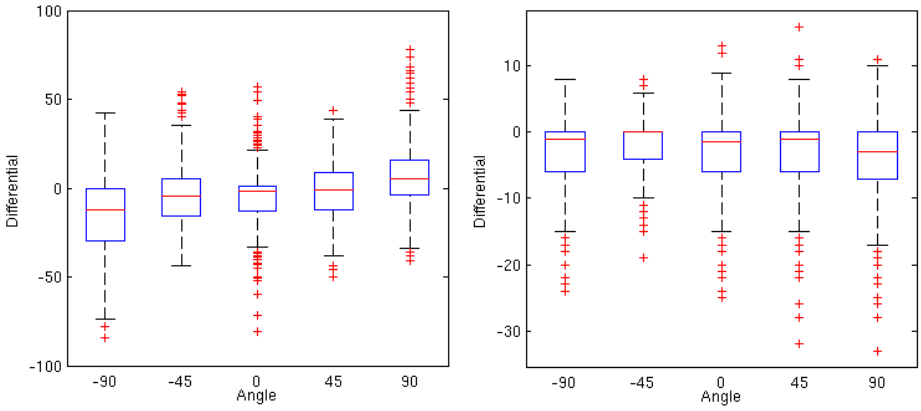
(c) Differential reading of the data from a)

Figure 8.7: Example set of FFT readings from the soundboard (a) and the corresponding amplitude (b) and differential (c). In a) the graph shows a single reading for a test frequency of 1470 Hz and DOA of $-90°$. b) is the sum of all microphone readings and c) is the left microphone minus the right microphone's FFT readings.



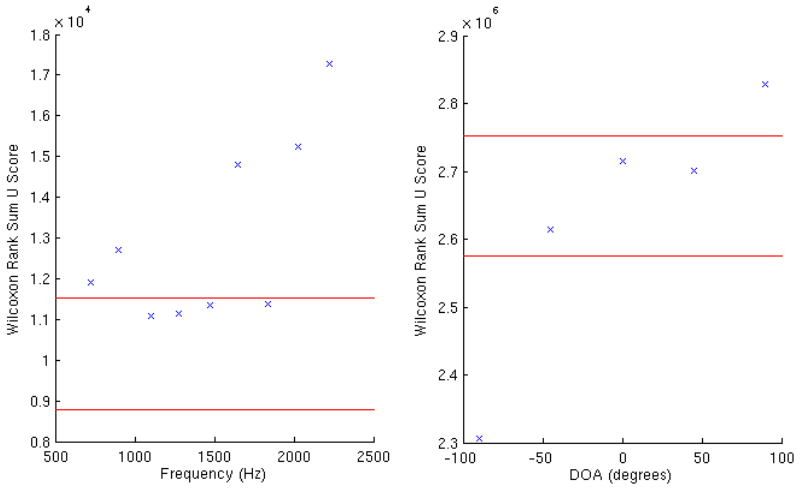
(a) (b)

Figure 8.8: Median amplitude readings from soundboard 1 (a) and soundboard 2 (b). Each line shows the median amplitude reading, over 140 measurements, for each test frequency. There are 9 lines on each graph, one for each test frequency,which is indicated by a peak in the median amplitude.



(a) (b)

Figure 8.9: The distribution of differential readings with respect to the test tone DOA for soundboard 1 (a) and soundboard 2 (b).



(a) (b)

Figure 8.10: The Wilcoxon rank-sum U value for the comparison of soundboard 1 and 2, for frequency (a) and DOA (b). Horizontal lines indicate the 95% confidence limits.

or amplitude readings from both soundboards can be characterised as being different at frequencies of 720, 900, 1650, 2030 and 2220 Hz. We have also shown that both soundboards are very poor at measuring the directionality of a tone, either when using DOA estimation as in section 8.3.2 or a left-right microphone differential as in section 8.3.3. Soundboard 1 has a poor differential response because there is considerable overlap between the spread of differential reading for each DOA. So when trying to guess DOA from a differential alone, as the e-pucks are ideally expected to do, a reading of "0" could indicate *any* of the five tested DOAs. However, board 1 does show an overall trend towards the mean of the differential readings from $-90°$ being lower and $+90°$ being higher, so there is an overall trend in the readings that is correlated with the actual DOA. Conversely with soundboard 2, whether using DOA estimation or the differential measure, there is nothing to distinguish a reading from one DOA from another. The readings from, for example, $-90°$ are indistinguishable from the readings for $0°$, $+90°$ or any other DOA. We are forced to conclude that soundboard 2 is incapable of measuring DOA.

The discrepancies between the two soundboards are most likely due to minor differences in the hardware. The FFT readings from both soundboards are subject to variations in microphone quality and the performance of the microphone pre-amplifiers. If a microphone is less responsive that the others on the board, it could cause a shift in the differential and amplitude readings. This shift would however be consistent across all frequencies and DOAs, which does not match the behaviour in the results we have observed. Furthermore, the soundboards were both hand soldered so it is possible that damage could have been done to components or the PCB tracks during construction. The most likely cause of these differences is that soundboard 2 has a slightly improved printed circuit board (PCB), with slightly wider tracks to make the soundboard PCBs easier to fabricate[1]. However, the circuitry, components and code were unchanged, so

[1]The facilities at the University of York allow us to fabricate small numbers of PCBs, but this is

such a substantial change in performance is unexpected. The manufacturing improvements to the PCB could have inadvertently caused attenuation in the signals from the microphones to the PIC.

A possible follow up test would be to compare two new soundboards using a different PCB design that doesn't have this problem. However, this would require us to design and test a new PCB design and then construct at least two more soundboards so that they can be compared. As mentioned at the beginning of section 8.3, the number of soundboards we can construct is limited by our resources and the amount of time available, and the construction and comparison of the soundboards is not the focus of this work. Consequently, no further tests to compare soundboards will be performed. All remaining soundboard tests will be carried out on the better performing soundboard and this data will then be used to inform our simulation.

8.3.5 Feedback into Simulation

The simulation will model two different soundboards: realistic and idealised. The realistic soundboards are based on measurements from board 1 and incorporate noise. The idealised soundboards give readings in the same range as the realistic ones, but they are not subject to noise or interference and the data they provide is always correct.

Deriving our soundboard model from the two soundboards we have measured in this section would make our simulation true to reality, but it would not help us to test our hypothesis. Our hypothesis requires us to compare communication of information where the tones have consistent meanings, against communication where the meaning of each tone is variable and inconsistent. If the soundboards are unreliable then the information received by each e-puck is inconsistent, regardless of the communication used, thereby making the simulation incapable of reliably testing the hypothesis. By simplifying our model of the soundboard, the simulation becomes less realistic, but more suited to our requirements for it.

8.4 DOA Measurement Accuracy

Calibration 10: The amount of noise on the tone direction measure.

In section 8.3.3 we showed how the differential reading from the soundboard changes according to DOA, but only angles $\pm 90°$, $\pm 45°$ and $0°$ are tested. In this test we measure the differential over a wider range of DOAs to see how noisy it is at different angles.

8.4.1 Experimental Method

As with section 8.3.1, we use an e-puck equipped with a soundboard to play a test tone at the soundboard. The e-puck plays a tone of 1470Hz from a distance of 20cm. The DOAs measured are $180°$, $\pm 135°$ and $10°$ increments between $-90°$ and $+90°$ as shown in figure 8.11. At each angle, 27 sets of FFT readings from the microphones are recorded and from these the left-right differential is taken. To measure the noise, the median and standard deviation for each DOA are taken. A larger standard deviation meaning more noise.

done by hand. Narrow tracks have a tendency to disintegrate during the fabrication process, meaning the PCB has to be remade.

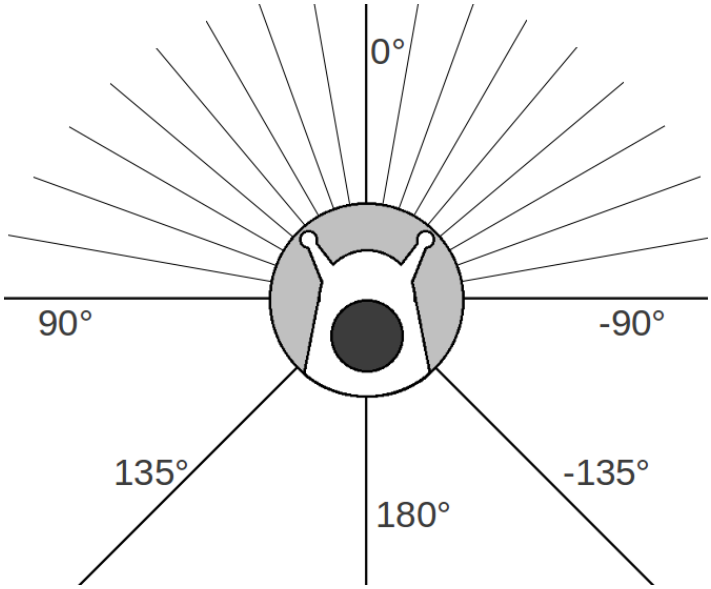


Figure 8.11: Angles measured in section 8.4 to test soundboard noise.

8.4.2 Results and Discussion

Figure 8.12 shows the results of our experiment. The data shown is the differential reading for the FFT band where the 1470 Hz tone would be measured. Each box shows spread of these readings at each of the tested DOAs. From $-90°$ to $-50°$, and $50°$ to $90°$ the results show the expected trend of increasing from negative to positive with increasing DOA.

Since our differential measure is comparing the comparative loudness at two microphones, we would expect $135°$ and $45,°$ to have similar distributions, and the same for the right hand side of the e-puck at $-135°$ and $-45,°$. Performing a Wilcoxon rank-sum on the data, using $\pm 50°$ instead of $\pm 45°$ (which was not measured in this test) does not indicate that these distributions were different [1] [2]. For the same reasons, we would expect the distributions of the $180°$ and $0°$ readings to be similar. However, performing a Wilcoxon rank-sum on the results gave $p = 5.13 \times 10^{-9}$, p being the probability of observing these results if the samples were both from the same data set[3].

Our results would indicate that the differential readings at extreme left and right DOAs ($\approx \pm 90°$) are more likely to indicate direction. This is because at these DOAs we should observe the maximum difference between left and right microphone readings. However, the increase in the differential at the extreme DOAs also causes more variance in the readings, as shown by figure 8.13.

The final thing of note in our results is the interesting anomaly in data at DOAs $-40°$ and $-30°$. At these angles the differentials were consistently much larger than expected, and consequently do not fit the general trend. We are unsure as to why these

[1] For angles $135°$ and $50°$, $U = 330$, $\mu = 364.5$ and $\sigma = 57.80$. Both data sets had 27 measurements for comparison. $p = 0.26$, so the $135°$ and $50°$ results are not shown to be different.

[2] For angles $-135°$ and $-50°$, $U = 343.5$. μ, σ and the data set sizes are the same as for $135°$ and $50°$. $p = 0.36$, so the $-135°$ and $-50°$ results are not shown to be different.

[3] $U = 33.5$, $\mu = 364.5$, $\sigma = 57.8$ with 27 samples in each data set.

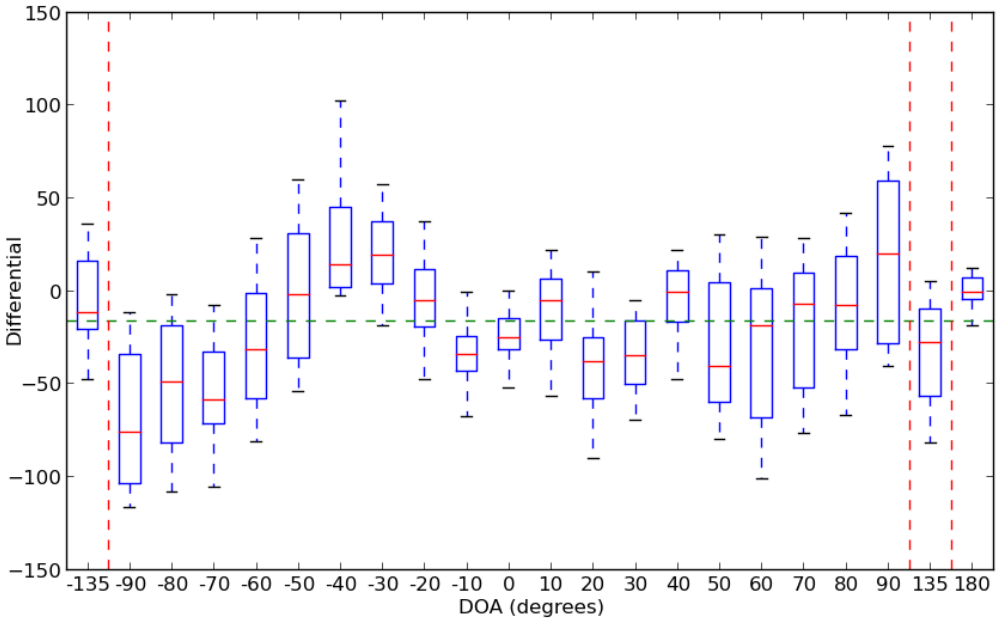


Figure 8.12: Differential results for each DOA. The horizontal, dashed line indicates the median over all readings (−16), the vertical, dashed lines indicate places where the gaps between boxplots is greater than 10°.

anomalies occurred. The increased differential reading at these angles was not observed in previous tests, as can be seen in figures 8.9a or 8.3b. We suspect that it is due to some transient problem in the soundboard, or perhaps environmental noise. However the test is performed under the same conditions as the test from section 8.3. This would indicate that the soundboard is somewhat erratic, and very sensitive to slight variations in the environment.

8.5 Two or More Simultaneous Tones

Calibration 12: The soundboard accuracy in the presence of two or more tones.

Assumption 10: Multiple tones of the same frequency combine as the mean of their polar coordinates.

We have already established in this chapter so far that the accuracy of the soundboard is very poor most of the time. In section 6.3.5 we made the assumption that if two tones of the same frequency were to be played by different robots simultaneously, their amplitudes and DOAs would combine as the average of polar coordinates (see section 6.3.5). As the soundboards have proven to be unreliable it is worth testing that this assumption is true.

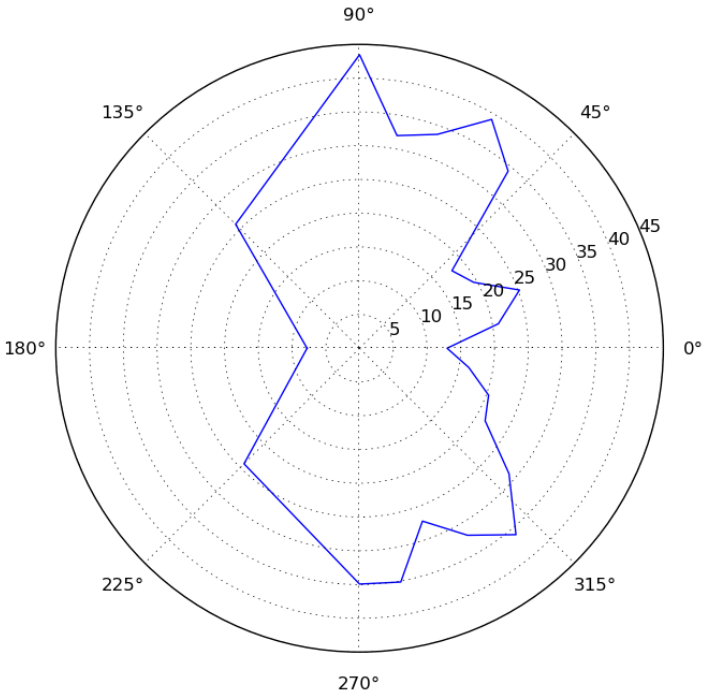
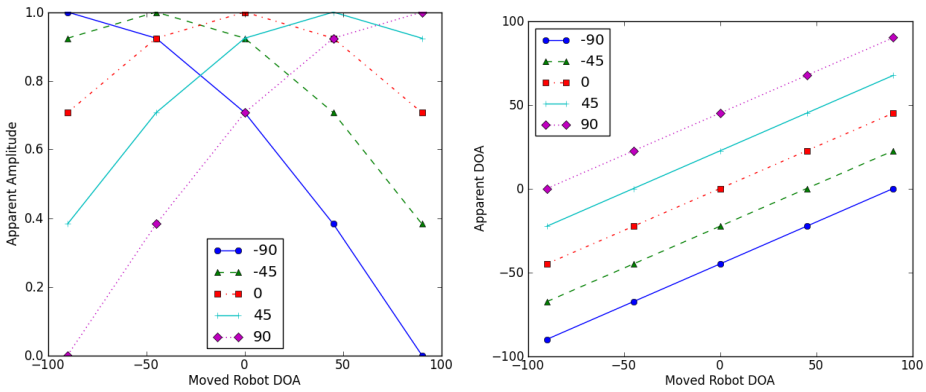


Figure 8.13: Polar plot of differential standard deviation (r axis) as it changes with DOA (θ axis).



(a) Ideal amplitude (b) Ideal DOA

Figure 8.14: The apparent amplitude (a) and DOA (b) of a tone when the two signals, equal distances from the receiver, are combined as polar coordinates. These results have not been converted into an expected amplitude or differential reading, but show the general shape the results ought to take.

8.5.1 Experimental Method

As with previous tests in this chapter, a soundboard is tested by playing test frequencies at it, using an e-puck equipped with a soundboard from several different DOAs. In this experiment there are two such e-pucks generating the test frequencies at the same time. The FFT readings from the tested soundboard are recorded and converted into amplitude and differential measures for our results.

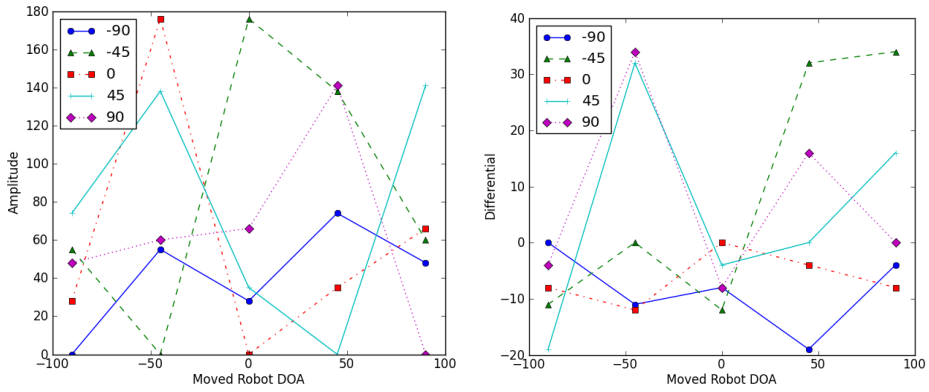
Three different test frequencies are used; low, medium and high at 720, 1470 and 2220 Hz respectively. The DOAs tested were $\pm 90°, \pm 45°$ and $0°$. To test each combination of DOAs one robot stays fixed at a particular DOA and the other robot is moved around all the other DOAs. Then the fixed position robot is moved and the process repeats until we have measurements for each combination of the two robots' DOAs.

8.5.2 Results

The results from this test are difficult to visualise because there are 3 degrees of freedom in the test conditions: the test frequency and the DOAs of each robot. The data is presented as line graphs where each line shows the amplitude or differential measured with the fixed position robot. The line changes along the x axis according to the DOA of the moved robot and the readings that are observed as a consequence of that movement.

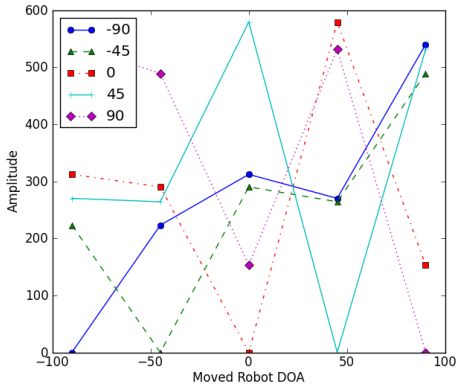
Figure 8.14 is an idealised graph showing the DOA that would be observed if the earlier assumption about combining tones of the same frequency as the mean of polar coordinates is true. This is contrasted with the results that were actually observed in figure 8.15.

It can be seen from figure 8.15 that our readings are quite noisy. Amplitude and differential results do not follow the same shape or trends as the idealised results in figure 8.14. The lines within each graph are not even correlated with each other.

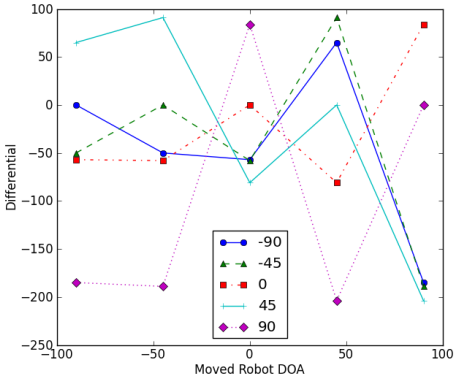


(a) Amplitude 720 Hz

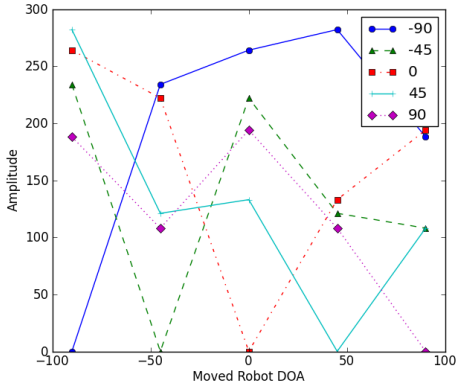
(b) Differential 720 Hz



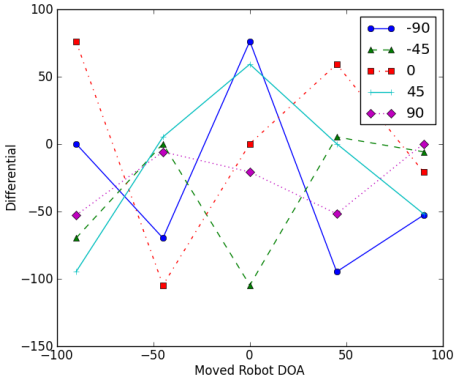
(c) Amplitude 1470 Hz



(d) Differential 1470 Hz



(e) Amplitude 2220 Hz



(f) Differential 2220 Hz

Figure 8.15: The amplitude and differential results of this experiment at 720, 1470 and 2220 Hz. Each data point on the graphs is the median of 37 different FFT readings. Legends on each graph show the DOA of the fixed position robot. At data points where the fixed robot and moved robot have the same DOA, data was not measured because it required the two robots to be in the same place at once. Consequently, results at these points are shown as zero.

8.5.3 Discussion and Feedback into Simulation

We can draw two conclusions from these results:

- the soundboard is insufficiently accurate to test the original assumption that tones of the same frequency can be modelled and combined as polar coordinates.
- This assumption is not a good enough model of the true behaviour of the soundboard for the purpose of our simulation.

To feed the information learnt in this section back into the model, the robot controller has been altered so that when a robot wishes to play a particular frequency tone, it first listens to check that another robot is not currently signalling with that frequency. If it detects another tone of the desired frequency, it waits until the other robot has finished before starting its own signal.

This check is subject to the maximum audio detection range of the soundboards, so it is still possible for two tones of the same frequency to be played at the same time when the first robot to play the tone is outside the second robot's detection range. In this situation, any listening robots will ignore the signalling robot which is furthest away. This is a simplification of actual sound physics because we assume the closer tone will seem much louder than the further one, but it is possible for the two sources to be equidistant. However, depending on the size of the arena and the audio range of the e-pucks, the eventuality of two tones being played at once may be quite rare.

8.6 Measuring Audio Range

Calibration 8: The range the e-puck can detect a tone.

Calibration 9: The amount of noise on the sound intensity measure.

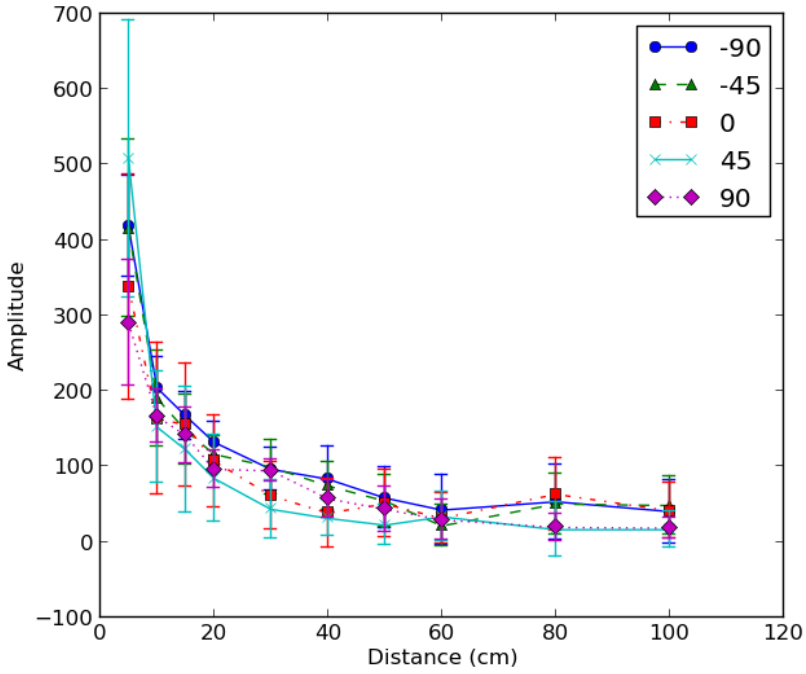
Calibration 11: The soundboard accuracy as it gets further from the sound source.

For our simulation to accurately reproduce audio communication we need to be able to model how the soundboard affects the detection of audio signals at distances other than 20cm. In this section we aim to measure:

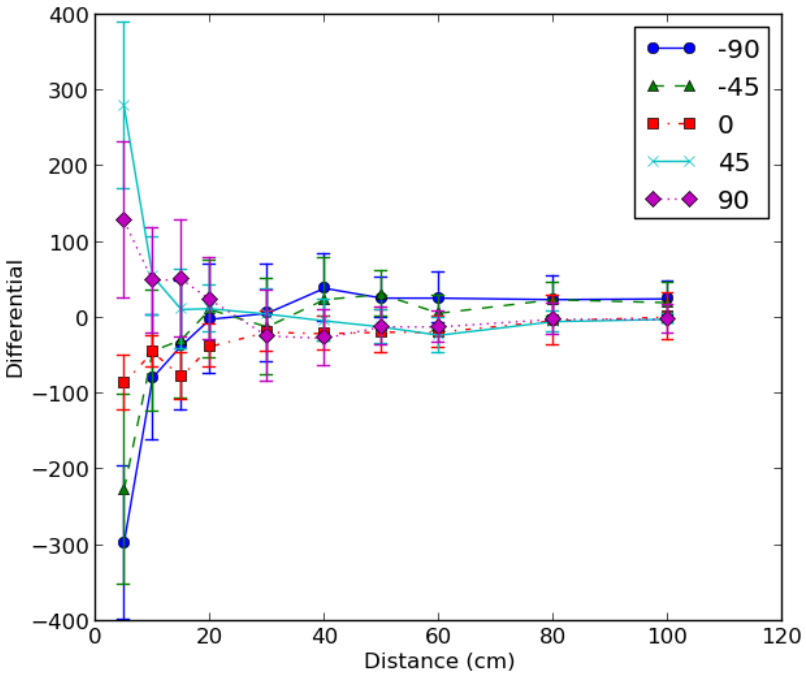
- The amplitude and differential of an audio signal as the source gets further away.
- The amount of variation in these readings.
- The maximum range at which an audio signal can be detected.

8.6.1 Experimental Method

The soundboard is tested by playing a test tone of 1470 Hz at it, using an e-puck equipped with a soundboard as a sound source. The sound source is placed at distances of 5, 10, 15, 20, 30, 40, 50, 60, 80 and 100cm at DOAs of $\pm 90°, \pm 45°$ and $0°$. A total of 50 different combinations of DOA and distance are measured. For each DOA and distance combination, the FFT readings from the soundboard are recorded and converted into an amplitude and differential measure for our results.



(a) Amplitude



(b) Differential

Figure 8.16: Amplitude and differential results per DOA for the ranging test. Each data point is the median of 27 readings, error bars indicate the 95% confidence limits.

8.6.2 Results

Figure 8.16 shows the results of this test. The amplitude graph in 8.16a shows that changing the DOA does not significantly affect the way the amplitude of the signal decreases with distance. The results are consistent across different DOAs, unlike in previous experiments (for example figure 8.15) where amplitude readings have varied significantly. This suggests that, at least for the amplitude measures, that at 1470 Hz and ranges of less than 2 lengths of the microphone array it may give useful results.

The differentials results in 8.16b showed that for distances smaller than 20cm the DOAs of $-90°$ and $-45°$ have a negative value medians and DOAs $+45°$ and $+90°$ have a positive value medians. Meaning that for distances less than 20cm it may be possible to determine directional information from the differential measures using thresholding. However, this may only be true when the signalling frequency is 1470 Hz. The board comparison test in section 8.3.4 showed that the soundboard's response is not the same over all signalling frequencies.

