Nature Inspired Algorithms for Prioritized Foraging

by

Jade Zoë Abbott

Submitted in partial fulfilment of the requirements for the degree
Master of Science (Computer Science)
in the Faculty of Engineering, Built Environment and Information Technology
University of Pretoria, Pretoria

November 2017

Publication data:

Jade Zoë Abbott. Nature Inspired Algorithms for Prioritized Foraging. Masters thesis, University of Pretoria, Department of Computer Science, Pretoria, South Africa ,December 2013.

Electronic, hyperlinked versions of this thesis are available online, as Adobe PDF files, at:

http://cirg.cs.up.ac.za/

http://upetd.up.ac.za/UPeTD.htm

Nature Inspired Algorithms for Prioritized Foraging

by

Jade Zoe Abbott
E-mail: jabbott@cs.up.ac.za

Abstract

Foraging is a major problem in swarm robotics, which has been applied to many areas such as agriculture, and search and rescue. This dissertation defines a variation of swarm robotics foraging, called prioritized foraging. Prioritized foraging differs from other foraging problems, in that there are two types of items: prioritized items and non-prioritized items. Furthermore, a novel honey bee inspired foraging algorithm is developed. An empirical analysis of three foraging algorithms (a naïve algorithm, and two nature inspired algorithms, namely a desert ant inspired algorithm and the novel honey bee inspired algorithm) is performed on the prioritized foraging problem. The analysis investigates each algorithm's performance on the prioritized foraging problem in terms of the major swarm robotics characteristics of efficiency, scalability, flexibility, and robustness as well as the behaviours that enable those characteristics. The work concludes that the honey bee algorithm is highly efficient, highly flexible, highly scalable in terms of the item density, and most robust in terms of redundancy due to the algorithm's division of labour between prioritized and non-prioritized items. The desert ant algorithm is almost as efficient as the honey bee algorithm. The naïve algorithm has very poor efficiency, compared to the other algorithms, but was the most scalable in terms of swarm size. Both the desert ant and naïve algorithm experience poor flexibility, scalability in terms of problem density, and robustness in terms of redundancy.

Keywords: Swarm Robotics, Foraging, Prioritized Foraging, Desert Ant, Honey bee
Supervisor : Prof. A. P. Engelbrecht
Department : Department of Computer Science
Degree : Master of Science

"It may be that. You never can tell with bees.. . . "

Winnie the Pooh by A. A. Milne

"The Three Laws of Robotics:

1: A robot may not injure a human being or, through inaction, allow a human being to come to harm;

2: A robot must obey the orders given it by human beings except where such orders would conflict with the First Law;

3: A robot must protect its own existence as long as such protection does not conflict with the First or Second Law;"

Isaac Asimov, I, Robot

Acknowledgements

I wish to express the greatest thanks to the following individuals and organisations for their support throughout my research:

- Prof Andries P. Engelbrecht for the amazing guidance and never giving up on me, despite my total loss of motivation, as well as the financial support.
- To my mother and father for their constant love, belief and encouragement.
- Bernard Frankel for sitting next to me and bringing me things, while I clawed my eyes out every Saturday and Sunday.
- Theo Crous for reminding me that the problem is not as big as my mind made it out to be and forcing me to get on with it.
- M-club (particularly Kristina Young, Christien Kroeze and Christopher Cleghorn) for the problem solving over wine and tears.
- Retro Rabbit for the financial support and giving me confidence in my abilities as a computer scientist.
- Wolves Cafe for always having good coffee, wifi, excellent seating, power and allowing me to sit there all day. It made research bearable.
- The National Research Foundation for the financial support. Opinions expressed in this thesis and arrived at, are those of the author and not necessarily the National Research Foundation.

Contents

List of Figures

List of Algorithms

List of Tables

Chapter 1

Introduction

Section 1.1 motivates the purpose of this thesis, and the objectives of this thesis are highlighted in Section 1.2. The contributions of the thesis are presented in Section 1.3, while Section 1.4 outlines the the structure of the thesis.

1.1 Motivation

Consider a search and rescue mission after a natural disaster or mining accident. These search and rescue missions are usually extremely dangerous for the human rescuers who are deployed to search for survivors. The use of swarm robotics to perform such search and rescue mission has been proposed and explored as an alternative to using human rescuers [1, 2].

Swarm robotics is the co-coordination of large numbers of relatively simple robots to perform a single collaborative function. Swarm robotics is inspired from the observation of social insects such as ants, termites, and bees [3]. An important activity of all natural swarms is foraging for resources. Foraging is defined as the search and collection of resources from sources in an environment and returning the resources to a collection point [4]. These resources could be food, water, or building materials. Foraging is an abstraction of the search and rescue problem where the trapped humans are items that robots need to locate and remove to a safe location.

In a search and rescue mission, the location of trapped humans is usually unknown

and humans can be potentially be blocked by rubble, which needs to be removed before the humans can be safely removed. A swarm of robots would need to search for the humans, as fast as possible, to avoid further danger, injury or loss of life. If robots cannot locate the humans, they will have to begin clearing debris in the hope of locating them, as well as clearing the route between the trapped humans and the safe zone. Locating and removing the humans is prioritized above removal of the debris, but often the debris must be removed, in order to locate the trapped humans. Thus there exists a resource prioritization from the perspective of the robot.

Resource prioritization is also relevant in mining problems where the metal ore is prioritized and the waste rock should be cleared to better access the valuable ore. In common gold mining techniques, the ore and waste rock are collected and transported to the surface and chemical techniques are used to separate them. The transport of the waste rock (which forms the majority of the load) to the surface is extremely expensive, so there is a cost–benefit to separate the ore and the waste rock beneath the surface and transport only the valuable ore to the surface.

The above search and rescue problem and mining problem can be abstracted as a foraging problem where there exists resources with differing priorities. In the case of search and rescue, the humans are the prioritized resource and the debris is the non-prioritized resource. In mining, the ore is the prioritized resource and the waste rock is the non-prioritized resource.

In nature, in times of stress, the collection of one resource may be prioritized over others - such as water during a drought or food before winter. Individuals in a natural swarm often adapt behaviour appropriately to enable greater collection of the prioritized item.

This study defines the prioritized foraging problem, and proposes three swarm robotics foraging algorithms to be evaluated on the prioritized foraging problem. The study proposes metrics to evaluate each algorithm's performance on the prioritized foraging problem in terms of foraging efficiency, flexibility, scalability, and robustness. The algorithms are developed for a simulated environment. Each algorithm's performance is evaluated on environments of a variety of complexities, with various swarm configurations.

1.2 Objectives

The primary objectives of this thesis are as follows: To

- conduct a survey of the swarm robotics field;
- conduct a survey of aspects of foraging in social insects that are relevant to development of swarm robotics algorithms;
- define the prioritized foraging problem;
- propose metrics for evaluating the performance of swarm robotics algorithms on the prioritized foraging problem;
- propose and develop different nature-inspired algorithms in a simulated swarm robotic environment, to be evaluated on the prioritized foraging problem; and
- evaluate the efficiency, flexibility, scalability, and robustness of the nature-inspired algorithms over different environments and different swarm configurations on the prioritized foraging problem.

1.3 Contributions

The thesis makes the following contributions:

- A novel variation of the foraging problem, i.e. prioritized foraging, as well as performance measures for the prioritized foraging problem are proposed.
- A novel honey bee inspired foraging algorithm for robot swarms is developed.
- A novel obstacle avoidance algorithm, inspired by the flocking behaviour of birds is developed.
- Methods for generating various types of foraging environments in a simulated environments are proposed.
- An analysis of the efficiency, flexibility, scalability, and robustness of each algorithm on the prioritized foraging problem is conducted.

1.4 Thesis Outline

This thesis consists of eight chapters, each handling a separate topic. The chapters are as follows:

- **Chapter 2** introduces the origin, motivation, development, and current state of swarm robotics.
- **Chapter 3** summarizes foraging behaviour of social insects and reviews existing foraging algorithms in swarm robotics.
- **Chapter 4** defines division of labour and summarizes strategies for division of labour employed by social insects. The chapter reviews where division of labour strategies have been employed by robot swarms.
- **Chapter 5** proposes a novel foraging variation called prioritized foraging. The chapter also proposes three swarm robotic foraging algorithms, inspired by social insects, to be evaluated on a prioritized foraging problem.
- **Chapter 6** presents an experiment designed to evaluate the proposed foraging algorithms in a simplified simulation environment, on the prioritized foraging problem. This chapter describes the environment and swarm parameters that the experiments use. The relevant performance measures to be used to evaluate the proposed algorithms on the prioritized foraging problem are also presented.
- **Chapter 7** analyses the results of applying the proposed algorithms to the prioritized foraging problem.
- **Chapter 8** summarizes the findings of this research.

The thesis contains two appendices, which are organised as follows:

- **Appendix A** lists and defines the mathematical symbols used in this thesis, divided according to the chapter in which they appear.
- **Appendix B** contains the publications derived from this work.

Chapter 2

Swarm Robotics

This chapter provides a broad view of swarm robotics in terms of its origin, motivation, development, and current state. Section 2.1 defines the concept of swarm robotics while the origin and a brief history of swarm robotics is provided in Section 2.4. Motivations for swarm robotics are outlined in Section 2.2, followed by Section 2.5 which expands on the current state of swarm robotics. Lastly, the chapter addresses the challenges faced by swarm robotics in Section 2.6.

2.1 What is Swarm Robotics?

Swarm robotics is the study of the co-coordination of large numbers of relatively simple robots in order to perform a single function, without the existence of central control. Swarm robotics focuses on how to create robotic algorithms that result in the emergence of a desired complex behaviour [5].

Swarm robotics typically draws inspiration from the observation of social insects, for example, ants [6], cockroaches [7], and bees [8] that exhibit collective behaviour in the growth and maintenance of their societies [9]. Social insect societies exhibit desired qualities of a robot swarm, namely robustness, scalability and flexibility. Swarm robotics also draws on concepts from other societies such as the amoeba's aggregation into slime [10], communication, and propulsion and sensing in bacteria [11, 12].

2.2 Motivations for Swarm Robotics

Swarm robots draws inspiration largely from swarms in nature due to the characteristics of robustness, flexibility, scalability, and to a lesser extent cost-effectiveness. The following sections describe the characteristics of natural swarms that all swarm robotics algorithms strive to achieve.

2.2.1 Robustness

In swarm robotics, robustness is defined as the ability of a swarm to continue to perform its function, despite failures or abnormalities of the respective individuals and environments. Three aspects of insect swarm algorithms have been identified as enabling robustness: redundancy, decentralized coordination, and multiplicity of sensing [5].

Swarm robotics algorithms achieve redundancy by giving all, or a portion of the robots in the swarm the same capabilities. In this way, if a percentage of the swarm malfunctions, the other robots have the capability to take the malfunctioning robots' place. The loss of a single individual is compensated by another individual in the swarm. Conversely, a swarm which assigns individual roles to each robot is not redudant, since the swarm is dependant on a single robot for a specific function and if that robot fails, then the swarm has no other robot to perform that function.

Decentralized coordination can be attained by creating algorithms that do not depend on the life span of any single individual or a few individuals of the swarm.

The use of a large number of individuals increases the total number of sensors in the swarm. As a result of the sensory multiplicity, the total signal-to-noise ratio is increased. If noise is experienced by a subset of the swarm, then the overall effect of the noise can be decreased or eliminated, by adequately aggregating the sensor data of the entire robot swarm.

2.2.2 Flexibility

A swarm robotics algorithm should exhibit the ability to adapt and adjust to a wide variety of different environments and tasks [13]. Ants and bees do this by having effective division of labour strategies. The individuals in many insect societies can take on

a variety of different roles required by the nest such as brooding, foraging or nest maintenance, due to changes in the environment [14]. The ability to take on different roles when requirements change, is known as division of labour [15]. Many swarm robotics algorithms make use of the division of labour strategies of social insects in order to adapt to changing requirements [16, 17, 18]. The use of division of labour thus assists the swarm in maintaining flexibility.

2.2.3 Scalability

Scalability refers to the ability of the robotic swarm to expand the self-organizational mechanism, by simply adding more robots to the swarm [13]. The performance of the algorithm should not be negatively effected by an increase in swarm size, and an increase in swarm size should adequately improve the performance of the swarm. Scalability studies should be performed in order to determine whether an algorithm's performance will increase or decrease at an adequate rate as the swarm size increases. Scalability studies of swarm robotics algorithms have been performed in [19, 20, 21]. Scalability consists of two dimensions: in terms of the scalability of the size of the swarm, or in terms the size of the problem being solved [13].

2.2.4 Cost-effectiveness

The idea that swarm robotics is more cost effective than traditional robotics runs off the premise that multiple inexpensive robots are likely to be cheaper to buy and maintain than a single, large, more complex robot.

2.3 Characteristics of Swarm Robotics

Drawing from inspiration from social insects, swarm robotics algorithms have a number of characteristics that distinguish them from other robotics algorithms. The characteristics are as follows:

1. **Quantity**: Studies are regularly focused with scalability as a potential characteristic of swarm robotics algorithms - even if real-robot experimentation is limited

to only a few individuals. In order to test the scalability of an algorithm, models or simulations are often built for experimentation purposes. Sahin proposes that 10 individuals are a reasonable lower bound for a group of robots to be considered a swarm [5]. A group of robots with less than 10 robots, is considered just that - a group of robots.

2. **Homogeneity**: A group of robots that has "too many" individuals with unique characteristics, is no longer considered a swarm, since it would likely violate the robustness requirements. Robustness would be violated since loosing a specific robot, that is the only individual of its kind capable of performing a specific function would mean that the swarm no longer can perform that function and is therefore no longer robust. It is of the author's opinion that heterogeneity can exist in a robot swarm provided that the redundancy of each type of robot in the swarm is high enough, or if the failure of that specific robot can be compensated for by the remaining types of robots, even if only to a lesser efficiency. The evaluation of the degree of homogeneity of a swarm of robots has been discussed and a potential measure of homogeneity is proposed in [22].

3. **Decentralization and autonomy**: There should be no single point of failure in the group of robots and the robots must have complete self-control of their actuation and sensors, without central control. Decentralization and autonomy of the robots are key factors enabling the robustness of the system. If a particular process requires a single leader to make a decision, such a process should include the ability to detect situations when a leader is no longer present (e.g. malfunction or destruction) and then to re-elect a new leader without human intervention, otherwise the swarm can no longer continue its function at the failure of a single individual.

4. **Localization**: Assume there exists a group of robots that are dependant on global sensors and communication. An example of such a global sensor would be where all inter-robot communications have to pass through a centralized server, or where all individuals receive the video feed of an overhead camera that each individual uses for navigation. The problem with global sensing or communication is that

the swarm has a single point of failure, thus violating decentralization. Thus, local sensory and communication abilities are required in order to uphold the decentralization requirement. If sensors are local to each robot, when a single robot's local communications fail, then the rest of the swarm can still communicate (albeit, in a deteriorated manner).

5. **Simplicity**: A single robot should be under-equipped to handle the task by itself. However, collaboration amongst a group of the same robots should assist the completion of the task. In the author's opinion, the level of simplicity of robots is a topic worth debating, since "simple" is a relative term and various complexities of robots have been used in swarms. For example, the kilobot project [23] has built very low cost simple robots to enable testing of collective behaviours on very large swarms. The kilobots do not even have wheels for motion but simply have three rigid legs with vibration motors for actuation. Sensors are limited in the form of a simple ambient light sensor and an infrared transmitter and receiver, and coloured LED lights for communication.

   On the other side of the spectrum, Melliner *et al* [24, 25] explore robotics algorithms for swarms of relatively advanced, expensive quadcoptors with a variety of top-class sensors such as magnetometers, accelerometers, gyros, barometer for altitude sensing, and two Zigbee transceivers for communication.

The field of swarm robotics has been applied to a large variety of problems such as search and rescue [26], item or garbage collection [27], autonomous inspection of machinery [28], and military formations for military application [29].

2.4 History

Swarm robotics is simply the latest buzzword for a concept that has been of interest since the 1980s. Swarm robotics has gone by many other names in the past, such as "multi-agent systems" or "collective robotics". Early swarm robotics research explored the use of robots as a tool with which to model and validate entomology research on social insects [30, 31, 32].

This section aims to briefly highlight the history of swarm robotics and its progress by addressing behaviour based robotics, the origins of multi-robot systems, and other early research.

2.4.1 Journey from classic robotics

The first step in the movement from symbol-based classical robotics to swarm robotics was the movement from complex symbolic systems towards more simpler architectures which began in the mid-to-late 1980s. In Brooks' iconic paper "Elephants don't play chess" [33], the movement from traditional artificial intelligence, towards what they brand "nouvelle" artificial intelligence, is discussed. Brooks explains that traditional symbol-based classical artificial intelligence made certain assumptions about how intelligence worked and those assumptions actually impeded traditional artificial intelligence techniques.

Brooks presents the physical grounding hypothesis which is the idea that intelligence is composed of individual modules that generate behaviour and the co-existence and co-operation of these modules allow for the emergence of complex behaviour. The physical grounding hypothesis states that the world is its own best model, and thus in order to accurately make decisions about the real world, one should add sensing and actuating capabilities to artificial intelligence agents.

On the other hand, classical or symbolic artificial intelligence (most notably used by systems such as Prolog) makes use of the symbol system hypothesis that states that all intelligence operates using symbols. The implication that symbols represent all world entities fails since symbols may not be up to date with the environment or have the ability to represent unseen things.

The physical grounding hypothesis lead to the development of behaviour-based robotics and more concretely the subsumption architecture [34] by tightly connecting perception to action. A subsumption program organises finite state machines into layered incremental networks.

Multiple systems were developed using the subsumption architecture with great success. Allen, a robot that has three layers of behaviours, namely an obstacle avoidance layer, a random wandering layer, and a search for the furthest point layer. These three

separate behaviours enable the robot to adequately explore an environment [34].

Herbert, the soda can collector robot, was a notable development. Herbert was required to complete the significantly more complex task of searching for and picking up empty soda cans from people's desks [35]. To solve the problem, 15 individual behaviours were developed, with no communication between the behaviour generating modules. Other subsumption implementations include Toto [36], Squirt [37] and Genghis [38].

The key feature that can be determined from the success of physically grounded systems is that interactions of simpler non-goal directed behaviour results in emergence of goal-directed behaviour. Arkin discusses the idea that behaviour-based robotics imposes a biologically bottom up approach with the need that intelligence must be reactive to the dynamic environment and that intelligence needs to generate robust results despite noisy complex real world environments [39].

Behaviour-based robotics is the idea that complex robot behaviour can emerge from a combination of simple behaviours. It is of the author's opinion that behaviour-based robotics signifies a change in focus of robotics research from complex indvidual robots to complex robots composed of simple behaviours (behaviour-based robotics), and finally to multi-robot systems consisting of simple robots composed of simple behaviours. Most work in swarm robotics began after the introduction of the behaviour-based robotics paradigm [40].

2.4.2 Evolution of the term "Swarm Robotics" and early research

Early research involving collaboration of multiple agents was originally a way of verifying entomology research on social insects [30, 31, 32]. Many researchers modelled and simulated aspects of the insect colonies that were being studied in order to learn more about their self-organizational mechanisms.

Early work included modeling the excavation behaviour and tunnel creation of ants in order to simulate the rate of excavation [41]. Agent simulation was used to validate a model of the spatial arrangement and diet overlap between colonies of desert ants [42].

Seeley *et al* [43] formulated a mathematical model of collective decision-making in bee colonies where digital simulations were used to determine validity of a model of collective

foraging in bees based on individual behaviour rules [44]. The foraging behaviour of ants has also been simulated in early research [45]. Such studies discovered many of the features about social insects that are exploited in swarm robotics today.

From the late 1980s until the mid 1990s, swarm robotics research emerged under many different guises: Collective robotics [46], cellular robotics [47], co-operative mobile robotics [48], distributed robotics [49], and multi-robot systems [50]. It is difficult to determine which terms were used first as they seem to have been in use concurrently. Essentially, all of these terms have converged to what we now know as swarm robotics.

In one of the earlier, more notable papers, Freund explored the design of the structure of multi-robot systems based on nonlinear control approaches. Freund demonstrated a design approach on the collision avoidance of two robots working on the same space [47, 51]. Around the same time, the multi-robot design paradigm ACTRESS was developed to also address the design of autonomous distributed robot systems [52].

In the late 1980s and early 1990s, Fukuda *et al* [53, 54] introduced work in the field of cellular or configurable robotics. Cellular robotics was defined as a robotic system that can reconfigure itself based on dynamic environmental requirements. The idea of cellular robotics is that cells of smaller, simpler robots can dock with each other to form a single, compound structure that can more effectively handle the pressures of the new environment. The research predominantly explored how the distributed communication between the cells would work. Fukuda *et al.* evaluated and described the CEBOT structure with an optimal knowledge allocation method to communicate between cells. Around the same time, Beni *et al* [55] explored the problems cellular robotics research would need to solve, in order for cellular robotics to become feasible in real-world applications.

The various ideas for the use of multiple robots to achieve a task have now been grouped under the umbrella term of swarm robotics. Since the 1990s, swarm robotics has become a popular field of research in artificial intelligence, and swarm robotics has peaked the interest of the world.

2.5 Current State Of Swarm Robotics

There are a number of axes of focus on swarm robotics research. The core dimensions of the field are modelling, behaviour design, communication, analytical studies, and applications. This section discusses those core dimensions in order to give an overview of the current state of swarm robotics research.

2.5.1 Swarm robot analysis

Macroscopic and microscopic models are often used to determine to what extent a property, such as scalability or performance, is satisfied or not satisfied. Microscopic models aims to model each robot individually where as macroscopic models model the entire swarm and are modelled in formal mathematics.

Microscopic models

Microscopic models are concerned with individual agents, and the models analyse interactions between each robot and between each single robot and the robot's environment. Microscopic models are implemented to varying levels of detail - some are simplified to a 2D grid world environment, where as others choose to opt for a continuous 3D environment with dynamic physics. The level of detail is dependant on the problem and what is being researched.

Predominantly, microscopic models take the form of simulators and are used to validate swarm robotics systems. Swarm-robotic specific simulators include Stage [56] and ARGoS [57], both of which focus on simulating a large number of agents. A concern of obtaining and maintaining a large number of agents is one of the reasons why simulators are used in swarm robotics instead of real world experimentation.

Macroscopic models

Macroscopic models are focused on a higher level view of the entire system and the individuals of the system are not analysed. A variety of macroscopic models exist as follows:

- **Rate and differential equations**: Rate equations are used to describe the change in the proportion of robots in a particular state over time. Rate equations were used to model a variety of swarm robotics problems such as clustering [58], stick pulling [59], foraging [60], chain formation [61], and multi-foraging [62]. However, modelling space and time is complex since robots' positions in space are not modelled explicitly.

  Similarly, differential equations [63] and systems of partial differential equations [64] have been used to model swarms as well as factors such as noise, stochasticity, and spatiality. Unfortunately, such systems are often computationally expensive and difficult to solve.

- **Classical control and stability theory** has been used to prove swarm properties [65, 66, 67]. Classical control methods have the advantage of being based on sound mathematics. However, they often rely on assumptions in order to simplify the modelling process. In reality, many of those assumptions are often violated. For example, Liu *et al* model a multi-agent foraging process, but make the assumption that there exists a resource gradient that is continuous and smooth. In real world scenarios environments are unlikely to have continuous and smooth resource gradients.

- **Other methods:** Many other mathematical modeling approaches have been used in a swarm robotics context, such as linear time temporal logic to define safety and liveness of swarm individuals [68], probabilistic model checking to verify swarm properties [69], or using branching processes to model communication of a swarm of aerial robots [70].

Real-robot analysis

Since it is implausible to attempt to simulate reality completely, real robot experimentation is integral to validating the behaviour of the swarm. Real-robot simulations usually occur in controlled environments that have the ability to control the level of noise and environmental disruption. The disadvantage of performing real robot analysis is that experimentation is generally more costly, complex, and time consuming than simulation

or modeling methods. The purpose of real robot analysis is to show that the prospective swarm behaviour is actually obtainable [13].

2.5.2 Behaviour design

In order to achieve emergent behaviours, a number of approaches for the design of robot controllers have been used. This chapter addresses the techniques of behaviour design, namely non-adaptive, learning, and evolutionary approaches.

Non-adaptive design

Non-adaptive behaviour design, in general, refers to techniques of behaviour design where by the algorithms have specifically been hand-crafted by the human designer. These techniques either utilize mathematical approaches or focus on how to combine simpler behaviours to achieve the desired emergent behaviour. Types of non-adaptive design are discussed as follows:

- **Subsumption Architecture:** The subsumption architecture is a robot architecture developed to aid the construction of robots that can interpret and respond to multiple environmental stimuli efficiently and correctly. In the subsumption architecture, each robot behaviour forms a separate module that can inhibit other behaviours [35]. These modules are arranged in a series of incremental layers connecting perception to action.

- **Probabilistic Finite State Automata**: Probabilistic finite state automata are a method to represent dynamical systems with finite state spaces. Each behaviour is a state and transistions between these states occur at specified probabilities based on external input [71, 72]. The algorithms addressed in this thesis take the form of probabilistic finite state automata.

- **Distributed Potential Field Methods**: In physics, potential energy is the energy possessed by a body resulting from position or configuration. For instance, a robot that is on top of a slope has greater potential energy than a robot at the bottom of the slope. A potential field is a collection of vectors that represents the

direction and force of the potential energy, i.e. the directions that a robot has the potential to move. Thus, each robot has a potential field which is made up of a vector combination of individual behaviours (for instance navigation behaviour resulting in a movement vector in one direction would be added to the vector from obstacle avoidance behaviour). A distributed potential field refers to the potential field of one robot to another robot in a robot swarm. Distributed potential fields are useful in building swarm behaviours that require spatial distribution between robots such as pattern formation, obstacle avoidance or motion coordination. Examples of using distributed potential field methods can be seen in [73, 74, 75].

Learning

Robots can have the ability to learn behaviours suited to solve certain problems. A number of techniques have been used to learn behaviours, most notably reinforcement learning and neural networks [76, 77].

Evolution

Neural network or tree-based controllers for swarms of robots can be evolved or optimized using a variety of algorithms from swarm to genetic techniques [78, 79]. Francesca *et al* define two evolutionary approaches, AutoMoDe-Vanilla and EvoStick [80, 81]. The experiments compared the effectiveness of evolving swarm robot controllers in comparison to human created controllers. The interesting result is that the evolved AutoMoDe-Vanilla controller was able to outperform the human created controller, thus showing the promise for using evolutionary techniques to design swarm controllers.

2.5.3 Interactions

A key element of swarm robotics is the interactions between robots in a swarm. Cao *et al* [48] classified the methods of information transfer between robots in their survey on swarm robotics. Interactions were segmented into three types:

- **Interaction via sensing**, where no direct communication between robots occurs. Robots simply obtain information from their senses, for instance the stick pulling

problem whereby a robot can sense another robot pulling the stick [82].

- **Interaction via the environment**, where the robots modify the environment in some way. Other robots then perceive and understand that modification, such as pheromones in ants. Robots have mimicked pheromone-like deposits in a number of ways, such as a substance distributor [83] or beacon deployer [84].
- **Interaction via communication**, where robots interact and transfer information by assigning meaning to specific signals or having a specific language [6]. The language has been in the form of light signals or even direct data transfer via bluetooth.

2.5.4 Behaviours

Brambilla *et al* [13] present a taxonomy of swarm behaviours. These behaviours can be used as ingredients to solve more complex problems.

Spatially Organizing Behaviours

The following behaviours involve agents positioning themselves in their environment in relation to the position of other agents:

- **Aggregation:** Aggregation is the movement of individuals towards one another to form a cluster and has become one of the fundamental swarm behaviours. In nature, aggregation aids protection from predators, the ability to defend against an unfavourable environment and to locate partners [85]. A variety of approaches to swarm robot aggregation have been investigated [86, 87, 88]. A more recent approach finds inspiration in honey bees [89, 90].
- **Pattern formation**: The goal of pattern formation is for deployed robots to maintain specific distances between each other in order to create a particular shape. Implementation usually utilizes virtual forces in order to co-ordinate positioning of the agents. A review can be found in [91, 92].

- **Path formation** is the creation of chains of robots in order to connect two or more points. The robot chains can serve as pheromone paths in order to navigate other robots through environments, because generally, robots are not equipped with pheromone sensors and deployers. Research in path formation forms part of environmental exploration and navigation [93].

  Path formation is useful as a navigational strategy when dealing with robot swarms since traditional complex forms of navigation do not scale well as the number of robots increases. Path formation techniques also have the advantage of being more failure tolerant since they do not rely on a specific individual.

  Research in path formation initially had the robots emit a signal communicating their position. Unfortunately, this introduces the problem of global localiation [94]. Later, path formation using real robots in a prey retrieval experiment, where the robots used physical contact to sense each other, was studied [95]. More recent approaches attempt to give directionality to the chains by giving the chains a cyclic directional pattern. The approach was tested with real robots to transport heavy objects [96]. Path formation has been used to connect two objects that are too far from each other to be perceived at the same time by a robot [93].

- **Self-assembly and morphogenesis**: Ants can create connections to each other to pull large prey and build bridges or walls [97, 98]. Self-assembly refers to individual robots with the ability to connect in order to create a compound structure of robots. Self-assembly was researched very early in the swarm robotics field, originally known as cellular robotics. In application, self-assembly has been used to stabilize robots on difficult terrains, and to combine forces to transport another object [13]. Self-assembly has remained one of the more complex behaviours, due to the challenge of morphogenesis and how to control the structure in a consistent manner. Reviews on self-assembly are presented by Groß and Dorigo [99].

- **Object clustering and assembling**: The goal of clustering is to group similar objects together, and the goal of assembling is to link objects in a required way.

  Challenges in the field of clustering and assembling are to do with interference from robots and obstacles near the cluster site. A variety of implementations of

clustering have been developed [100].

Aggregation differs from object clustering and assembling, in that the goal of aggregation is to bring the robots themselves closer together, where as object clustering and assembling involves robots moving items in specific ways.

Navigation behaviours

Robot navigation is a difficult task and navigational complexity increases as the number of robots increases due to the increase in inter-robot interference. In order to decrease interference, knowledge can be gained by sharing information about navigation between robots. Navigation behaviours are outlined in the following section:

- **Collective exploration**: Collective exploration is the use of a swarm to search and map out features of an area. Brambilla *et al* [13] break collective exploration into two behaviors, namely area coverage and swarm-guided navigation. The goal of area coverage is to deploy robots with the aim of creating a grid of communicating robots over a space. Area coverage is usually approached by using virtual physical forces [101, 102]. Swarm-guided navigation is the process of discovering the most efficient route to a goal location using swarms. Swarm navigation is usually achieved using probabilistic finite state machines [103, 104].

- **Coordinated motion** enables robots to move together like a swarm of locusts, a school of fish, or a flock of birds. Navigating effectively as a swarm reduces interference between agents, increases the safety of individuals, and reduces energy consumption [105]. Research performed in this domain is focused around maintaining a constant distance between each robot as well as on a means of calculating the overall heading direction of the swarm [106, 107, 78].

- **Collective transport** has become a benchmark for swarm robotics since it requires numerous collaborative aspects such as task allocation, conflict resolution, and communication. Mataric *et al* use a simple communication protocol on real robots to compensate for noise-ridden sensing to use a group of robots to move a box [50]. Hiroshi *et al.* do not utilize explicit communication, but rather allow

robots to infer the intention of other robots via their behaviour in order to enable the robots to correctly place desks in an environment. The study indicates the benefit of collaboration [108]. Gerkey *et al* implement a dynamic task allocation algorithm for a robot with an auction-based mechanism using publish/subscribe communication also to perform a box pushing exercise [109]. The SWARM-BOTS project involved substantial work with s-bots. S-bot robots have the ability to link to each other. Experimentation showed that the chained s-bots were able to transport items more effectively than s-bots that were not chained [110, 111, 112].

Collective decision-making

With any multi-agent system, the ability of the system to collectively make decisions results in a far more complicated system than in traditional robotic systems. This section discusses and provides examples of types of collective decision making behaviours as follows:

- **Consensus achievement** is the process of how a group of robots come to a decision among a variety of alternative choices in order to maximise system performance. This is observed in ant colonies to determine the shortest path [85] and is used by bees to decide on the next best site for a nest [113].

- **Division of Labour** occurs naturally in insect colonies [114], most notably in bees and ants. Division of labour is a key problem in swarm robotics. Most tasks require different behaviours in order to reach completion. The goal of division of labour in swarm robotics is to most effectively assign behaviours to robots in the swarm in order to more optimally achieve some final goal. Division of labour is usually addressed as part of a more complex problem such as foraging where by multiple roles are required for efficient foraging to occur such as foragers and scouts and the ratio of foragers to scouts. Another common division of labour problem is to find and maintain the optimal number of robots to perform an activity as opposed to resting. Too many robots performing a task may increase the number of collisions, and increased collisions decreases the overall efficiency of the swarm. In the same line, too many robots performing a task leads to wasting of energy, thus the need

for effective division of labour between active and passive robots arises [115, 116].

2.6 Challenges

Sahin *et al.* suggesed a number of challenges faced by swarm robotics [117]. These challenges are discussed below:

- **Emergent Algorithm Design**: As the field stands, there is no real way to actually design individual behaviour in order to guarantee a specific emergent behaviour. Hamman *et al.* attempted to devise a mathematical model to assist in modelling swarm behaviour and communication in order to help bridge the local behaviour to global behaviour [63]. Unfortunately, Hamman *et al.*'s model is limited to specific sensory or actuation abilities.

- **Experimentation**: In order to properly study swarm robotics, the appropriate infrastructure is required - from robot hardware to cameras with which to monitor the actions of the robots. The need for real robots can be alleviated by use of simulation environments. A number of simulators have been developed in order to enable easier experimentation at different levels of accuracy [56, 57, 118]. Some were only 2D grid-world models while others attempted to mimic real life as accurately as possible [119]. It is of the author's opinion that the field suffers from a lack of real robot experimentation and that little validation of the accuracy of simulators has been performed.