8.6.3 Discussion

At beginning of section 8.6 we state that the aim of this test is to get data that can be used for calibrating simulation; to get the audio detection range of the soundboard and "accuracy" of the soundboard as the distance from the sound source changes.

The results have shown that from the amplitude readings the detection range is between 30 and 40 cm. Above this distance the amplitude tends to level out. For the differential measure the detection range is around 20 cm at which point the differentials for the different DOAs appear to converge. This means that between 20 and 40 cm it is possible to detect that a tone has been played, without being able to determine the DOA.

For calibrating the accuracy of the soundboard over distance we now have measurements of the median and spread of the amplitude and differential data. The feedback of these results, and the results of previous tests, into the model, is described in the next section.

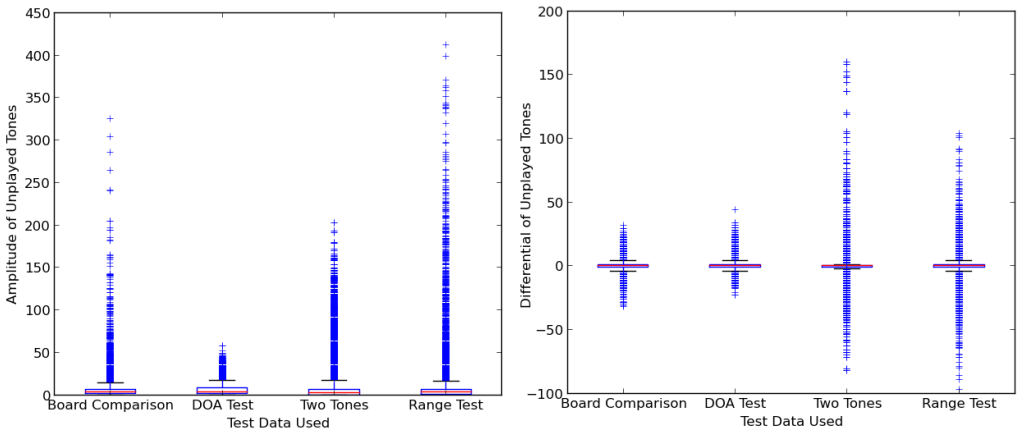
8.7 Calibrating the Model

In section 8.3.5 we conclude that the soundboard is very unreliable and that to test our hypothesis we should model both idealised and realistic boards. This is so that we can have a realistic simulation, by using the realistic model for the soundboards, and one where we know that the e-pucks are communicating the frequency and DOA of the tones effectively. The latter is important so that we can guarantee noise in the sound transmission does not cause incorrect audio readings. Only the communication coupling and signal-meaning mapping affects the information passed between robots.

For both soundboard models we need to approximate:

- The amplitude and differential observed when there is no tone present.
- The amplitude and differential observed when a single tone is played at a signalling frequency.

We avoid the situation where there are two or more tones of same frequency, as described in section 8.5.3. If there are several tones but they all have different frequencies, they are assumed not to affect each other (see section 6.3.5).



(a) Amplitude (b) Differential

Figure 8.17: Boxplot showing the distribution of amplitude and differential readings for FFT bands where a tone was not present. Each box shows the distribution for the tests described in this chapter.

8.7.1 Amplitude and Differential with No Tone

For an idealised soundboard both amplitude and differential when there is no tone present can simply be modelled as zero.

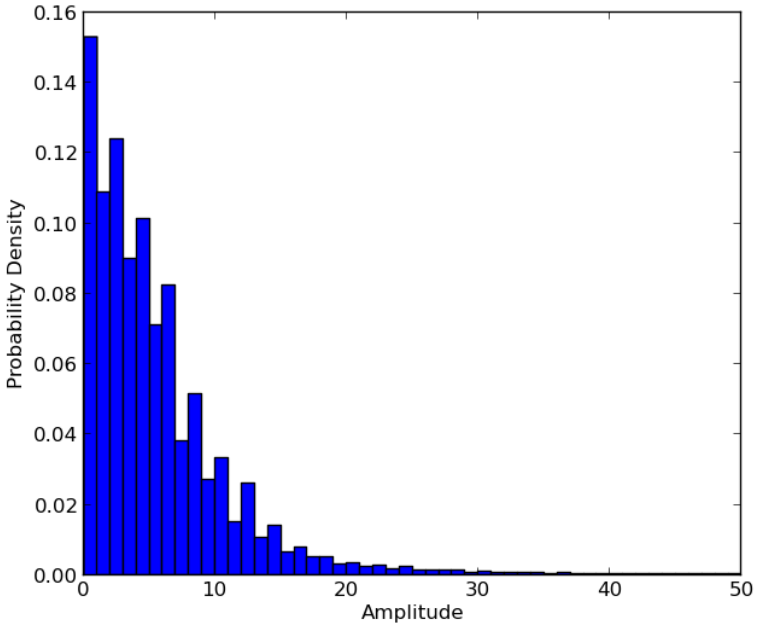
To model realistic board we use data taken from the tests in this chapter. The amplitude and differential readings are taken from each experiment for all FFT bands where no tone would be observed. In tests where the test frequency, DOA or distance was varied, all data readings are shown together. This is because we are investigating tones that were *not* played to the soundboard, therefore they have no frequency, DOA or distance to the source. These results are given in figure 8.17.

For each test, the amplitude and differential distributions have similar median and quartiles. For the amplitude readings some tests had more outliers than others, but the distribution of the majority of the data is the same in each case. The difference in the number of outliers may be due to slight variations in the environmental conditions when performing the experiments. Figure 8.18 shows histograms of the amplitude and differential readings for when no tone is present using all the data from figure 8.17.

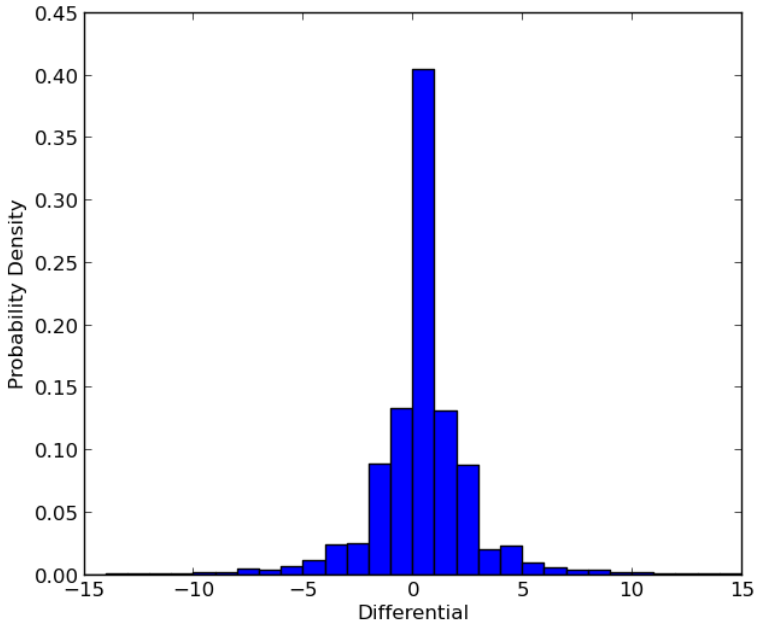
Figure 8.18 shows that the differential has normal distribution, and the amplitude is the absolute value of a normal distribution centred around zero. To generate realistic amplitude and differential readings we therefore use a normally distributed random number generator and then transform the random numbers to have the same mean and standard deviation as the data we require.

The differential data shown in figure 8.18b has $\mu = -0.07$ and $\sigma = 3.26$ (2 d.p.). To get the mean and standard deviation for the amplitudes, we reflect figure 8.18a around 0 to get figure 8.19. The mean of figure 8.19 is 0 and the standard deviation is 11.13 (2 d.p.). In our realistic model these values are converted into an amplitude and differential reading using the following equations, where $\mathcal{N}$ is a normally distributed random number with $\mu = 0$ and $\sigma = 1$:

$$\text{amplitude} = |11.13\mathcal{N}| \tag{8.1}$$



(a) Amplitude



(b) Differential

Figure 8.18: Histogram showing the distribution of all the amplitude and differential readings for FFT bands where a tone was not present. This is the combination of readings from every test in this chapter, covering a total of 226554 data points.

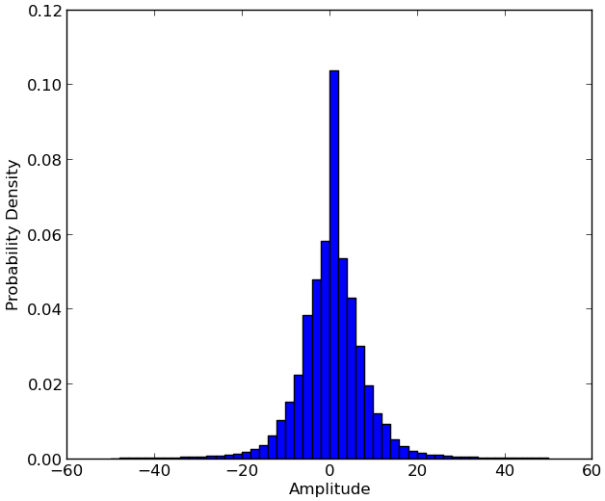


Figure 8.19: Amplitude histogram reflected around zero.

$$\text{differential} = 3.26\mathcal{N} - 0.07 \tag{8.2}$$

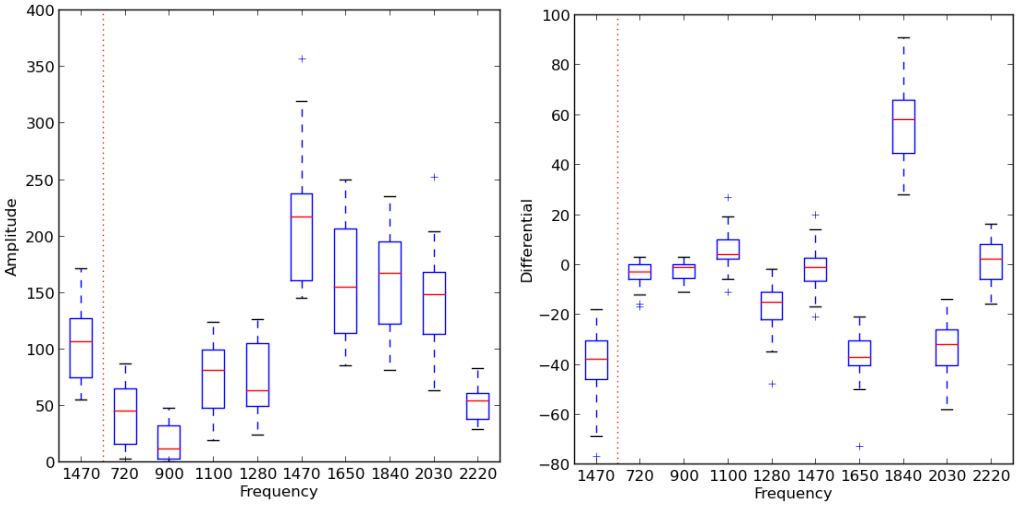
8.7.2 Amplitude and Differential with One Tone

Realistic Model

To realistically model the soundboard's response to a single tone, we model the results from the range measurement test in section 8.6, figure 8.16. From the graphs in figure 8.16 we can use the distance between a sound source and the listening robot to estimate the amplitude and differential that would be observed in reality. However, figure 8.16 shows the results of only one test frequency. Using this data in our model assumes that these results are representative of all the signalling frequencies we used in the experiment.

This is a significant simplification of the actual soundboard's behaviour. Figure 8.20 shows how the test tone frequency affects the distribution by comparing the amplitude and differential readings from the audio ranging test to the soundboard comparison test. It can be seen that the audio range test, at 1470 Hz, has a different distribution to the other test results, even at the same frequency. It also shows that, within the soundboard comparison test, the distribution of data varies a lot between different test frequencies. This is further evidence that the soundboard is highly sensitive to environmental conditions, since both tests were performed in the same arena, using the same e-pucks and soundboards, but on different days. The soundboard is so erratic that any attempt to model it would either be intractably complicated or over-simplified.

The audio ranging results are not entirely representative of the data the soundboard produces, but the soundboard is so erratic that it would be very difficult to derive data that was representative of it. The model we use is an over-simplification, but it is at least based on actual observations, so gives results which *might* be observed in reality. Without running the experiment on real robots we cannot know if our approximate model of the e-puck with a soundboard would act similarly to the real ones. For the



(a) Amplitude (b) Differential

Figure 8.20: The ranging results compared to the soundboard comparison results. All boxes show the readings for a distance of 20cm, and DOA of 0°. The leftmost box shows the spread of amplitude (a) and differential (b) readings for the audio range test, the other boxes show the soundboard comparison test results with different test frequencies.

purposes of this work, we assume that our approximation is good enough. The validation of our model against the experiment using embodied e-pucks is left as future work.

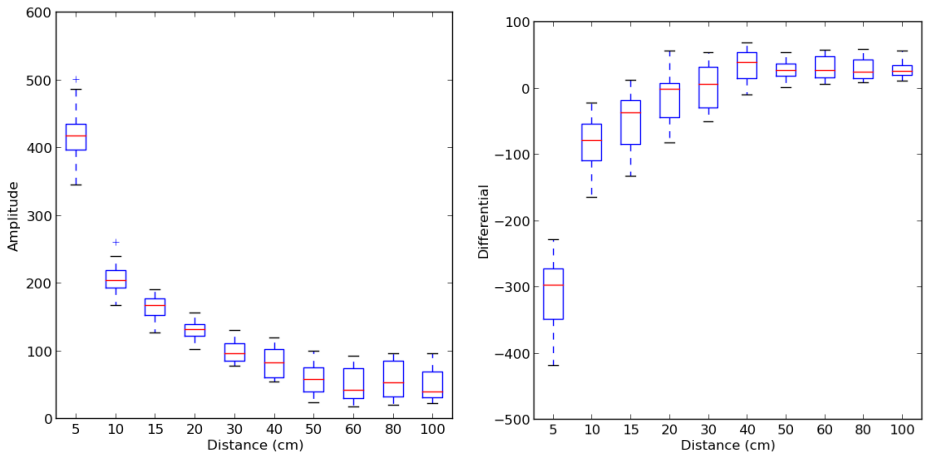
Building the Realistic Soundboard Model To model the soundboard's response to one signal tone, we take a similar approach to modelling the lack of a tone in section 8.7.1. Firstly by finding the shape of the data distribution as it changes with distance and DOA, deriving parameters which describe it, then finally programming the model to generate data that fits the distribution described by the derived parameters.

Examining the distribution of the audio range test results (figure 8.21) it can be seen that for each distance measured, the boxes mostly show a normal distribution. We therefore assume the amplitude and differential readings given by the soundboard are normally distributed. To model the results we must find the mean and standard deviation of the amplitude and differential as they change with distance and DOA.

This information is shown in figure 8.22, as measured during the audio range test. For each graph there are 5 DOAs measured, meaning that there are 20 curves in total that we need to model in order to get the mean and standard deviations required for reproducing the amplitude and differential readings.

For each curve in figure 8.22 we used the MATLAB curve fitting tool to fit a cubic polynomial to its shape, because preliminary tests showed that cubic polynomial gave a good enough fit for the data. Cubic polynomials are equations in the form $ax^3 + bx^2 + cx + d$, so the curve fitting tool derived the coefficients a, b, c and d for each of the 20 amplitude and differential mean and standard deviations.

From the simulation we can measure the true distance and DOA between the source and receiver robots. If the distance between the two is larger than the soundboard's maximum range of 20 cm our model assumes the tone is unheard and return a reading



(a) Amplitude (b) Differential

Figure 8.21: The results from the audio ranging test, for DOA of $-90°$, showing the distribution of the data.

for an unheard tone. Otherwise, we use the true distance and DOA to get the mean and standard deviation of the amplitude and differential from the fitted cubic polynomial curves.

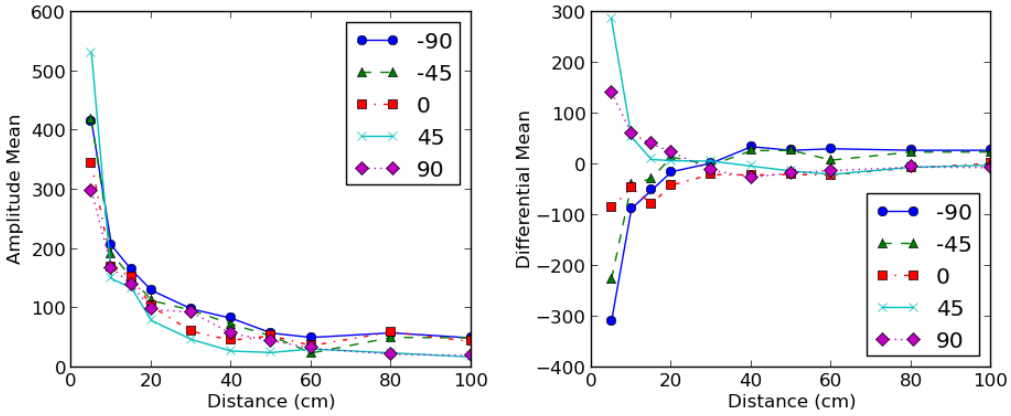
In the model there are only curves for DOAs of $\pm 90°, \pm 45°$ and $0°$, since these were the only DOAs measured, but the DOA in the simulation could be any value. The true DOA from the simulation needs to be converted into a set of coefficients to describe the mean and standard deviation curves.

First there are cases where the true DOA falls outside the range $-90°$ to $+90°$. As a result of the differential being derived from the difference between the left and right microphone readings, the differential cannot distinguish between sound sources in front or behind the robot. This was observed in the DOA accuracy test in section 8.4.2, when the distribution of $-135°$ differentials was found to be similar to those of $-50°$ (and $+135°$ similar to $+50°$). To account for this if the true DOA is less than $-90°$ the DOA is flipped to be between $-90°$ and $0°$. Similarly, a true DOA of more than $+90°$ is flipped to be between $0°$ and $+90°$. For example, if the DOA is $-135°$, the DOA used to find the mean and standard deviations will be $-45°$, or if the DOA was $170°$ it gets flipped to $10°$.

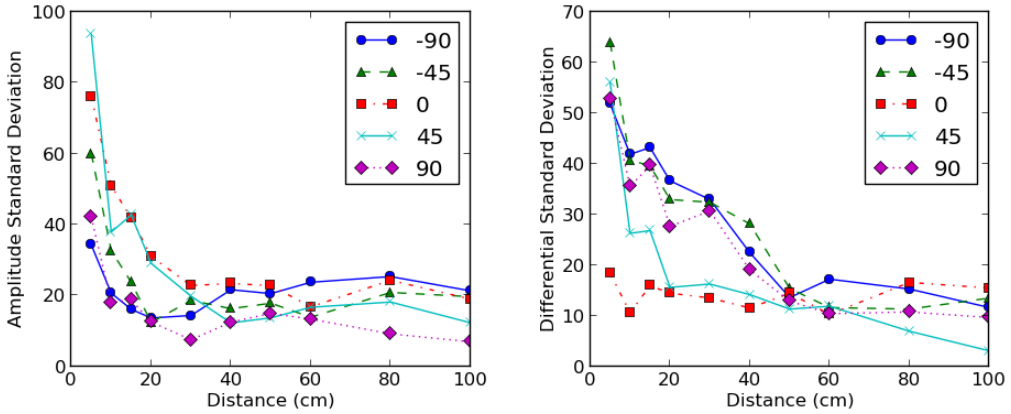
The DOA used to calculate the soundboard data is now in the range $-90°$ to $+90°$, but it might not be exactly one of the DOAs that was measured. If this is the case then the model linearly interpolates the cubic polynomial coefficients used to get the mean and standard deviation from the two nearest measured values.

Once the co-efficients have been found for the simulated DOA, the amplitude and differential mean and standard deviation can be found using the equations 8.3 to 8.6, where r is the distance between the receiving robot and the sound source, a_μ is the amplitude mean, a_σ the amplitude standard deviation, and d_μ and d_σ are the differential mean and standard deviation:

$$a_\mu = a_{a\mu} r^3 + b_{a\mu} r^2 + c_{a\mu} r + d_{a\mu} \tag{8.3}$$



(a) Mean



(b) Standard Deviation

Figure 8.22: The mean (a) and standard deviation (b) of the amplitude and differential readings as the distance to the sound source increases. Each line shows the mean or standard deviation at a different DOA.

$$a_\sigma = a_{a\sigma} r^3 + b_{a\sigma} r^2 + c_{a\sigma} r + d_{a\sigma} \tag{8.4}$$

$$d_\mu = a_{d\mu} r^3 + b_{d\mu} r^2 + c_{d\mu} r + d_{d\mu} \tag{8.5}$$

$$d_\sigma = a_{d\sigma} r^3 + b_{d\sigma} r^2 + c_{d\sigma} r + d_{d\sigma} \tag{8.6}$$

Finally, the model can generate realistic amplitude and differential readings using a normally distributed random number $\mathcal{N}$:

$$\text{amplitude} = a_\sigma \mathcal{N} + a_\mu \tag{8.7}$$

$$\text{differential} = d_\sigma \mathcal{N} + d_\mu \tag{8.8}$$

Idealised Model

In chapter 6, section 6.3.5 we simulated sound dissipation over distance using equation:

$$I_{\text{at robot}} = \frac{W_{source}}{2\pi r^2} \tag{8.9}$$

where $I_{\text{at robot}}$ is the sound intensity at the robot, W_{source} is the watts used at the source to generate the sound, and r is distance from source to robot in metres. Given we now know the hardware characteristics of the soundboard we can calculate W_{source} to be 1.36 Watts[1].

Using equation 8.9 gave a value of 4.33 at 5cm, whilst the smallest of the actual readings at 5cm was 232. W_{source} is not a big enough scale factor to reproduce amplitude. However, we found that equation 8.10 was a closer approximation to the shape of the observed results, as shown by figure 8.23.

$$\text{Ideal Amplitude} = \frac{150}{2\pi r} \tag{8.10}$$

We model ideal amplitude using equation 8.10. To model the ideal differential we use the true DOA, and scale it by $(2\pi r)^{-1}$:

$$\text{Ideal Differential} = \frac{\text{DOA}}{2\pi r} \tag{8.11}$$

As with the realistic model, if the true DOA is behind the robot, then it is flipped about the front/back axis of the robot as with the actual soundboard. So a DOA of $-100°$ would become $-80°$, and $+100°$ becomes $+80°$. Figure 8.24 shows the ideal differentials, and figure 8.25 shows the idealised differential plotted against the mean of the observed differentials. It can be seen that the idealised curves quite closely match the actual ones.

[1]Maximum voltage at speaker is 3.3V and speaker resistance is 8Ω. Power $= V^2 R^{-1} = 3.3^2 8^{-1} =$ 1.36125 Watts.

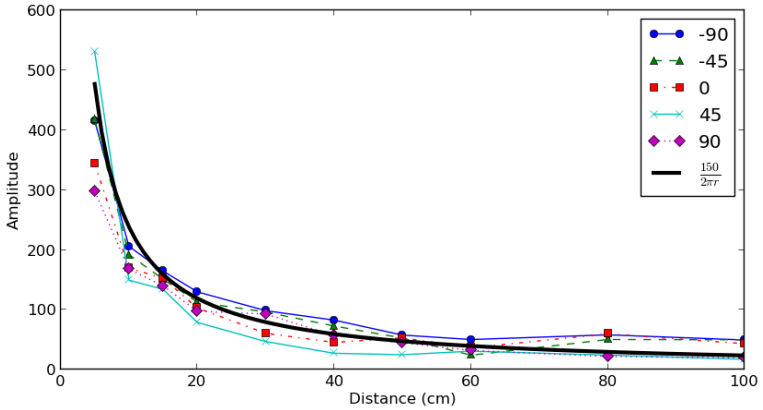


Figure 8.23: Approximation of curve amplitudes using equation 8.10

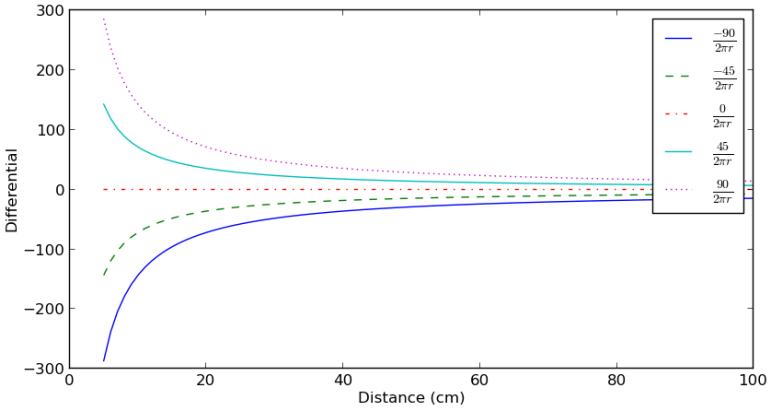


Figure 8.24: Approximation of differentials using equation 8.11.

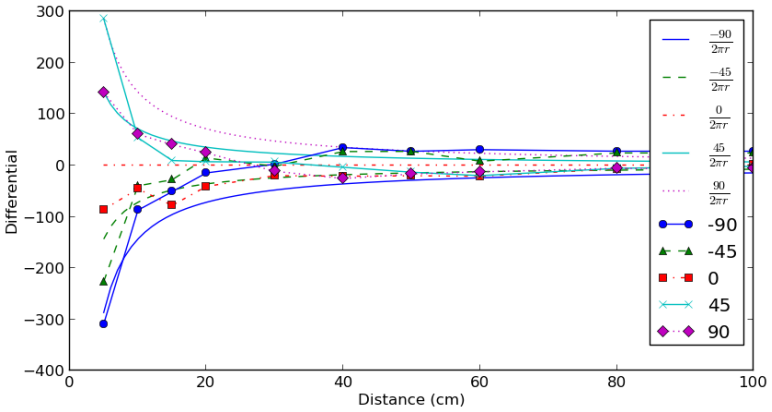


Figure 8.25: The idealised differentials compared to the mean measured differential readings.

8.8 Conclusion

In this chapter we address the calibration points raised in chapters 6 and 7.

Several tests are performed to measure the board's audio characteristics. When two similar soundboards are compared under the same test conditions, both soundboards are found to be very poor at measuring the directionality of a tone, and one board was incapable of measuring directionality. The final test on the soundboard was to find the maximum range at which the soundboard would be effective. This showed that the soundboard is able to give consistent readings at ranges of less than 20 cm.

The results from this chapter are used to calibrate two different models of the soundboards. One is realistic and returns noisy readings based on actual measurements from section 8.6 of this chapter. The other model is idealised and does not implement any signal or hardware noise into its results. Both models are simplified versions of the real soundboard.

Due to the erratic nature of the hardware, an exact reproduction of the kind of results that have been observed in this chapter would introduce inconsistency into the information being transmitted between robots. This would make it difficult for the robot, through evolution, to distinguish whether tones have been played or which direction the tone arrived from. By simplifying the model we hope to prevent these extra inconsistencies so that we can more reliably test the effects of different communication strategies on the rate of evolution in a robotic swarm. In part III we use the model developed in chapter 6 with the idealised soundboard model from chapter 8 to run the experiment from chapter 5.

Further work is required to run the experiment from chapter 5 using the real e-pucks and soundboards, to validate the simulated experiment against the embodied experiment. This is left as future research.

By following the CoSMoS process and documenting the process of modelling the robot swarm in chapter 6, we made a list of assumptions to show where our model is weakest, and a list of areas to calibrate later when more can be found out about the real, engineered system. These points are calibrated in this chapter. This documentation makes it easier for future researchers to use this model or to see an example of developing a model of an engineered complex system.

Part III

Results and Conclusions

Chapter 9

Initial Results From Testing the Hypothesis

9.1 Introduction

In this work we aim to test our hypothesis, from chapter 4, that evolutionary swarm robots will evolve collaborative behaviours faster if the swarm consists of *independent* agents sharing *stable* information. In chapter 5 we propose an experiment for testing this hypothesis, and in part II we develop a model for running that experiment in such a way that result from the model could be extrapolated to a swarm of real, embodied robots. In this chapter we present the results of the experiment, performed using the model.

To summarise chapter 5, the robot swarm performs a foraging task. Robots collect food from various food sources in the environment, larger food sources give more food, but require more robots to be present before distributing food. Robots are able to signal to each other about the food sources, so that other robots can decide which food sources to forage from. We test our hypothesis by trying different methods of mapping between signal and meaning, in order to control the coupling between robots:

1. **Tightly coupled, fixed mapping**. 9 signals, each tone expresses a different combination of food source size and occupancy.

2. **Loosely coupled, fixed mapping**. 6 signals, two signals are combined sequentially to express a food source's size and occupancy.

3. **Loose coupled, evolved description mapping**. 6 signals, sequentially combined as with (2). 3 signals always refer to a food source's size, and the other 3 to an occupancy. Within those 3 signals the signal to meaning mapping for size or occupancy can be evolved.

4. **Loose coupled, evolved mapping**. 6 signals, sequentially combined as with (2). The mapping between signals and meanings is evolved.

5. **Tight coupled, evolved mapping**. 9 signals as with (1), the mapping between signals and meaning is evolved.

Section 5.3.1 contains a full explanation of why these signal to meaning mappings have been chosen. The hypothesised result is that the loose coupled, fixed mapping communication would lead to better solutions sooner. However, the results show that all robot swarms evolve solutions which do not use communication, and that no intentional collaboration occurs under any of the tested coupling strategies or experimental conditions.

Several different measures are taken to make it easier for the GA to evolve effective communication and effective foraging. These are shown to alter the swarm's overall ability to forage, but do not cause collaboration or communication between robots to evolve. Consequently, the hypothesis can not be tested because it is unclear whether coupling has no effect, or the experiment has design flaws preventing coupling from being effective.

Finally it is concluded that the task used in this chapter experiment is not suitable for testing the hypothesis because it can be completed without communication or collaborative behaviour. Robots are encouraged to cooperate by rewarding cooperation with more food; if the robots work together they can collect from the larger food sources. However, there need to be multiple collaborating robots in the swarm before collaboration can be rewarded more than individualism, but until collaboration is better rewarded it is difficult for that behaviour to propagate into the genome population. However, selfish behaviour is easier to evolve because a single robot can collect small amounts of food, giving its genome a small fitness. Collaborative behaviour requires multiple robots in the swarm trying to cooperate before food can be collected. If the collaboration is successful the robots are rewarded more than the selfish robots, but if unsuccessful it is not rewarded at all. It is difficult for collaborative behaviour to propagate through population, because multiple robots need to be collaborating before it become better rewarded. However, multiple robots will not be cooperating unless the collaborative behaviour has propagated in the population.

9.2 Experimental Method

The experiment presented in this section is described in detail in chapter 5. In this section we describe the implementation details of the experiment, and how the results are measured. The experimental parameters are broken down into the following categories:

- Grammatical Evolution
- Genetic Algorithm
- Food Distribution
- Communication

When applicable, a parameter's initial value is based on what we would expect that parameter to be in a real robot swarm.

9.2.1 GE Grammar

In section 5.4.2 we stated that grammatical evolution (GE) [77] should be used in this experiment to evolve a controller. The following GE grammar is used in the experiment:

```
<code>              ::=  <if> | <if><code>
<if>                ::=  IF <condition> THEN <action> ELSE <action>
<condition>         ::=  <tone><op><tone>
                       | <tone><op><constant>
                       | <condition><logical><condition>
                       | <food data><op><percent constant>
                       | isAtTree
<action>            ::=  <move> | <collect>
<tone>              ::=  <tone distance> | <tone bearing>
<op>                ::=  < | >
<logical>           ::=  AND | OR
<move>              ::=  forwards <time delay>
                       | left <time delay>
                       | right <time delay>
<collect>           ::=  collect <time delay>
<tone amplitude>     ::=  tone1 amplitude ...
<tone differential> ::=  tone1 differential ...
<food data>         ::=  food source size
                       | food source occupancy
<constant>          ::=  0 | 10 | 20 | ... | 260
<time delay>        ::=  500ms | 1000ms | ... | 4000ms
<percent const>     ::=  0 | 1 | ... | 100

START = <if><code>
```

The GE grammar is designed to create sequential `IF - THEN - ELSE` statements, so that the robots are essentially running a sense - think - act loop as their controller.

In swarms where the communication mapping must be evolved, the genome contains an extra S genes, where S is the number of signals that are used. These extra genes control the signal frequency or frequencies a robot plays upon encountering a food source. The mapping is implemented using the following pseudo code:

```
make list of the S signal frequencies to use, called signals
for each meaning m:
     value = gene % number items remaining in signals
     map meaning m to signals[value]
     remove signals[value] from list of signals
     gene = next mapping gene from genome
```

Depending on the communication coupling used, the meanings are allocated in the follow order:

```
loose coupling =
  0%< size, 33%< size, 66%< size, 0%< full, 33%< full, 66%< full
tight coupling =
   0%< size:0%< full,  0%< size:33%< full,  0%< size:66%< full,
  33%< size:0%< full, 33%< size:33%< full, 33%< size:66%< full,
  66%< size:0%< full, 66%< size:33%< full, 66%< size:66%< full
```

In the case where we evolve only between the descriptions, the mapping code is run once for size and again for occupancy signal mappings.

Initialisation

Genomes are initialised with random integers to a length of 80 genes. For genomes which also contain a signal to meaning mapping there are an additional S genes which are separate from the main genome.

9.2.2 Genetic Algorithm Parameters

In the simulated environment we have ten foraging robots and three food source robots. Ten is a small number of robots to have in an evolving population, but swarm size is limited by the number of real e-pucks available to use if this experiment were to be performed on a real robot swarm.

With a small population there is little genetic diversity, so it is harder to perform a wide search of the potential solutions. To give the GA a better change at finding a suitable controller, we run the experiments for 1000 generations. This means that, although each generation only performs a small search of the potential solutions, there are many generations, so there is more opportunity to find a good solution. Each generation lasts 120 seconds to give the robots a chance to find a food source and collect from it. Consequently, over 1000 generations each experiment lasts 33 hours and 20 minutes. However, by testing our hypothesis in simulation rather than on real robots the experiments do not have to be run in real time, and so we can obtain results much more quickly.

For each experimental run, the foraging and food source robots are given the randomly generated starting positions shown in figure 9.1. After each generation the foraging robots are randomly repositioned in the environment. This is to prevent evolving behaviour where the robots is stationary and waits at a food source waiting for more food to be generated for it to immediately collect. By moving the foraging robots each generation, we are trying to steer the GA towards generating solutions where the robots can both find a tree and collect from it.

Genetic Operators

Selection Selection of genomes to create the next population is done by tournament. Two different genomes are taken from the population and their fitnesses are compared. The genome with the highest fitness is put into the new population.

Crossover The crossover function is applied with a 10% probability. Crossover takes two genomes from the new population, and randomly picks a single, complete `IF - THEN - ELSE` statement. These statements are then swapped into the vacated position in the other genome. The length of the genome is not preserved since one statement may require more genes to complete. The genomes are limited to have between 20 and 120 genes. if the genome is too short, random genes are added to the end of the genome, if the genome is too long, genes are removed.

To crossover the signal to meaning mapping in two genomes (if present), a random position along the length of the mapping genes is picked. The mapping genes after this point are swapped. The length of the mapping genes is preserved so that there are always enough genes to specify a meaning for each signal.

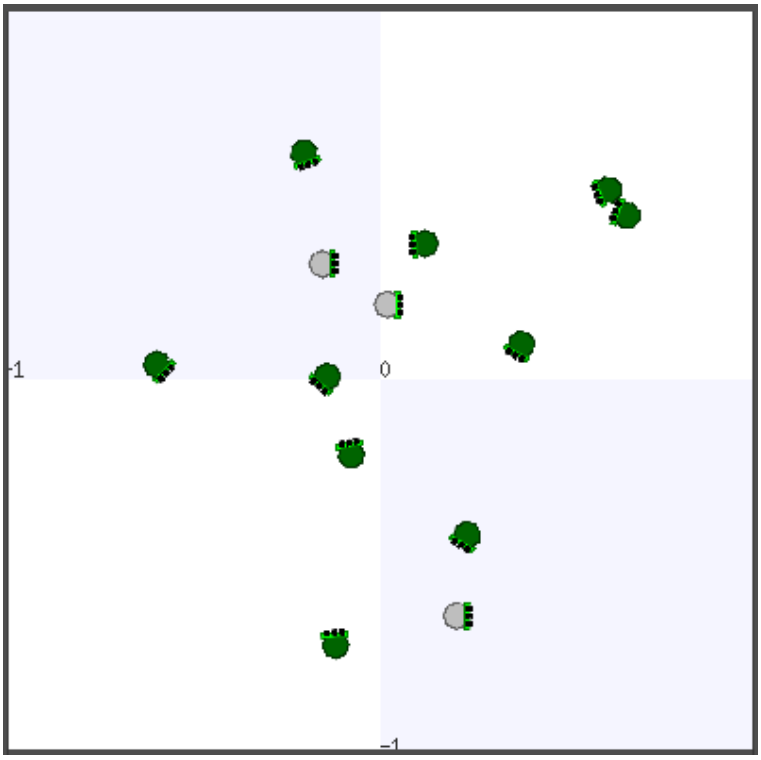


Figure 9.1: Simulation environment showing collector and food source initial positions. The arena is a 2×2 metre square. Robots acting as food sources are coloured in light grey.

Mutation To mutate a genome, the algorithm checks every gene of the genome and changes it to a new, random integer with 10% probability. The same is done for the mapping genes, if present.

9.2.3 Food Distribution Parameters

The way that food is rationed in the environment is an important factor in how much food the robots can collect, and consequently has a large effect on the swarm fitness. In section 5.2 we describe how the size of each food source gets smaller over time, and when a food source is emptied it replenishes to some random size after a short time. The parameters used to control food distribution are as follows:

- **Maximum food size**: 5. Food source sizes are random and can be any integer value between 1 and the maximum. We have chosen a maximum of 5 for our initial experiments so that the task is not too difficult for a swarm of 10 robots.

- **Replenish time**: When a food source is depleted or collected it is given a new size after 4 seconds.

- **Food degradation rate**: The food sources reduce a size every 15 seconds. With a generation duration of 120 seconds this means a food source can deplete up to 8 times. This is another measure to prevent the robots waiting at a food source until it shrinks enough to be collected. If the food sources deplete slowly, the robots must wait longer before the source can be collected when it may be more rewarding to move to another source.

9.2.4 Communication Parameters

To start with, the experiments use the idealised model of the soundboard, developed in section 8.7.2, to test the hypothesis. This way, if the swarm fails to communicate or collect food, then we know this is due to failings in the experimental setup or hypothesis rather than the model. If we get promising results from the experiments then the experiments can be repeated with the realistic soundboard model (see section 8.7.2) to see how it affects the results.

Each tone has a duration of 500ms. If the swarm is using loosely coupled communication then each signal is a sequence of two tones, for a total length of 1000ms. When a robot listens for signals it only takes a measure of the tones currently playing in the environment, and does not have a memory of the previous measurements.

Section 5.3 of chapter 5 describes how the communication is implemented. To summarise, as soon as a robot detects a food source, it broadcasts the size and occupancy description of that food source to all other robots within audio range. There are D_{size} descriptions that the robot can use to describe the food source's size and D_{occ} for its occupancy, $D_{size} = D_{occ} = 3$. These descriptions are equivalent to small, medium, big, and empty, slightly occupied, mostly occupied; although they are calculated as a percentage of the maximum possible size or occupancy, as given by equations 5.1 and 5.2 on page 36. We only investigate cases where D_{size} and D_{occ} are the same size, so for the rest of this work we use D as notation for this coupling parameter, such that $D_{size} = D_{occ} = D$.

9.2.5 Measuring Results

To test our hypothesis, we evaluate the results of the GA on the following metrics:

1. The fitness of the evolved robot controllers, compared to randomly generated controllers.
2. The quality of the evolved controllers.
3. The speed with which a "successful" solution is reached.

For each metric, we will use the Wilcoxon rank-sum and A-test to compare fitness results. The p values for the Wilcoxon rank-sum results are measured, and if $p < 0.05$, this means that the two compared distributions are different, with 95% confidence. However, what p does not tell us, is how much the two distributions diverge. For this we use the A-test [91], which measures the "effect size" of the difference in the distributions, i.e. how much the two distributions overlap. Vargha and Delaney [91] define A by equation 9.1:

$$A = P(X > Y) + 0.5P(X = Y) \tag{9.1}$$

where X and Y are samples from two different distributions of data.

Vargha and Delaney [91] give boundaries for A which indicate the effect size of the distributions' overlap:

- **No effect**. The distributions are the same: $0.5 \leq A < 0.56$
- **Small effect**. The distributions mostly overlap: $0.56 \leq A < 0.64$
- **Medium effect**. The distributions have some overlap: $0.64 \leq A < 0.71$
- **Large effect**. The distributions do not overlap or only have a small overlap: $0.71 \leq A$

From equation 9.1 it can be seen that the A-test is double-sided, so $A < 0.5$ if on average $Y > X$. The above boundary values show how much A has to deviate from 0.5 to indicate a distribution difference. In our experiment we can therefore use the A value to tell which experiment produced results with higher fitnesses.

Where A values are reported, they are calculated using equation 9.1 with the current experiment's data for X and the comparison experiment for Y. Therefore if $A > 0.5$ it means that the experiment being reported gave higher fitnesses than the comparison. For example, table 9.4 in section 9.4 reports A values greater than 0.5 when comparing that section's evolved genome fitnesses with random, indicating that the evolved controllers have higher fitnesses than random. In cases where $A < 0.5$, or only the magnitude of the effect size is relevant, A is also given normalised to the range $0.5 \leq A < 1$. This normalised A value is denoted by A'.

Comparison to Random

The fitness of evolved controllers are compared to the fitnesses of randomly generated ones with the same communication couplings under the same experimental conditions. This is to test whether the experimental conditions are somehow preventing effective evolution from taking place.

The distribution of evolved controller fitnesses from the final 50 generations of each GA run, are compared to the fitnesses of 50 generations of randomly generated controllers using the Wilcoxon rank-sum and A-test. If the Wilcoxon rank-sum shows that the GA performs no differently to random, or they are different but the effect size is only small (i.e. if $A < 0.64$), then it is concluded that no significant evolution occurred and the experiment is unsuccessful.

Quality of Solution

The quality of the evolved solutions is not as relevant to testing the hypothesis as the speed of the GA for finding the solutions, nevertheless it is interesting to know if coupling affects solution quality.

The quality of the evolved solutions is measured using at the mean best fitness (MBF) of each coupling. The MBF is measured as either the best fitness from the final generation averaged over all runs, or alternatively the best fitness found throughout all generations, averaged over all runs ([27] p245). In this thesis we will measure the MBF both ways.

Speed of Solution

The speed with which the GA finds a "successful" solution is measured by the average number of evaluations it takes to find a solution ([27] p245). In this chapter we measure the number of evaluations as the first generation that finds a successful solution, and then take the mean over all runs for each coupling. We also look at how the first generations to success are distributed for each coupling, and compare with Wilcoxon rank-sum and A-test to determine the effect size of the coupling on the speed of finding a solution.

Before the average number of generations to success can be measured, we first need to define what fitness value is "successful". The success threshold is calculated by taking the best fitness found in each random experimental run and average that over all runs and couplings to get an overall MBF for equivalent, randomly generated controllers. This threshold is chosen because a successful controller should be at least as good as the best controller that can be randomly generated, and the MBF of the random GA gives an easily calculable lower bound on what should be achievable within the experiment.

9.3 Developing Test Benchmarks

9.3.1 Randomly Generated Controllers

As mentioned in section 9.2.5, the evolved solutions should be compared to random ones. If the GA then evolves solutions that are the same or worse than random, the GA can be shown to be unsuccessful. First though we must measure the fitnesses of some randomly generated controllers to get a minimum baseline for how well the evolved controllers should perform.

A population of 10 robots is run for 50 generations. Each robot evaluates a randomly generated genome for one generation. At the end of each generation a new random genome is assigned to each robot and the process repeats. This is repeated for 40 experimental runs.

Coupling	MBF of last generation	MBF over all generations
Tight, Fixed	3.20	10.47
Loose Fixed	2.83	10.05
Loose, Evolved Descriptions	3.08	11.70
Loose Evolved	3.10	11.53
Tight, Evolved	3.00	10.93

Table 9.1: Quality metrics from the randomly generated robot controllers.

The results shown in figure 9.2, and table 9.1 give the quality metrics for fitness of the random generated robot controllers. The MBF over all generations and couplings is 10.94. Consequently, we use a success threshold of 11 to determine whether a genome is successful.

9.3.2 Designing A Controller

In this section we test a controller we have designed ourselves. This is done to in order to get some measure of what a "good" fitness might be and to see whether it is possible to complete the task given the GE grammar presented in section 9.2.1. By comparing these results to our evolved results we can also find out whether it is possible for evolution to develop solutions which outperform one designed by hand.

The following behaviours are expected to feature in a high fitness foraging controller:

- All robots in the swarm should share a common signal to meaning mapping.
- Robots should try to collect from the largest food source.
- Robots should try and move towards signals indicating a large food source.
- If a medium to large food source is in range, or a source that is nearly fully occupied, then the robots should collect for a long period of time to maximise the chances of receiving food.
- If any food source is in range, the robot should try collecting for a short time, in case it is successful.
- If no food sources are within collection range then the robot should move away for a short distance to search for food elsewhere.

The following controller has been written to perform these behaviours:

```
IF (largest size signal) amplitude > 10
  AND (largest size signal) differential > 130
      THEN turn left 45°
      ELSE collect for 500ms
IF (largest size signal) amplitude > 10
  AND (largest size signal) differential < 130
      THEN turn right 45°
      ELSE collect for 500ms
IF nearest tree size > 60%
```

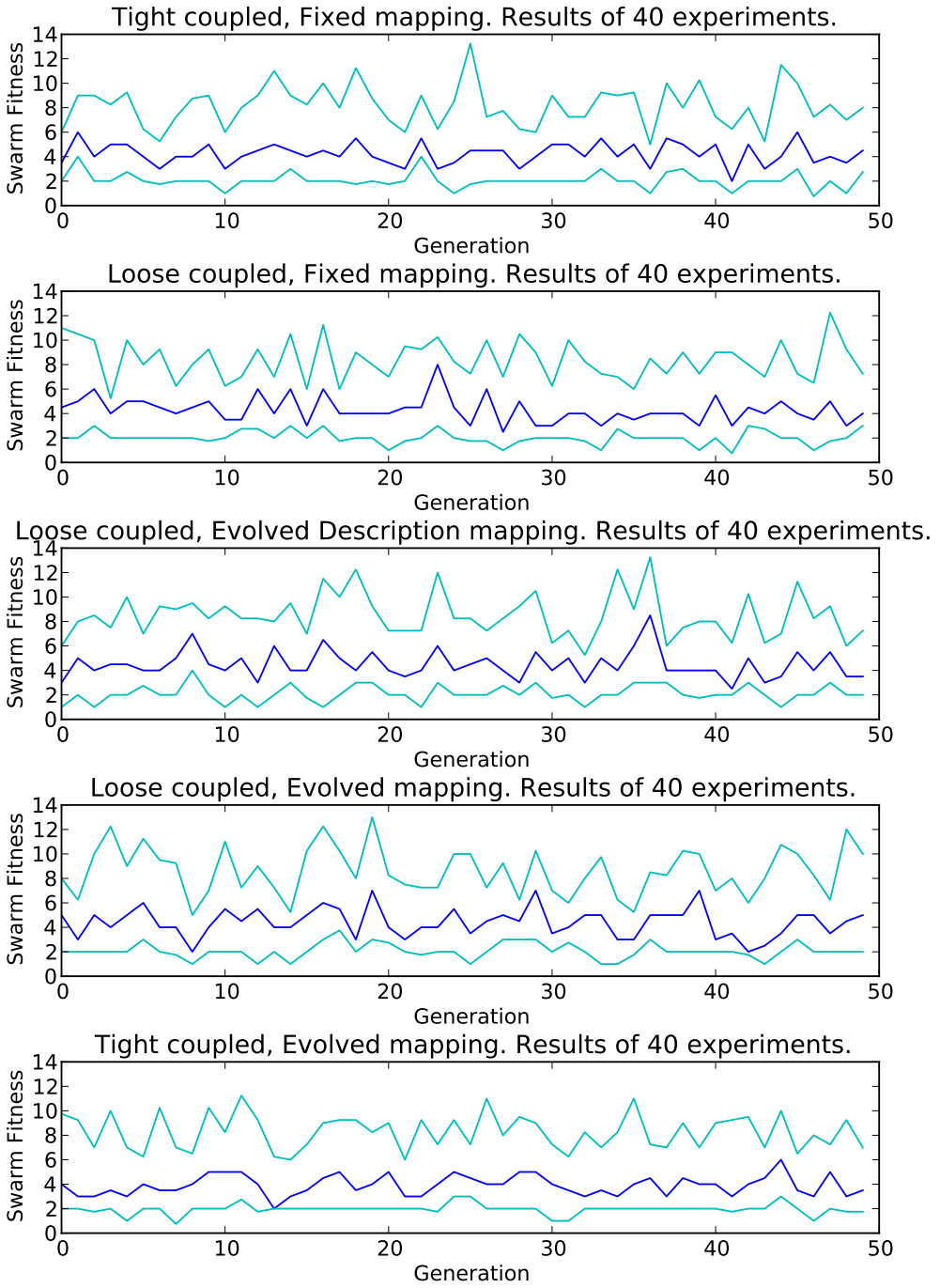


Figure 9.2: The total swarm fitness at each generation from randomly generated controllers. The lines indicate 25th, 50th and 75th percentiles.

```
  OR nearest tree occupancy > 60%
      THEN collect for 4000ms
      ELSE move forward for 500ms
IF Food source in range
      THEN collect for 1500ms
      ELSE move forward for 500ms
```

In couplings where the signal to meaning mapping should be evolved we use the same mapping as the fixed couplings. Consequently only the "Loose, Fixed" and "Tight, Fixed" couplings are necessary in this section.

The first two `IF - THEN - ELSE` statements perform basic phonotaxis towards a signal indicating the largest possible sized food source. If a signal of any frequency is undetectable, then the differential reading for that frequency defaults to 128 and the amplitude to 0. The conditional statements check whether a signal for a large food source is to the left (differential > 128) or the right (differential < 128). If a signal is detected then the robot will turn 45° towards the signal. If no signal is detected then the amplitude should not be more than 10, so the condition is false. There are no `ELSE-IF` statements in the GE grammar, so instead the robot performs an arbitrary actions for as short a time as possible; `collect` is used here because collecting can lead to food being awarded, so is a useful default action. The signal used for the basic "phonotaxis" is the signal indicating a large food source. For loose couplings this is straight-forward since there is only one tone that expresses this information. For tight couplings there are three signals indicating a large food source and these signals also give information about the occupancy. Preliminary experiments were tried where the tight coupling would check for all three of these signals, but this results in lower fitnesses because the number of times the robot must check the audio environment is tripled, and this has a detrimental effect on performance.

To measure the fitness of this controller, each of the 10 robots in the population is given a copy of the controller to evaluate. The genetic algorithm is then run for 50 generations without applying any genetic operators, so that the population remains the same and homogeneous throughout the whole run. The fitness of the robots after each generation is recorded and the experiment is repeated for 40 runs for each coupling type. These fitnesses are measured using the quality metrics from section 9.2.5, and the results are given in tables 9.2 and 9.3.

Table 9.2 compares the fitnesses from the designed controller against randomly generated controllers. It shows that $p < 0.05$ meaning the distributions are different, and that $0.64 \leq A < 0.72$ indicating that changing from random to the designed controller has a medium to large effect on the resulting fitnesses.

9.4 Results

Figure 9.3 shows the results of the experiment presented in order of coupling strength as given in section 5.3.1 in chapter 5. It can be seen that, as with the random controllers from figure 9.2, the swarm fitness is variable from one generation to the next but does not improve over the course of the experimental run.

Table 9.4 shows the Wilcoxon rank-sum and A-test results when the evolved and randomised robot controller fitnesses are compared. Although the Wilcoxon rank-sum $p < 0.05$ indicates that there is a statistically significant difference in the evolved and random fitness distributions, the A-test result $A' < 0.56$ indicates that the effect of this

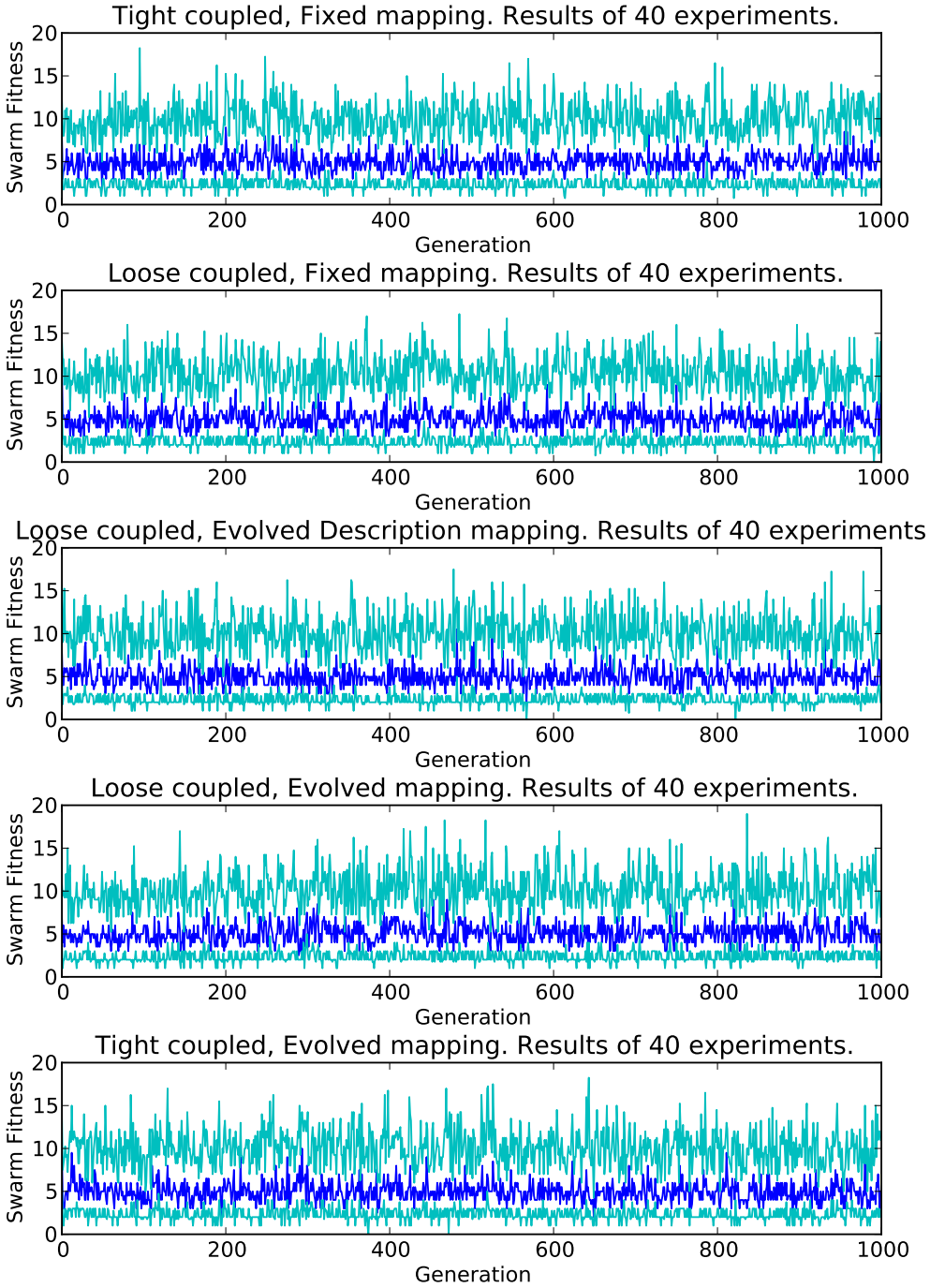


Figure 9.3: The total swarm fitness at each generation from evolved controllers. The lines indicate 25th, 50th and 75th percentiles.

Coupling	p	A
Tight Coupling	0.000	0.696
Loose Coupling	0.000	0.713

Table 9.2: Comparison of the fitnesses from the designed controller against randomly generated controllers. For both couplings tested $A > 0.5$ meaning the designed solution performed better than random. The tight coupling $A' > 0.64$, meaning there are "medium differences", but the loose coupling gave $A' > 0.71$ meaning that the designed solution had a "large" effect on the fitness distributions.

Coupling	MBF of last generation	MBF over all generations
Tight, Fixed	3.55	11.53
Loose Fixed	4.30	11.95

Table 9.3: Quality metrics from the robot controller designed by hand.

Coupling	p	A
Tight, Fixed	6.77×10^{-6}	0.513
Loose Fixed	1.52×10^{-5}	0.513
Loose, Evolved Descriptions	5.14×10^{-4}	0.510
Loose Evolved	2.63×10^{-5}	0.512
Tight, Evolved	4.47×10^{-6}	0.513

Table 9.4: Comparison of evolved and randomised robot controllers from the experiment. Results show the Wilcoxon rank-sum and A-test for the last 50 generations of each coupling type.

difference is too small to be of interest. We can conclude from these results that there is no significant evolution occurring in the experiment.

9.4.1 Discussion

It can be seen that the swarm fitness is highly variable from one generation to the next. Successful behaviours are learnt and then quickly forgotten 1 or 2 generations later. Successful behaviour is then re-learnt and the cycle of forgetting and re-learning continued until the end of the run. In figure 9.3 this cycle manifests itself as a very variable fitness which stays at approximately the same level throughout the entire run.

The inability to "remember" good solutions is most likely caused by the indirect mapping between the genome and the evolved controller. In GE, when a good genome is found, similar genomes do not always translate to similar controllers. This is discussed in section 5.4.2. Although we cannot change this aspect of GE, we can try to improve our results by improving the GA's ability to remember information.