- **Analysis and Modelling**: While linear swarm robotic systems do exist, swarm robotic systems are most often stochastic non-linear, and often highly-coupled systems. It is difficult to prove properties as well as optimize any required parameters of stochastic non-linear systems. Thus, models for swarm robotics have made assumptions about the swarm, or the environment, to simplify the swarm robotics system so that it was easier to prove specific properties of the swarm or to optimize swarm parameters. The risk of making assumptions about the robots and the environment is that the conclusions achieved from analysis of the models may not reflect reality.

- **Robotic systems are challenging in themselves**. Swarm robotics will also still be weighed down by the challenges faced by traditional robotics. Even if the swarm element is removed from the system, robotic systems require expertise from a variety of very different fields such as physics, programming, artificial intelligence, and mechanics.

The study of the process of how to achieve reliable real-world swarm systems and how to address the above challenges, is known as swarm engineering [13].

2.7 Summary

This chapter defined swarm robotics as the study of the co-coordination of large numbers of relatively simple robots in order to achieve a single goal, without the existence of central control. The history of swarm robotics was traced from its potential origins in classical robotics. The guises that swarm robotics went by in its early days were discussed and consolidated into a single concept. Motivations for swarm robotics were discussed in order to contextualize the relevance of swarm robotics and its recent popularity. This chapter addressed the current state of swarm robotics. The current state of swarm robotics was broken down into sub-fields of study, namely, analysis, behaviour design techniques, interaction types, and the many swarm behaviours. Finally, the challenges that swarm robotics faced were highlighted. The next chapter discusses foraging in nature and swarm robotics.

Chapter 3

Foraging

In swarm robotics, robot foraging has become a benchmark problem due to its complex nature involving the coordination of numerous sub-tasks. Robot foraging is also one of the swarm robotics problems that has some very obvious useful applications. This chapter aims to provide an in-depth review of what foraging is, how different social insects forage as well as to provide a review of existing foraging algorithms in swarm robotics. A definition of foraging in swarm robotics is provided in Section 3.1, while Section 3.2 describes foraging behaviours observed in nature, specifically in bees and ants. Section 3.3 presents a taxonomy for swarm robot foraging and discusses existing swarm robotics foraging algorithms. Prioritized foraging is defined and discussed in Section 5.1, and the chapter is summarized in Section 3.4.

3.1 Definition of Foraging

Foraging was initially studied by biologists, particularly the foraging behaviour of ants [120, 121], bees [32] and bacteria [122]. Foraging has a variety of real-life applications such as search and rescue [123, 124] and waste clean-up [27]. Numerous swarm robotics algorithms have been developed to improve robustness, time, and energy efficiency of the foraging process for multiple variations of the foraging problem as outlined in [4].

Foraging is the act of searching for and collection (or capturing) of food for storage and consumption [4]. Foraging is an important problem that was first addressed by

biologists in the examination of nature, particularly with the foraging behaviour of ants and bacteria. In a robot context, foraging is defined as the search and collection of scattered objects in an environment and returning those objects to a collection point.

Foraging sub-tasks include the efficient search and collection of food, homing whilst carrying food to the nest site, and then depositing of the food item in the nest before returning to the foraging task. Efficient foraging requires indirect or direct co-operation between individuals in order to transport food items too large for individual transport. The foraging process has been adapted for numerous foraging problems and for different types of robots.

3.2 Foraging in Nature

As with many swarm robotics problems, the inspiration for foraging comes from nature, in particular, social insects such as ants and honey bees. This section describes the biological inspiration for foraging. Section 3.2.1 describes foraging in ants, while foraging activities of bees are discussed in Section 3.2.2. Section 3.2.3 discusses other types of foraging techniques seen in nature.

3.2.1 Ant Foraging Behaviour

As with many swarm robotics problems, the inspiration for foraging comes from nature, in particular, social insects such as ants and honey bees. This section describes the biological models that are used in the algorithms derived by this thesis.

The study of ants has revealed that impressive emergent activities can be achieved with simple interacting agents with only a few simple rules. Many ant species use a form of communication known as stigmergy [125]. Ants perform indirect communication between ants by depositing a substance known as pheromone. In foraging, pheromone is deposited on the paths between the nest and the food source. Other ants detect the deposited pheromone and will prefer to follow paths which have a greater amount of pheromone deposit. Ant pheromones have been modelled in numerous swarm intelligence algorithms such as ant systems [126, 127].

Although algorithms based on ant foraging behaviour are used to solve optimization problems, the algorithms are often difficult to replicate in a real-life robot environment, since most of the algorithms need to model pheromone dropping and detection. A robotics algorithm that uses pheromone requires robots to be equipped with a substance-distributors, beacon-deployers or complex communication that simulates pheromone deposition and trail-following [6].

The desert seed-harvester ant (*Pogonomyrmex ant*) does not make use of stigmergy to forage, because pheromone deposited on desert sand would be blown away by the wind [128, 129]. Instead, desert ants use a technique called path integration (PI) for navigation to relocate food sources that have already been found by random exploration [130, 131]. Fig 3.1 demonstrates path integration. The black solid lines represent the path of the ant and the blue dotted lines show how the direction to the nest is maintained as the ant explores. The ant is constantly monitoring the change in heading from the original heading such that, at the destination, the ant has a direction directly back to the sink. As a result, a shortcut back to the nest is calculated in order to minimize the heat stress caused by walking through the hot desert. The shortcut back to the nest is known as the home vector [132]. Path integration, more commonly known as dead reckoning, is a key evolutionary aspect gained from living in the hot barren desert. When path integration fails, the ant resorts to using landmarks, such as the sun, for navigation [130].

Due to the lack of pheromone, the desert seed-harvester ant was simpler to model for real-robot interaction. Desert seed-harvester ant foraging has been modelled in [133, 134]. Hecker *et al* [134] performed experiments to compare the performance of two foraging algorithms: The one algorithm was based on desert ant foraging which has no pheromone-like communication and the other algorithm included pheromone-like communication. It was shown that communication improved performance; however, the desert ant based algorithm still performed comparatively well.

3.2.2 Bee Foraging Behaviour

Bees have an impressive set of abilities considering the simplicity of a single individual. They have the ability to remember the colour and shape of flowers [135], and in terms of navigation, they are able to learn local features and routes due to well-developed learning

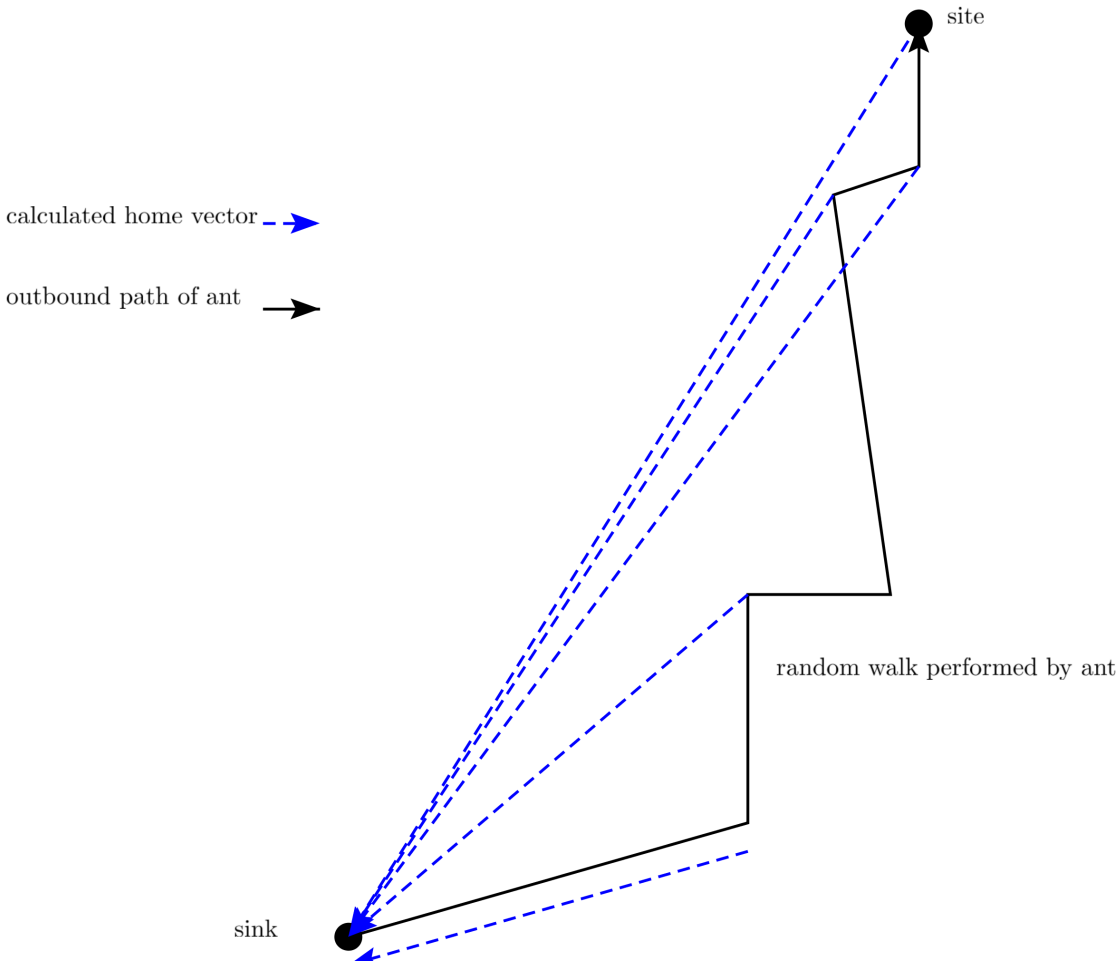


Figure 3.1: Path Integration

and memorizing capacities [136]. Honey-bees are even time aware [137].

Honey bees have efficient division of labour between different functions in the hive, such as foraging and brood-care. Within the foraging activity itself, honey bees perform foraging-specific division of labours. Honey bee foraging is made up of three different roles namely employed foraging, unemployed foraging, and scouting [32]. Scouts explore the environment to locate new food sources. Once a source has been found, scouts return to the hive to communicate information about the located food source. To communicate the foraging information, scout bees perform a waggle-dance at the hive. The waggle-dance communicates information about the distance and bearing of the resource [138].

Recruitment [32] refers to the process of communicating high quality foraging sites to the swarm.

Unemployed foragers wait on the dance floor, and evaluate the dances of the scout bees. An unemployed bee selects a location described by the scout bees, and become an employed forager. Employed forager bees use the information about the resource, attempt to locate the source, load themselves with food, and return to the hive where unemployed foragers are ready to offload the food. Jansen [139] suggests that unemployed bees become exploring scouts when they do not detect any dancing scout bees.

3.2.3 Other Foraging Behaviours

This thesis focuses on ant and bee foraging behaviours. Other foraging behaviours exist, some of which are briefly discussed in this section.

The *E.coli* bacteria forage for carbohydrates in the human gut. Bacteria have a unique search mechanism whereby they move in the same direction for a period of time (which is called a "run") and then change direction using their complex flagella (known as a "tumble"). Bacteria exhibit the behaviour of continuing to move in a direction if the nutrient gradient is positive in that direction. Bacteria stop moving in a direction if the nutrient gradient of that direction is negative in order to avoid toxic substances. Under certain conditions, the bacteria secrete an attractant substance which attracts the bacteria to one another for protection. The foraging behaviour of *E.coli* bacteria inspired a computational intelligence algorithm [140].

In nature, a predator animal persues a prey animal in order to forage. Predator-prey foraging behaviour has been extensively studied in the field of optimal foraging theory, which aims to predict the behaviour of an animal searching for food [141]. Predator-prey models have been used in other swarm intelligence techniques in multi-objective optimization problems [142].

Particle swarm optimization is loosely inspired by the flocking behaviour of birds when foraging for food [143].

3.3 Foraging in Swarm Robotics

This section addresses foraging in a swarm robotics context. Firstly, a taxonomy of robot foraging is presented in Section 3.3.1 and then existing nature-inspired swarm robotics foraging algorithms are discussed in Section 3.3.2. The chapter concludes with a discussion of the challenges faced by swarm robotics in Section 3.3.3.

3.3.1 Taxonomy of Robot Foraging

Foraging is a popular problem due to the vast potential opportunities of foraging for application to a variety of real-world problems, from search and rescue [123] to toxic waste retrieval to mining. It is beneficial to present a taxonomy of robot foraging that can be used to classify existing research about foraging as well as to contextualize the specific foraging problems addressed by this thesis. One should note that the foraging taxonomy presented also includes non-swarm robotic foraging.

Multiple taxonomies have been developed as robot foraging research has progressed [144, 145]. However, a more recent and complete foraging taxonomy is presented in [4]. The taxonomy classifies foraging solutions based on four major axes: the foraging environment, the capabilities and types of robots used for foraging, performance aspects, and the foraging strategy used in a foraging algorithm. Each of these major axes has a number of minor axes which are outlined in Table 3.1. The following sections discuss each of the major and minor axes in more detail.

Table 3.1: Winfield's Robot Foraging Taxonomy

Major Axis	Minor Axis	Value
Environment	Search space	Unbounded Constrained
	Source Areas	Single limited Single unlimited Multiple
	Sinks	Single Multiple

Major Axis	Minor Axis	Value
	Object types	Single static Multiple static Single active
	Object placement	Fixed known locations Uniform Distributions Clustered
Robot(s)	Number	Single Multiple
	Type	Homogenous Heterogenous
	Object Sensing	Limited range Unlimited range
	Localization	None Relative Absolute
	Communications	None Near Infinite
	Power	Limited Forage Unlimited
Performance	Time	Fixed Minimum Unlimited
	Energy	Fixed Minimum Unlimited

Major Axis	Minor Axis	Value
Strategy	Search	Limited range Geometrical pattern Trail following Follow other robots In teams
	Grabbing	Single Cooperative
	Transport	Single Cooperative
	Homing	Self navigation Home on beacon Follow trail
	Recruitment	None Direct Indirect
	Coordination	None Self-organized Central control Master slave

Environment Axis

The environment axis is divided into minor axes as follows:

- **The type of search space**: The type of search space can be either unbounded or constrained by a boundary wall. Goldberg and Mataric [146] implement a foraging method for an unbounded environment. Schneider [147] develops a foraging algorithm where a known constrained environment is split into bounded areas. Each robot is assigned an area to forage.

- **Item sources**: Item sources can be of different types and quantities. A single limited source is a source where all items come from the same source and the

source has a limited capacity for items [148]. A foraging environment with a single unlimited source has one source that an unlimited number of items can be foraged from. A foraging environment can also have multiple sources, such as in [146].

- **The number of sinks**: The environment can have a single or multiple sinks.
- **Item distribution**: Item distributions in the environment can vary. Items can be placed in the environment, in known locations, uniformly distributed, or in clusters.
- **Item types**: Items can be classified into types in terms of their activity (active or static), homogeniety (homogenous or heterogenous), and quantity (single or multiple) [149, 62, 115].

Robot Axis

The robot axis refers to qualities such as the number of robots and the sensory, power, and actuation capabilities of the robots, discussed as follows:

- **The number of robots used in the foraging problem**: If only a single robot or few robots are used to forage, then that would not be considered swarm robotics.
- **Type**: Robots can either be all identical (homogenous) or have different capabilities (heterogenous). Homogenous swarms are the more common type of robots in swarm foraging, but research using a heterogenous swarm does exist. In particular, the Swarmanoid project uses a heterogenous swarm to forage an item [150]. The Swarmanoid swarm consists of three types of robots with three separate responsibilities: the eye-bots locate the item, the foot-bots transport the hand-bot to the item, and the hand-bot climbs until it can grab the item.
- **Object Sensing**: The sensors of the robots can be of a limited range or global sensors with unlimited range. Foraging research that uses global sensors is not considered swarm robotics. Therefore, foraging algorithms using global sensors are not addressed in this literature study.

- **Localization** refers to the robot's ability to position themselves in the environment. Localization could be non-existent, relative, or absolute. Absolute localization, such as a global positioning system (GPS), is not considered swarm robotics because the swarm will be dependant on a centralized source of information, violating the swarm robotic principle of decentralization. Relative positioning is the use of local information in order for a robot to position itself in an environment. Relative positioning is suited to swarm robotics foraging problems, since relative positioning is decentralized. Relative localization techniques such as path integration are used in this thesis.

- **Communication capabilities** of the robots to each other can have an infinite range, a near range, or not exist at all. Infinite range communication is out of scope for swarm robotics research since local communication is a requirement of swarm robotics. Near range communication can occur via light-based signals [148] or local direct communication methods such as Bluetooth. The communication occurs in differing topologies such as broadcast, direct, tree, or graph topologies [151].

- **Power**: Robots can have an infinite power source, a limited power source, or an energy source which can be sustained by foraging resources. Studies focused on energy efficiency or automatic-charging have robots with limited energy sources or an energy source that can be sustained by foraging energy resources [152].