9.5 Elitism and Steady State GA

In order for the swarms to improve over time, the cycle of learning and forgetting successful behaviours must be broken. In this section we re-run our experiment comparing

Coupling	p	A
Tight, Fixed	6.29×10^{-14}	0.522
Loose Fixed	1.76×10^{-14}	0.522
Loose, Evolved Descriptions	8.00×10^{-11}	0.519
Loose Evolved	1.16×10^{-9}	0.518
Tight, Evolved	1.76×10^{-11}	0.519

Table 9.5: Comparison of evolved and randomised robot controllers from the elitism GA experiment. Results show the Wilcoxon rank-sum and A-test for the last 50 generations of each coupling type.

elitism, and a steady state GA to see if results are improved.

9.5.1 Elitism

Introducing elitism into the genetic algorithm may help the successful behaviours to persist in the population from one generation to the next, because it gives the swarm a short memory of the best solution from the previous generation.

To implement elitism in our experiments, the best genome from each generation is copied into the next generation, without any crossover or mutation being applied.

Results

Each experiment is run for 250 generations. Figure 9.4 shows the total population fitness at each generation for each coupling, over 40 experimental runs.

The fitness of evolved genomes are compared with randomly generated genomes from section 9.3.1, and the Wilcoxon rank-sum and A-test results are given in table 9.5. The A score region indicating that any difference in distributions shown by the Wilcoxon rank-sum has no effect size is $0.50 \leq A' < 0.56$. It can be seen that A scores are all at the lower end of this region, so the genomes evolved with elitism have slightly higher fitnesses overall, but differences between evolved and random controllers are very small. As with section 9.4 we can therefore conclude that no significant evolution took place in this test.

9.5.2 Steady State GA

In a steady state GA, instead of generating a new population at the beginning of each generation, only a few members of the population are updated at any time so that some genomes persist in the population whilst others are updated ([27] p58).

In this experiment, robots must evolve their response to the environment, but they must also discover where the food sources are. At the beginning of each generation, the robot controllers are changed and the robot is randomly repositioned, so the swarm has to re-discover where the food sources are *every* generation. By updating and moving only some robots at any one time, the knowledge of where food is located can persist in the environment because the unchanged robots are able to remain at a food source. This knowledge can still be lost over time as more genomes are updated and their robots repositioned, potentially losing the discovery of the food's location. As the population evolves to become better at foraging, the rate the swarm is able to discover food locations should begin to balance the rate at which this knowledge is lost.

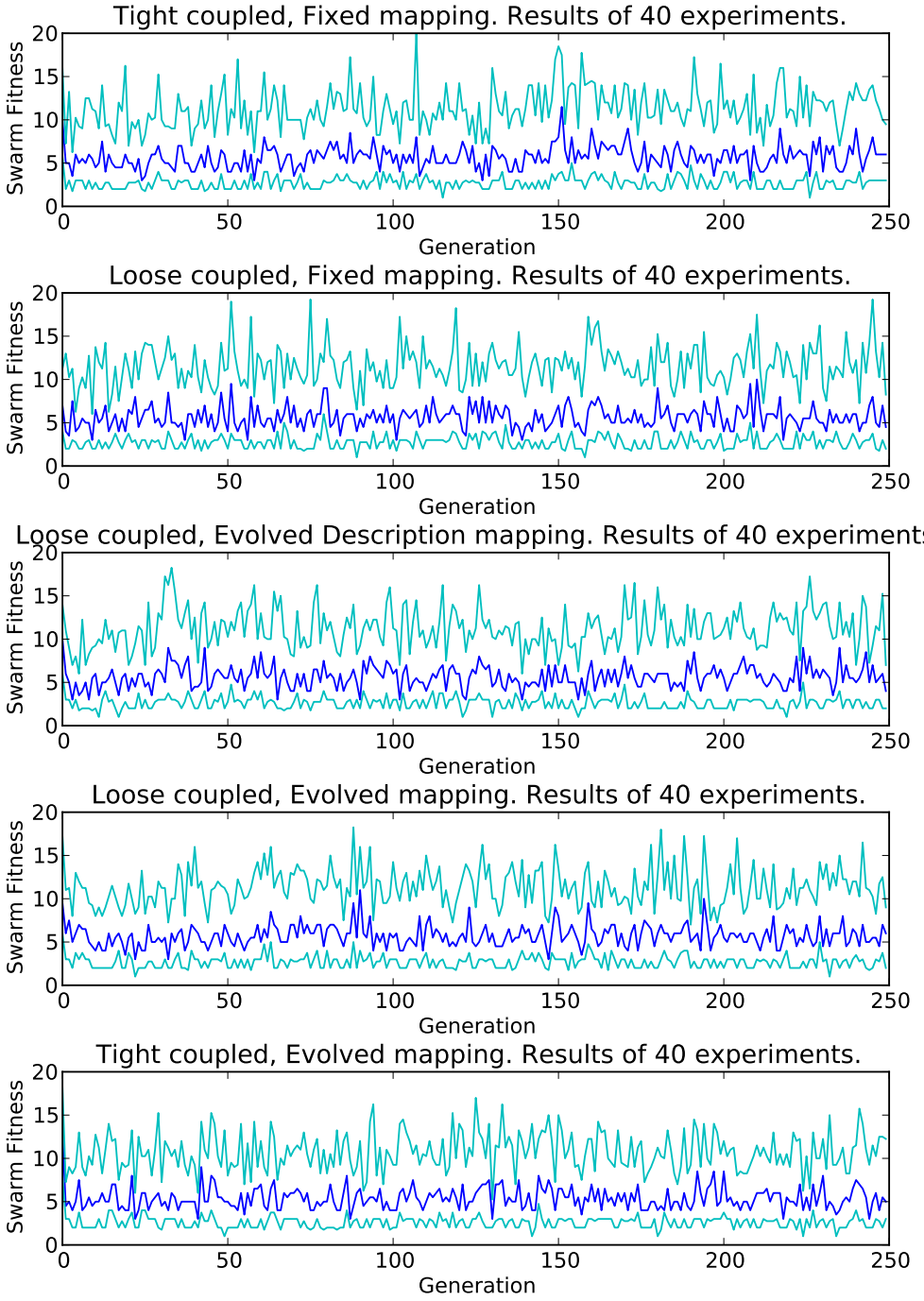


Figure 9.4: The total swarm fitness at each generation when using elitism in the GA. The lines indicate 25th, 50th and 75th percentiles.

Coupling	MBF of last generation	MBF over all generations
Random, Tight, Fixed	9.55	35.67
Random, Loose Fixed	8.85	38.33
Random, Loose, Evolved Descriptions	9.95	36.92
Random, Loose Evolved	8.68	35.60
Random, Tight, Evolved	9.18	35.60
Designed, Tight Coupling	33.67	65.70
Designed, Loose Coupling	35.02	63.58

Table 9.6: Solution quality benchmark metrics for the steady state GA from randomly generated genomes the designed controller from section 9.3.2.

The steady state GA is implemented in this experiment by updating the oldest population member every 12 seconds. Previously, each generation lasts 120 seconds and there are 10 members of the population. Updating one genome every 12 seconds means that each genome is evaluated for 120 seconds before being updated, and it takes 120 seconds to update the entire population. To perform crossover, the genome being updated is crossed with a random other genome in the population, using the same crossover probability as in previous experiments (10%). However, the other genome is not affected by the crossover. This is because updating members of the population asynchronously means that the other genome is still being evaluated, and if we changed the genome during evaluation then the fitness measure we get for it will be wrong. Mutation is not affected by changing to the steady state GA since mutation does not require any other genomes.

Re-evaluating Test Benchmarks

Using a steady state GA makes it easier for the robots to forage by helping locational knowledge to persist for multiple generations. The random and designed controller fitnesses must therefore be re-evaluated so that the population is updated in the same way as the steady state GA, and there can be a fair comparison between the random, designed and evolved GAs. All other test conditions remain the same as described in section 9.3.

Table 9.6 shows the re-evaluated solution quality metrics for both designed and randomly generated controllers. It can be seen that both random and designed controller tests show much higher MBFs than in section 9.3. In section 9.3.1 the randomly generated controllers' MBF over all couplings was 10.94, but using steady state GA the MBF is 36.43, giving over a three times increase fitness. This result shows that updating to a steady state GA makes the foraging task easier to perform.

The overall MBF of the designed controller from section 9.3.2 increased from 11.74 to 64.64, which is over a five times increase in fitness; a greater increase in fitness than the random controllers showed. This suggests that the change to a steady state GA has some other effect that is beneficial to the controller we designed. Possibly the steady state GA makes communication more beneficial. If there are always some robots around a food source then there will always be signals being broadcast about that food source. If, on the other hand, the population updates generationally, then at the start of each generation the swarm must re-discover the food sources, and so it is less likely that

Coupling	p	A
Tight, Fixed	1.11×10^{-26}	0.583
Loose Fixed	6.91×10^{-36}	0.605
Loose, Evolved Descriptions	1.78×10^{-23}	0.578
Loose Evolved	1.08×10^{-17}	0.565
Tight, Evolved	1.09×10^{-56}	0.629

Table 9.7: Comparison of evolved and randomised robot controllers from the steady state GA experiment. Results show the Wilcoxon rank-sum and A-test for the last 50 generations of each coupling type.

Coupling	p	A	A'
Tight, Fixed	0.00	0.269	0.731
Loose Fixed	0.00	0.280	0.720
Loose, Evolved Descriptions	0.00	0.236	0.764
Loose Evolved	0.00	0.233	0.767
Tight, Evolved	0.00	0.245	0.755

Table 9.8: A Wilcoxon rank-sum and A-test comparison of the evolved steady state results and the designed controller steady state results.

there will other robots in the swarm signalling about the food. Consequently, reacting to signals is not as worthwhile in the generational GA as it is in the steady state because there are fewer signals, so our designed controller does not perform as well.

For the steady state GA experiment we use a success threshold of 36, which is the MBF of the randomly generated controllers over all coupling types.

Results

The evolved fitnesses are shown in figure 9.5. The fitnesses are much larger than in previous experiments, and improve over time. Table 9.7 shows the Wilcoxon rank-sum and A-test comparison of the steady state experiment against the equivalent random experiment. As with sections 9.4 and 9.5.1, the p and A scores show that the evolved and random distributions are different, but the effect is only small. When the evolved fitnesses are compared to the designed controller fitnesses (table 9.8) the evolved fitnesses are significantly lower, with A' consistently greater than 0.710 for all couplings[1]. However, A is larger than in previous experiments and the solutions produced have higher fitnesses, so even though this test shows that evolution has no significant effect, the speed and quality metrics described in section 9.2.5 are presented in table 9.9. These metrics shows that there is not much difference between the couplings for both solution speed and quality.

Performing a Wilcoxon rank-sum and A-test on the first successful generations from each coupling run, compared against other couplings gave the results shown in table 9.10. This gives an indication of the couplings' comparative speed at finding a successful solution. No comparisons gave $p < 0.05$ or $A > 0.64$, which indicates that the commu-

[1]Where p is stated as zero, the true p value is less than can be represented by a floating-point number on a 64-bit computer. This minimum value is 2.23×10^{-308}, and any p value less than this number is approximated to zero.

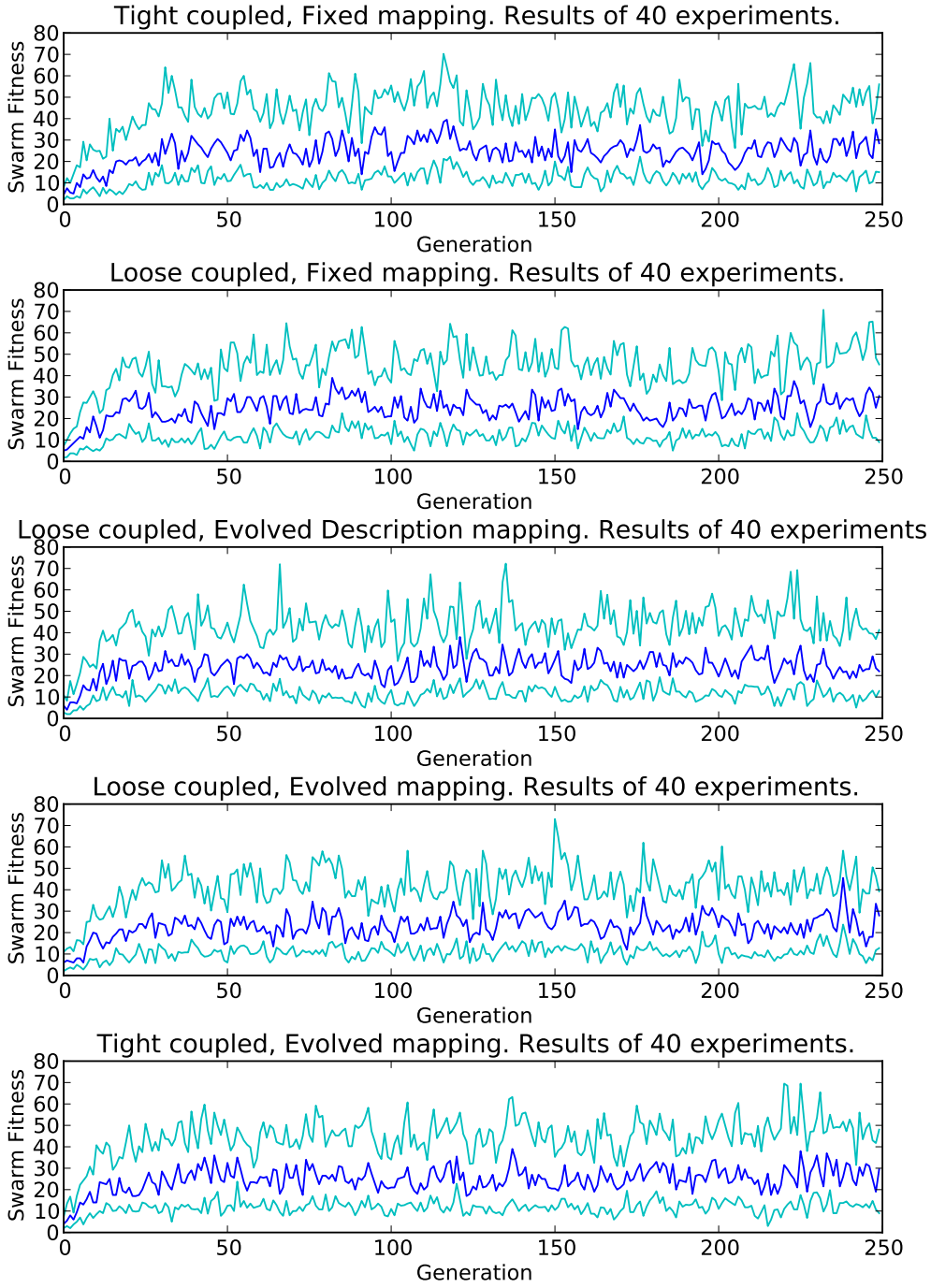


Figure 9.5: The total swarm fitness at each generation when using a steady state GA. The lines indicate 25th, 50th and 75th percentiles.

Coupling	Mean Generations to Success (Number Successes)	MBF of last generation	MBF over all generations
Tight, Fixed	109.50 (34)	12.88	43.40
Loose Fixed	103.68 (31)	11.53	44.42
Loose, Evolved Descriptions	113.70 (33)	11.20	42.80
Loose Evolved	92.81 (26)	11.00	42.12
Tight, Evolved	92.75 (28)	11.72	42.17

Table 9.9: Speed and quality metrics from the steady state GA experiment. Success threshold is 36.

$p/\boldsymbol{A'}$	Tight, Fixed	Loose Fixed	Loose, Evolved Desc	Loose Evolved
Tight, Evolved	0.154/ **0.589**	0.189/ **0.559**	0.089/ **0.571**	0.391/ **0.521**
Loose Evolved	0.130/ **0.541**	0.198/ **0.549**	0.087/ **0.560**	
Loose, Evolved Desc	0.423/ **0.526**	0.307/ **0.544**		
Loose Fixed	0.374/ **0.501**			

Table 9.10: Wilcoxon rank-sum and normalised A-test results for the steady state GA experiment, comparing the distribution of the first generation to find a successful solution from each run.

nication coupling has no effect on the speed of finding a solution in this experiment. Doing a similar comparison comparing the fitnesses from the last 50 generations of each test run (described in section 9.2.5) gave $A' < 0.56$ for each possible coupling comparison. This means that the distributions are the same, and therefore that coupling also does not affect the quality of evolved solutions.

9.5.3 Discussion

In this section, we first try elitism to see if keeping the best solutions in the population for at least one more generation would improve the fitness of the population. However, the results from section 9.5.1 demonstrate that this is not the case for this experiment. When these results are compared to our initial results from section 9.4, the A-test showed that A was always less than 0.56 for each coupling type. Therefore the distributions of the results from this section and section 9.4 have no significant differences, and so elitism has no effect in this experiment.

The experiment with the steady state GA gives much better fitnesses than previous experiments because the population does not lose information about the location of food sources as quickly. Although, as with elitism, controllers evolved with a steady state GA were not able to significantly out-perform randomly generated controllers.

From the elitism experiments of section 9.5.1, the best controller was found with a

fitness of 32 and the loose coupled, fixed signal to meaning mapping:

```
IF Food source in range
      THEN Collect for 2500ms
      ELSE Move forwards for 3500ms
IF Food source in range
      THEN Collect for 4000ms
      ELSE Move forwards for 2000ms
```

The best controller evolved by the steady state GA experiments has a fitness of 69, and used the tight coupled, evolved signal to meaning mapping:

```
IF Food source in range
      THEN Collect for 1000ms
      ELSE Collect for 3500ms
IF Nearest food source occupancy < 72%
  OR 2030Hz Differential > 220
      THEN Move forwards for 4000ms
      ELSE Collect for 1500ms
```

In both cases, the GA was able to evolve controllers with the ability to both collect food and explore for new food sources. Both examples also appear not to use communication as part of the foraging behaviour to any significant degree. The fittest controller from the elitism experiment does not check any audio signals as part of its behaviour, relying solely on its own environmental observations for decision making. The fittest controller evolved by a steady state GA does check one differential reading, but it is unlikely to have much effect on the overall behaviour. If a signal of any frequency is not detectable, then the differential defaults to 128 and the amplitude to 0. signals to the left of the robot will have differential greater than 128 and to the right will be less than 128. Therefore, differential(2030) > 220 checks if there is a signal of 2030Hz to the extreme left of the robot, and will be true only in the event that such a signal is observed. Consequently, the conditional `IF` statement usually simplifies to just the occupancy check.

These experiments may not have produced results that were better than random because the task the robots are performing is very difficult. To test our hypothesis, the population needs to evolve communication so that we can compare the effects of the different communication couplings. However, for the swarm to evolve communication it must also evolve the ability to move towards sounds and prioritise some tones over others. It is probably easier for the GA to evolve foraging behaviour which ignores the audio signals and uses only the information that the robot can observe, even if the resulting fitnesses are perhaps not as high.

9.6 With Phonotaxis

To improve the results from the steady state GA experiment, in this section we add a phonotaxis behaviour to the GE grammar, so that the swarm no longer needs to evolve the ability to move towards a tone, and the task of evolving foraging which uses communication should be slightly easier. A steady state GA is still used to evolve possible controllers for the swarm robots.

The updated grammar is as follows, with the new additions underlined:

```
<code>              ::=  <if> | <if><code>
<if>                ::=  IF <condition> THEN <action> ELSE <action>
<condition>         ::=  <tone><op><tone>
                       | <tone><op><constant>
                       | <condition><logical><condition>
                       | <food data><op><percent constant>
                       | isAtTree
<action>            ::=  <move> | <collect> | <phonotaxis>
<tone>              ::=  <tone distance> | <tone bearing>
<op>                ::=  < | >
<logical>           ::=  AND | OR
<move>              ::=  forwards <time delay>
                       | left <time delay>
                       | right <time delay>
<collect>           ::=  collect <time delay>
<phonotaxis>        ::= phonotaxis <tone const><time delay>
<tone amplitude>    ::=  tone1 amplitude ...
<tone differential> ::=  tone1 differential ...
<food data>         ::=  food source size
                       | food source occupancy
<constant>          ::=  0 | 10 | 20 | ... | 260
<time delay>        ::=  1 | 2 | ... | 8
<tone const>        ::= 1 | 2 | ... | alphabetsize
<percent const>     ::=  0 | 1 | ... | 100

START = <if><code>
```

The phonotaxis behaviour allows the robots to move towards a sound of a given frequency for a time period specified by the genome. If no tone of that frequency can be detected then the robot will instead wander for that time period.

We compare the swarm fitnesses using the above grammar when the number of food source descriptions, $D = 3$ and when $D = 4$. Until this section, we have been using $D = 3$, giving the loose coupled communication 6 signals, and the tight coupled communication 9 signals. With $D = 4$, there are 8 loose coupled signals and 16 tight coupled signals (see equations 5.3 and 5.4 on page 37). With more descriptions there is a much larger difference in the number of signals required for loose and tightly coupled communication. According to the hypothesis, if communication is used, the difference between the swarm fitnesses of each coupling type should be more pronounced.

9.6.1 Simplifying the Audio and Soundboard Model

The audio communication in the model needs to be accurate so that phonotaxis can work properly, otherwise the phonotaxis behaviour is too similar to random wandering. By working in simulation, we can give the robots abilities that they would not have in reality, due to the limitations of the soundboards. This makes it possible to run this experiment, but makes it harder for the results to be extrapolated to a real swarm. However, we have available a higher fidelity model than the one used for this experiment, so it is possible to add realism back into the experiment later to validate the results against a real swarm of robots.

Coupling	MBF of last generation	MBF over all generations
Random, Tight, Fixed	2.48	16.80
Random, Loose Fixed	2.67	15.57
Random, Loose, Evolved Descriptions	3.02	18.18
Random, Loose Evolved	3.20	17.45
Random, Tight, Evolved	3.02	16.38
Designed, Tight Coupling	30.68	58.25
Designed, Loose Coupling	18.43	44.80

Table 9.11: Quality metrics from the random GA with phonotaxis.

For the experiments in this section the audio range is increased so that sound does not decay with distance, giving every tone infinite range. The differential is also idealised so that it doesn't get smaller with distance and gives the correct DOA of the tone. This means that robot has access to all tones currently being played, and can use phonotaxis behaviour to move towards it. However, it also means that comparative distance information in the amplitudes is lost since tones which are closer have the same amplitude as a tone which is very far away.

9.6.2 Re-evaluating Test Benchmarks

In section 9.3.2 we design a controller that reacts to signals about large food sources. A basic phonotaxis is used which checks the signal is present and then uses the signal's differential to determine whether to turn left or right. The GE grammar has been updated to include a phonotaxis behaviour, so the controller can now be simplified to:

```
IF (largest size signal) amplitude > 10
      THEN phonotaxis (largest size signal) for 2000ms
      ELSE collect for 500ms
IF nearest tree size > 60%
  OR nearest tree occupancy > 60%
      THEN collect for 4000ms
      ELSE move forward for 500ms
IF Food source in range
      THEN collect for 1500ms
      ELSE move forward for 500ms
```

The quality metrics for the updated controller and the random GA are presented in table 9.11. The overall MBF from random GA was 16.88 so in this section we will use 17 as the success threshold.

9.6.3 Results

Figure 9.6 shows the total swarm fitness per generation for each coupling where $D = 3$. The fitnesses do not get better over time, and a comparison to the random GA (table 9.12) shows that there were no significant differences between the evolved and random fitness distributions. When compared to the designed controller (table 9.13) the evolved controller produces much lower fitnesses, and the effect size is large for all couplings.

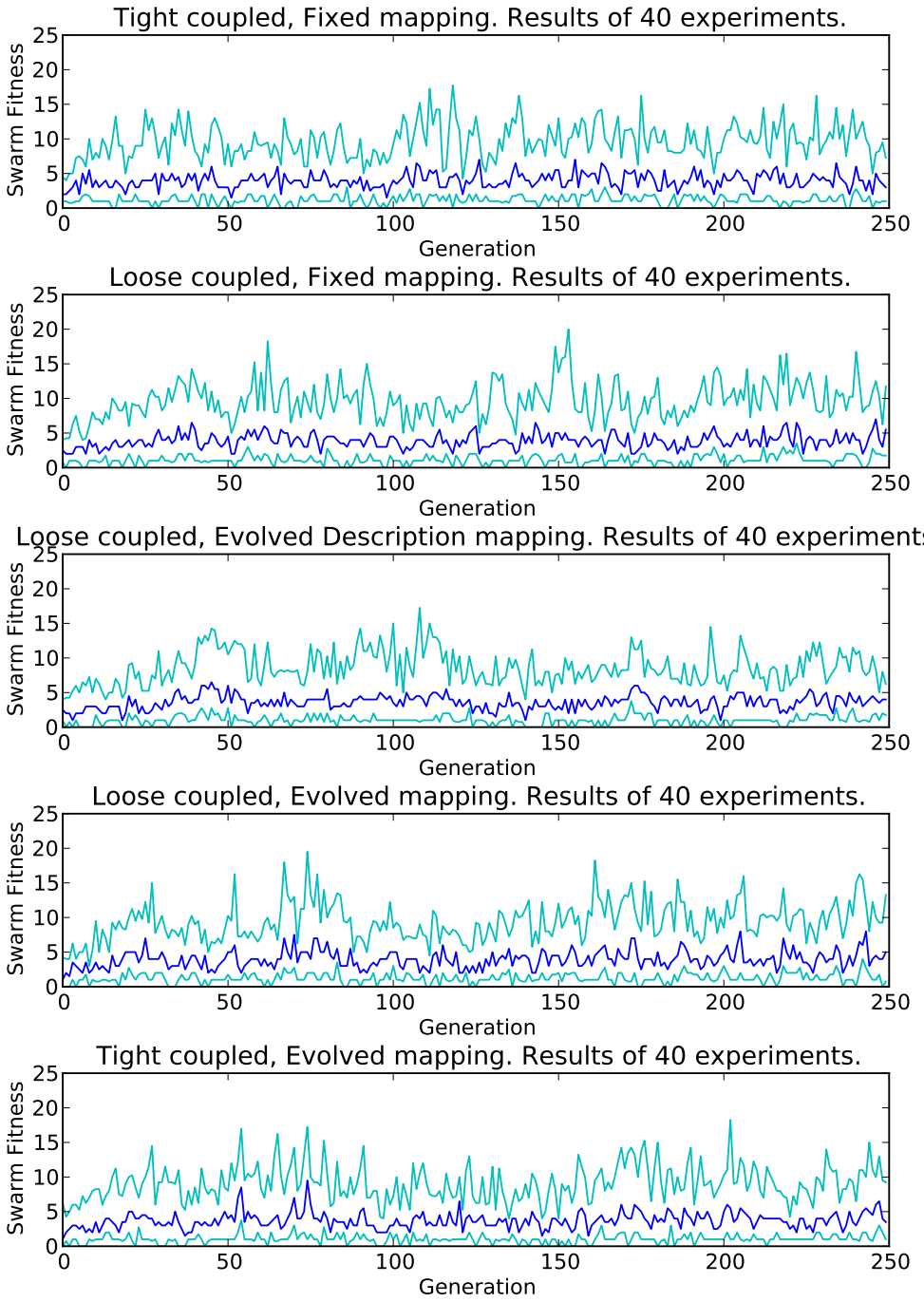


Figure 9.6: The total swarm fitness at each generation from evolved with phonotaxis behaviour in the grammar. The lines indicate 25th, 50th and 75th percentiles, and $D = 3$.

Coupling	p	A
Tight, Fixed	1.87×10^{-44}	0.526
Loose Fixed	2.52×10^{-39}	0.525
Loose, Evolved Descriptions	6.11×10^{-26}	0.520
Loose Evolved	1.18×10^{-38}	0.525
Tight, Evolved	5.65×10^{-22}	0.518

Table 9.12: Results of Wilcoxon rank-sum and A-test of the phonotaxis experiment fitnesses where $D = 3$ compared against random.

Coupling	p	A	A'
Tight, Fixed	0.00	0.221	0.779
Loose Fixed	0.00	0.291	0.709
Loose, Evolved Descriptions	3.41×10^{-83}	0.241	0.759
Loose Evolved	3.78×10^{-48}	0.315	0.685
Tight, Evolved	3.61×10^{-146}	0.160	0.840

Table 9.13: Results of Wilcoxon rank-sum and A-test of the phonotaxis experiment fitnesses where $D = 3$ compared against the designed controller.

When the alphabet size was changed to $D = 4$, the Wilcoxon rank-sum and A-test again showed that the distribution of fitnesses for the last 50 generations is statistically no different to random. According to the hypothesis, the speed to find a successful solution should diverge more between couplings in the $D = 4$ experiment, since tight coupling uses 16 signals, and loose coupling uses only 8 when $D = 4$. With twice as many signals to interpret, the tight coupled communication strategy should be slower to evolve successful solutions. However, a Wilcoxon rank-sum and A-test comparing the first generation to find a successful solution (table 9.14) showed no statistical difference between the couplings.

To see if an increase in alphabet size affects the GA speed, the distribution of first successful generations for each experimental run is compared between the $D = 3$ and $D = 4$ phonotaxis experiments. The Wilcoxon rank-sum and A-test results are given in table 9.15. These results show that the smaller alphabet size was slightly faster to evolve successful solutions, and $p < 0.05$ and $A > 0.64$ in three out of the five couplings. However, since neither test shows signs of successful evolution, no definite conclusions about increased solution speed can be drawn from these figures.

9.6.4 Discussion

From these results, it is clear that adding phonotaxis to the GE grammar causes the GA to evolve genomes with lower fitnesses. Comparing the speeds of the phonotaxis experiment from this section against the steady state without phonotaxis, from section 9.5.2, shows that the two experiments are as fast as each other to reach a successful solution, meaning that the additional phonotaxis action gives no speed advantage to the GA. The Wilcoxon rank-sum and A-test results of this comparison are given in table 9.16.

The best controllers evolved by the GA do not make much use of the new phonotaxis behaviour. The following controller used the "Tight, Fixed" coupling where $D = 3$, with a fitness of 49:

$p/\boldsymbol{A'}$	Tight, Fixed	Loose Fixed	Loose, Evolved Desc	Loose Evolved
Tight, Evolved	0.470/ **0.577**	0.432/ **0.566**	0.927/ **0.502**	0.414/ **0.571**
Loose Evolved	0.863/ **0.512**	0.882/ **0.502**	0.388/ **0.558**	
Loose, Evolved Desc	0.227/ **0.589**	0.453/ **0.562**		
Loose Fixed	0.850/ **0.508**			

Table 9.14: Wilcoxon rank-sum and normalised A-test results for the phonotaxis experiment where $D = 4$, comparing the distribution of the first generation to find a successful solution from each run. These results show that no coupling was significantly faster than any other.

Coupling	p	A
Tight, Fixed	0.038	0.667
Loose Fixed	0.242	0.580
Loose, Evolved Desc	0.030	0.647
Loose Evolved	0.018	0.689
Tight, Evolved	0.501	0.549

Table 9.15: A Wilcoxon rank-sum and A-test comparison of the first generations to find a successful genome from each run. The numbers show the comparison between the phonotaxis experiment when $D = 3$ and $D = 4$, with $A > 0.5$ meaning that the former found a successful genome sooner. Success threshold is 17 for both experiments.

Coupling	p	A	A'
Tight, Fixed	0.768	0.473	0.527
Loose Fixed	0.485	0.485	0.515
Loose, Evolved Desc	0.936	0.489	0.511
Loose Evolved	0.311	0.575	0.575
Tight, Evolved	0.853	0.562	0.562

Table 9.16: A Wilcoxon rank-sum and A-test of the first generation to find a successful solution in the phonotaxis $D = 3$ when compared to the steady state experiment (section 9.5.2). Where $A > 0.5$, indicates the steady state experiment was slower than with phonotaxis. Success threshold is 17 for the phonotaxis experiment and 36 for the steady state.

```
IF Food source in range?
   THEN Collect for 500ms
   ELSE Collect for 1000ms
IF Nearest food source size > 54%
   THEN Collect for 3000ms
   ELSE Phonotaxis (1470Hz) for 1500ms
```

This controller has some of the features of a good controller, specified in section 9.3.2. It collects from any nearby food source for a short amount of time, then tries to collect from a medium sized food source for a longer time. In the tight, fixed coupling a 1470Hz signal indicates a food source $33\% \leq \text{size} < 66\%$ and $33\% \leq \text{occupancy} < 66\%$. So if there are no large enough food sources nearby then it will either move towards a different food source's signal, or if no such signal is detected then it wanders the arena.

The second highest genome evolved in this section used the "Loose, Evolved" coupling with $D = 3$ and its fitness was measured as 45:

```
IF Food source in range?
   THEN Collect for 4000ms
   ELSE Collect for 4000ms
IF 1100Hz Amplitude < 1650Hz Amplitude
   THEN Collect for 4000ms
   ELSE Collect for 1000ms
```

This genome makes no use of the phonotaxis action and was successful only because the robot happened to have started the generation within range of a food source.

This leaves the question of *why* adding phonotaxis to the grammar makes the fitnesses worse. The only option from the `<action>` grammar that directly awards food is `collect`. With the addition of the `phonotaxis` option the chance of a genome selecting the `collect` action is reduced from 50% to 33%, so from this change alone we might expect fitness to reduce by a third. The MBF of the random steady state GA without phonotaxis was 43, so with the extra phonotaxis action we might expect a random MBF of approximately 29, however the random MBF was measured as 23.63. The addition of the extra action should not affect the designed controller's fitness, because we do not need to randomly select from the `<action>` options, yet this also had a lower fitness when using phonotaxis. Consequently the phonotaxis must have some additional negative effect on the robots' foraging ability.

One possible explanation is that the phonotaxis behaviour makes the robots better at exploration, but also forces them to explore, even when the reward would be easier to get by waiting at the food source. Another possibility is that the communication is not used, and introducing actions which force robots to react to signals may lead to the robots moving around without good cause. If there is no tone present, phonotaxis will force the robot to wander, so phonotaxis could cause a robot to lose a food source it has discovered. Since the results showed no significant differences between couplings, this supports the theory that the communication was unused for the foraging task. Both of these possibilities have the knock on effect of meaning that there are fewer robots at the food source signalling to other robots. there are then fewer audio signals for the robots to respond to, making it harder to locate food sources using phonotaxis.

9.7 Conclusion

In this chapter we try several approaches to running the experiment described in chapter 5, none of which adequately test the hypothesis of this thesis.

One factor in the failure of these experiments is that the fitness function does not sufficiently reward collaboration. In our experiment, collaboration is encouraged by the food sources giving out more food when there are multiple robots collecting from it. However, the robots are able to collect food individually and collaboration happens by robots accidentally, rather than purposefully, collecting from the same source. In this experiment, a robot that tries to collaborate with others would prioritise large food sources over small ones where it can quickly collect food. This behaviour would be well rewarded if other robots in the swarm have the same behaviour. However, the swarm is heterogeneous, and so other robots might not be so helpful. Individualism is easier to evolve because a single robot can collect small amounts of food, giving its genome a small fitness. Collaborative behaviour requires multiple robots in the swarm trying to cooperate before food can be collected. If the collaboration is successful the robots are rewarded more than the selfish robots, but if unsuccessful it is not rewarded at all. It is difficult for collaborative behaviour to propagate through population, because multiple robots need to be collaborating before it become better rewarded. However, multiple robots will not be cooperating unless the collaborative behaviour has propagated in the population. Such problems could be avoided by using a homogeneous swarm to evaluate each genome.

Throughout this chapter, we evaluate the designed controller using a homogeneous swarm, so all the robots in the swarm prioritise large food sources. However in the first experiment of this chapter (section 9.4), the designed controller performed as well as the random and evolved solutions, as shown in tables 9.1 and 9.3. This suggests that the GA has a search space filled with mediocre solutions where it is very difficult to find an outstanding genome. A flat topology of the GA search space would make it difficult to evolve any form of collaborative behaviour, since no genome gives any particular advantage over another.

The main reason the experiment is unsuccessful in testing our hypothesis is that the task can be accomplished by the robots without needing to explicitly use communication to share information about food sources. The swarm performs well enough without communication in order for this to be the evolved solution each time an experiment is run. The lack of collaboration does not encourage the evolution of communication. Robots foraging alone just explore until they find a food source. No communication is required for this behaviour. In the steady state experiment, information about food source location is able to persist in the environment. However this is due to the robots being *physically* located around the food source, making it easier to collect larger foods, rather than because robots communicate that information to each other. This is why the evolved solutions were not significantly better than random. Consequently, the effects of different coupling strategies on the swarm fitness cannot be measured, because they have no effect on the behaviour of robots in the swarm.

It would seem from the evidence in this chapter that the current experiment is not appropriate for testing our hypothesis. In subsequent tests we should test both homogeneous and heterogeneous swarms, improve our fitness measure, and change the swarm's task to something that cannot possibly be performed without using communication. Only when communication is essential to the task being performed, can we be certain to test the effects of different communication strategies.

Chapter 10

Coupling in Swarms Using Indirect Communication

10.1 Introduction

In chapter 9 we attempted to test out hypothesis on a swarm of foraging robots, but found that the robots are able complete the foraging task by ignoring communicated information and acting independently. In this chapter we try a similar foraging experiment, but make it impossible to collect food without making use of communication.

Heterogeneous and homogeneous swarms are compared, and it is found that the homogeneous swarms tend to evolve controllers which are better at collecting food and collaborating with other robots in the swarm. Comparing the couplings shows that the loose coupled communication strategy gives higher fitness solutions than the tight coupled strategy, and may be quicker to evolve successful solutions. However, an uncoupled communication strategy using only a single tone to describe all food sources gives higher overall fitnesses more quickly than either the loose or tight couplings.

Further investigation shows that when only information about the size of the food sources is shared, swarms are able to collect the most food. Conversely, sharing information about the food source's occupancy results in low fitness solutions. Consequently some of the information communicated to the robots is shown to be less useful than other information, causing differences in the swarm fitnesses and GA speeds under different communication couplings. We conclude that there is a trade-off between the cost of interpreting information and the benefit gained from having that information, but using loose coupling could make the interpretation easier.

In section 10.2 we describe the experiment we used to ensure that the robots would evolve to react to communicated information, and how the robots are evaluated on their ability to interpret and act on this information. In section 10.3 we develop benchmarks against which the experimental results can be tested. In section 10.4 we present the results of the experiment and go on to test the swarm fitness using additional couplings in section 10.5. Finally in sections 10.6 and 10.7 we discuss the significance of our findings and suggest potential areas of future research.

10.2 Experimental Method

In this experiment it should be impossible to complete the task without acting on communicated information, so that the hypothesis can be tested.

As with chapter 9 the robots forage for food from food sources of size n, requiring n collector robots to be present before giving each collector n "foods". Each food source has a size and occupancy level that it communicates to the robots.

The following measures have been taken to ensure that the robots use communication to perform foraging:

- There is no direct, robot to robot communication. The food sources signal the size and occupancy information about themselves. Consequently, there is a fixed signal to meaning mapping and the GA must evolve controllers to react to this mapping.
- Different communication couplings are tested by changing the signal to meaning mapping of the food sources.
- Robots can not collect food unless they are reacting to the signal currently being played by the food source in range. If a robot happens to be wandering within range of a food source it is not given food.
- The evolved controller is much simpler and the corresponding genome is much shorter. The new controller maps between a list of which signals are currently playing, and a high level behaviour to use in response to the signals.
- The GA is evaluated using heterogeneous and homogeneous swarms to see whether homogeneity encourages collaboration between robots in the swarm.

In the rest of this section we describe in detail the experimental setup for this chapter.

10.2.1 Task

There are D food sources in the environment, each is a different size such that each food source uses a different frequency signal to describe its size. Food source size remains constant over the experimental run. The objective of the task is to have the collector robots distribute themselves around the sources, so that by the end of every generation each food source is fully occupied and all available food can be collected. The food sources are positioned in the environment so that it is not possible for one robot to be within collection range of two sources at the same time, so as to avoid ambiguity.

At the end of the generation, each food source distributes food if there are enough collectors. For a food source size n with c collector robots:

- if $c < n$: then no collectors are awarded food.
- if $c \geq n$: then n collectors at the food source are given n food each. Remaining collectors receive no food.

For a robot to count toward c, it must be both using phonotaxis to move towards the food source *and* be within collection range of the source at the end of the generation. Without this provision, the robots would be able to collect food by random wandering alone, albeit with only limited success.

Swarm fitness is measured as the total amount of food collected by the swarm. By not awarding food if there aren't enough collectors we encourage the swarm to explore and find sources that are occupied by other robots. if there are too many collectors then the potential to collect more food is wasted by robots not being at other sources, encouraging robots to find less occupied sources.

10.2.2 Communication

In our previous set of tests, the collector robots signalled to each other whenever they found a food source, describing the source's size and occupancy status. In some cases we used evolution to determine what frequency signal the collector should use to express that information.

In this experiment the food sources are the ones signalling their own status to the robots in the swarm. Consequently we do not test evolved communication couplings because the food sources are not subject to evolution. Instead we test our hypothesis using the following communication strategies:

- **Single Signal, Uncoupled**. Each food source uses the same frequency signal to indicate its location only, so the signal contains no size or occupancy information. The only real coupling between information and the signal used is that a signal can be interpreted as "A food source is here".
- **Loose Coupling**. There are $D + D$ signals, one for each size level and one for each occupancy level. Food sources alternate between playing the signal that describes its size and its occupancy.
- **Tight Coupling**. There are $D \times D$ signals, one for each combination of size and occupancy level. Food sources play the signal that describes their current size and occupancy.

In the uncoupled and loosely coupled communication, it is possible to have two tones of the same frequency playing at the same time. However, in section 8.5.3 of chapter 8 we establish that the soundboard model is not capable of simulating this conflict, and this is solved by having the model instead return the simulated amplitude and DOA of the nearest tone only. For loose coupled communication the food sources are different sizes but the occupancy might be the same, causing conflicts in the occupancy signalling. We can minimise this by making the food sources alternate between expressing the size and occupancy signal asynchronously to attempt to reduce the number of conflicts. For uncoupled communication conflict is unavoidable because all the food sources use the same frequency to signal, so a robot can only detect the closest food source. Consequently, as with the previous chapter, the swarm fitness is dependent on where robots are positioned at the start of each generation.

Hypothesised Outcome

According to our hypothesis we would expect the tight coupled communication to cause the robot fitnesses to improve over time the slowest of all three couplings tested. It requires the most signals to express the food sources' information, so the robot controllers have the maximum number of signals to interpret and prioritise. Loose coupling is hypothesised to have a more rapid improvement in swarm fitness over time, because it expresses the same size and occupancy information using fewer signals. The uncoupled

Tone Number	Priority	Action
1	3	phonotaxis
2	7	wander
3	1	phonotaxis
4	9	wander
5	8	phonotaxis
6	6	wander
7	5	phonotaxis
8	2	wander
9	4	wander

Table 10.1: An example controller for when $D = 3$.

communication is expected to perform poorly because information about food source is not available to the robots. However, there is only one signal that the controllers need to react to, so the uncoupled communication is expected to show the most rapid initial improvement but result in the worst performing swarms.

10.2.3 Evolved Controller

For each tone, the evolved controller dictates which action to associate with the tone, and the priority to give that action. Possible actions are `wander` or `phonotaxis`. The `wander` behaviour causes the robot to randomly explore the environment. The `phonotaxis` behaviour is as described in section 9.6, using a simplified model of the audio communication with infinite signalling range and accurate DOA estimation.

Every 5 seconds, the collector robot listens for tones in the environment. If a robot detects a single tone it automatically perform the associated action, and if it can detect multiple different tones then it performs the action associated to the tone with the highest priority. If no tones are detected the robots automatically perform the `wander` action because there are no tones for the phonotaxis action to respond to. An example of a possible controller is given in table 10.1.

In the three types of coupling we test (given in section 10.2.2), the GA evolves the action and priority for D^2 tones. This is the number of tones used in the tightly coupled communication strategy. In this way, the robot swarms for each tested coupling are equivalent and each GA must evolve the same amount of genes. Our hypothesis is tested by changing the way the information is presented to the robots by the food sources.

This new controller addresses some of the problems encountered in our previous chapter. The last controller presented the robots with very low level sensor inputs and actuator outputs, and required them to evolve communication, phonotaxis and exploration. With this new controller there are only high level, pre-processed inputs and pre-programmed outputs. The GA must evolve the linkage between them, and in doing so finding the best way to react to the information being communicated.

Genome to Robot Controller Mapping

The controller is represented by a genome which is a fixed-length sequence of $2D^2$ integers, representing the priority and action for each tone. These priority and actions are stored in the following order in the genome:

$$\text{priority}_1, \text{action}_1, \text{priority}_2, \text{action}_2 \ldots \text{priority}_{D^2}, \text{action}_{D^2}$$

The action is is represented by either a 0 or 1, where 0 = `wander` and 1 = `phonotaxis`. The priority is a number between 1 and D^2, with larger numbers meaning a higher priority. No two priority genes are allowed to have the same value, otherwise it would cause a conflict when two tones of the same priority are detected. As an example, the genome given in table 10.1 is represented by the sequence:

```
3, 1, 7, 0, 1, 1, 9, 0, 8, 1, 6, 0, 5, 1, 2, 0, 4, 0
```

10.2.4 Genetic Operators

To derive the next generation of genomes from the current population, the following genetic operators are performed:

- Tournament selection.
- Position based crossover.
- Random and swap mutation.

Our representation for the tone priorities is similar to a "permutation representation". This is when genome has fixed, non-repeating values and the evolved information is the order, or permutation, of those values within the genome [27]. An example is travelling salesman problem, where each city must be visited once, but the *order* the cities are visited is the information being evolved. In this experiment, the priority genes have fixed non-repeating values, but their position in the genome is more important that the order which the priorities occur. For example, the genome 5 1 2 3 4 is completely different to 1 2 3 4 5 because no two values have the same position. However if the order was important, as with the travelling salesman problem, they would be similar or the same because the ordering is mostly preserved.

The non-repeatability of the priority genes poses a problem when applying crossover or mutation, as these operations could lead to invalid controllers. Permutation representation GAs have various crossover and mutation methods designed to prevent gene repetition. In the remainder of this section we describe the crossover and mutation technique used in this experiment.

Tournament Selection

As with chapter 9, the parents for the next generation are selected by tournament. Two random genomes from the current population are picked. The one with the highest fitness is copied into the next generation. This is repeated until the next generation's population is full.

Position Based Crossover

Position based crossover [88] is a permutation representation crossover technique intended for preserving positional information during crossover [85], without causing genes to repeat. The crossover process is shown in figure 10.1. First, half of the genes from the first parent are randomly selected and copied into the same position of the offspring. Next the genes that appear in the offspring are removed from the second parent, and

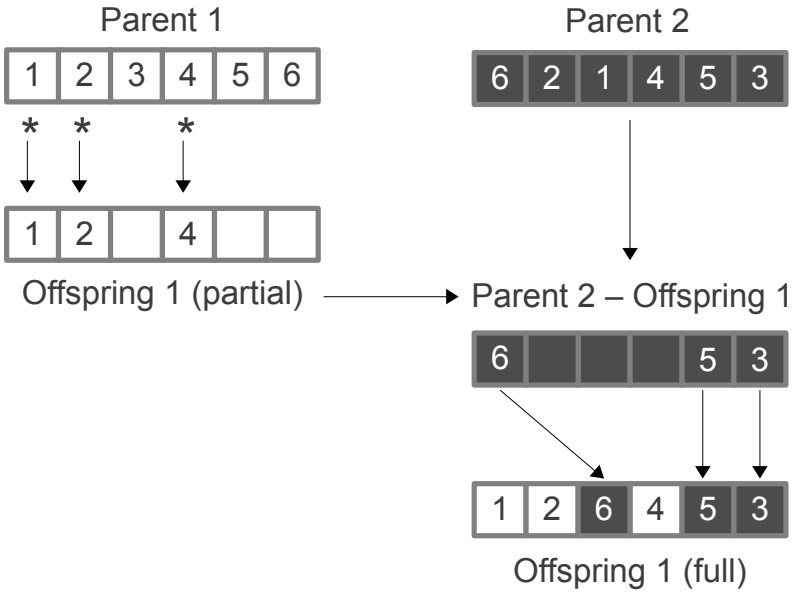


Figure 10.1: Position based crossover of two genomes. To get the second offspring the process is repeated starting from parent 2.

the remaining genes are then placed into the offspring in the order they appear in the second parent.

In this experiment, we group the priority and action pair for each tone and perform crossover on the sequence of pairs. Our implementation of position based crossover is therefore:

1. Copy randomly selected priority-action pairs from the first parent into the offspring.
2. Any pairs in the second parent which have the same priority as ones in the offspring are removed.
3. The remaining pairs from the second parent are copied into the empty spaces of the offspring.

Figure 10.2 shows an example of this crossover on genomes with three priority-action pairs.

Random and Swap Mutation

After tournament selection and crossover, each gene in each genome is mutated with a 10% probability.

To mutate a gene representing an action, its value is replaced randomly with either a 0 or 1 representing the wander or phonotaxis actions respectively. To mutate a priority gene, its position is swapped with another randomly selected priority gene from the genome. Unlike with crossover, the corresponding action genes do not swap positions. Figure 10.3 shows an example of both types of mutation.

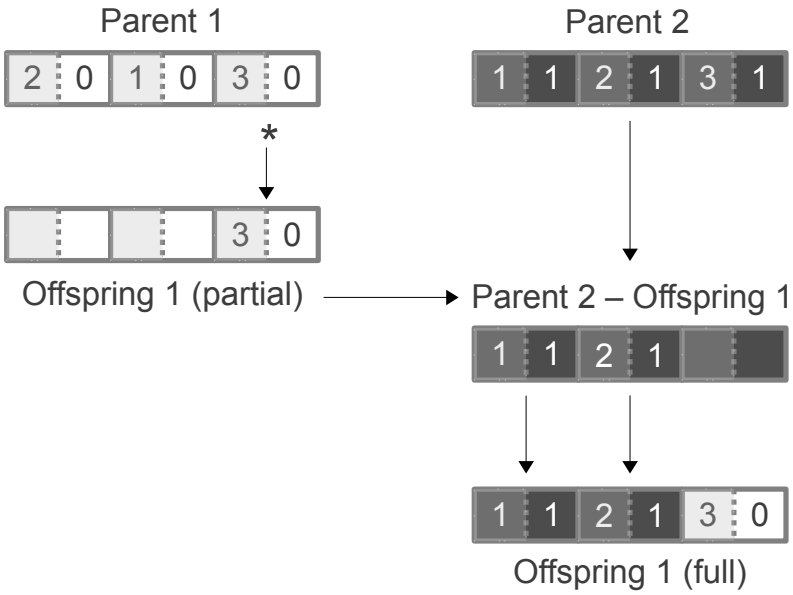


Figure 10.2: Demonstration of how position based crossover is implemented in this chapter. Truncated genomes are used in this example, of the form: priority_1, action_1, priority_2, action_2, priority_3, action_3.

Two different types of mutation are used because there are strict limitations on the priority genes, so a more restrictive mutation method is necessary to prevent invalid genomes being generated. Conversely, the actions selected by mutation or crossover can take any value and it does not matter if the same action is used twice, so more flexibility is needed to allow the genome to use any permutation of actions for the D^2 tones.

10.2.5 Implementing Collective Evolution

In section 5.4.1 we compare heterogeneous and homogeneous robot swarms in the context of our research. A heterogeneous swarm, evaluating genomes in parallel is used in the experiment described in chapters 5 and 9. This is because genomes can be evaluated in parallel with each other making the genetic algorithm much quicker to run, and also because it is easier to decentralise a heterogeneous swarm if the experiment is to be repeated using embodied robots.

In this chapter we test both types of swarm to avoid the problem of fitness measures not necessarily being representative of a genome's performance, and to try and encourage the evolution of collaborative behaviour.

Heterogeneous Swarm

In the heterogeneous swarm each genome in the population controls a robot in the population, as shown in figure 10.4a. Therefore the number of genomes in the GA population must be the same as the number of robots in the experiment. Each robot

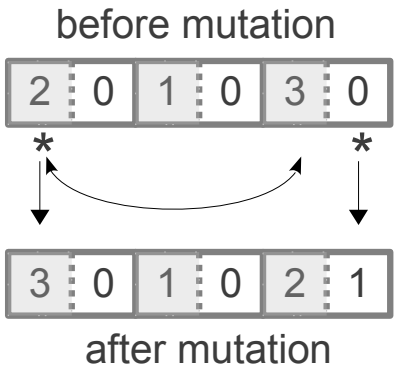
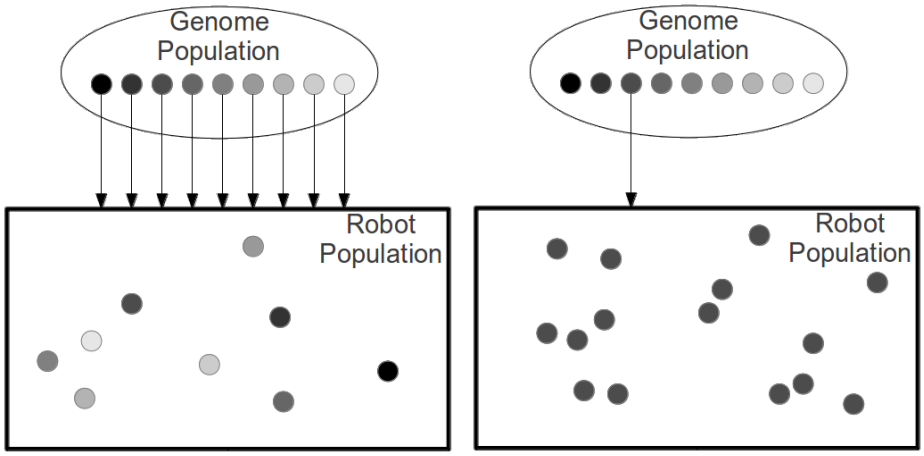


Figure 10.3: Example of mutation applied to an action and a priority gene (shaded). Mutated priority genes swap position with a random other priority gene, whilst the mutated action gene keeps its position but changes its value.



(a) Heterogeneous Swarm (b) Homogeneous Swarm

Figure 10.4: In a heterogeneous swarm (a) each genome is assigned to a different robot, meaning there must be the same number of robots as genomes in the population. In a homogeneous swarm (b) all robots are assigned the same genome, the genome population does not need to be the same size as the number of robots.

evaluates a different genome, so the robots are all executing different controllers and interpreting the information in the environment in different ways.

In chapter 9 we conclude that one of the reasons the previous experiment fails to produce conclusive results, is that the fitness function does not reward collaborative behaviour well enough. Consequently, for this experiment we measure a robot's fitness as the amount of food it collected plus the mean amount of food collected by the robots in the swarm. The formula for calculating a robot's fitness is given by equation 10.1, for a swarm of N robots where fitness_i is the fitness and f_i the amount of food collected by robot i:

$$\text{fitness}_i = f_i + \frac{1}{N}\sum_{j=1}^{N} f_j \tag{10.1}$$

Once the fitness of each genome in the current population has been calculated, the robots are randomly repositioned in the arena ready for the next generation of genomes to be evaluated.

Homogeneous Swarm

In a homogeneous swarm, each genome in the GA population is assessed by putting its derived controller onto all robots in the swarm, as shown in figure 10.4b. The fitness of a genome is then measured by making the robots perform the task and summing the amount of food collected by all the robots. Equation 10.2 shows the fitness function of a homogeneous swarm, where fitness_i is the fitness of genome i, N is the number of robots and f_j is the amount of food collected by robot j.

$$\text{fitness}_i = \frac{1}{N}\sum_{j=1}^{N} f_j \tag{10.2}$$

The fitness is measured for all the genomes in the population this way, and selection, crossover and mutation are applied as with the heterogeneous swarms.

10.2.6 Other Experiment Parameters

There are 10 collector robots in both types of swarm, and the homogeneous swarm has a GA population of 10 genomes. There are 3 food sources in the arena of sizes 1, 3 and 5, meaning that at least 9 robots are required in order to collect all the available food. The food sources play each signal tone for 1.5 seconds. After this period, their size and occupancy is re-evaluated and the corresponding signal tone is played. For the loose coupled communication the food sources alternate between expressing size and occupancy every 1.5 seconds. Table 10.2 shows the remaining GA parameters used in the experiment.

Figure 10.5 shows the arena used in this experiment. The arena is 2×2 metres in size and the food sources are all spaced 1 metre apart in an equilateral triangle. The food sources have a collection range of 0.3 metres, so it is not possible to be in collection range of two food sources at the same time. Collector robots are randomly positioned at the start of each generation, with a uniform distribution across the arena.

Parameter	Value
GA Population Size	10
Number of Generations	500
Generation Time	100 seconds
Coupling Parameter D	3
Tournament Size	2
Crossover Probability	10%
Mutation Probability	10%

Table 10.2: Genetic algorithm parameters.

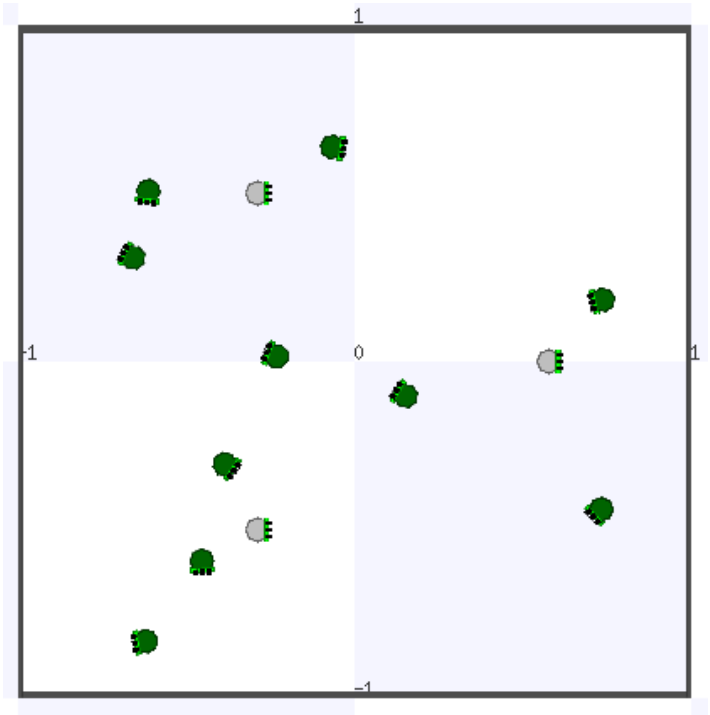


Figure 10.5: Arena containing 3 food sources and 10 collector robots, food sources are coloured light grey.