Performance Axis

The performance axis characterises robot foraging on the methods used to evaluate the performance of a foraging algorithm, such as energy usage, or the time taken to forage each item. Each minor axis is described as follows:

- The performance of a foraging algorithm can be classified in terms of the time taken to forage. A foraging experiment can be allowed an infinite time to forage, a fixed time for foraging, or may attempt to minimize the time taken to forage.

- The energy that a robot can spend on foraging can be a fixed amount of energy,

an unlimited amount of energy, or a foraging algorithm can attempt to minimize the energy used to forage [152].

Strategy Axis

The strategy axis characterizes algorithms by the techniques that a foraging algorithm implements, such as the type of search used, or the type of recruitment technique used in a foraging algorithm. Each type of strategy is discussed as follows:

- The first stage of most foraging algorithms is the search for items. There are various techniques for searching for items, such as random search, trail following [153], following other robots [150, 95], or groups of robots looking for items together [154].

- Once an item is found, it must be picked up. The techniques used by a robot for grabbing an item can be used to classify the algorithm. An item can be grabbed by a single robot or a group of robots can attempt to co-operatively grab an item. Closely related to grabbing an item is the techniqe for transorting an item to the sink. Transport of an item back to the sink can be performed by a single robot or by a a group of co-operating robots. A review of co-operative transport and grabbing techniques is presented in [155].

- In order for a robot to move from its current location to the sink, the robot can employ one of many homing techniques. Examples of homing techniques include beacon homing (where the robot first changes its direction to face a beacon at the sink, and then move towards the beacon), following a trail or odometry [156].

- Recruitment has been defined as activities that bring individuals to a location where work is required [120]. Robots can recruit other robots to forage certain areas. Direct recruitment is where a robot explicitly communicates to another robot to forage an item source [157, 17]. Indirect recruitment occurs when a robot is recruited to forage by implicit markers, such as an increase in the number of foragers waiting by the sink, or an increase in the density of other foragers moving towards a particular source of the environment [158].

- There exist various techniques for coordination of tasks between the robots in a swarm. A centralized control mechanism can be used for controlling all robots. Centralized control is not considered to be swarm robotics, and thus foraging algorithms that use centralized control are not addressed in this thesis. Master-slave configurations, where one robot leads the others, is another option for coordination of robots. A master–slave configuration with the ability to select a new master in the case that a master is destroyed, is considered as swarm robotics, since the swarm is then robust to individual failures [159, 160]. A swarm with a master–slave configuration that does not have the ability to select a new master is not swarm robotics. Swarms can have self-organized coordination, which means that each robot is in control of its own activities, and the self-organised coordination is an emergent property of the swarm.

3.3.2 Nature Inspired Swarm Robotics Foraging Algorithms

Foraging is an animal activity performed in order to retrieve food. Social insects, in particular, have found very efficient means of foraging despite their individual simplicity. This section discusses the swarm robotics algorithms that have specifically been inspired by social insects, in particular, those models inspired by ant foraging and bee foraging.

Ant Inspired Foraging Algorithms

Vaughan developed a swarm robotic algorithm allowing for robust transportation of items from a single source to a single sink in an environment with spatial constraints [161]. The algorithm presented makes use of both ant-like and bee-like foraging techniques. The ants broadcast the landmarks in the area for odometric localization as well as uses a form of the honey bee "waggle dance" that communicates the direction of the food source from the robot that has found the food source to other robots. The robots use path integration, utilized by both ants and bees in order to maintain position and heading estimates. Using these techniques, the robots can communicate multi-segment paths to item sources. The dance communication occurs globally so that every robot knows the multi-segment path to the item source. The path integration technique suffers

from accumulation of localization errors over time, which is worsened by global communication, since localization errors experienced by an individual robot get communicated to every single other robot.

Hoff [6] classified different types of ant-inspired pheromone-based foraging algorithms by how the pheromones are represented. Pheromones can be represented as physical marks on the environment [162] or by sharing data that represents pheromones over wireless networks.

Labella *et al* [17] proposed a foraging model inspired by the foraging behaviour of ants, where individual robots adapt to the environment using only locally available information, in an attempt to preserve energy. The probability, p_1, that a robot will change from rest to searching is increased by a constant when the robot returns to the sink and deposits an item. If a robot fails to forage any item after searching for a specified maximum time, then the foraging attempt is considered a failure, p_1 is decremented by a constant, and the robot returns and rests. The study showed an improvement in energy efficiency and the algorithm was able to regulate the number of active foragers. An advantage of this algorithm is its simplicity, since communication is required to regulate the number of robots. The study also showed that there was a global negative effect on overall foraging efficiency at large swarm sizes, despite an increase in energy efficiency.

Hoff [6] proposed two ant-based foraging algorithms which do not require marking of the physical environment. Instead, robots themselves become pheromone beacons. The purpose of pheromone beacon robots is to act as pheromone. When many pheromone beacon robots line up, the robots form a pheromone trail that other robots can follow to reach a destination. The algorithm enabled the creation of pheromone trails from the sink to the item source and back.

Honey Bee Inspired Foraging Algorithms

Bee swarms have been used in swarm robotics for problems such as path planning [163], aggregation [164], and collective perception [165].

A foraging algorithm was developed in [166] that simulated simplified honey bee recruitment strategies. The algorithm consisted of two phases: an exploratory phase followed by an exploitive phase. Initially, all robots are located around the sink. In

the exploratory phase, all robots performed a random search using a Lévy distribution until an item source was found by a robot. The exploitation phase began when the robot that located the item returned to the sink and communicated the location to other members of the robot swarm using wi-fi. The other robots were recruited and began to commute between the hive and the item source, transporting items back to the sink. This algorithm exhibited three behaviours of bees: waiting around the sink, exploring the environment, and foraging. Unlike bees, the algorithm did not consider the division of labour between foragers, explorers, and waiting robots. The robots used path integration in order to remember the location of the sink source, and used beacon homing in order to locate the sink. The path integration vector was communicated to the other robots using wi-fi, which is used to guide them to the sink source.

3.3.3 Challenges in Swarm Robotic Foraging

Foraging for humans is a near trivial problem. Most children can walk around and retrieve blocks and return them to a single source. The seemingly simple sub-activities required by foraging such as identification and grasping are often relatively complex for robots to perform. Unlike other simpler swarm robotics problems, such as aggregation, foraging is made up of a group of non-trivial sub-tasks. Some of the aspects which are problematic for robots in foraging are discussed below:

- **Mechanically challenging interactions**: Despite how far robotics has come as a field, there are still tasks that are trivial for humans, but still very complex for most robots. For instance, the challenge of picking up items [167] and moving them to another location requires high quality sensors, sensitive actuators, and time-consuming calibration [168].

- **Complex noisy environments**: Although research can be performed in a controlled environment, for robust foraging to be efficiently applied in real-life, environmental factors have to be taken into consideration. Environmental factors, such as light [169, 170] and quality of the surface the robots forage on [171], often make a robotic solution complex, non-viable, non-robust, or expensive. To cope with real-life environments, the robots require adaptive algorithms and more complex

hardware for their sensors and actuators for basic navigation and object detection in unknown terrains with unknown lighting and weather conditions.

- **Localization, navigation and obstacle avoidance**: Localization is the ability for a robot to depict its position in the environment. In swarm robotics, global knowledge about location (such as GPS) can not be used by a robot to determine its position and where it is going. Non-trivial algorithms must be designed in order for a robot to position itself in the environment as in [172, 173, 158].

Swarm robotics researchers often choose to use simulated robot platforms, such as Stage [56] or Argos [57], instead of real robots in order to remove or regulate the amount of environmental noise. If a swarm robotics experiment uses real-life robots, the environments are usually simplified by creating an environment where temperature, lighting, and moisture are regulated [17, 96]. Due to the discussed complexities of foraging, existing swarm robotic research often simplifies some of the complexities such as environmental changes, sensor noise, and the use of physical robots in order to focus on a single aspect of foraging.

3.4 Summary

This chapter reviewed foraging in nature and in swarm robotics. In nature, social insects have efficient emergent foraging behaviours. Ants generally lay pheromone to guide the item search process, but the desert ant does not use pheromone, but instead uses path integration to relocate a food source. Honey bees employ more complex foraging behaviour including division of labour and communication.

Winfield's foraging taxonomy was presented and discussed so that research can be contextualized in the field. Lastly, existing swarm robotics foraging algorithms that are based on ants and the honey bee were discussed.

Chapter 4

Division of Labour

As explained in Section 2.2.2, division of labour is a strategy that is used by social insects. This chapter defines division of labour and discusses different types of division of labour that occur in social insects. Lastly, division of labour strategies employed by different swarm robotics algorithms are described.

4.1 Definition of Division of Labour

Oster *et al* [144] defined division of labour as a "stable pattern of variation" of the repertoire of tasks that individuals perform. Each individual specialized on a subset of the complete repertoire of tasks. The subset of the complete repertoire of tasks was called the specialization of an individual and the specialization of individuals varied across the swarm.

Robinson *et al* [174], more simply, defined division of labour as the adjustment of ratios of workers engaged in different tasks based on external and internal stimuli.

There are two types of division of labour in social insects [15]:

- **Temporal polyethism**, where the pattern of tasks being performed by a worker correlates to the age of the worker. In nature, the younger workers may perform tasks within the nest while the older workers perform tasks outside of the nest.
- **Morphological polyethism**, where a worker with extreme physical features in terms of size or shape will specialize more in particular tasks. The more extreme

a particular physical feature is, the narrower the repertoire of the worker. For example, soldier ants are larger in order to defend the nest, while the smaller workers are involved with foraging.

4.2 Models of Division of Labour from Biology

An overview of existing division of labour models from entomology is provided in this section, since insect-like division of labour is used by the algorithms developed in this thesis. The section focuses on common models and models used by the algorithms described in this thesis. Section 4.2.1 discusses the response threshold model, while foraging for work is presented in Section 4.2.2. Self-reinforcement models are presented in Section 4.2.3, while Section 4.2.4 discusses social inhibition models. Finally, Section 4.2.5 discusses network models for division of labour.

4.2.1 Response Threshold Model

The response threshold model gives each individual of a swarm a response threshold for each task that can be performed. The task-specific thresholds vary across the swarm [175].

Suppose R is the set of tasks that can be performed by individuals of the swarm. An individual γ will only perform task $\nu \in R$ when the environmental stimuli for task ν, S_ν, exceeds the response threshold, $r_{\gamma,\nu}$. Each response threshold for a task, for an individual is constant. If an individual has a lower threshold for a specific task, compared to other individuals in the swarm, then it is likely that the individual will become a specialist in that task [175].

Response threshold models have been modelled formally by Page *et al* [176] and Bonabeau *et al* [177]. Response thresholds have explained temporal polyethism in honey bees where the response thresholds of the honey bees changed as the honey bees aged [178].

4.2.2 Foraging for Work

Foraging for work assumes that individuals are intrinsically identical and thus performance of individuals at a task is dependent on opportunity to perform a task, rather than intrinsic task preferences [179].

Foraging for work has two main principles:

1. Individuals repeat the same task when possible.

2. Individuals actively seek work when they have no task.

Foraging for work also assumes that tasks are radially spatially localized within the nest [180]. Foraging for work shows that temporal polyethism does not require age-related differences in the mechanism of task choice and can simply stem from an individual's proximity to the nest. Foraging for work is controversial since it shows that task organization might occur within a social group in absence of selection efforts or an intrinsic mechanism of task performance [179].

Foraging for work can be seen as a special case of the threshold model where all individuals have identical thresholds. In foraging for work, temporal polyethism is generated by spatial organisation, where as the response threshold model generates temporal polyethism by differences in internal thresholds. Unlike threshold models, foraging for work does not suffer from inactive individuals as all workers are either performing or seeking tasks [15].

Foraging for work is controversial in the fact that it induces division of labour without any explicit mechanism, but instead as a natural result of the environment [15]. However, it has been shown that, in insect colonies, there is intrinsic variation in each individuals' response to environmental stimuli, rather than from opportunity alone [181].

4.2.3 Self-Reinforcement Models

Self-reinforcement models adjust an individual's probability of performing a task, based on prior success or failure to perform that task. Initially, the probability of performing all tasks are equal. If a task is performed successfully or there is no opportunity to perform a task, then the probability of performing that task again is reduced. Individuals that

continuously successfully perform a specific task become specialists in that task. Task success is directly proportional to the probability of doing the task again [182, 183]. Self-reinforcement has been attributed to division of labour in many systems in nature, such as ants and bees [184].

4.2.4 Social Inhibition Models

With social inhibition models, individuals change tasks as they age. However, the process of aging is inhibited by social environments [185]. The model is based on the activator-inhibitor behaviour present in bee swarms.

In honey bee swarms, juvenile bees work in the hive while older bees forage. When a juvenile bee ages, it transitions from a worker bee into a forager bee. The rate of growth of the juvenile bee is determined by two hormones: the activator hormone and the inhibitor hormone. The activator hormone promotes the growth of the juvenile bee into a forager bee. The activator hormone is released by the juvenile bee. The inhibitor hormone inhibits the growth of a juvenile bee into a forager bee [185]. The inhibitor hormone is released by forager bees. The inhibitor hormone counteracts the effect of the activator hormone.

The existence of many forager bees in the hive means that inhibitor hormine would be released. Thus, the growth of the juvenile bees would be slowed, and therefore juvenile bees remain worker bees. If forager bees are destroyed, then less inhibitor hormone would be released and more juvenile bees would becomes forager bees, until the number of forager bees have been replenished.

4.2.5 Network Models of Task Allocation

Network models assume that swarm individuals are identical, and that division of labour is induced by effective communication between individuals of the swarm, about the number of individuals that are required per task [186]. A number of network models exist [186, 187].

Network models are similar to foraging for work models in that division of labour is generated by changes in local information encountered by each individual, rather than

by intrinsic differences in individuals.

4.3 Division of Labour in Robot Swarms

Division of labour has been used extensively in swarm robotics. This section provides an overview of the use of division of labour in swarm robotics.

Agassounon *et al* [188] designed and demonstrated three response threshold-based methods for division of labour. The robots had to perform a clustering task with multiple object sites. Division of labour techniques were employed to avoid inter-robot interference that occur more often when too many robots attempt to cluster the same site.

The experiment determined that the robot swarm benefited from threshold division of labour since the robot swarms that employed division of labour performed similarly or better than robot swarms that set a fixed task group size per site. The experiment also determined that local communication improved swarm performance.

Labella *et al* [17] developed a division of labour strategy for prey retrieval based on Deneubourg's ant self-reinforcement model, discussed in Section 4.2.3. Experimentation was performed on simulated and real robots. The robots switched from search behaviour to rest behaviour with probability z. This probability was incremented by a fixed value when a robot successfully returned to the nest with an item and decremented by the same fixed value when a robot returns to the nest without an item. Labella *et al* classified the foragers into different types, namely, loafers, foragers, or undecided, based on the final value of z. Experiments showed that the values of z tended towards extreme values, showing that robots tended to become either loafers or foragers, while only a few became undecided. It was shown that a simple self-reinforcement division of labour model can improve the performance of a swarm of robots at a task, without the need for communication. However, scalability was still a concern, since the swarm experienced negative performance as swarm size increased.

Liu *et al* [18] introduced three mechanisms for adapting the number of resting robots to foraging robots in a simple foraging problem. The mechanisms consisted of variations on how a robot perceives the worker demand. The mechanisms were used to adjust time

specific thresholds related to the length of time to wait before returning to work and how long work should be performed for. The mechanisms were as follows:

- The internal success of an individual at a specific task.
- The collective success of the swarm at a specific task.
- The amount of environmental interference experienced by an individual while performing a task, most importantly the number of collisions with other robots. This technique is a social inhibition model.

Four combinations of these mechanism were evaluated as well as compared to a naïve foraging approach. The performance of the swarm was measured as the total energy spent by the robot swarm. The efficiency was evaluated over different robot quantities and food densities.

The experiments discovered that the use of the mechanisms improved performance significantly compared to swarms without such mechanisms. It was also shown that the mechanisms resulted in emergent division of labour, regulating the number of foraging and resting robots. The experiments resulted in robustness in environmental changes related to the density of items. The experiments using collective success of the swarm achieved the highest net energy income to the swarm and also had the fastest adaptation of the ratio of foragers to resting robots when food density changes. Liu *et al* [18] noted that scalability was not tested in the experiments.

The majority of research in division of labour for swarm robotics was concerned with regulating the number of active robots in a swarm. The ability of a swarm to adjust the number of individuals actively participating in a task with the number of robots at rest enabled robot swarms to adapt to changes in the environment. Existing research also concluded that division of labour improved swarm robustness, by replenishing the number of active robots when active robots are destroyed.

Existing research did not test scalability of the division of labour techniques, since the maximum swarm size was usually ten or less individuals. This thesis will evaluate the scalability of the division of labour techniques used. There exists scope for investigating other types of division of labour in swarm robotics, such as network models, but this is not within the scope of this thesis.

4.4 Summary

Division of labour is the adjustment of ratios of workers engaged in different tasks, based on external or internal stimuli. Division of labour is used in social insect societies such as bees and ants. Multiple models for division of labour have been explored by biologists. Extensive research has been performed about response threshold models, foraging for work, network models and self-reinforcement models. Despite the large number of models, little verification of these models have been performed in real or simulated environments.

The use of division of labour in swarm robotics was reviewed. Most of the reviewed research focused around the problem of using division of labour to regulate the number of active robots to the number of inactive robots. The techniques used in swarm robotics employ a variety of division of labour models such as response threshold models, self-reinforcement, and social inhibition models.

The author notes that the reviewed research did not adequately address the scalability of the division of labour techniques.

Chapter 5

Nature inspired algorithms for prioritized foraging

This chapter presents a foraging variation known as prioritized foraging as well as three algorithms whose performance is to be evaluated on a prioritized foraging problem. The three algorithms are based upon phenomena observed in nature. The first model is a simple foraging algorithm, called naïve foraging, which will form the benchmark algorithm for the experiments. Two novel swarm robotics foraging algorithms, inspired by the foraging behaviour of desert ants and honey bees, are also presented. Each of these algorithms have different capabilities when it comes to memory, communication, division of labour, and navigation. The naïve foraging algorithm is presented in Section 5.2. Section 5.3 introduces the novel foraging algorithm based on desert ants. Section 5.4 presents a novel prioritized foraging algorithm inspired by honey bees. The algorithms are summarized in Section 5.5.

5.1 Prioritized foraging

The prioritized foraging problem, illustrated in Fig.5.1, is a modified version of the multi-foraging problem as discussed in Section 3.3.1. In prioritized foraging, an environment has two types of items: prioritized items and non-prioritized items. The goal is to forage all the items of the prioritized type. The possibility exists that prioritized items

become trapped among non-prioritized items and thus the non-prioritized items need to be removed from the environment to clear an access route to the prioritized items.

The prioritized foraging problem has increased difficulty compared to traditional multi-foraging problems, due the fact that foraging the non-prioritized item more than required will result in a waste of time and energy. The goal of research in prioritized foraging is to develop an algorithm to efficiently adapt the number of robots searching for prioritized items to those moving non-prioritized items out the way.

The prioritized foraging problem has similarities to the problem of search and rescue. For example, in the case of a collapsed building, robots will need to get to the survivors as quickly as possible. However, it is important that some robots move waste material in order to reach the trapped survivors. Prioritized foraging could be applied to the gold mining problem where gold needs to be foraged as a priority, and waste needs to be moved out of the way.

As per Winfield's classification discussed in Section 3.3.1, the prioritized foraging problem has the following characteristics: The environment has a bounded search space, with multiple source areas for items which can be placed in multiple sinks. Multiple object types exist in the environment - the prioritized items and the non-prioritized items. The primary performance measure for the prioritized foraging problem is time, in terms optimizing the time taken to forage each type of item.

5.2 Naïve foraging algorithm

Naïve foraging consists of the following tasks: searching for an item, grabbing an item, returning home with the item, and storing the item at the sink. The steps performed by the algorithm are illustrated in Figure 5.2, and described below:

1. Robots perform a random walk until they find an item.

2. On locating an item, the robot grips the item. If the item has been moved before the robot is able to pick it up, the robot will continue to explore; otherwise, the robot returns the item to the correct sink using a beacon-based homing algorithm.

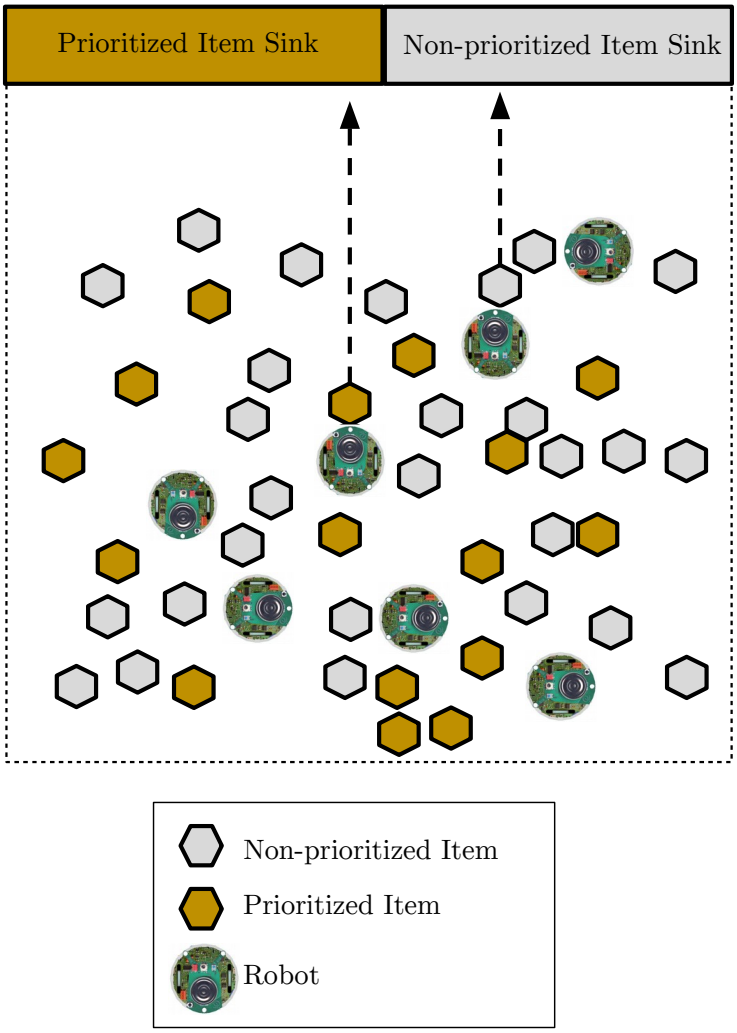


Figure 5.1: Prioritized Foraging Problem

Naïve foraging includes only the most minimal set of foraging actions and is included as a baseline for comparison to evaluate how novel techniques, such as the desert-ant foraging or honey-bee foraging, compare to a standard model [145, 6].

The following random walk is used: A robot chooses a random direction, σ, and a random distance, $m \in (0, M)$, where M is a chosen maximum path length. Both σ and m are selected from a uniform distribution. The robot walks in direction σ for distance m, or until the robot reaches a boundary. The robot then chooses new values for σ and

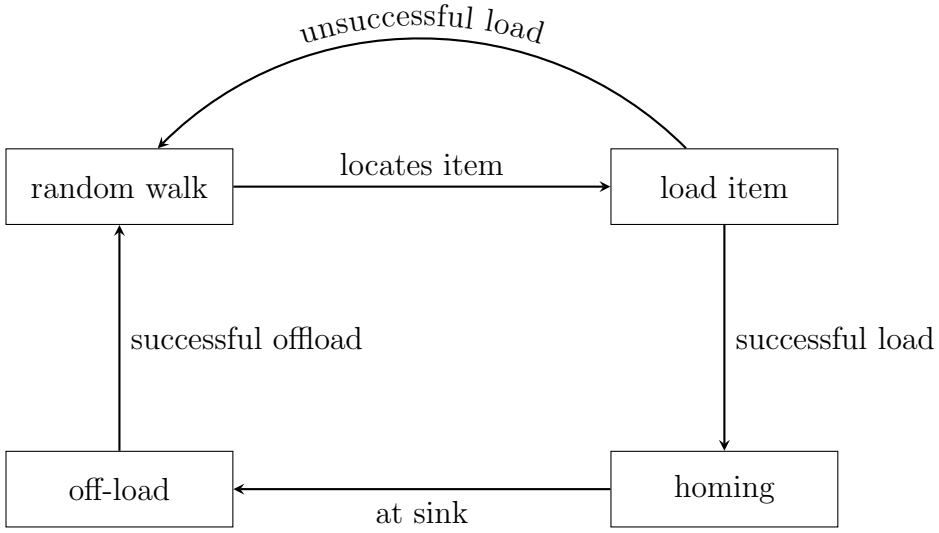


Figure 5.2: Naïve Foraging State Diagram

m.

5.3 Desert ant foraging

As discussed in Section 3.2.1, desert ant foraging behaviour is a very suitable model for robot foraging, since no pheromone depositors or pheromone mimickry is required. Pheromone depositors or pheromone mimickry are quite complex to perform in robot environments. Instead of pheromone, desert ants use PI to memorize the location of an existing food source and later to return to the memorized source to find more food, as discussed in Section 3.2.1. The notion of returning to a previously explored site is known as site fidelity [189]. The desert ant algorithm does not require communication between robots or the dispersal of beacons, and is thus simpler than many other swarm robotic foraging algorithms. The desert ant algorithm was defined by Hecker *et al* [134]. The desert ant foraging robots can be in the following states:

1. **Exploration State**: A robot in the exploration state performs a random walk with PI. The random walk used is the same random walk as discussed in Section 5.2. The purpose of the exploration state is to explore the environment to locate items.

2. **Loading State**: On finding an item, the robot switches into the loading state. In the loading state, the robot loads the item and memorizes the current PI vector. The PI vector is memorized so that the robot can use it to return to the sink and then later to return to the site where the item was found. If loading of the item fails (perhaps due to another robot loading the item), then the robot returns to the exploration state; otherwise, the robot moves into the homing state.

3. **Homing State**: In the homing state, the robot uses the PI vector to move to the sink. The use of the PI vector will enable the robot to follow the most direct route back to the sink.

4. **Offloading State**: When the robot arrives at the sink, the robot switches into the offloading state, where the robot simply offloads the item into the sink.

5. **Locating State**: Once the robot has offloaded the item, the robot switches to the locating state. In the locating state, the robot follows the memorized PI vector to the location of the previous item. The premise of returning to the site where the previous item was found is that more items may exist where the previous item was located in order to locate more items. If another item is found, the robot moves into the loading state; otherwise, the robot returns to the exploration state in the search of new items.

All robots begin at random positions adjacent to the sink in the exploration state. The desert ant foraging states and transitions are illustrated in Figure 5.3.

5.4 Honey bee foraging

The honey bee foraging algorithm presented in this section is based on the foraging behaviour of honey bees as described in Section 3.2.2. The algorithm described requires robots to take on one of three roles, namely, scout robots, unemployed forager robots, or employed forager robots. The roles of the robots and the transitions to and from these roles are described in this section.

Figure 5.4 provides a simplified diagram for the honey bee prioritized foraging algorithm, illustrating the roles and the transitions between each role and the states within

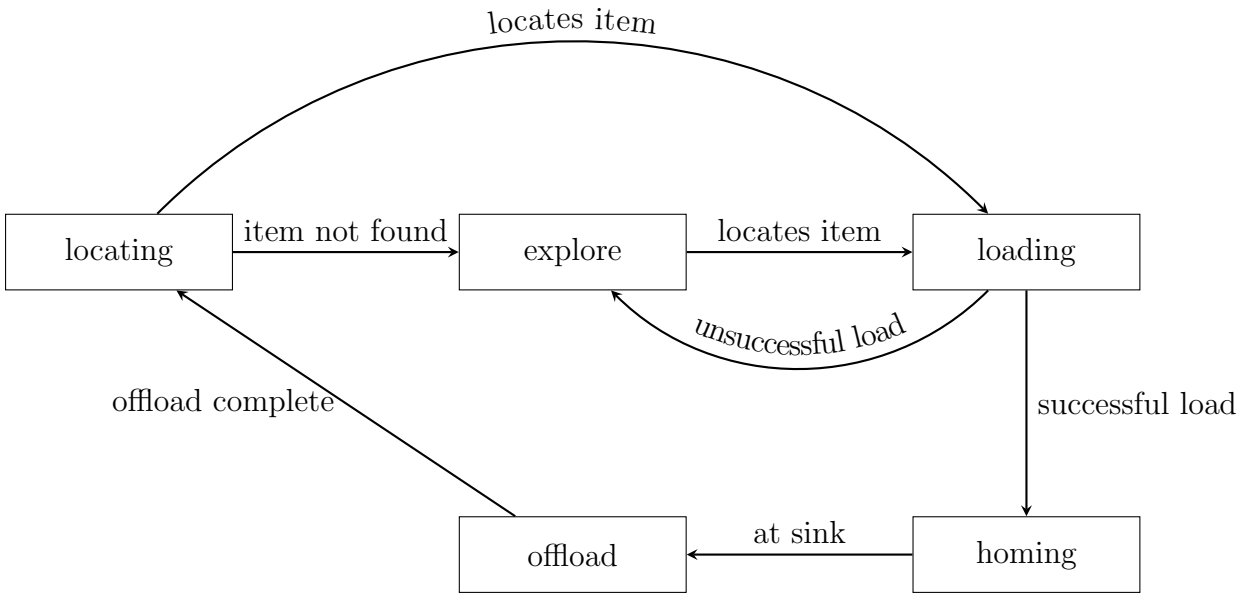


Figure 5.3: Desert Ant Foraging State Diagram

these roles. The dance and explore states of the scout robots are shown separately for clarity and the employed forager states are outlined separately in Figure 5.5.

To more succinctly define the honey bee prioritized foraging algorithm, the algorithms for each behavioural state, within each role, are provided below. For each algorithm, note that the current state of the robot is denoted *state*, and the *state* is modified during the behaviour to denote the next state of the robot. The current time step is denoted as i. The algorithm for a single state is executed once per time step i.

Section 5.4.1 presents the behaviour of scout robots, while the behaviour of forager robots is described in Section 5.4.2. The initial state of the swarm is described in Section 5.4.3, while Section 5.4.4 describes the division of labour mechanism used by the honey bee algorithm.

5.4.1 Scout Robots

Scout robots mimic the scouting behaviour of the scout honeys bees as discussed in Section 3.2.2. The purpose of the scout honey bees is to locate high quality sites of resources. If the discovered site is of a high enough quality, then the scout will broadcast

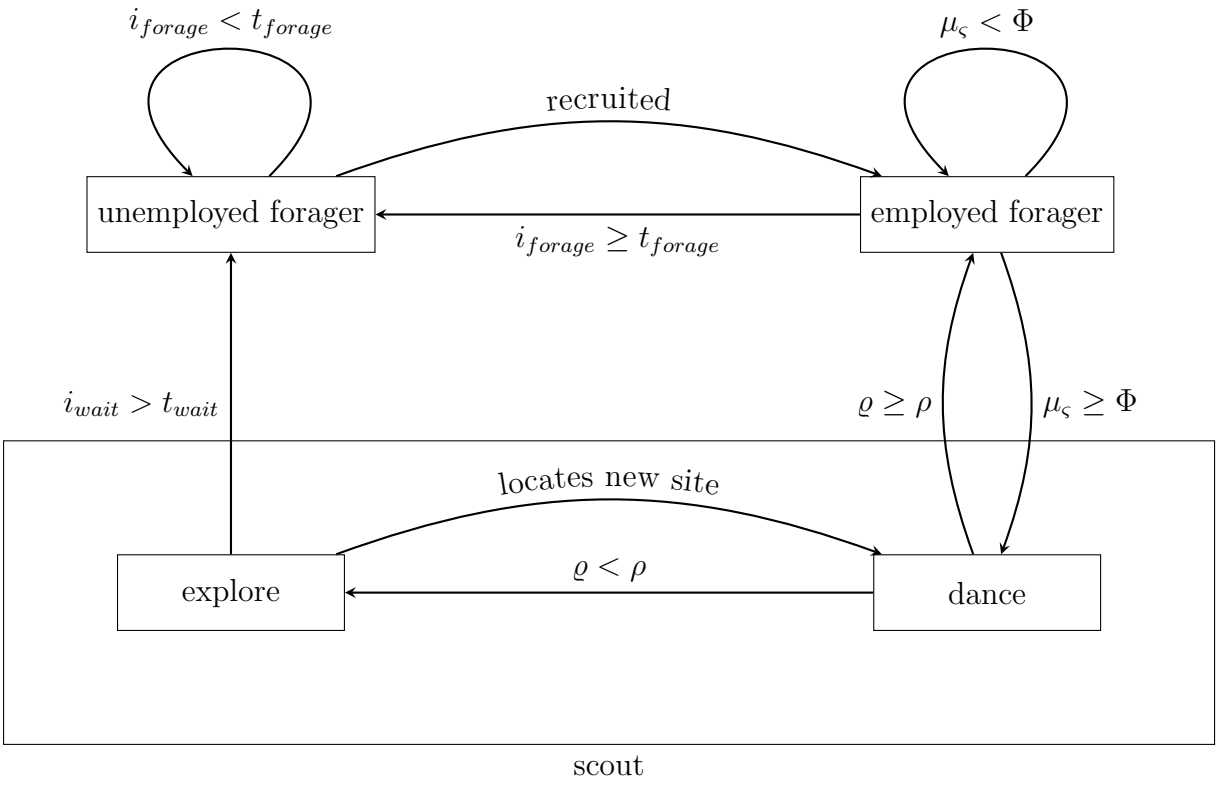


Figure 5.4: Honey Bee Foraging State Diagram

the location of the site to the unemployed forager robots at the sink.