10.2.7 Measuring the Results

In this chapter we use the same assessment metrics as the previous chapter, described in section 9.2.5. To summarise, these metrics are:

- Test that evolution is effective by comparing the distribution of evolved genomes' fitnesses, to the fitnesses of randomly generated genomes. If a Wilcoxon rank-sum test shows that the two distributions are different to a 95% confidence level, and the effect size is medium or large, then the results is statistically significant; i.e. if the Wilcoxon rank-sum $p < 0.05$ and the A-test $A > 0.64$.
- Measure the mean best fitness (MBF) of each coupling to compare the quality of the evolved solutions.
- Measure the GA speed by examining the earliest generation that found a "successful" genome in each experimental run. These are then compared for each coupling to see if the coupling affects the speed.

In this experiment there is a fixed amount of food available to the swarm each generation. We can define a successful genome as one that manages to collect from the largest food source, which in this experiment is size five. The total swarm fitness if a large food source is collected is at least 25. For the heterogeneous swarm, substituting these values into equation 10.1 gives a success threshold of 7.5. For a homogeneous swarm, using equation 10.2, the success threshold is 2.5.

Comparing the heterogeneous and homogeneous GAs is not as straightforward, because the different fitness functions mean that fitnesses are on a different scale to each other. For the both swarms a maximum of 35 "food" items can be collected[1]. In the homogeneous swarm each genome's fitness is the mean amount of food collected by the 10 robots in the swarm, so each genome has a maximum possible fitness of 3.5 and a maximum population fitness of 35 at each generation. The fitness function used for the heterogeneous swarm means the maximum fitness for any single genome is 8.5. Which would be a genome whose corresponding robot that collected from the largest food source in a swarm that collected all the food[2]. The total population fitness is double the amount of food collected; giving an upper limit of 70 for the total heterogeneous swarm fitness.

Consequently, the heterogeneous and homogeneous GA fitnesses are not directly comparable. However, the task the swarms are performing is the same, so we can measure comparative success using the amount of food collected over the course of the GA as a percentage of the amount it was possible to collect. Additionally, by using different success thresholds for each swarm type, the average number of generations until the GA finds a "success" gives values that can be compared with each other.

10.3 Developing Test Benchmarks

in this section we present the results of randomly generated GAs that will be used as benchmark to test the evolved GAs. In section 10.3.2 we describe some designed "ideal" controllers for each different coupling, and present the results when those controllers are tested on the robot swarm.

[1] Each food source can give its size squared in food each generation, $1^2 + 3^2 + 5^2 = 35$.

[2] The largest food source is 5, and the total amount of collectable food is 35: $5 + \frac{35}{10} = 8.5$.

Coupling	MBF of last generation	MBF over all generations
Heterogeneous, Single	1.60	7.60
Heterogeneous, Loose	0.69	4.93
Heterogeneous, Tight	0.69	4.71
Homogeneous, Single	1.38	3.50
Homogeneous, Loose	1.28	2.82
Homogeneous, Tight	0.93	2.66

Table 10.3: Quality metrics for the random heterogeneous and random homogeneous GAs.

10.3.1 Randomly Generated Controllers

To generate the random fitnesses, a population of 10 genomes is run for 500 generations for each coupling type, then repeated for 40 runs. Table 10.3 shows the MBF measurements for the heterogeneous and homogeneous robot swarms. It can be seen that in both swarms the single coupling results in much higher fitnesses than loose or tight. This may be because only one signal is ever present in the environment, so only the action gene relating to that tone affects the resulting controller behaviour. The remaining genes are redundant, and can take any value without affecting behaviour.

10.3.2 Designing A Controller

In this experiment, some features we would expect to find in a good robot controller are:

- A robot's reaction to any tone it has detected should be to phonotaxis towards it. This gives the maximum chance of collecting from a food source by the end of the generation.
- Robots should prioritise the largest food source because that gives the largest reward.
- High occupancy food sources should be prioritised over low occupancy ones because they have a greater chance of successful collection.

The designed controller's genome is tested under the same experimental conditions as the random genomes. Each coupling type has a slightly different genome because a different amount of tones are used in each coupling and the tones have different meanings. Consequently a genome designed for one coupling would not be the best genome for another coupling.

Single Tone Coupling

In this coupling, there is only 1 tone in environment for the robots to react to. The designed, single tone coupling controller has that tone's action set to phonotaxis, and is given the highest priority. The remainder of the genome is unused so the genes can take any value.

Tone	Priority	Action
66% < size	9	phonotaxis
66% < full	8	phonotaxis
33% < size	7	phonotaxis
33% < full	6	phonotaxis
0% < size	5	phonotaxis
0% < full	4	phonotaxis
unused	3	phonotaxis
unused	2	phonotaxis
unused	1	phonotaxis

(a) Loose coupling controller.

Tone	Priority	Action
66% < size 66% < full	9	phonotaxis
66% < size 33% < full	8	phonotaxis
66% < size 0% < full	7	phonotaxis
33% < size 66% < full	6	phonotaxis
33% < size 33% < full	5	phonotaxis
33% < size 0% < full	4	phonotaxis
0% < size 66% < full	3	phonotaxis
0% < size 33% < full	2	phonotaxis
0% < size 0% < full	1	phonotaxis

(b) Tight coupling controller.

Table 10.4: Controllers used for the designed loose and tight coupling benchmarks.

Coupling	MBF over last generation	MBF over all generations
Heterogeneous, Single	3.95	8.48
Heterogeneous, Loose	2.33	7.78
Heterogeneous, Tight	2.09	7.52
Homogeneous, Single	2.67	3.50
Homogeneous, Loose	2.44	3.30
Homogeneous, Tight	2.14	2.58

Table 10.5: Quality metrics for the designed controllers.

Loose Coupling

For loose coupling, all actions are set to phonotaxis. The largest size signal is given the highest priority, followed by the fullest occupancy signal, then the second largest size signal and second fullest occupancy, and so on until all tone priorities are set. This controller, for $D = 3$ signals is shown in table 10.4a.

Tight Coupling

All tones are assigned the phonotaxis action, and then prioritised primarily by size and secondarily by occupancy as shown in table 10.4b.

Benchmarking Results

Table 10.5 gives the quality metrics for the designed solutions. Compared to the random GA MBFs in table 10.3, the heterogeneous results and the loose coupling from the

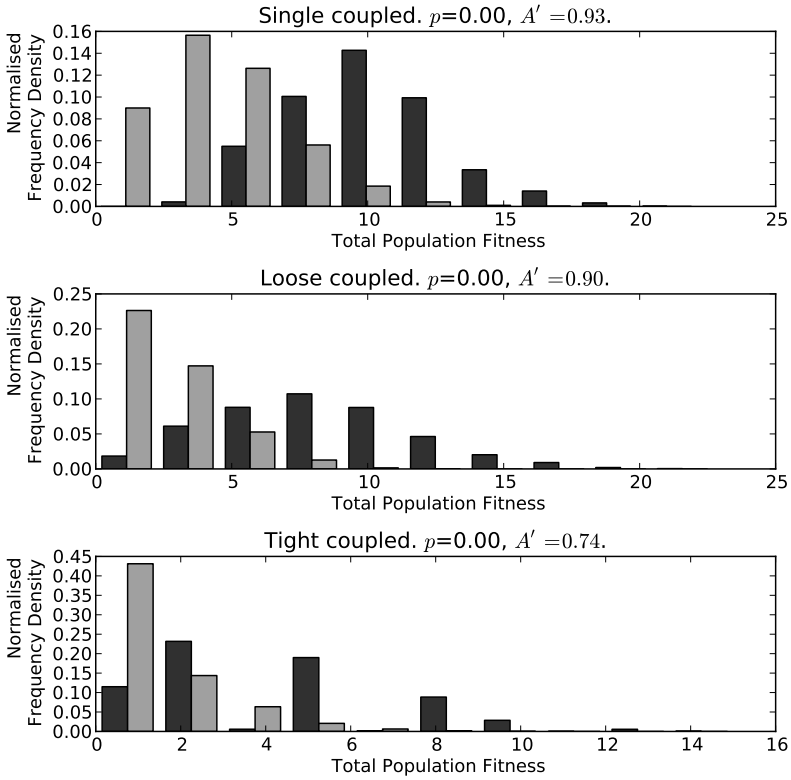


Figure 10.6: Histogram of the total population fitness at each generation, comparing the fitness of homogeneous designed and random controllers. Each distribution contains 4000 samples.

Coupling	p	A
Heterogeneous, Single	0.00	0.667
Heterogeneous, Loose	3.43×10^{-22}	0.526
Heterogeneous, Tight	1.86×10^{-53}	0.540
Homogeneous, Single	0.00	0.729
Homogeneous, Loose	0.00	0.621
Homogeneous, Tight	5.97×10^{-278}	0.604

Table 10.6: Comparison of evolved and randomised fitnesses for each swarm type and coupling. Results show the Wilcoxon rank-sum and A-test for the last 50 generations of each coupling type.

homogeneous swarm are all improved. Figure 10.6 shows a histogram of the total population fitnesses from the homogeneous designed and random GAs. The population fitness is improved in the designed solution with $p \approx 0$ and $A' > 0.72$, meaning the distributions are statistically different and the effect size is large for all couplings.

The combination of similar MBFs and significantly different population fitnesses from the designed and random experiments, indicates that the random GA was able to generate a small number of high fitnesses controllers in each experimental run, but its overall performance is worse than the designed GA.

10.4 Results

The initial results of the experiment are shown in figures 10.7 and 10.8. Figures 10.7 and 10.8 show the mean total *population* fitness of the evolved and random GAs, and the mean best *individual* fitness, as it changes over the course of the experiment. It can be seen that the coupling where the food sources used only one tone performed considerably better than the other two, and that for both kinds of swarm the loosely coupled communication performed better than the tight coupled.

Table 10.6 gives the Wilcoxon rank-sum and A-test scores for the evolved fitnesses compared to the random GA fitnesses from section 10.3.1. For all couplings and swarm types $p \approx 0$, meaning that evolution has an effect on the observed fitnesses, although only in single coupling is the effect size large enough to be significant ($A > 0.64$). Interestingly, the evolution has a greater effect size when applied to the homogeneous swarm compared to the heterogeneous swarm, indicating that the homogeneous swarm may be better at evolving collaborative foraging.

The results presented so far in this section contradict what is hypothesised in section 10.2.2. None of the results in figures 10.7 and 10.8 show improvement in population fitness after the first few generations, and the coupling which presents no information about the food sources consistently gives the best performance. However, these results do suggest that there is a relationship between the amount of information shared by the food sources, and the resulting fitness of the swarm. To investigate further, in section 10.5 we measure the effect of expressing size and occupancy information on the fitness of the swarm.

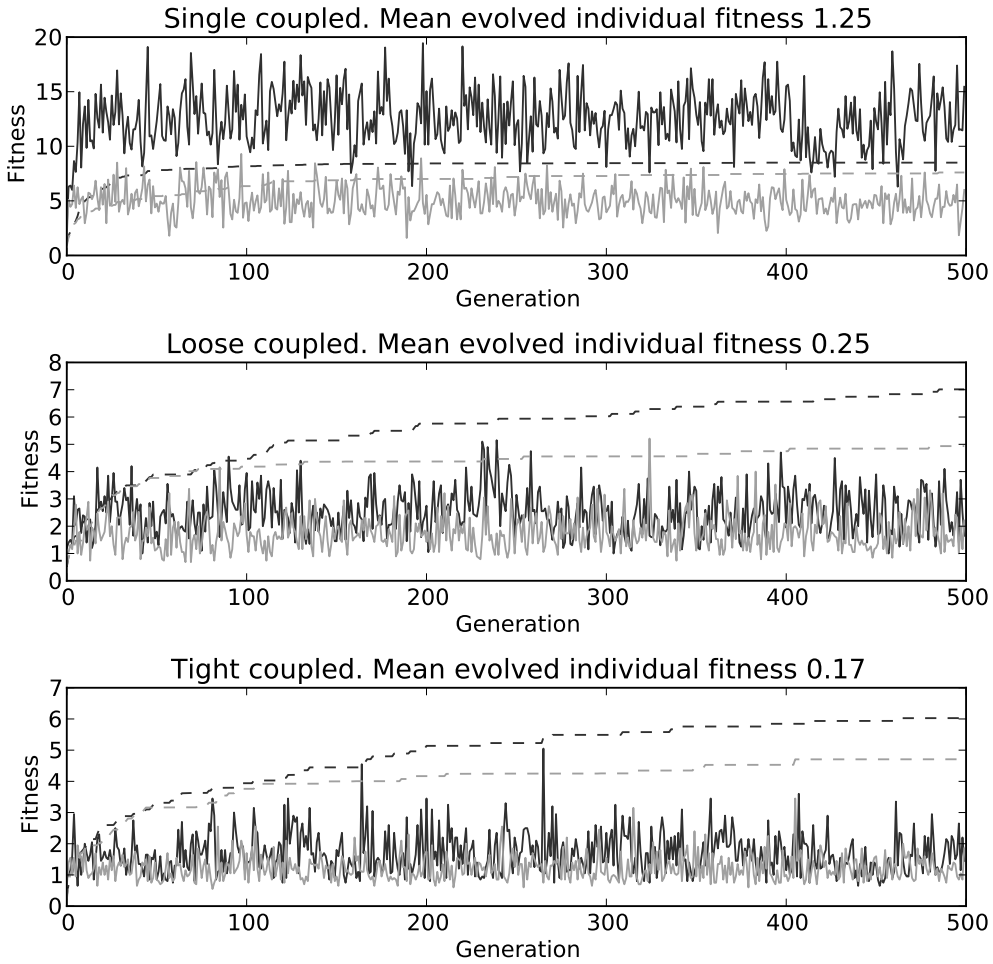


Figure 10.7: Mean total population fitness and MBF for the heterogeneous evolved and random GAs. Solid lines indicate the mean *total* population fitness. The dashed lines are the best *individual* fitness found up to the generation along the x-axis, averaged over all experimental runs. The dark lines are the evolved results and the light grey lines are the random results.

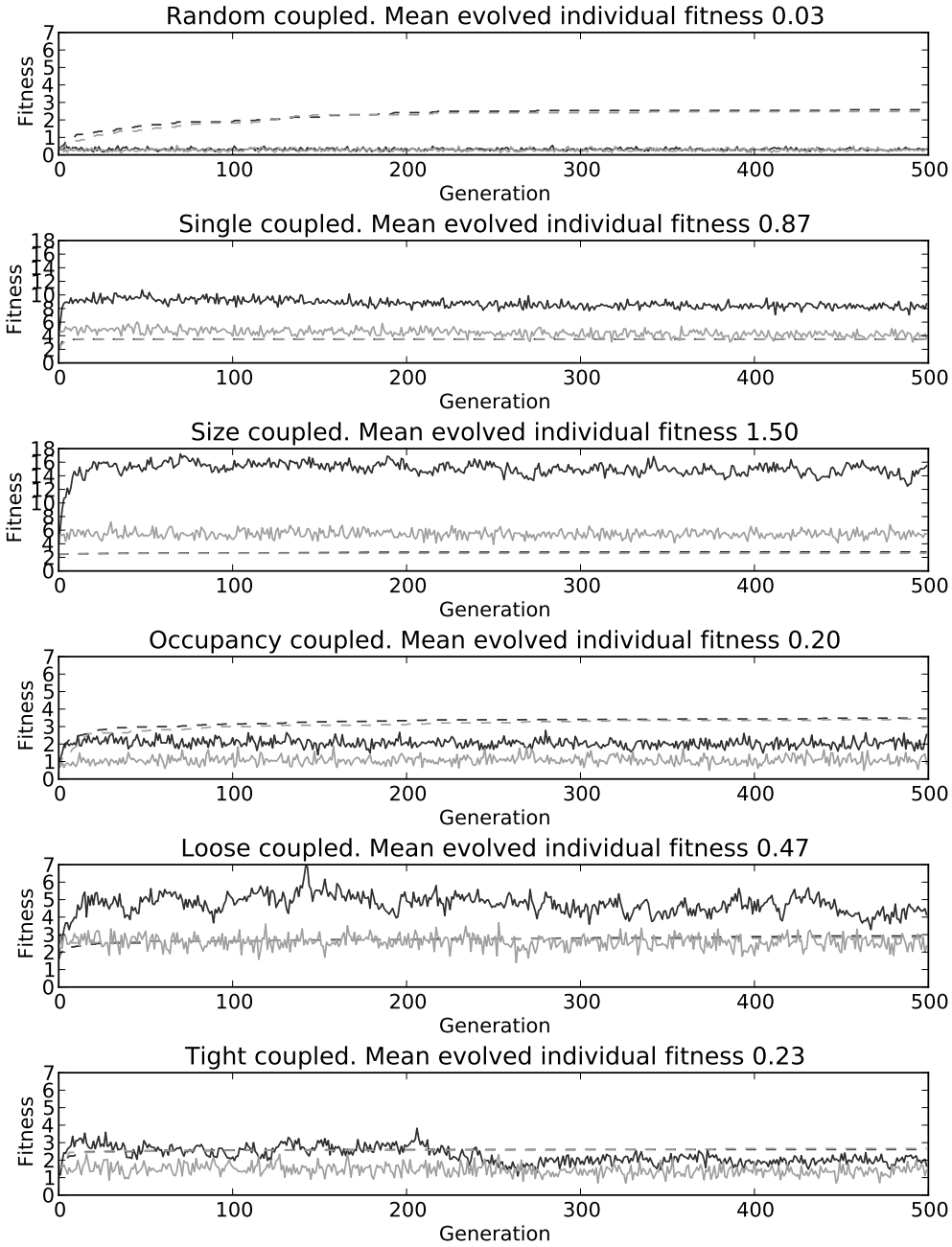


Figure 10.8: Mean total population fitness and MBF for the homogeneous evolved and random GAs. Solid lines indicate the mean *total* population fitness. The dashed lines are the best *individual* fitness found up to the generation along the x-axis, averaged over all experimental runs. The dark lines are the evolved results and the light grey lines are the random results.

Tone	Priority	Action
66% < size	9	phonotaxis
33% < size	8	phonotaxis
0% < size	7	phonotaxis
unused	6	phonotaxis
unused	5	phonotaxis
unused	4	phonotaxis
unused	3	phonotaxis
unused	2	phonotaxis
unused	1	phonotaxis

(a) Size only coupling, designed controller.

Tone	Priority	Action
66% < full	9	phonotaxis
33% < full	8	phonotaxis
0% < full	7	phonotaxis
unused	6	phonotaxis
unused	5	phonotaxis
unused	4	phonotaxis
unused	3	phonotaxis
unused	2	phonotaxis
unused	1	phonotaxis

(b) Occupancy only coupling, designed controller.

Table 10.7: Designed controllers for the size and occupancy couplings.

10.5 Testing Additional Couplings

To investigate how the information shared by the food sources affects the swarm fitness we test the following communication strategies:

- **Random, uncoupled**. Every 1.5 seconds the food source plays a random signal, informing collector robots of the food source's location, but lacking any other meaning. Compared to the single tone uncoupled communication from section 10.2.2 the same amount of information is being transmitted, but the robots need to interpret D^2 signals instead of just one. Effective behaviour is therefore harder to evolve.
- **Size**. Each food source only shares information about its size. There are D different size levels, and D corresponding signals the food sources can use. Each food source has a different size which does not change over the course of a generation. Consequently each of the D signal tones is constantly repeated whilst the robots are foraging.
- **Occupancy**. Each food source only shares information about its occupancy. There are D occupancy levels and D corresponding signals the food sources can use. The food sources' occupancy is 0 at the start of each generation and changes as robots discover them. Meaning the signals are not constant throughout a generation, and several food sources could be transmitting the same signal at the same time.

10.5.1 Benchmarks

The designs for the size and occupancy coupling controllers are given in tables 10.7a and 10.7b. In the random coupling, the tones do not convey any specific meaning, and they are equally as likely to be played by the food sources. The best design for the random coupling would therefore set every action as phonotaxis so that the robot will move to any tone it detects. The priorities are arbitrarily assigned, giving priorities ascending with tone frequency, but kept consistent throughout the benchmark testing.

Testing the new couplings under the same conditions as those from section 10.3, gives the random GA metrics in table 10.8 and the designed controller metrics from

Coupling	MBF of last generation	MBF over all generations
Heterogeneous, Random	0.20	2.93
Heterogeneous, Single	1.60	7.60
Heterogeneous, Size	1.20	7.54
Heterogeneous, Occupancy	0.54	4.70
Heterogeneous, Loose	0.69	4.93
Heterogeneous, Tight	0.69	4.71
Homogeneous, Random	0.22	2.48
Homogeneous, Single	1.38	3.50
Homogeneous, Size	2.06	2.60
Homogeneous, Occupancy	0.52	3.43
Homogeneous, Loose	1.28	2.82
Homogeneous, Tight	0.93	2.66

Table 10.8: Quality metrics from the random heterogeneous and homogeneous controllers. Results from table 10.3 are included for reference.

Coupling	MBF of last generation	MBF over all generations
Heterogeneous, Random	0.40	4.03
Heterogeneous, Single	3.95	8.48
Heterogeneous, Size	4.50	7.62
Heterogeneous, Occupancy	1.25	6.32
Heterogeneous, Loose	2.33	7.78
Heterogeneous, Tight	2.09	7.52
Homogeneous, Random	0.38	2.14
Homogeneous, Single	2.67	3.50
Homogeneous, Size	2.50	2.96
Homogeneous, Occupancy	0.86	2.84
Homogeneous, Loose	2.44	3.30
Homogeneous, Tight	2.14	2.58

Table 10.9: Quality metrics from the designed heterogeneous and homogeneous controllers. Results from table 10.5 are included for reference.

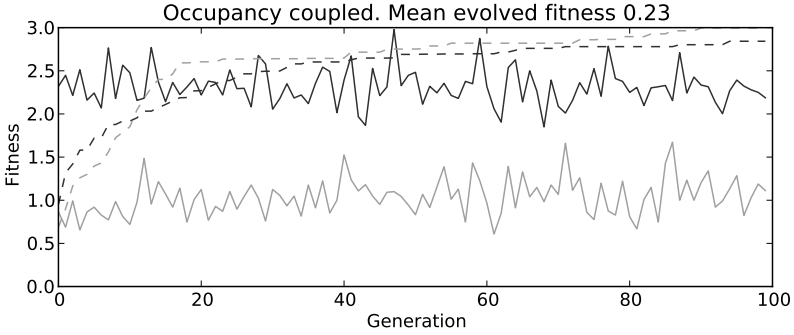


Figure 10.9: The mean fitness and MBF for the first 100 generations of the homogeneous occupancy coupling designed and random test. Solid line indicates the mean population fitness, the dashed line is the mean best fitness. The dark lines are the evolved results and the light grey lines are the random results.

table 10.9. As expected, the designed controller performs better than random in most cases. The only exception is the homogeneous occupancy coupling where the MBF over all generations is 3.43 in the random GA but 2.84 for the designed controller test. Figure 10.9 shows the total population fitness and MBF for the homogeneous designed and random tests as it changes over the first 100 generations. The designed occupancy controller gives a higher mean fitness, but the random GA produces higher peak fitnesses, suggesting that the designed controller is better than most possible controllers, but is not optimal.

10.5.2 Results

Figures 10.10 and 10.11 show the random and evolved mean population fitness and MBF for each coupling and swarm type, averaged over 40 runs of each experiment. In all couplings there was no improvement in the population fitnesses after the first 25 generations. Table 10.10 gives the Wilcoxon rank-sum and A-test comparison of the evolved GA results against the random GA. For both swarms and all couplings, $p \approx 0$, so evolution does affect the fitness with a 95% confidence level. For the heterogeneous swarm, only the single coupling had a large enough effect size that the difference is significant. The effect size of the evolution on the homogeneous swarm is larger than the heterogeneous. The single, size and occupancy couplings all give $A > 0.64$, indicating a medium or large effect size, and a significant improvement over the random GA fitnesses.

The evolved fitnesses are compared to designed controller fitnesses in table 10.11. The Wilcoxon rank-sum shows that for all couplings except the homogeneous, loose coupling the designed and evolved controllers give statistically different fitnesses. The effect size varies, but only the tight coupled tests performed better than the designed controller, and the effect size is almost large enough to be a significant difference. This means that the designed controller for the tight couplings is not optimal and the GA is able to evolve controllers which are able to collect more food.

Table 10.12 gives the speed and quality metrics for the evolved GAs. For both swarm types, loose coupling gives higher fitnesses than tight coupling, but there is little difference in the speed to success. Doing a Wilcoxon rank-sum and A-test on the distribution of first generation to find a successful genome from all 40 runs, gives

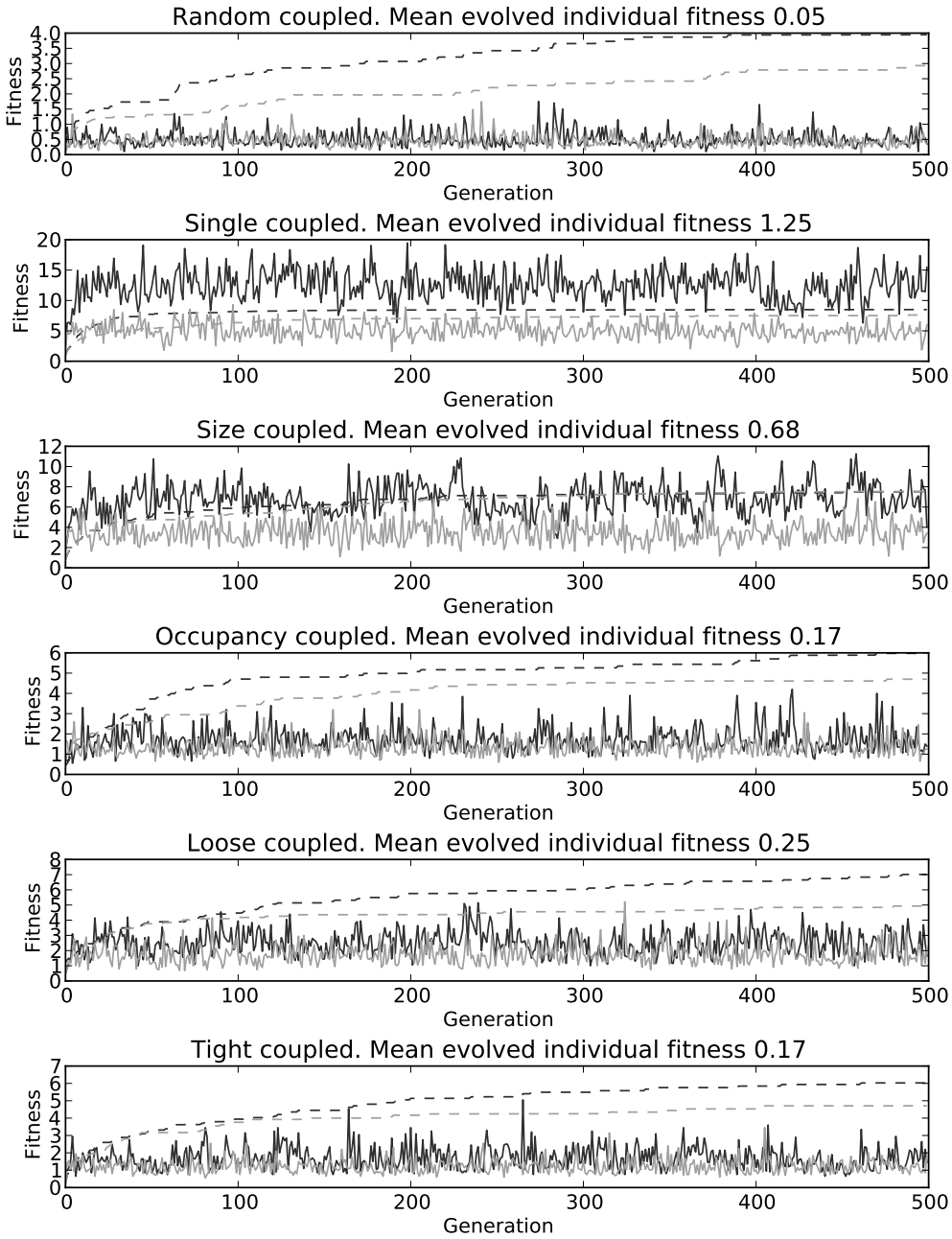


Figure 10.10: Mean total population fitness and MBF for the heterogeneous evolved and random GAs. Solid lines indicate the mean *total* population fitness. The dashed lines are the best *individual* fitness found up to the generation along the x-axis, averaged over all experimental runs. The dark lines are the evolved results and the light grey lines are the random results. Results from figure 10.7 have been included for comparison.