Each scout robot begins in the explore state where the scout robot performs a random walk. Algorithm 1 provides a step-by-step explanation of the explore state of the scout robot. Variable ς saves the robot's item specialization as prioritized or non-prioritized. The random walk performed is the same random walk as discussed in Section 5.2. As the scout robot moves, the robot maintains a PI vector v as explained in Section 3.2.1. Upon finding an item ϑ of priority type ς at site ξ, the robot loads the item and then performs an evaluation of the quality, μ_ς, of the site ξ for the item type ς.

The quality, μ_ς, of site ξ, for a robot scouting items of type ς is calculated as the estimated density of items of type ς in the local vicinity of the found item ϑ. The robot has distance sensor values $k_j \in [0, 1]$ for $j = 1, ..., n$, where n is the number of distance sensors. A distance sensor reading of 0 means that nothing is detected in the sensor's range and a distance sensor reading of 1 indicates that the robot is touching an item.

Algorithm 1 Explore State of Scout Robot

function EXPLORE($role, state, v, \varsigma, i$)
    perform a single random walk step from the current location
    update path integration vector v
    if item ϑ of priority ς is detected in vicinity **then**
        calculate quality μ_ς of site ξ
        $\omega \leftarrow v$
        load item ϑ
    else if $i_{explore} > f_{max}$ and $i_{explore} \leq t_{explore}$ **then**
        $\varsigma \leftarrow N$
    else if $i_{explore} > t_{explore}$ **then**
        $state \leftarrow homing$
    end if
    $i = i + 1$
end function

The sensor value k_j for item type ς, denoted k_{j_ς}, is calculated as

$$k_{j_\varsigma} = \begin{cases} k_j & \text{if detected item is type } \varsigma \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

The site quality of type ς, μ_ς, is calculated using

$$\mu_\varsigma = \frac{1}{n} \sum_{x=1}^{n} k_{x_\varsigma} \tag{5.2}$$

Before returning to the sink with the item, the scout memorizes the PI vector v in site location variable ω. The scout robot then switches to the homing state given in Algorithm 2. Using PI vector ω, the scout returns the item ϑ to the sink. When the scout robot has returned to the sink, S_ς, the scout robot offloads the item ϑ. If the quality μ_ς of the visited site ξ is larger than site quality threshold, Φ, the scout robot switches into the dance state, which is summarized in Algorithm 3. If the quality, μ_ς, of the visited site ξ is less than site quality threshold, Φ, the robot takes on the role of an employed forager and switches into the locate state, as outlined in Algorithm 4.

Algorithm 2 Homing State of Scout Robot

```
1: function SCOUT HOMING(role, state, v, ς, i, ω)
2:     if robot is at sink S_ς and robot is loaded then
3:         robot offloads item ϑ
4:         if μ_ς > Φ and ς = prioritized then
5:             state ← dance
6:         else
7:             role ← employed forager
8:             state ← locate
9:         end if
10:    else
11:        calculate direction σ to sink S_ς from current location
12:        if robot can move in direction σ then
13:            robot moves a step in direction σ
14:        else
15:            robot does not move
16:        end if
17:    end if
18:    i = i + 1
19: end function
```

In the dance state, the scout robot communicates the location, ω, and quality, μ_ς, of the previous site, ξ, to the unemployed workers at the sink. The communication is akin to the waggle dance performed by honey bees in nature, as discussed in Section 3.2.2. A scout robot's "dance" takes the form of a localized broadcast communication between the scout and the unemployed forager robots near the sinks. The scout robot communicates the site quality and location for the unemployed foragers for t_{dance} time steps.

After the scout robot has completed broadcasting site details to the unemployed foragers, the scout robot must decide to either stay a scout robot and switch back to the explore state, or to become an employed forager robot and begin foraging the previous site. The scout robot becomes an employed forager robot with probability of ρ. If a

Algorithm 3 Dance State of Scout Robot

```
1: function DANCE(role, state, v, ς, i, ω, μ_ς)
2:     if i_dance < t_dance then
3:         broadcast ω and μ_ς to robots at the sink
4:     else
5:         ϱ = random(0, 1)
6:         if ϱ < ρ then
7:             role ← employed forager
8:             state ← locate
9:         else
10:            state ← explore
11:        end if
12:    end if
13:    i = i + 1
14: end function
```

random number $\varrho \in [0, 1]$ is sampled from a uniform distribution such that ϱ is less than ρ, then the scout robot remains a scout and begins to explore the environment. If ϱ is greater than or equal to ρ then the scout becomes an employed forager. Increasing the site quality threshold, Φ, will make the scout robots more selective about the sites they broadcast. Decreasing Φ will result in scout robots being less selective about the sites they broadcast.

5.4.2 Forager Robots

There are two types of forager robots: The unemployed foragers and employed foragers. The unemployed forager robots take on the role of unemployed foragers from foraging honey bees discussed in Section 3.2.2. Unemployed forager robots remain at the sink and await dance behaviour from a scout robot. This wait state is described in Algorithm 5.

A scout robot dances at the sink after locating an item ϑ at some site ξ. Each unemployed forager decides whether to listen to the scout robot with a probability of α. If an unemployed forager robot chooses to accept the details communicated by the

Algorithm 4 Locate State of Employed Forager

1: **function** FORAGE($role, state, v, \varsigma, i, \omega, \mu_\varsigma$)
2: get direction σ using path integration vector ω
3: robot moves a step in direction σ
4: **if** (robot has finished path integration) OR (robot can see item of type ς) **then**
5: $state \leftarrow load$
6: **end if**
7: $i = i + 1$
8: **end function**

scout, then the unemployed foraged robot has been recruited and becomes an employed forager robot. The unemployed forager takes on the same item specialization, ς, as the dancing scout robot. The unemployed forager will thus only search for items of type ς.

The employed forager robots are modelled based on the employed forager bees as discussed in Section 3.2.2. The purpose of the employed forager robot is to forage the sites communicated by scout robots. When an unemployed forager robot takes on the role of an employed forager robot after recruitment by a scout robot, the employed forager starts in the locate state, described in Algorithm 4. In the locate state, the employed forager robot uses the PI vector ω to relocate the site ξ. Once the site ξ has been relocated, the employed forager robot switches into the load state, as described in Algorithm 6. If an item ϑ of type ς is detected in the vicinity of the located site, then the item is loaded. After successfully loading the item, the robot switches to the homing state. If the employed forager robot does not detect an object in the vicinity of the site ξ, then the employed robot switches to the local search state in order to perform a brief local search for items nearby.

The local search is performed for a limited number of time steps, t_{ls}. At each time step, i_{ls}, the robot checks if an item of priority ς is nearby and if the item can be loaded. If an item an item of type ς can be loaded, then the robot loads the item and switches to the homing state. If the employed forager robot can see an item of type ς, but it is not close enough to be loaded, then the robot moves in the direction of the item. If the employed forager can not see an item in the vicinity, then the robot moves in a random

Algorithm 5 Wait State of Unemployed Forager

function WAIT($state, v, \varsigma, i, \omega, \mu_\varsigma$)
  if $i_{wait} \geq t_{wait}$ **then**
    $role \leftarrow scout$
    $\varsigma \leftarrow P$
    $state \leftarrow explore$
  else if scout robot is broadcasting at sink **then**
    receive site location ω and site quality μ_ς
    $role \leftarrow$ employed forager
    $state \leftarrow locate$
  end if
  $i = i + 1$
end function

direction, σ. The local search state is summarized in Algorithm 7. If i_{ls} is greater than t_{ls}, then the foraging site is depleted and so the employed forager robot must return to the sink without an item.

When in the homing state, the employed forager robot moves towards the appropriate sink, S_ς, in order to off-load item ϑ. Once at sink S_ς, if the employed forager robot is loaded, it offloads the item and switches back to the locate state and thus repeats the steps of foraging the site ξ. If the employed forager robot is not loaded, the employed forager robot takes on the role of an unemployed forager and switches into the wait state. The reason for switching from an employed forager to an unemployed forager is because an item could not be found at the site ξ, and the site has likely been depleted. Thus the employed forager switches to an unemployed forager in order to await recruitment by a scout robot.

The states and transitions of the employed forager are shown in more detail in Figure 5.5.

The unemployed forager robot role allows the number of active robots in the environment to be regulated so that there are not too many robots attempting to forage or explore at once. An environment with too many employed foragers would result in more

Algorithm 6 Load State of Employed Forager

```
1: function LOADING(role, state, v, ς, i, ω, μ_ς)
2:     if item ϑ of priority ς is detected in vicinity then
3:         LOAD(ϑ)
4:         state ← homing
5:     else
6:         state ← local_search
7:     end if
8: end function
```

collisions between robots, which would mean that the employed foragers take longer to forage items. Also, if there are too many employed foragers in an environment which is sparse in items, then the employed foragers are not only causing collisions but they are also wasting energy by exploring areas unnecessarily.

Unemployed forager robots become scout robots if no scout robot broadcasts are detected for t_{wait} time steps. The control parameter t_{wait} is the maximum waiting time an unemployed forager can spend in the waiting state before switching to a scout robot, and i_{wait} is the time spent by a robot in the waiting state. Decreasing t_{wait} results in more scout robots exploring the environment and less unemployed foragers that a scout robot, who may have found quality sites, can recruit. Increasing t_{wait} results in a greater number of unemployed forager robots waiting to be recruited. The greater number of unemployed foragers can form a large work force for a recruiting scout. However, too many unemployed foragers result in a smaller work force in the foraging environment.

5.4.3 Initial States

A portion of the robots are initialized as scout robots in the explore state and the rest as unemployed forager robots. All robots are initialized adjacent to the sink. The percentage of robots initialized as scout robots is $X \in (0, 100)$. Unemployed forager robots require a scout robot to recruit them to become employed forager robots. At initialization, the scout robots do not have site location details available, and therefore robots can not be initialized as employed forager robots.

Algorithm 7 Local Search State of Employed Forager

function LOCAL_SEARCH($role, state, v, \varsigma, i, \omega, \mu_\varsigma$)
  if $i_{ls} < t_{ls}$ **then**
    if item ϑ of priority ς is nearby and can be loaded **then**
      LOAD(ϑ)
      $state \leftarrow homing$
    else if item ϑ of priority ς can be seen but is not close enough **then**
      select direction σ to move towards item ϑ
      robot moves a step in direction σ
    else
      select a random direction σ
      robot moves a step in direction σ
    end if
  else
    $state \leftarrow homing$
  end if
  $i = i + 1$
end function

5.4.4 Division of Labour

The honey bee algorithm has two levels of division of labour. The first level is the division of labour between the scout, employed forager, and unemployed forager roles.

Another level of division of labour exists to deal with division of labour between foraging items with different priorities. Item-type division of labour is defined in this thesis as the division of labour between foraging prioritized item types and non-prioritized item types.

In nature, in times of drought, bees prioritize water over nectar or pollen. Honey bees would be sent out to forage for water but may encounter pollen while searching for water. If the foraging honey bee happens to encounter pollen, it will forage the pollen but will not communicate the discovery of the pollen site to the unemployed foragers [32]. Using the bee's prioritization of resources as inspiration, the following rules for

Algorithm 8 Homing State of Employed Forager Robot

```
1: function EMPLOYED FORAGER HOMING(role, state, v, ς, i, ω)
2:     if robot is at sink S_ς then
3:         if robot is loaded then
4:             robot offloads item ϑ
5:             state ← locate
6:         else if robot is not loaded then
7:             role ← unemployed forager
8:             state ← wait
9:         end if
10:    else
11:        calculate direction σ to sink S_ς from current location
12:        if robot can move in direction σ then
13:            robot moves a step in direction σ
14:        end if
15:    end if
16:    i = i + 1
17: end function
```

item-type division of labour are defined:

1. A scout robot that is set to search for the prioritized type (i.e. ς is set to prioritized items) will forage a non-prioritized type only if a prioritized item can not be located for the maximum time period, f_{max}. If $i_{explore} > f_{max}$ then ς will switch to foraging the non-prioritized type ς = non-prioritized. The described rule is given explicitly in Algorithm 1.

2. An employed robot foraging the non-prioritized item type will forage the non-prioritized item type until the robot fails to relocate a previously located non-prioritized item type site, or until the robot locates a prioritized item. For both of these cases, the robot will switch to foraging the prioritized item type ς = non-prioritized.

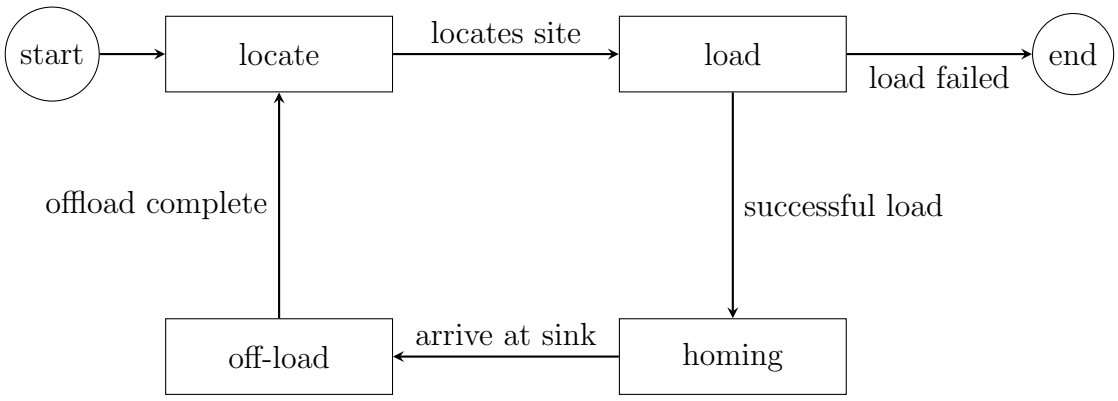


Figure 5.5: State Diagram of an Employed Forager Robot

3. A robot foraging a non-prioritized item will not communicate the location of the non-prioritized item site by dancing.

For the purposes of this study, each robot of each algorithm is assigned an initial item type to forage. The robots of the desert ant foraging algorithm and the naïve algorithm do not have item-type level division of labour and thus will continue to forage the same item-type that they were assigned throughout the experiment. The robots in the honey bee algorithm may switch what item-types they forage during the experiment, due to the item-type division of labour.

5.5 Summary

This chapter introduced two novel algorithms for foraging robot swarms. A desert ant inspired foraging algorithm and a honey bee inspired foraging algorithm are defined. A benchmark algorithm, called naïve foraging, is also presented. The desert ant algorithm uses path integration to memorize the location of an item site and returns robots to the site to forage more items. The honey bee algorithm is substantially more complex with three roles for robots: scout, unemployed forager, and employed forager. The scout robots in the honey bee algorithm uses path integration to memorize the location of a site and to evaluate the quality of the site. The location and quality of a site is communicated by the scout to unemployed foragers through direct communication. The unemployed

foragers are recruited and become employed foragers. The honey bee algorithm also exhibits division of labour between prioritized and non-prioritized items. An overview of the properties of the algorithms are given in Table 5.1.

Table 5.1: Properties of the foraging algorithms used in this study

Property	Naïve	Desert ant	Honey bee
Memory	✗	✓	✓
Communication	✗	✗	✓
Division of Labour	✗	✗	✓

Chapter 6

Experimental setup

This chapter describes the experimental setup for the simulations that were performed for the purposes of this study. Section 6.1 describes the robots used in the study, while the simulation environment is described in Section 6.2. The environments used in the experiment are discussed in Section 6.3, and Section 6.4 defines the parameters for the robot swarm. Section 6.5 defines how experiments were performed and performance measures used to evaluate the performance of the algorithms on the prioritized foraging problem are proposed in Section 6.6. The chapter is summarized in Section 6.7.

6.1 Robots

Foraging robots occur in all shapes, sizes and capabilities. Some robots have powerful GPS capabilities and advanced long distance sensors, while others are much simpler. This chapter defines the capabilities of the simulated robots to be used in this study. The robots are described in Section 6.1.1, while Section 6.1.2 outlines the navigational capabilities of the robots.

6.1.1 Robot Description

The artificial robots modelled in this study are based on e-puck robots [190], with additional grippers. Each simulated robot is equipped with a 360 degree camera to identify objects around the robot, as well as eight local distance sensors equally spaced around

the circular perimeter of the robot. Both camera and distance sensors have a depth of view of five times the robot's size. Robots use local communication which can occur in a radius of five times the robot's size. The sensor and communication range is sufficiently localized with respect to the size of the environment. Each simulated robot can forage a single item at a time. The robots do not have a global positioning system (GPS) capability to locate items to position themselves in the environment. A robot can not see occluded items. As a result, the simulated robots have to explore the environment to find the prioritized items.

6.1.2 Navigation and Obstacle Avoidance

The environments used in the simulation have a variety of complexities: Some environments are very sparse, while others have large zones of non-prioritized items that must be navigated around or foraged to clear a route to the prioritized items. Due to the environmental complexity, an advanced navigation and obstacle avoidance technique is required.

The robots in the experiments for this thesis use a navigation and obstacle avoidance technique inspired by a congestion avoidance technique developed for communication congestion avoidance in wireless sensor networks [191]. Antoniou *et al* use inspiration from the flocking behaviour of birds in order to efficiently route messages around congested areas in wireless sensor networks. In flocking behaviour of birds, birds are attracted by a global magnetic attractor to the birds' final destination, while a local attractor pulls flocking birds away from areas of congestion. In the congestion avoidance algorithm, the final destination of the message being sent on the wireless sensor network is a combination of the global attractor and the local attractor.