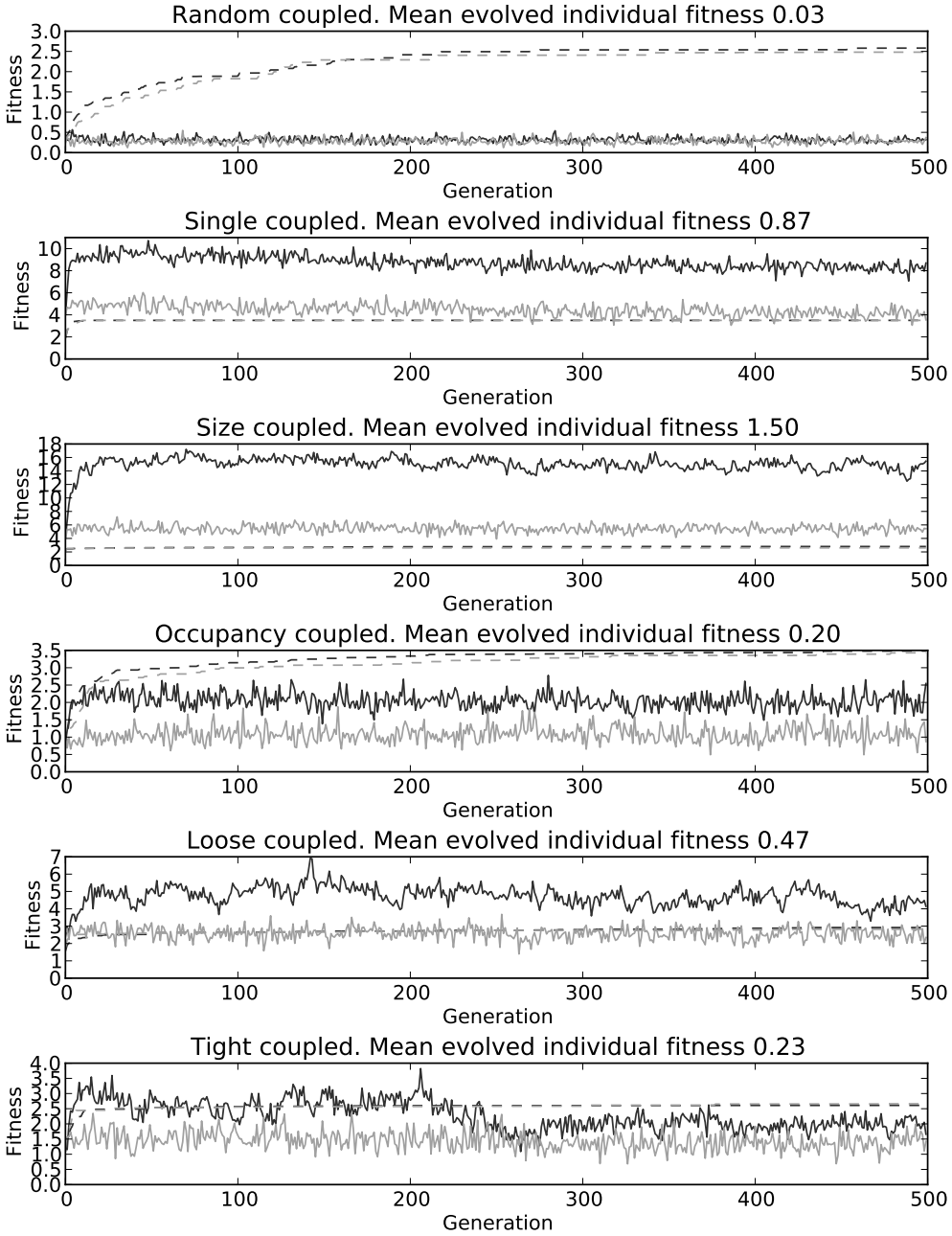


Figure 10.11: Mean total population fitness and MBF for the homogeneous evolved and random GAs. Solid lines indicate the mean *total* population fitness. The dashed lines are the best *individual* fitness found up to the generation along the x-axis, averaged over all experimental runs. The dark lines are the evolved results and the light grey lines are the random results. Results from figure 10.8 have been included for comparison.

Coupling	p	A
Heterogeneous, Random	3.19×10^{-6}	0.509
Heterogeneous, Single	0.00	0.667
Heterogeneous, Size	1.49×10^{-264}	0.594
Heterogeneous, Occupancy	0.125	0.503
Heterogeneous, Loose	3.43×10^{-22}	0.526
Heterogeneous, Tight	1.86×10^{-53}	0.540
Homogeneous, Random	0.004	0.505
Homogeneous, Single	0.00	0.729
Homogeneous, Size	0.00	0.712
Homogeneous, Occupancy	0.00	0.669
Homogeneous, Loose	0.00	0.621
Homogeneous, Tight	5.97×10^{-278}	0.604

Table 10.10: Comparison of evolved and randomised fitnesses for each swarm type and coupling. Results show the Wilcoxon rank-sum and A-test for the last 50 generations of each coupling type. Results from table 10.6 have been included for comparison.

Coupling	p	A	A'
Heterogeneous, Random	3.02×10^{-76}	0.460	0.540
Heterogeneous, Single	0.00	0.371	0.629
Heterogeneous, Size	0.00	0.341	0.659
Heterogeneous, Occupancy	0.00	0.299	0.701
Heterogeneous, Loose	7.81×10^{-13}	0.481	0.519
Heterogeneous, Tight	0.00	0.633	0.633
Homogeneous, Random	4.19×10^{-158}	0.443	0.557
Homogeneous, Single	2.56×10^{-150}	0.430	0.570
Homogeneous, Size	5.99×10^{-11}	0.484	0.516
Homogeneous, Occupancy	0.00	0.385	0.615
Homogeneous, Loose	0.448	0.500	0.500
Homogeneous, Tight	0.00	0.611	0.611

Table 10.11: Comparison of evolved and designed controller fitnesses for each swarm type and coupling. Results show the Wilcoxon rank-sum and A-test for the last 50 generations of each coupling type.

Coupling	Mean Generations to Success (Number Successes)	MBF of last generation	MBF over all generations
Heterogeneous, Random	274.00 (1)	0.19	3.95
Heterogeneous, Single	20.57 (40)	3.10	8.50
Heterogeneous, Size	117.51 (39)	1.30	7.52
Heterogeneous, Occupancy	216.86 (22)	0.60	5.98
Heterogeneous, Loose	213.82 (34)	0.76	7.01
Heterogeneous, Tight	224.26 (23)	0.65	6.03
Homogeneous, Random	93.47 (40)	0.19	2.58
Homogeneous, Single	0.80 (40)	2.45	3.50
Homogeneous, Size	0.20 (40)	2.46	2.80
Homogeneous, Occupancy	5.88 (40)	1.10	3.48
Homogeneous, Loose	5.05 (40)	1.39	2.92
Homogeneous, Tight	4.42 (40)	0.74	2.61

Table 10.12: Speed and quality metrics from the evolved heterogeneous and homogeneous GAs.

$p = 0.271$ $A' = 0.520$ for the heterogeneous swarm and $p = 0.092$ $A' = 0.596$ for the homogeneous swarm. Since $p > 0.05$ and $A' < 0.64$ in both cases, there are no significant differences in GA speed between the loose and tight couplings.

Similarly, a Wilcoxon rank-sum and A-test on the distribution of fitnesses from the last 50 generations of the loose and tight couplings gives $p = 1.01 \times 10^{-52}$ $A' = 0.540$ for the heterogeneous swarm and $p = 0.000$ $A' = 0.618$ for homogeneous. In both cases $p < 0.05$, so the distributions are different, but $A < 0.64$ so although loose coupling giving slightly higher fitnesses, the effect size of these differences is not large enough to be significant.

Comparing the heterogeneous and homogeneous swarm types, the speed metrics from table 10.12 show the homogeneous swarm is much faster at finding successful solutions. In section 10.2.7 we define a successful genome as one which caused its corresponding robot or robot swarm to collect from the largest food source. All 40 runs of each coupling type from the homogeneous swarm were able to find at least one successful genome, unlike the heterogeneous experiments, indicating that there was a much larger tendency towards cooperative behaviour in the homogeneous swarm. This observation is supported by analysis of what food sources were collected, in table 10.13. The homogeneous swarm collects a greater percentage of the larger food sources than the heterogeneous swarm does, so the total food collected, as a percentage of available food, is larger for the homogeneous experiments. This is especially noticeable for the homogeneous size coupling, which prioritises the largest food source and so collected a greater percentage of available food than any other coupling.

10.6 Discussion

In the previous chapter we speculate that using homogeneous swarms would improve collaboration between robots, because a robot attempting to cooperate with others to collect from a larger food source is less likely to have the cooperation attempt reciprocated in a heterogeneous swarm. Whereas in the homogeneous swarm, other robots are

Coupling	Food Size 1 (%)	Food Size 3 (%)	Food Size 5 (%)	Total Food Collected (%)
Heterogeneous, Random	20.34	0.56	0.01	0.73
Heterogeneous, Single	90.22	39.84	7.06	17.86
Heterogeneous, Size	69.55	24.32	2.03	9.69
Heterogeneous, Occupancy	51.85	3.25	0.20	2.46
Heterogeneous, Loose	58.38	6.46	0.37	3.59
Heterogeneous, Tight	49.39	3.42	0.17	2.41
Homogeneous, Random	15.97	1.43	0.11	0.90
Homogeneous, Single	89.36	52.21	12.54	24.94
Homogeneous, Size	10.55	12.19	55.21	42.87
Homogeneous, Occupancy	56.18	11.62	1.75	5.85
Homogeneous, Loose	30.63	31.07	6.54	13.54
Homogeneous, Tight	29.00	16.43	2.07	6.53

Table 10.13: Percentage of all runs in which the different food sources was collected from, and the percentage of available food that was collected.

just as inclined to collect from the same sized source, and all are consequently rewarded with more food. The results observed in this chapter do not contradict this theory since the collection percentage for the medium and large food sources are higher in the homogeneous swarm than the heterogeneous swarm (table 10.13).

Our results also suggest that size information is more useful than occupancy because, with the exception of the random coupling, occupancy coupling had the lowest mean fitness. Whereas the single and size couplings, where occupancy information is not transmitted, had the highest mean fitness. One possible cause of occupancy signals not being as useful as size signals is that the occupancy levels of the food sources are affected by the success of the robot population, causing a possible feedback loop as the population becomes better at locating food sources. As the population gets better at reacting to "empty" signals they become better at finding the empty food sources. The occupancy levels of the food sources increase causing the occupancy signal to change, so the occupancy signal the population was using to find food is not as prevalent, and the swarm is no longer as able to find the food. This could make it harder for swarms to improve over time.

In chapter 4 we hypothesise that loose coupling would show a faster initial improvement in swarm fitnesses compared to tight coupling, but it might not necessarily find genomes with better fitnesses. However, results from this chapter show that for both swarm types there is no significant difference in speed for the loose or tight coupling. The analysis so far has focused on the evolved results, but the results from the random benchmarking tests support the hypothesis. A Wilcoxon rank-sum and A-test comparison of the random GA results from the loose and tight coupled homogeneous swarms gives fitness results of $p = 2.13 \times 10^{-194}$, $A' = 0.591$. Meaning the random loose coupled test gave better fitnesses but the effect is only small. However, comparing the number of generations before a successful solution is found, gives $p = 0.029$, $A' = 0.688$, so not only is loose coupling faster than tight, but the effect size is large enough to be significant. If loose coupling is capable of making a random search GA able to find better solutions faster than tight coupling, this indicates that changing the coupling also changes the nature of the foraging task to make it easier to find food. We theorise

this is because loose coupling makes it easier to disregard the less useful occupancy information shared by the food sources. With tight coupling, the size and occupancy information is linked together into one signal so it is harder for the robots receiving the signal to separate the pieces of information.

The loose coupling did not perform as well as the coupling with just the size information. This indicates that there is a cost associated with interpreting the extra signals in the environment, and that there may be a cost to filtering out the less useful occupancy information. With tight coupling, these costs are high because there are more signals and it is hard to separate signal and meaning. These costs are not balanced by the usefulness of the occupancy information, since the tight coupling was only able to collect a small percentage of the available food. With the loose coupling, the costs are lower because there are fewer signals and they have distinct meanings, but these costs are still not balanced by any benefit from the information, since the loose coupled fitnesses are lower than if only the size information had been shared. The lower costs of interpreting signals come at the expense of information transmission speed. The loosely coupled communication takes twice as long to deliver the same information as a single signal in a tightly coupled communication strategy. Consequently the robot might be reacting to information that is no longer valid, and taking longer to react to a change in a food source's state.

10.7 Conclusion

In this chapter our aim is to test hypothesis of chapter 4 that loose coupled communication helps swarm behaviour to evolve sooner than it would using tight or no coupling. In section 10.4 we tested the hypothesis using heterogeneous and homogeneous swarms of robots and found that, in the context of our experiment, a homogeneous swarm tended to evolve fitter solutions with a higher prevalence of collaboration. Furthermore, the homogeneous swarm is able to evolve "successful", collaborative controllers using fewer fitness evaluations than the heterogeneous swarm. This matches the findings of Floreano *et. al.* [29] cited in section 5.4.1, who were able to evolve signalling and foraging behaviour in a homogeneous swarm that outperformed the behaviour evolved by an equivalent, heterogeneous swarm.

The results of the hypothesis testing show that loose coupling performed slightly better than tight coupling both in terms of speed and quality (table 10.12). The single tone coupling, where the robots are given the smallest amount of information needed to complete the task, the location of food sources, outperforms both the loose and tight coupling strategies. Upon further investigation of the relationship between the information transmitted and the swarm fitness, it is found that combining size and occupancy information into a loose or tight coupling leads to lower fitnesses than if just the size information is shared. With only the size information shared by the food sources the robots in the homogeneous swarm are able to evolve controllers that favour larger food sources, and consequently they collect a higher percentage of the available food compared to the same swarm with the single tone coupling.

The most important finding of this chapter is that *the cost of interpreting information can be greater than the benefit gained from the information.* Loose coupling gave lower fitnesses than size only coupling, because it has to interpret more signals and ignore the less useful occupancy information. Tight coupling has lower fitness than loose coupling and collected less food overall because it has more signals to interpret and it

is much harder to separate the size and occupancy information. Loose coupling can reduce cost of interpreting multiple pieces of information in return for slower message transmission, but the evolved solutions are not as fit as they would be if the unnecessary information is not shared. Knowing what information is and is not relevant *before* evolving the robots is a problem we have not addressed.

10.7.1 Further Work

In our work so far we have not investigated whether the idea of loose coupling can be generalised to other swarm robotic systems. Could loose coupling be incorporated into a swarm's communication system, and if so is it beneficial? It would be worthwhile to compare different couplings when the robot swarm is doing a different task that requires robot to robot communication in order to be completed or effective. For example, shape forming, where robots in a swarm must communicate in order to organise themselves into a shape, or box pushing, where robots must cooperate to push a box across an arena.

Further investigation could be done into the usefulness of information being transmitted and the resulting effect on the swarm fitness. How does transmitting information which is known to be meaningless affect how the robots interpret the signals when using different couplings? Instead of measuring a food source's actual occupancy we could try selecting a random occupancy level, to see whether the results are similar to those observed in this chapter. Or alternatively communicating another piece of information, which is known to be meaningless, in addition to the existing size and occupancy information.

Chapter 11

Conclusion

11.1 Introduction

The aim of this thesis is to investigate whether it is beneficial to evolve a robotic swarm in a loosely coupled way, in order to exploit the speed of evolution in complex systems. We define coupling as the amount of influence that each robot has on another; uncoupled robots are completely independent of each other and do not collaborate, with tight coupling the robots are very co-dependent, and have a strong influence on each other. Loosely coupled robots have some influence on each other but are comparatively independent.

The hypothesis is tested using swarms of foraging robots, with food sources which dispense food only when there are enough collecting robots present. The more robots a food source needs in order to dispense food, the more food it will give to each collecting robot. Robots can share information about a food source's size (the number of robots needed for it to dispense food), and the source's occupancy, a measure of how many more robots are needed before food can be dispensed.

To test the effects of coupling on the fitness of a robot swarm, we compare cases where the swarm is uncoupled, loosely coupled and tightly coupled. The couplings are varied by manipulating the communication between robots. The robots use sound signals to express information, and coupling is varied by manipulating the mapping between sound signal frequency and the meaning the robot wants to express. Loosely coupled robots have one signal for each size and occupancy amount, and any message plays these signals sequentially to describe the food source. With tight coupling, there is one signal for each *combination* of size and occupancy amounts. Finally, uncoupled robots must each evolve their mapping between signal and meaning.

In chapter 6 we build a model of the experiment following the CoSMoS process, so that we can test the hypothesis in simulation, but have results that can be generalised to embodied robots. The evolutionary swarm robotic (ESR) system that we model does not exist in reality, because the e-pucks cannot detect sound or sound directionality without additional sensors. Consequently in chapter 6 we build a model that cannot be validated against a real system, but is flexible enough to be easily calibrated later once the real system is built. Following the CoSMoS process we detail all the aspects of the system that *are* known, and record all the assumptions made about the system during the modelling process. Areas of the real system that are unknown are recorded as calibration points to be addressed in later stages of the modelling. In chapter 7 we build

the soundboard add-on to the e-pucks so that they can detect the frequency, amplitude and direction of signalling audio tones. The model is then calibrated to reproduce the behaviour of the soundboard, by addressing each of the calibration points from chapter 6. The behaviour of the soundboard is tested and found to be erratic and highly sensitive to environmental conditions, so both realistic and idealised models of the soundboard are built for the simulation.

In chapter 9 we use the completed model to test couplings when the robots must communicate with each other to locate food sources. For loose and tight couplings, each robot is given the same signal to meaning mapping, which does not change over time. For uncoupled swarms, each robot must evolve its own signal to meaning mapping. The results show that in all tested couplings the robots do not evolve to use communication and do not collaborate to collect from large food sources, usually preferring to stay stationary for the whole generation.

Building on these results, we try a similar experiment where it is the food sources that constantly express information about themselves. Loose and tight couplings are implemented as in the earlier experiment, for the uncoupled swarm we use random signalling and signalling with only one tone. We also test swarms where the food sources just signal their size or their occupancy, but not both. Our results show that the best fitnesses are found when only the size information is shared or when the food sources all used the same tone to signal to the collectors. These couplings are also the fastest to find "success" solutions. Loose coupling performed slightly better than tight coupling for both finding higher fitness solutions and finding successful solutions sooner. Although with a randomised solutions rather than evolved ones, the loose coupled tests became significantly faster than tight coupled tests.

From these results we conclude that the occupancy information is not as useful for the completion of the task as the size information. Furthermore, *the evolutionary cost of interpreting the occupancy information is greater than any benefit that was gained from having that information.* It is difficult to know what information will be useful before beginning the evolution, and results indicate that being given extra information, in case some of it is useful, can be a detrimental solution. The use of loose coupling can negate the effects of sharing the extra information, but there is still an evolutionary cost associated with interpreting the extra signals. Tightly coupled swarms perform badly because of the strong effect of the less useful occupancy information on the signal being broadcast. By coupling the pieces of information being broadcast, the experiment demonstrates an interesting relationship between how the information is shared and the swarm fitnesses.

With regards to the hypothesis, the results show that loose coupling leads to better fitnesses than tight coupling. However, this is not necessarily achieved by having more rapidly improving fitnesses as hypothesised, but because loose coupling makes the task of interpreting signals easier. The best fitnesses are observed when there is only size information being shared, or when only one signal tone is ever used. The size-only coupling has a shared signal to meaning mapping, so there is some degree of coupling in the environment, and because no extraneous information is shared the task of interpreting messages is easier. The single tone coupling also has a fixed signal to meaning mapping, in that a tone indicates the location of a food source. However, the robots only need to evolve their reaction to one tone and the rest of the genome is redundant, making the task of evolution much easier. Additionally, because the robots cannot distinguish between food sources with this coupling, they can only learn to distribute themselves evenly between the food sources, and this is a good tactic for completing the given task.

11.2 Impact of this Research

The research in this work can be most readily applied to swarms of robots doing tasks which require explicit robot to robot communication.

The alphabetisation of information into simple, atomic descriptive primitives is beneficial because it discretises information, making it easier to transmit. By demonstrating that loose coupling gives better swarm fitnesses than tight coupling, it suggests that the sequential transmission of several alphabetised descriptive primitives can be easier to interpret than a large repertoire of many expressive signals from which only one signal is needed at once to express all the information.

These findings are relevant not just to ESR but any form of swarm robotics where robots directly communicate with each other, because it affects the design of the communication. Sequential transmission of alphabetised data makes it easier for receiving robots to extract relevant information from the message, but there may be situations where short messages are preferable. It is an extra consideration when designing a swarm robot experiment or when applying ESR to complete a task; asking what is the best way of implementing the communication between robots for this application? Inappropriately designed communication leads to robots not being properly able to share information, and so not performing the task as well as they could.

Our research has also shown evidence that sharing more information between robots does not make the task easier for them to complete. Future researchers should be aware of this when designing experiments or swarm robot communication, taking care to make sure any information being shared is useful. This has limitations if the task or environment are likely to change over time because the data shared may become irrelevant. In those circumstances, sharing as much information as possible in a loosely coupled way may be a viable approach to designing the communication. An alternative solution is to have the swarm robots evolve both the meanings and the signal to meaning mapping, so that new meanings can be expressed as they become more useful [86]. However, in chapter 9 we show that when robots are able to evolve their signal to meaning mapping, even with a fixed set of expressible meanings, reliable communication and collaborative behaviour is difficult to evolve. It would be worthwhile to run short experiments to see how useful any shared data is, before running the evolutionary algorithm. Although, in embodied robots this could take a long time, so it may also be worth building a simulation of the robot swarm to help determine the best conditions.

The CoSMoS process has been applied to modelling for the purpose of learning more about the system being modelled (the domain), for example [32, 75]. However, this is the first application of the CoSMoS process for modelling engineered complex systems. Although we have not been able to validate our results against real robots, we do recommend following the CoSMoS process. Noting the modelling assumptions is a useful process, because it shows others where the model is assumed to be "good enough" and any justifications for setting it that way. Assumptions that are later found to be erroneous are much easier to address, especially if the original modeller is not available. Another useful exercise has been to note what aspects of the modelled system are unknown, so that they can be calibrated later when the domain is created. By co-developing the model and domain, the features and limitations of the domain can be fed back into the model, and the constraints of the modelling can inform requirements of the domain.

11.2.1 Original Contributions

Major Contributions

- Applying loose coupling to evolutionary swarm robotic systems, and finding it leads to higher fitnesses of robot controllers than other couplings by making the communicated signals easier to interpret.
- Demonstrating that making more data available to the robots does not always result in better performance.
- Providing a case study for the application of the CoSMoS process to engineered systems.

Minor Contributions

- Building a model of e-pucks communicating with audio signals.
- Development of the e-puck soundboard.
- Testing of phased array beamforming under size and processing limitations of e-pucks.
  - It is shown that these limitations are too great for beamforming to be effective.
- Comparing the performance of homogeneous and heterogeneous swarms

11.3 Further Work

In section 10.7.1 we suggest further experiments that lead directly on from our findings in chapter 10. In this section we suggest further work which builds on our research.

One of our aims is to provide a case study of applying the CoSMoS process to engineered complex systems. Although we have developed a model and used it to perform experiments, we have not yet validated the model against similar experiments using real e-pucks. In chapter 10 we tested our hypothesis on simulated e-pucks with idealised audio measuring and transmission. This was to test whether the hypothesis would hold in an ideal environment, because if the hypothesis could be rejected there is little point in time-consuming experimentation on real robots. Chapter 10 shows that our hypothesis is able to hold in ideal conditions, so the same experiment needs to be repeated using the realistic soundboard model, developed in chapter 8, and with real e-pucks. After this has been done, the model can be validated against reality. The simulated and embodied experiments can be compared to see whether fitnesses under different couplings show the same trends in both, and whether the evolved controllers are similar. If the two tests are found to be different, the assumptions of the model can be addressed to try and make the model a closer representation of reality.

In this work we have not investigated circumstances where tightly coupled communication might be better than loose. The sequential transmission of alphabetised data works well in chapter 10. However, if the robots must respond to signals quickly, or if the communication is noisy or unreliable it may be preferable to send tightly coupled information as one short signal that can be repeated frequently. With unreliable transmission of loosely coupled information there is a higher chance of one alphabetised

descriptive primitive getting lost in transmission, meaning the whole message must be repeated before that information is re-sent. If the message being transmitted contains irrelevant information then there is a risk of the useful information being lost, impacting a robot's ability to act appropriately. Unreliable data transmission could be tested by repeating the experiment from chapter 10 using the realistic soundboard model. If the audio signal transmissions have a maximum range, it is possible for robots to move in and out of range, meaning signals can be lost. Consequently, we may find that tightly coupled communication gives higher fitnesses than loose coupled swarms.

Swarm robotics is inspired by swarm intelligence in insects, but the evolution used in the genetic algorithms of our work is not the same as the evolution of swarm insects. Traditionally, genetic algorithms have a population of individuals where the individuals with the highest fitnesses survive to produce the next generation. It is possible for individuals to cooperate, but their ability to reproduce is judged only on individual, rather than collective, fitness. With swarm insects, most members of a colony are sterile. These sterile insects work together to collect food and protect the colony from danger. The only ants which reproduce and carry on their genetic material to the next generation are queens and male ants. These ants reproduce with other queens or males from other colonies. if a colony is good at collecting food and avoiding danger, it can devote more energy to producing more reproductive ants, thereby giving it more chances to reproduce [37]. Consequently, it is *colonies* that evolve, not individual insects. Evolutionary selection is applied to the ability of the reproductive ants to successfully mate before dying, but also to the performance of the colony.

It would be interesting future research to try applying swarm evolution to ESR rather than the more traditional "survival of the fittest" evolution. This could be implemented using a genetic algorithm with two genome populations, one of queens and one of males. The robot controller genomes are then each derived from the crossover and mutation of a queen and a male genome pair, and the robot swarm performs its task. Some number of genomes are created from the two parents and placed into the next generation's reproductive population, this number is proportional to the performance of the robot swarm. Once enough genomes are created in this way, then next generation is evaluated.

By evolving controllers in this way, the *group* performance is used as a measure of evolutionary fitness. The *swarm* of robots is evolved rather than individual robot controllers, much like when homogeneous robot swarms were used in chapter 10. However, all robot controller genomes in a swarm are derived from the same "parents" and then mutated, so individuals in the swarm are heterogeneous but genetically similar. This has the advantage of encouraging collaborative behaviour, but maintaining diversity so that the swarm is more adaptable and able to cope with environmental changes.