The described congestion avoidance technique has been adapted to form a simple but effective navigation and obstacle avoidance technique. Robots are pulled to a global attractor, which is the intended destination, while a local attractor directs robots away from local obstacles while maintaining a course to the destination.

Figure 6.1 illustrates the navigation and obstacle avoidance method used by the robots. The navigation and obstacle avoidance algorithm achieves the effect of the global attractor by setting the robots' field of view towards the direction of the desired

destination. The destination is determined by a homing beacon or by the robot's path integration vector. The direction at the centre of the field of view is the direction to the destination.

The effect of the local attractor is modelled by evaluating each direction in the field of view to select the most desirable direction. Desirability, d, of a direction, q, is a metric quantifying the how well a direction achieves a balance between clarity of the path and directness of the direction to the destination. The clarity, κ_q, in a direction q, is a normalized reading from the proximity sensor or camera such that $\kappa_q \in [0, 1]$. Clarity indicates the distance to the next nearest obstacle. If no obstacles exist in the depth of view v, then $\kappa_q = 0$. If there exists an obstacle immediately next to the robot, then $\kappa_q = 1$. The directness of a direction q, $\iota_q \in [0, 1]$ is calculated as the angular deviation from the direction of the destination, where $\iota_q = 0$ occurs when the direction i is the same as direction to the destination, dir, and a $\iota_q = 1$ occurs when the direction is at the edge of the field of view, f. Desirability d_q of direction q is defined as

$$d_i = \lambda\kappa_q + (1 - \lambda)\iota_q \tag{6.1}$$

where λ determines whether clarity, κ_q, or directness, ι_q, of direction q, has a greater effect on desirability. The described navigation and obstacle avoidance technique is used with all algorithms in the experiment and for all algorithms, λ is set to 0.5.

The obstacle avoidance technique was developed to be used in a continous environment. In order to adapt the technique for the 2D grid environment used in this thesis, the technique was discretized.

6.2 Simulator

A spatially discrete 2-dimensional grid world simulator has been developed and used in this thesis in order to accelerate computation. Discrete 2-dimensional grid world simulators are also used in [148, 129]. In real robot experiments, algorithm performance is sensitive to the amount of time taken to load items and manoeuvre the robots [145]. The 2-dimensional grid world simulator allows for movement and loading time to be standardized across all algorithms for effective comparison.

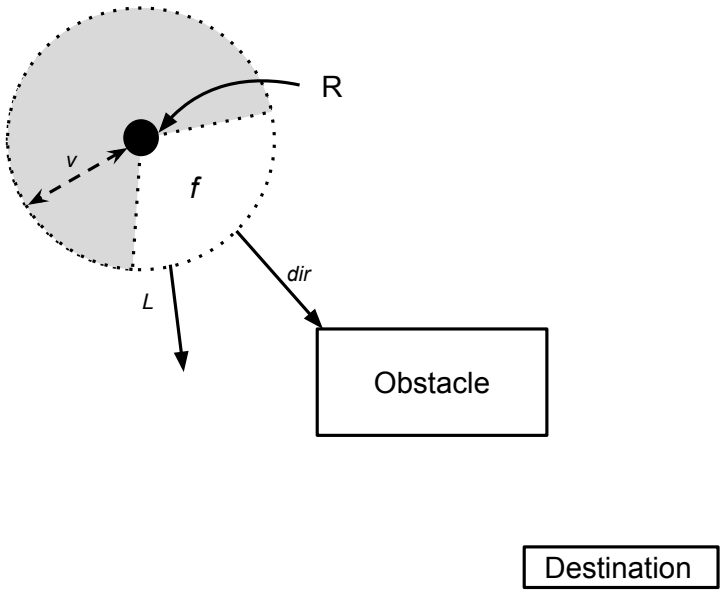


Figure 6.1: Navigation and obstacle avoidance, where v is depth of view, f is the field of view, R is the robot, *dir* is the direction of the destination and L is a possible value of local attractor

The simulated robots function as follows:

- Each robot fits into one grid block and each item takes up one grid block.
- Only a single object can occupy a grid block at a time. An object is either a robot or an item. Since only one object can occupy a grid block at a time, collisions and congestion can occur.
- Each robot can move to an adjacent cell in any direction.
- Robots can load, transport and offload a single item at a time.
- If a robot cannot pick up an item, the item is an obstacle that a robot may have to navigate around in order to reach its destination.

The prioritized and non-prioritized sinks were placed next to each other, on a single side of the environment. The sinks were marked by beacons that all robots can detect and navigate towards. The reason why the sinks were not placed in the centre of the

environment, as is commonly found in swarm robotics research [17], is because the prioritized foraging problem is inspired from using a swarm of robots to rescue trapped miners in mining tunnels, discussed in Section 5.1. A mining tunnel has a single entrance where the gold and waste must be moved to, in order to be transported to the surface [192]. Since there is only a single entrance at the beginning of a tunnel, the sinks need to occur at the beginning of the tunnel so that items can be easily exported.

6.3 Environments

In order to test flexiblity on different environment types, many types of environments should be examined. This section defines the various environments and presents how they were generated. Section 6.3.1 discusses the parameters for the environments, while different environment distributions, and the algorithms to generate those distributions are presented in Section 6.3.2. Finally, the accessible environment is defined in Section 6.3.3.

6.3.1 Environmental Parameters

The experiments were run on different environments where each environment has different item distributions, environment sizes, item densities, and different ratios of prioritized to non–prioritized items. All the environments are square. The length and width of the square environment grid is denoted as Λ, where $\Lambda \in \{50, 100, 200, 300, 500\}$.

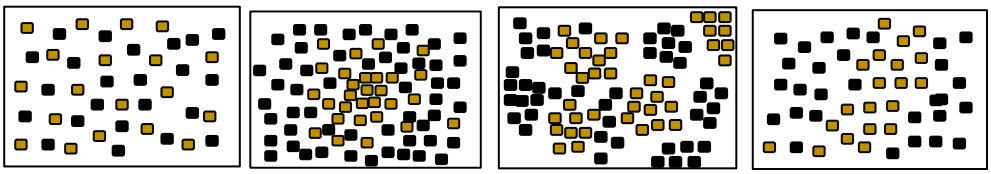
Different values for the density, p, of the items on the grid were chosen, such that if $p = 0.9$, then 90% of the grid cells are occupied by items, where $p \in \{0.05, 0.2, 0.5, 0.7, 0.9\}$. Environments with a larger item density are more complex to forage, because there exists a higher probability that items will occlude each other. Thus, a robot may not be able to access items of the type that it is specialized to forage, since its access to those items is occluded by items of a different type. For this reason, we refer to the density of items on the grid as the problem complexity.

The environment type ratio, r, is defined as the ratio of prioritized to non-prioritized items in an environment, where $r \in \{0, 0.2, 0.25, 0.33, 0.5, 0.67, 0.75, 0.8, 1\}$. When $r = 0$, an environment contains no prioritized items, and when $r = 1$, only prioritized items

exist in the environment. Environments with a small r value have an abundance of non-prioritized items which increases the likelihood of non-prioritized items preventing access to prioritized items.

6.3.2 Environment Distributions

The environment distribution, ϵ, is defined as the distribution of prioritized and non-prioritized items in an environment, such that $\epsilon \in \{uniform, clustered, gaussian, vein\}$. Each environment distribution was chosen in order to examine different characteristics of the algorithms. The item distributions are illustrated in Figure 6.2, where each lighter shaded square is a prioritized item and a non-prioritized item is shown by a darker shaded square. The four environment distributions were generated as follows:



(a) Uniform (b) Gaussian (c) Clustered (d) Vein

Figure 6.2: Environment Classes

1. The position of each item in a uniformly distributed environment is selected from a uniform distribution (refer to Figure 6.2a). The uniformly distributed environment has uniform concentrations of prioritized and non-prioritized items across the environment and thus is used as a control environment. Pseudo-code for generation of uniform environments is provided in Algorithm 9.

2. For the Gaussian environments, the positions of the prioritized items are sampled from a Gaussian distribution. The mean of the Gaussian disitribution is the centre of the grid. The deviation of the Gaussian distribution was selected as $deviation = S * E_p/2$ to ensure that generation took a reasonable amount of time (that the same positions are not reselected regularly), and that the non-prioritized items are densely concentrated (refer to Figure 6.2b). Pseudo-code for generation of Gaussian

Algorithm 9 Uniform Distributed Environments

```
1: function UNIFORM(numberItems, r, S)
2:     nonPrioritizedItemsLeft = ceil((1-r)*numberItems)
3:     prioritizedItemsLeft = floor( r*numberItems)
4:     while prioritizedItemsLeft > 0 do
5:         x ← uniform(0, S)
6:         y ← uniform(0, S)
7:         if gridCell (x,y) is empty then
8:             Place prioritized item at (x,y)
9:             Decrement prioritizedItemsLeft
10:        end if
11:    end while
12:    while nonPrioritizedItemsLeft > 0 do
13:        x ← uniform(0, S)
14:        y ← uniform(0, S)
15:        if gridCell (x,y) is empty then
16:            Place prioritized item at (x,y)
17:            Decrement nonPrioritizedItemsLeft
18:        end if
19:    end while
20: end function
```

environments is provided in Algorithm 10. The positions of the non-prioritized items are selected from a uniform distribution, after placing the prioritized items.

In Gaussian distributed environments, prioritized items occur in high concentration towards the center of the environment. More non-prioritized items occur on the outskirts of the environment, surrounding the prioritized items in the centre.

The Gaussian environments are used to examine whether each algorithm will enable the robot swarm to forage or navigate past the non-prioritized items to reach the high concentration of prioritized items in the environment's centre.

3. Environments with a vein distribution resemble the patterns observed in naturally

occurring gold reefs [193] (refer to Figure 6.2d). In a gold reef, molten gold fills planar fractures between rock resulting in a vein of gold. Inspired by gold reefs, vein distributed environments have a long thin vein of prioritized items running from one side of the environment to another. The vein of prioritized items was surrounded by non-prioritized items.

The vein environments aimed to test whether a swarm of robots could forage the continuous length of the vein of prioritized items, before foraging non-prioritized items. A swarm that is able to detect the location of the vein and return to the vein's location after foraging an item should forage more prioritized items initially than an algorithm that cannot detect and remember the location of the vein. Pseudo-code for generation of vein environments is provided in Algorithm 11.

4. Environments with a clustered item distribution have a random number of clusters of items of the same type (refer to Figure 6.2c). After clusters have been generated, each cluster is labelled randomly (with a Bernoulli disitribution with probability equal to the item ratio, r, required for that environment), as either a cluster of prioritized items or a cluster of non-prioritized items.

   The goal of performing experiments in a clustered environment was to test an algorithm's ability to exploit areas which are rich in prioritized items. Clustered environments are also used to test whether an algorithm can either navigate around or forage non-prioritized items. A clustered environment can also test an algorithm's ability to remember locations of areas which have a high density of prioritized items in order to aid more efficient access to prioritized items. Pseudo-code for generation of clustered environments is provided in Algorithm 12 and Algorithm 13.

To summarize, the challenges introduced by the more complex distributions (the Gaussian, clustered, and vein environment) aim to test a swarm's ability to:

- navigate past obstacles efficiently, or alternatively, to forage obstacles efficiently, and to
- return to areas rich in prioritized items to forage these areas.

Algorithm 10 Gaussian Distributed Environments

```
1: function GAUSSIAN(numberItems, r, S)
2:     Calculate environment centre point (x_c, y_c)
3:     prioritizedItemsLeft = ceil(r*numberItems)
4:     nonPrioritizedItemsLeft = floor((1-r)*numberItems)
5:     deviation = S*r/2
6:     while prioritizedItemsLeft > 0 do
7:         x ← floor(gaussian(x_c, deviation))
8:         y ← floor(gaussian(y_c, deviation))
9:         if gridCell (x,y) is empty and is valid then
10:            Place prioritized item at (x,y)
11:            Decrement prioritizedItemsLeft
12:        end if
13:    end while
14:    while nonPrioritizedItemsLeft > 0 do
15:        x ← uniform(0, S)
16:        y ← uniform(0, S)
17:        if gridCell (x,y) is empty then
18:            Place non prioritized item at (x,y)
19:            Decrement nonPrioritizedItemsLeft
20:        end if
21:    end while
22: end function
```

6.3.3 Accessible Environment

To aid discussions about the algorithms' performance on different types of environments, the concept of an **accessible environment** should be introduced. For the purposes of this thesis, an accessible environment is the parts of the environment that can be foraged by the swarm and are not blocked by other items. Items that are hidden behind other items are not accessible to the robot swarm and are thus not part of the accessible environment. The accessible environment is much larger in a less densely populated

Algorithm 11 Vein Distributed Environments

function VEIN($numberItems, r, S$)
    Select two random sides from the grid
    Select a random points on each sides, (x_0, y_0) and (x_1, y_1)
    Calculate gradient of vein, $m = (y_0 - y_1)/(x_0 - x_1)$
    Calculate c of equation for line vein, $c = (y_0 - m * x_0)$
    prioritizedItemsLeft = ceil(r*numberItems)
    nonPrioritizedItemsLeft = floor((1-r)*numberItems)
    while $prioritizedItemsLeft > 0$ **do**
        $x \leftarrow uniform(min(x_0, x_1), max(x_0, x_1))$
        $y \leftarrow m * x + c$
        if gridCell (x,y) is valid and gridCell (x,y) is empty **then**
            Place prioritized item at (x,y)
            Decrement $prioritizedItemsLeft$
        end if
    end while
    while $nonPrioritizedItemsLeft > 0$ **do**
        x $\leftarrow$ uniform(0, S)
        y $\leftarrow$ uniform(0, S)
        if gridCell (x,y) is empty **then**
            Place nonprioritized item at (x,y)
            Decrement $nonPrioritizedItemsLeft$
        end if
    end while
end function

environment. This also introduces the concept of the environment ratio of a given accessible environment. For example, an environment may have a high ratio of prioritized items overall, but if the prioritized items are surrounded completely by non-prioritized items, then the accessible environment consists only of non-prioritized items and has an environmental item type ratio of 0. The environment ratio of the accessible environ-

Algorithm 12 Clustered Distributed Environments (Part 1)

1: **function** CLUSTERED($numitems, r, S$)
2: Generate number of clusters, $total = uniform(3, 15)$
3: Calculate number of prioritized clusters, $clusters_p = ceil(r * total)$
4: Calculate number of non-prioritized clusters, $clusters_{np} = floor((1 - r) * total)$
5: Calculate number of prioritized items, $total_p = ceil(r * numitems)$
6: Calculate number of non-prioritized items, $total_{np} = floor((1 - r) * numitems)$
7: $ave_p = total_p / clusters_p$
8: $ave_np = total_{np} / clusters_{np}$
9: $clustersToGenerate_p = clusters_p$
10: $clustersToGenerate_{np} = clusters_{np}$
11: **while** $clustersToGenerate_p > 0$ **do**
12: sample centroid for cluster C (x,y) uniformly from grid
13: $num_p = uniform(ave_p/2,\ 2 * ave_p)$
14: Calculus radius, r, of cluster C as $r = \sqrt{num_p/\pi}$
15: **if** cluster C does not collide with existing clusters **then**
16: Save centroid and radius of cluster C to list
17: Decrement $clustersToGenerate_p$
18: $itemsToGenerate_p = num_p$
19: **while** $itemsToGenerate_p > 0$ **do**
20: $x_p = gaussian(x, r)$
21: $y_p = gaussian(y, r)$
22: **if** position (x_p, y_p) is valid **then**
23: Place prioritized item at (x_p, y_p)
24: Decrement $itemsToGenerate_{np}$
25: **end if**
26: **end while**
27: **end if**
28: **end while**

Algorithm 13 Clustered Distributed Environments (Part 2)

```
29:     while clustersToGenerate_np > 0 do
30:         sample centroid for cluster C (x,y) uniformly from grid
31:         num_np = uniform(ave_np/2, 2ave_np)
32:         Calculus radius, r, of cluster C as r = sqrt(num_np/π)
33:         if cluster C does not collide with existing clusters then
34:             Save centroid and radius of cluster C to list
35:             Decrement clustersToGenerate_np
36:             itemsToGenerate_np = num_np
37:             while itemsToGenerate_np > 0 do
38:                 x_p = gaussian(x, deviation = r)
39:                 y_p = gaussian(y, deviation = r)
40:                 if position (x_np, y_np) is valid then
41:                     Place non-prioritized item at (x_np, y_np)
42:                     Decrement itemsToGenerate_np
43:                 end if
44:             end while
45:         end if
46:     end while
47: end function
```

ment changes differently per environment disitribution type. In high density Gaussian environments, the accessible environment will initially consist of mostly non-prioritized items, but as robots forage towards the centre, the accessible environment will consist largely of prioritized items. For uniform environments, all accessible environments would have a similar or same environment item ratio as the entire environment. In clustered environments, depending on the configuration of the clusters, the environment ratio of the accessible environments would vary. For vein environments, at the beginning of foraging, the accessible environment ratio would be low in high density environments if the vein is perpendicular to the sink since only the "entrance" of the prioritized vein would be exposed to the robots and the rest of the prioritized items would be stacked behind

those items.