Appendix A

Supplementary Soundboard Information

A.1 Microphone Pre-Amplifier Schematic

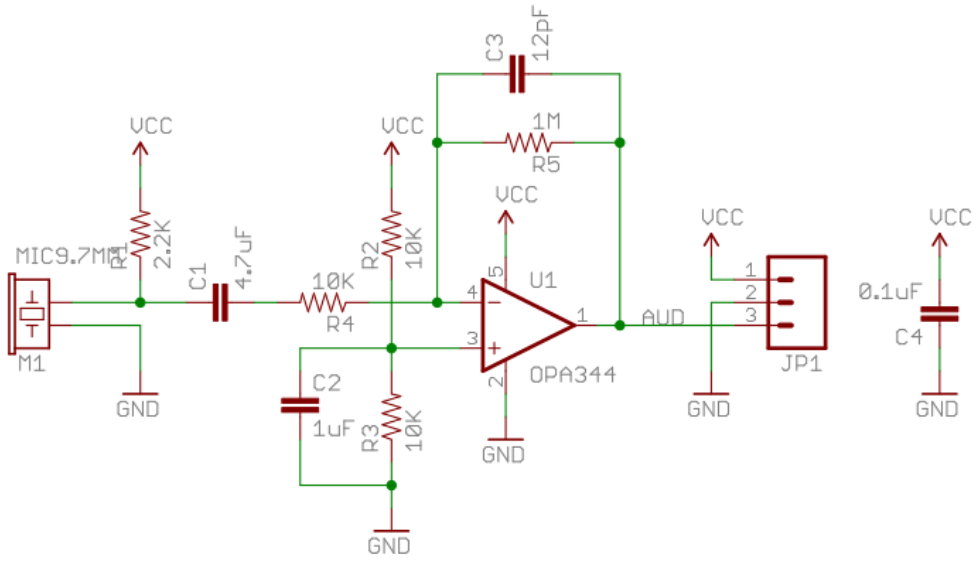
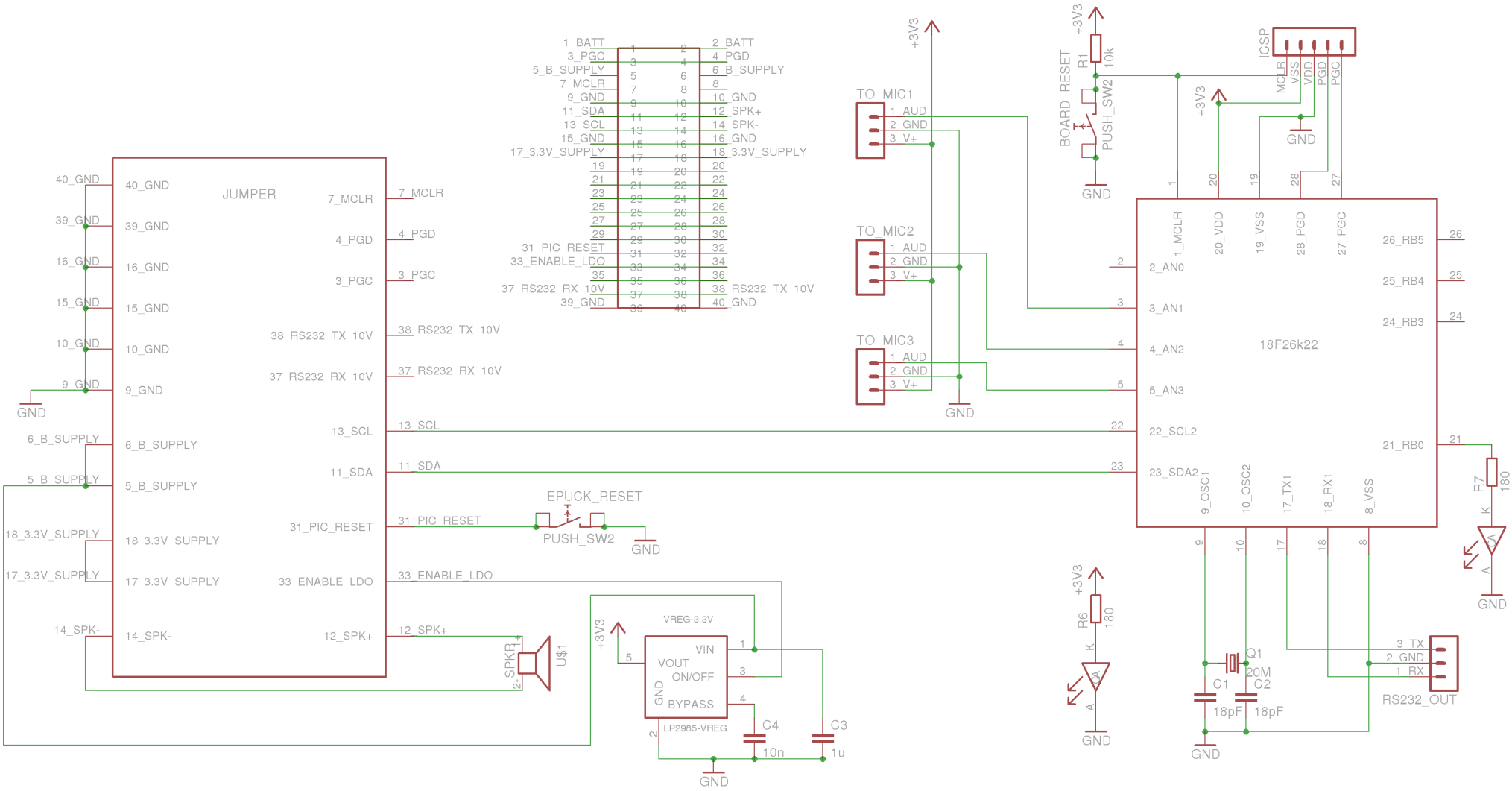


Figure A.1: Circuit diagram of microphone pre-amplifier used in the soundboard. Image reproduced from [28].

A.2 Soundboard Circuit Diagram


JUMPER
40_GND
39_GND
16_GND
15_GND
10_GND
9_GND
6_B_SUPPLY
5_B_SUPPLY
18_3.3V_SUPPLY
17_3.3V_SUPPLY
14_SPK-
7_MCLR
4_PGC
3_PGC
38_RS232_TX_10V
37_RS232_RX_10V
13_SCL
11_SDA
31_PIC_RESET
33_ENABLE_LDO
12_SPK+
GND
1_BATT
2_BATT
3_PGC
4_PGC
5_B_SUPPLY
6_B_SUPPLY
7_MCLR
9_GND
10_GND
11_SDA
12_SPK+
13_SCL
15_GND
16_GND
17_3.3V_SUPPLY
18_3.3V_SUPPLY
31_PIC_RESET
33_ENABLE_LDO
37_RS232_RX_10V
38_RS232_TX_10V
39_GND
40_GND
TO_MIC1
TO_MIC2
TO_MIC3
1_AUD
2_GND
3_V+
+3V3
BOARD_RESET
PUSH_SW2
R1
10k
ICSP
MCLR
VSS
VDD
PGD
PGC
18F26k22
1_MCLR
2_AN0
3_AN1
4_AN2
5_AN3
20_VDD
19_VSS
28_PGD
27_PGC
22_SCL2
23_SDA2
26_RB5
25_RB4
24_RB3
21_RB0
9_OSC1
10_OSC2
17_TX1
18_RX1
8_VSS
EPUCK_RESET
PUSH_SW2
SPKR
U$1
VREG-3.3V
VIN
VOUT
ON/OFF
BYPASS
LP2985-VREG
C4
10n
C3
1u
R6
180
R7
180
Q1
20M
C1
C2
18pF
18pF
3_TX
2_GND
1_RX
RS232_OUT

A.3 Derivation of Microphone Phased Array Frequency Response

The frequency response of a microphone phased array is the summation of the signals at each microphone. These waves are not in phase so their amplitudes cannot simply be added, as shown by figure A.2.

To sum waves, they are first represented as vectors on a complex plane as shown in figure A.3. Then we sum the vectors, and the resultant gives the combined waveform. The real and imaginary parts of the wave vector can be described by the following equations, where A is the wave amplitude:

$$\Re(wave) = A\cos(\phi) \tag{A.1}$$

$$\Im(wave) = A\sin(\phi) \tag{A.2}$$

Consequently, the sum of all the waves is the sum of the real and imaginary parts of all the composite waves. A wave can also be represented as a function of its frequency f and time t by the expression $\sin(2\pi ft)$. For each microphone, the time value will be slightly different because the wave will have travelled different amounts, depending on the phase, as illustrated in figure A.4.

Figure A.4 shows that the distance the wave must travel before reaching each microphone is $L\sin(\theta)$, so the time delay for each microphone is the distance divided by the speed of sound c. This gives the real and imaginary parts of the wave as:

$$\Re(wave) = A\cos(2\pi f\frac{L\sin(\theta)}{c}) \tag{A.3}$$

$$\Im(wave) = A\cos(2\pi f\frac{L\sin(\theta)}{c}) \tag{A.4}$$

Since $e^{jx} = \cos(x) + j\sin(x)$:

$$wave = A\exp(2\pi f\frac{L\sin(\theta)}{c}) \tag{A.5}$$

To sum this over all microphones, we just need to calculate the new value of L to get the time delay for each microphone. If there are N microphones evenly spaced L metres apart, this gives the equation:

$$\frac{1}{N}\sum_{m=0}^{N-1}\exp(\frac{j2\pi fmL\sin\theta}{c}) \tag{A.6}$$

The summation is normalised between 0 and 1 by dividing by N. The equation assumes the microphones receive the wave perfectly across all frequencies and that the the wave is not attenuated whilst being transmitted from the source to the array.

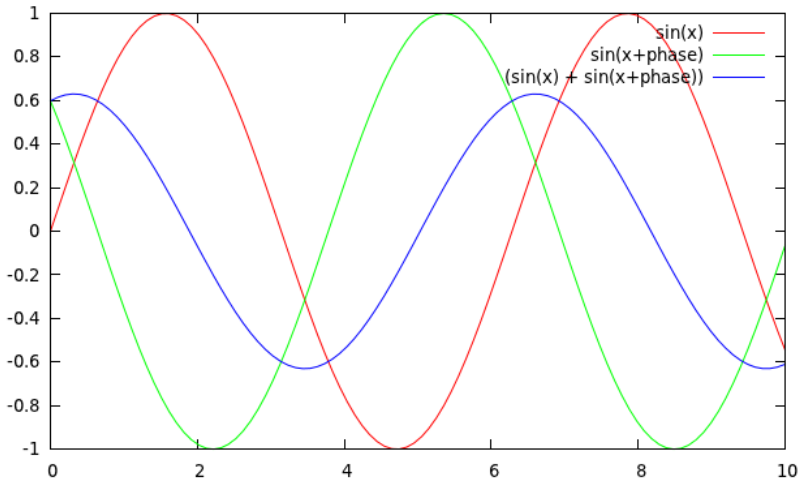


Figure A.2: When two waves are out of phase, their amplitudes do not add linearly. In this example, the combined wave strength is smaller than it is individually.

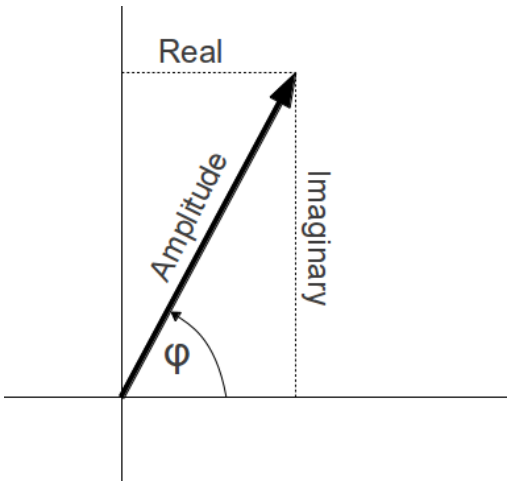


Figure A.3: Signal represented as a vector in a complex plane. The length of the vector is the amplitude of the signal and ϕ is its phase.

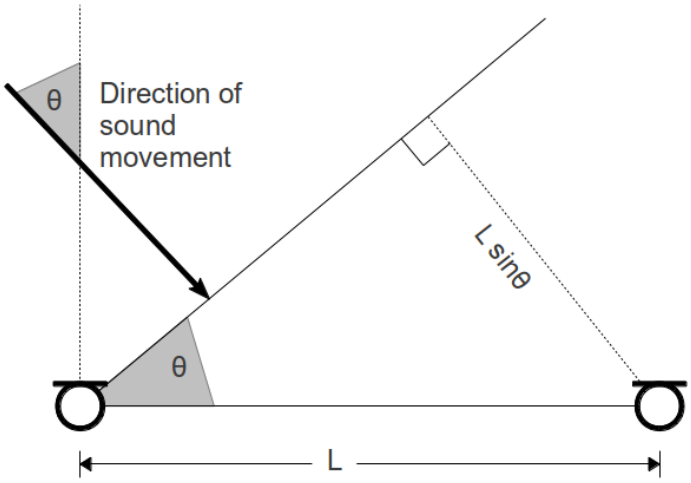


Figure A.4: The extra distance a wave must travel to reach each microphone is $L\sin(\theta)$, where θ is the DOA and L is the distance between the microphones.

Bibliography

[1] Epuck education robot website. `http://www.e-puck.org`, 2011.

[2] ABComponents. ABS-216-RC specification sheet. `http://www.farnell.com/datasheets/1662526.pdf`.

[3] PW Anderson. More is different. *Science*, 177:393–396, 1972.

[4] Paul S. Andrews, Fiona Polack, Adam T. Sampson, Jon Timmis, Lisa Scott, and Mark Coles. Simulating biology: Towards understanding what the simulation shows. In Susan Stepney, Fiona Polack, and Peter Welch, editors, *Proceedings of the 2008 Workshop on Complex Systems Modelling and Simulation, York, UK*, pages 93–123. Luniver Press, 2008.

[5] Paul S. Andrews, Fiona A. C. Polack, Adam T. Sampson, Susan Stepney, and Jon Timmis. 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.

[6] Paul S. Andrews, Susan Stepney, Tim Hoverd, Fiona A. C. Polack, Adam T. Sampson, and Jon Timmis. CoSMoS process, models, and metamodels. In Susan Stepney, Peter Welch, Paul S. Andrews, and Carl G. Ritson, editors, *Proceedings of the 2011 Workshop on Complex Systems Modelling and Simulation*, pages 1–13. Luniver Press, August 2011.

[7] Paul S. Andrews, Susan Stepney, Jon Timmis, Fiona Polack, Adam Sampson, Peter Welch, and Frederick Barnes. The CoSMoS process: Simulations for scientific exploration. *ECCS 2009: European Conference on Complex Systems*, September 2009. extended abstract.

[8] Peter J. Angeline. Genetic programming and emergent intelligence. In Kim Kinnear, editor, *Advances in genetic programming*, pages 75–97. MIT Press, Cambridge, MA, USA, 1994.

[9] Aristotle. *Metaphysics*, volume H. Oxford University Press, 1924, 350 BC. Translated by W. D. Ross.

[10] R.D. Beer and J.C. Gallagher. Evolving dynamical neural networks for adaptive behaviour. *Adaptive Behavior*, 1:91–122, 1992.

[11] Gerado Beni. The concept of cellular robotic system. In *Proceedings of the IEEE International Symposium on Intelligent Control*, pages 57–62, 1988.

[12] Gerado Beni and Jing Wang. Swarm intelligence in cellular robotic systems. In *Proceedings NATO Advanced Workshop on Robots and Biological Systems*, volume 102, pages 703–711, 1989.

[13] Gerardo Beni. From swarm intelligence to swarm robotics. In Erol Şahin and William M. Spears, editors, *Swarm Robotics*, volume 3342 of *Lecture Notes in Computer Science*, pages 1–9. Springer, 2004.

[14] Eric Bonabeau, Marco Dorigo, and Guy Theraulaz. *Swarm intelligence: from natural to artificial systems*. Oxford University Press, Inc., New York, NY, USA, 1999.

[15] Rodney A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6(1&2):3–15, June 1990.

[16] R. Burbidge, J.H. Walker, and M.S. Wilson. Grammatical evolution of a robot controller. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 357–362. IEEE, 2009.

[17] J. Byrne, M. O'Neill, J. McDermott, and A. Brabazon. An analysis of the behaviour of mutation in grammatical evolution. *Genetic Programming*, pages 14–25, 2010.

[18] S. Camazine, J.L. Deneubourg, N.R. Franks, J. Sneyd, G. Theraulaz, and E. Bonabeau. *Self-organisation in Biological Systems*. Princeton University Press, 2001.

[19] James P. Crutchfield and Karl Young. Inferring statistical complexity. *Physical Review Letters*, 63(2):105–108, July 1989.

[20] R.A. Dain. Developing mobile robot wall-following algorithms using genetic programming. *Applied Intelligence*, 8(1):33–41, 1998.

[21] Tom De Wolf and Tom Holvoet. Towards a methodology for engineering self-organising emergent systems. In H. Czap, editor, *Self-Organization and Autonomic Informatics*, volume 135, pages 18–34. IOS Press, 2005.

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

[23] J. L. Deneubourg, S. Goss, N. Franks, A. Sendova-Franks, C. Detrain, and L. Chrétien. The dynamics of collective sorting robot-like ants and ant-like robots. In *Proceedings of the first international conference on simulation of adaptive behavior From animals to animats*, pages 356–363. MIT Press, 1990.

[24] René Descartes. *Discourses on the Method of Rightly Conducting One's Reason and of Seeking Truth in the Sciences, Part V*. Project Gutenberg, 1637. Translator not given.

[25] M. Dorigo and M. Birattari. Swarm intelligence. *Scholarpedia*, 2(9):1462, 2007.

[26] S Dumbacher, J Blough, D Hallman, and P Wang. Source identification using acoustic array techniques. In *Proceedings of the SAE Noise and Vibration Conference*, volume 2, pages 1023–1035, May 1995.

[27] A.E. Eiben and J.E. Smith. *Introduction to Evolutionary Computing*. Natural Computing Series. Springer, 2003.

[28] Sparkfun Electronics. Amplified mic electret v14 schematic. `http://www.sparkfun.com/datasheets/BreakoutBoards/Amplified-Mic-Electret-v14.pdf`.

[29] Dario Floreano, Sara Mitri, Stéphane Magnenat, and Laurent Keller. Evolutionary conditions for the emergence of communication in robots. *Current Biology*, 17(6):514–519, 2007.

[30] N. R. Franks, N. Gomez, S. Goss, and J. L. Deneubourg. The blind leading the blind in army ant raid patterns: Testing a model of self-organization (hymenoptera: Formicidae). *Journal of Insect Behavior*, 4(5):583–607, 1991.

[31] Phillip Garnett, Susan Stepney, and Ottoline Leyser. Towards an executable model of auxin transport canalisation. In Susan Stepney, Fiona Polack, and Peter Welch, editors, *Proceedings of the 2008 Workshop on Complex Systems Modelling and Simulation*, pages 63–92, 2008.

[32] Teodor Ghetiu, Robert D. Alexander, Paul S. Andrews, Fiona A. C. Polack, and James Bown. Equivalence arguments for complex systems simulations – a case-study. In *Complex Systems Simulation and Modelling Workshop (CoSMoS 2009)*, pages 101–140, 2009.

[33] Robert B. Glassman. Persistence and loose coupling in living systems. *Behavioural Science*, 18:83–98, 1973.

[34] F. Gruau. Automatic definition of modular neural networks. *Adaptive Behavior*, 3(2):151–183, September 1994.

[35] Andrew Guest. personal communication, 2010.

[36] John H. Holland. Complex adaptive systems. *Daedalus*, 121(1):17–30, 1992.

[37] B. Hölldobler and E.O. Wilson. *The ants*. Belknap Press of Harvard University Press, 1990.

[38] Bert Hölldobler and Edward O. Wilson. *Journey to the Ants: A Story of Scientific Exploration*. Belknap Press, 1994.

[39] D.M. Howard and J. Angus. *Acoustics and Psychoacoustics, Third Edition*. Focal Press. Elsevier Science & Technology, 2006.

[40] Mattias Jacobsson, Sara Ljungblad, and Johan Bodin. Glowbots website. Glowbots Economy Library for the E–Puck.

[41] Nicholas R. Jennings. An agent-based approach for building complex software systems. *Communications of the ACM*, 44(4):35–41, 2001.

[42] Chris Jones and Maja J Matarić. Adaptive division of labor in large-scale minimalist multi-robot systems. In *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1969–1974, 2003.

[43] J Kennedy and R Eberhart. Particle swarm optimisation. *Proceedings of the IEEE International Conference on Neural Networks*, 4:1942–1948, 1995.

[44] Andrey Kolmogorov. Three approaches to the quantitive definition of information. *Problems of Information Transmission*, 1:1–17, 1965.

[45] J. Koza. *Genetic Programming*. MIT Press, Cambridge, MA, 1992.

[46] J.R. Koza. Genetic programming as a means for programming computers by natural selection. *Statistics and Computing*, 4(2):87–112, 1994.

[47] J.R. Koza and J.P. Rice. Automatic programming of robots using genetic programming. In *Proceedings of the national conference on artificial intelligence*, pages 194–194. JOHN WILEY & SONS LTD, 1992.

[48] H Krim and M Viberg. Two decades of array signal processing research: the parametric approach. *IEEE Signal Processing Magazine*, 13(4):67–94, 1996.

[49] C. Ronald Kube and Hong Zhang. Collective robot intelligence. In *Proceedings of the second international conference on From animals to animats 2 : simulation of adaptive behaviour*, pages 460–468, Cambridge, MA, USA, 1993. MIT Press.

[50] L. Lach, C.L. Parr, and K.L. Abbott. *Ant Ecology*. Oxford biology. Oxford University Press, 2010.

[51] Robert Lacoste. PICFFT18 library. `http://www.alciom.com/images/stories/downloads/fftpic18-v14.zip`, June 2004.

[52] W. Langdon and R. Poli. Fitness causes bloat: Mutation. *Genetic Programming*, pages 37–48, 1998.

[53] Chris G. Langton. Computation at the edge of chaos: Phase transitions and emergent computation. *Physica D: Nonlinear Phenomena*, 42(1-3):12–37, 1990.

[54] Kristina Lerman, Chris Jones, Aram Galstyan, and Maja J Matarić. Analysis of dynamic task allocation in multi-robot systems. *International Journal of Robotics Research*, 25(3):225–241, 2006.

[55] Wenguo Liu and Alan F.T. Winfield. Open-hardware e-puck linux extension board for experimental swarm robotics research. *Microprocessors and Microsystems*, 35(1):60–67, February 2011.

[56] S.M. Manson. Simplifying complexity: A review of complexity theory. *Geoforum*, 32(3):405–414, 2001.

[57] Humberto R. Maturana and Francisco J. Varela. *Autopoiesis and Cognition: the Realization of the Living*. D. Reidel, 1980.

[58] Warren Mcculloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biology*, 5(4):115–133, December 1943.

[59] Microchip. PIC18(L)F2X/4XK22 data sheet. `ww1.microchip.com/downloads/en/DeviceDoc/41412F.pdf`.

[60] Mark M. Millonas. Swarms, phase transitions and collective intelligence. In Christopher G. Langton, editor, *Artificial Life III*, pages 417–446. Addison-Wesley, 1994.

[61] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1996.

[62] Daniel L. Moody. Metrics for evaluating the quality of entity relationship models. In *ER '98: Proceedings of the 17th International Conference on Conceptual Modeling*, pages 211–225, London, UK, 1998. Springer-Verlag.

[63] R Mucci. A comparison of efficient beamforming algorithms. *IEEE Transactions on Speech and Signal Processing*, 32(3):548–558, 1984.

[64] Ulrich Nehmzow. *Scientific Methods in Mobile Robotics*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[65] S Nolfi and D Floreano. *Evolutionary Robotics: The Biology, Intelligence, and Technology of Self-Organizing Machines*. Cambridge, MA: MIT Press/Bradford Books, 2000.

[66] Andrea Omicini. Soda: Societies and infrastructures in the analysis and design of agent-based systems. In P. Ciancarini and M. J. Wooldridge, editors, *Agent-Oriented Software Engineering*, pages 311–326. Springer-Verlag, 2000.

[67] Jennifer Owen, Susan Stepney, Jonathan Timmis, and Alan F. T. Winfield. Exploiting loose horizontal coupling in evolutionary swarm robotics. In *Proceedings of the 7th international conference on Swarm intelligence*, ANTS'10, pages 432–439, Berlin, Heidelberg, 2010. Springer-Verlag.

[68] H. Van Dyke Parunak, Robert Savit, and Rick L. Riolo. Agent-based modeling vs. equation-based modeling: A case study and users' guide. In *Proceedings of the First International Workshop on Multi-Agent Systems and Agent-Based Simulation*, pages 10–25, London, UK, 1998. Springer-Verlag.

[69] Ognen Paunovski, George Eleftherakis, and Tony Cowling. Framework for empirical exploration of emergence using multi-agent simulation. In Susan Stepney, Fiona Polack, and Peter Welch, editors, *Proceedings of the 2008 Workshop on Complex Systems Modelling and Simulation, York, UK*, pages 1–32. Luniver Press, 2008.

[70] Stuart L. Pimm. The complexity and stability of ecosystems. *Nature*, 307(6):321–326, January 1984.

[71] Fiona A. C. Polack, Paul S. Andrews, Teodor Ghetiu, Mark Read, Susan Stepney, Jon Timmis, and Adam T. Sampson. Reflections on the simulation of complex systems for science. In *ICECCS 2010: Fifteenth IEEE International Conference on Engineering of Complex Computer Systems*, pages 276–285. IEEE Press, March 2010.

[72] Fiona A.C. Polack, Tim Hoverd, Adam T. Sampson, Susan Stepney, and Jon Timmis. Complex systems models: Engineering simulations. In *ALife XI*, pages 482–489. MIT Press, August 2008.

[73] Jim Pugh and Alcherio Martinoli. Multi-robot learning with particle swarm optimization. In *International Conference on Autonomous Agents and Multiagent Systems*, pages 441–448, May 2006.

[74] Jim Pugh and Alcherio Martinoli. Parallel learning in heterogeneous multi-robot swarms. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 3839–3846, 2007.

[75] Mark Read, Paul S. Andrews, Jon Timmis, and Vipin Kumar. A domain model of experimental autoimmune encephalomyelitis. In Susan Stepney, Peter Welch, Paul S. Andrews, and Jon Timmis, editors, *Proceedings of the 2009 Workshop on Complex Systems Modelling and Simulation, York, UK, August 2009*, pages 9–44. Luniver Press, 2009.

[76] Justinian P. Rosca and Dana H. Ballard. Hierarchical self-organization in genetic programming. In *Proc. 11th International Conference on Machine Learning*, pages 251–258. Morgan Kaufmann, 1994.

[77] C. Ryan, JJ Collins, and M. Neill. Grammatical evolution: Evolving programs for an arbitrary language. *Genetic Programming*, 1391:83–96, 1998.

[78] Erol Sahin. Swarm robotics: From sources of inspiration to domains of application. In *Swarm Robotics*, volume 3342 of *LNCS*, pages 10–20. Springer, 2004.

[79] Erol Sahin, Sertan Girgin, Levent Bayındır, and Ali Emre Turgut. Swarm robotics. In Christian Blum and Daniel Merkle, editors, *Swarm Intelligence*, Natural Computing Series, pages 87–100. Springer, 2008.

[80] Robert G. Sargent. Verification, validation, and accreditation: verification, validation, and accreditation of simulation models. In *WSC '00: Proceedings of the 32nd conference on Winter simulation*, pages 50–59, San Diego, CA, USA, 2000. Society for Computer Simulation International.

[81] R.O. Schmidt. Multiple emitter location and signal parameter estimation. *IEEE Transactions on Antennas and Propagation*, 34(3):276–280, 1986.

[82] H. A. Simon. *The Sciences of the Artificial*. The MIT Press, Cambridge, MA., 1969.

[83] Herbert A. Simon. The architecture of complexity. *Proceedings of the American Philosophical Society*, 106(6):467–482, 1962.

[84] Herbert A. Simon. The organization of complex systems. In H. H. Pattee, editor, *Hierarchy Theory*, pages 1–27. George Braziller, 1973.

[85] T. Starkweather, S. Mcdaniel, D. Whitley, K. Mathias, D. Whitley, and Mechanical Engineering Dept. A comparison of genetic sequencing operators. In *Proceedings of the fourth International Conference on Genetic Algorithms*, pages 69–76. Morgan Kaufmann, 1991.

[86] Luc Steels. Evolving grounded communication for robots. *Trends in Cognitive Sciences*, 7(7):308–312, July 2003.

[87] Susan Stepney, Robert E. Smith, Jonathan Timmis, and Andy M. Tyrrell. Towards a conceptual framework for artificial immune systems. In G Nicosia and et al, editors, *Third International Conference on Artificial Immune Systems*, number 3239 in LNCS, pages 53–64. Springer, September 2004.

[88] Gilbert Syswerda. Schedule optimisation using genetic algorithms. In L. Davis, editor, *Handbook of Genetic Algorithms*, VNR computer library. Van Nostrand Reinhold, 1990.

[89] Vito Trianni. *Evolutionary Swarm Robotics: Evolving Self-Organising Behaviours in Groups of Autonomous Robots*. Springer, 2008.

[90] Jean-Marc Valin, François Michaud, Jean Rouat, and Dominic Létourneau. Robust sound source localisation using a microphone array on a mobile robot. In *Proceedings International Conference on Intelligent Robots and Systems*, pages 1228–1233, 2003.

[91] András Vargha and Harold D. Delaney. 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, 2000.

[92] Richard Vaughan. Massively multi-robot simulation in Stage. *Swarm Intelligence*, 2:189–208, December 2008.

[93] Richard A. Watson, Sevan G. Ficici, and Jordan B. Pollack. Embodied evolution: Embodying an evolutionary algorithm in a population of robots. In Angeline, Michalewicz, Schoenauer, Yao, and Zalzala, editors, *Congress on Evolutionary Computation*, pages 335–342. IEEE, 1999.

[94] Karl E. Weick. Educational organizations as loosely coupled systems. *Administrative Science Quarterly*, 21(1):1–19, March 1976.

[95] A. F. T. Winfield and J. Nembrini. Safety in numbers: fault-tolerance in robot swarms. *Int. J. Modelling, Identification and Control*, 1(1):30–37, 2006.

[96] AFT Winfield, W Liu, and JD Bjerknes. Functional and reliability modelling of swarm robotic systems. In Paul Levi and Serge Kernbach, editors, *Symbiotic Multi-Robot Organisms*, Cognitive Systems Monographs, pages 56–79. Springer-Verlag, 2010.

[97] Alan Winfield, Wenguo Liu, Julien Nembrini, and Alcherio Martinoli. Modelling a wireless connected swarm of mobile robots. *Swarm Intelligence*, 2(2):241–266, December 2008.

[98] Stephen Wolfram. *A New Kind of Science*. Wolfram Media, January 2002.

[99] Franco Zambonelli. Challenges and research directions in agent-oriented software engineering. *Journal of Autonomous Agents and Multiagent Systems*, 9(3):253–283, 2004.

[100] Dong Zhen, Fengshou Gu, and Andrew Ball. The study of acoustic source localization using a small microphone array for condition monitoring. In Gary Lucas and Zhijie Xu, editors, *Future Technologies in Computing and Engineering: Proceedings of Computing and Engineering Annual Researchers' Conference 2010*, pages 14–19. University of Huddersfield, Huddersfield, December 2010.