6.4 Swarm parameters

For all algorithms, each robot swarm was initialized with a swarm specialization ratio, $\tau \in \{0, 0.2, 0.25, 0.333, 0.5, 0.667, 0.75, 0.8, 1\}$. The swarm specialization ratio is the ratio of robots foraging prioritized items to non-prioritized items. When no robots are set to initially forage prioritized items, then $\tau = 0$ and when all robots are set to initially forage prioritized items, then $\tau = 1$.

The ability of an algorithm to adapt the value of τ appropriately for a given r indicates the flexibility of the algorithm.

The swarm density, c, is defined as the number of cells occupied by a robot, as a percentage of the grid size Λ (the length of the side of an environment), where values of $c \in \{0.1, 0.3, 0.5, 0.7, 1\}$ were evaluated. The swarm density is varied in order to test the scalability of the algorithm. The swarm density is also varied to test each algorithms' ability to adjust the number of actively foraging robots to the density of the items in the environment, c, as well as to adjust the number of robots actively foraging an item type, τ, to the item type density, r.

The honey bee algorithm specific parameters were selected based on [32], where $t_{wait} = 200$ time steps, $f_{max} = 100$ time steps, $\Phi = 0.8$ and $\rho = 0.1$. Testing the effect of each of the parameters specific to the honey bee algorithm was not in the scope of this thesis.

6.5 Experimentation

The initial position of each robot was randomly selected, adjacent to the sink. For the naïve algorithm and desert ant algorithm, all robots began in the exploration state. For the honey bee algorithm, all unemployed foraging robots were initialized in the waiting state, and the scout robots were initialized in the explore state. An experiment is defined as the 30 independant runs for each algorithm, with a specific set of swarm parameters, and on an environment which has a specific set of environmental parame-

ters. An experiment is run for every possible combination of swarm and environmental parameters.

6.6 Performance measures

The performance of swarm robotics algorithms can be measured by examining the algorithm's ability to perform a task in terms of efficiency, scalability, flexibility, and robustness.

6.6.1 Foraging Efficiency

The foraging efficiency, E_P, of an algorithm is defined as the ratio of the number of prioritized items collected by all robots in a fixed time period on a specific environment to the total number of prioritized items that exist in the environment. The foraging efficiency metric is similar to the efficiency metric used by Hecker *et al* [129]. The metrics used in experiments performed by Hecker *et al* evaluate the performance of foraging algorithms and are thus appropriate for use in the experiments of this study. The foraging efficiency measure, E_P, only considers the prioritized items for the prioritized foraging problem.

6.6.2 Flexibility

As stated in Section 2.2.2, flexibility refers to the effect of variations in the environment and tasks on the co-ordination mechanisms of swarm robotics algorithms. An algorithm that is highly flexible is one that has been optimized for a specific environment, but is equally efficient on a different environment [129].

The flexibility performance measures used in this study are based on the flexibility performance measures used in [129], which have been adapted for the prioritized foraging problem by only taking into account the prioritized item. The flexibility study addresses two aspects: Flexibility with respect to different prioritized item ratios, r, and flexibility with respect to different environment distributions.

Notation

In order to more succinctly define the flexibility performance measures, the following notation is defined:

Let $\overline{r}$ be the list containing all considered elements for environment item type ratio, r, and let n_r be the number of elements in the list. Let $\overline{\tau}$ be the list containing all considered elements for initial swarm specialization ratio, $\tau(0)$, and let the number of elements in the list be n_τ. Similarly, let $\overline{\epsilon}$ be a list of each environment distribution and let n_ϵ be the number of elements in the list.

Let Z be a matrix of dimensions $n_\tau \times n_r$. Define each entry of the matrix Z_{ij} as the average foraging efficiency, E_P, over all experiments, for a specific environment ratio r_j, where r_j is the jth element of $\overline{r}$, and for a specific initial swarm specialization ratio $\tau_i(0)$, such that $\tau_i(0)$ is the ith element of $\overline{\tau}$.

Similarly, define matrix D with dimensions $n_\tau \times n_\epsilon$, such that each entry of the matrix D_{ij} is the average foraging efficiency, over all experiments with environment distribution, ϵ_j, and with the initial swarm initialization ratio, $\tau_i(0)$.

Flexibility with respect to different prioritized item ratios

The goal of examining flexibility over different prioritized item ratios is to determine how well the optimum value for initial swarm specialization ratio, $\tau(0)$, for a specific environment item ratio, r_u, generalizes across all other environment item ratios, r_v, where $u, v = 1, ..., n_r$ and $u \neq v$. The optimum value for $\tau(0)$ for environments of a specific environment item ratio is the value that yields the greatest average efficiency for those environments.

To do so, a list I of size n_r is defined, where each entry I_j is set to the index k that yields the maximum value for Z_{ij}, for $i = 1, ..., n_\tau$, for environments with a specific environment type ratio r_j. More succinctly:

$$I_j = (k | Z_{kj} = \max_{i=1,...,n_\tau} \{Z_{ij}\}) \tag{6.2}$$

The moment, F_{r_u}, around the maximum foraging efficiency for a specific environment item type ratio, r_u, is defined as follows:

$$F_{r_u} = \sum_{v=1}^{n_r} \frac{|Z_{I_u u} - Z_{I_u v}|}{Z_{I_u u}} \tag{6.3}$$

The final performance measure is the normalized sum of the above moments around the maximum foraging efficiency for every environment item type ratio in $\overline{r}$:

$$F_r = \frac{1}{n_r} \sum_{u=1}^{n_r} F_{r_u} \tag{6.4}$$

The performance measure, F_r, measures how well swarm parameters, which are optimized to yield the best efficiency for environments with a specific prioritized item ratio generalize across environments with different prioritized item ratios. The less the foraging efficiency varies when an algorithm that is configured with swarm parameters optimized for a specific environment ratio, is run on all other environment ratios, the smaller F_r will be. Therefore, the smaller F_r is, the more flexible an algorithm is considered to be. F_r is a macro performance indicator which is run on the results of all experiments.

Flexibility with respect to different environment distributions

This performance measure evaluates flexibility by analysing how well swarm parameters, which are optimized to yield the best efficiency for environments with a specific environment distribution generalize across environments with different environment distributions. The performance measure, F_ϵ, is defined similar to F_r, except over environment distributions, rather than environmental item ratios. A list Y of size n_ϵ is defined as follows:

$$Y_j = (k|D_{kj} = \max_{i=1,\ldots,n_\tau} \{D_{ij}\}) \tag{6.5}$$

The performance measure, F_ϵ, is defined as follows:

$$F_\epsilon = \frac{1}{n_\epsilon} \sum_{u=1}^{n_\epsilon} \sum_{v=1}^{n_\epsilon} \frac{|D_{Y_u u} - D_{Y_u v}|}{D_{Y_u u}} \tag{6.6}$$

F_ϵ is interpreted in a similar manner to F_r, where the smaller the value for F_ϵ is, the more flexible an algorithm is considered to be over environment distributions.

6.6.3 Scalability

Scalability is described in Section 2.2.3. The scalability performance measures used in this thesis are based on the measures used in [129]. This study analyses the scalability of each of the algorithms with respect to two performance measures, namely, swarm density scalability and problem complexity scalability. Swarm density scalability and problem complexity scalability are defined in the following sections. Both swarm density scalability and problem complexity scalability are evaluated on the largest environment size (i.e. $\Lambda = 500$) in order to prevent the possibility that an environment runs out of items to forage.

Notation

To simplify the explanation of the scalability performance measures, the following notation is defined:

Let $\overline{c}$ be the list of all considered values for swarm density, c, with length n_c. Let c_{min} be the minimum value considered for c and c_i be the ith value of $\overline{c}$. Let E_{c_i} be the average foraging efficiency, E_P, over all experiments, with swarm density c_i.

Similarly, define the list of all considered values for problem complexity, p, as $\overline{p}$ with length n_p. The smallest problem complexity is p_{min} and E_{p_i} is the average foraging efficiency over all experiments, with problem complexity p_i.

Swarm density scalability

The efficiency of a robot swarm should be relatively undisturbed by changes in group sizes. This property is known as swarm density scalability. Ideally, foraging efficiency should increase linearly as swarm density increases, which is known as linear scalability. However, due to increased inter-robot interference in larger swarm densities, scalability is often sub-linear (i.e. logarithmic) [60]. Many swarm robotics algorithms focus on the use of division of labour, navigation or communication in order to decrease inter-robot interference [60, 147].

In order to analyse swarm scalability, one could simply examine the average efficiency of each algorithm, over all experiments, at each swarm density. However, to ensure

that each algorithm is compared on scalability alone, the individual efficiencies of each algorithm must be eliminated. To eliminate the individual efficiencies of each algorithm, the average efficiency, E_{c_i}, for each swarm density, c_i, is normalized by the average efficiency, , $E_{c_{min}}$, for the smallest sized swarm for that algorithm, as follows:

$$S_{c_i} = \frac{E_{c_i} - E_{c_{min}}}{E_{c_{min}}} \tag{6.7}$$

To analyse how each algorithm scales, the normalized average foraging efficiency, S_{c_i}, for each value in $\overline{c}$ can be plotted for each algorithm to determine whether an algorithm scales linearly, sublinearly, or superlinearly.

The total swarm scalability, S_c, provides a single value per algorithm, which can be used to compare overall swarm scalability:

$$S_c = \sum_{i=1}^{n_c} S_{c_i} \tag{6.8}$$

The larger the value of S_c, the better foraging efficiency scales with respect to swarm density.

Problem complexity scalability

Ideally, foraging efficiency should be indirectly proportional to problem complexity. As the problem complexity increases, the foraging efficiency should decrease linearly or super-linearly. An algorithm is scalable when performance decreases linearly or super-linearly as problem complexity increases. Similar to swarm density scalability (described in Section 6.6.3) problem complexity scalability is often sub-linear, due to increased environmental interference.

In order to evaluate the problem scalability of each of the algorithms presented, the efficiency of each algorithm is examined over a variety of environment item densities, p, defined in Section 6.3. Similar to swarm density scalability performance measures S_{c_i} and S_c, the problem complexity scalability measures, S_{p_i} and S_p, are defined as follows:

$$S_{p_i} = \frac{E_{p_i} - E_{p_{min}}}{E_{p_{min}}} \tag{6.9}$$

$$S_p = \sum_{i=1}^{n_p} S_{p_i} \tag{6.10}$$

6.6.4 Behavioural Performance Measures

Despite not being one of the more typical performance measures for swarm robotics, a number of measures are included to enable further understanding of the obtained quantitive performance measures.

Items foraged over time

To give insight into the effects of environmental and swarm parameters on overall performance, and to characterise the effects of certain behaviours, the following metrics have been recorded:

- The total number of prioritized items foraged over time, E_P^t. The total number of prioritized items foraged are recorded every 200 time steps and are normalized with the total number of prioritized items that exists in the specific environment.
- The total number of non-prioritized items foraged over time, E_{NP}^t. The total number of non-prioritized items foraged are recorded every 200 time steps and are normalized with the total number of non-prioritized items that exist in the specific environment.

Swarm specialization ratio over time

Swarm specialization ratio stays constant for the desert ant algorithm and the naïve algorithm, but the swarm specialization ratio changes for the honey bee algorithm, due to the division of labour mechanisms. In order to gain a better understanding of how the division of labour mechanism performs, the swarm specialization ratio over time, $\tau(t)$, is tracked. The swarm specialization ratio is recorded every 200 time steps.

Time spent performing recruitment

Unlike the desert ant algorithm and the naïve algorithm, the honey bee algorithm robots perform behaviours that are not directly involved in retrieving items - in particular, the recruitment activities form part of the division of labour mechanism. In order to better understand the influence of the division of labour mechanisms, the average percentage of the time that each robot spents on each division of labour activity is recorded as follows:

- t_{wait}, the average percentage of the total number of time steps spent by each robot in the waiting state.
- $t_{recruitment}$, the average percentage of the total time number of steps spent by each robot in the recruitment state.

6.7 Summary

The robots used in this study are based on the e-puck robots, but with gripper capabilities. The robots used a navigation and obstacle avoidance technique based on the flocking behaviour of birds where a global attractor force attracts the robot in a specific direction, while the local attractor force guides a robot around localized obstacles. A simple 2-dimensional grid-based simulator is used. Only a single robot or item occupies a single grid cell at any point in time.

Different types of environment distributions were generated for experimentation. The environment types that have been created are uniform, Gaussian, clustered, and vein distributions. The following environmental parameters were varied: item density, environment size, and item type ratio. The following swarm parameters chosen for each of the algorithms were presented: initial swarm specialization ratio, swarm density, and honey bee specific parameters.

Performance measures were selected to evaluate the efficiency of the algorithms, as well as the scalability and flexibility of the algorithms. Lastly, a set of behavioural performance measures are presented in order to gain insight into individual behaviours of the algorithms, namely, the items foraged over time, the swarm specialization ratio over time, and the time spent performing recruitment activities.

Chapter 7

Results

The analysis of the results of the experiments is organised with respect to four major characteristics of swarm robotics algorithms, namely efficiency, flexibility, robustness, and scalability. Section 7.1 addresses foraging, while the flexibility of the algorithms over different environment distributions and item type ratios is analysed in Section 7.2. Section 7.3 evaluates the scalability of each algorithm, and robustness is analysed in Section 7.4. Section 7.5 summarizes the findings of the analysis.

7.1 Foraging efficiency

Despite the fact that determining the most efficient algorithm is not a main focus of this study, this section does a brief comparison of the foraging efficiency, E_P (defined in Section 6.6.1), of each algorithm.

This is done by comparing the performance of the three algorithms over all environments and all swarm parameters giving a total at 121500 unique experiment configurations each run for 30 independant samples. A Wilcoxin test, at a confidence level of 0.05, was applied over the 30 independant samples, for each pair of algorithms, for each experiment, to determine if a statistically significant difference in foraging efficiency of the pair of algorithms exists. If no statistically significant difference is indicated between the two algorithm experiment samples, then nothing is done. If a statistically significant difference is indicated between the two algorithm experiment samples, then two one-

tailed Mann-Whitney U tests were performed at a significance level of 0.05 to determine which algorithm had a greater foraging efficiency on a particular experiment. Suppose A and B are two of the selected algorithms. If the first one-tailed Mann-Whitney U test indicates that A is significantly more efficient than B, then a "win" is counted for algorithm A. If the second one-tailed Mann-Whitney U test indicates that the E_P for A is significantly less efficient than B, then a "loss" is counted for algorithm A.

To determine an algorithm's foraging efficiency in comparison to each other algorithm, the wins and losses for each algorithm are summed per experiment. This statistical analysis approach has been used in [194]. The percentage of wins per algorithm and the percentage of losses per algorithm is calculated, because percentages are more informative than counts. The percentage of wins per algorithm and the percentage of losses per algorithm are summarized in Table 7.1. The desert ant foraging performed better than the naïve algorithm. The honey bee algorithm out-performed both the naïve algorithm and the desert ant algorithm. The following sections analyse the swarm robot properties of each algorithm, and explain why the foraging efficiency of the algorithms is as shown in Table 7.1.

Table 7.1: Pairwise one-tailed Mann Whitney U wins and loss counts, for the foraging efficiency, E_P, for each algorithm, over all environments and swarm parameter values

Algorithms	**Wins**	**Losses**
Naïve	1.3%	66.5%
Desert ant	47.4%	16.4%
Honey bee	51.2%	17.1%

7.2 Flexibility

This section analyses the flexibility of each of the considered algorthms in terms of the prioritized item ratio and the environment distribution types, over all experiments. Section 7.2.1 analyses the flexibility of each algorithm in terms of the prioritized item ratio of the environment. Section **??** discusses the flexibility of each algorithm in terms

of environment distribution types.

7.2.1 Flexibility in terms of prioritized item ratio

The performance measure, F_r (defined in Section 6.6.2), measures how well the swarm parameters, which yield the best efficiency for environments with a specific prioritized item ratio, generalize across environments with different prioritized item ratios. Therefore, the smaller F_r is, the more flexible an algorithm is considered to be. F is a macro performance indicator which is run on the results of all experiments.

Table 7.2 summarizes the flexibility of each foraging algorithm in terms of environment type ratio, F_r. According to Table 7.2, the honey bee algorithm is the most flexible in terms of F_r, followed by the naïve algorithm, with the desert ant algorithm exhibiting the worst flexibility.

Table 7.2: Flexibility in terms of prioritized item ratio, F_r, and flexibility in terms of environment distribution, F_ϵ, for each algorithm

	Naïve	Desert ant	Honey bee
F_r	1.186	3.124	0.828
F_ϵ	1.112	0.507	0.458

The honey bee algorithm is the most flexible in terms of prioritized item ratio. To understand why this is the case, the mechanisms used specifically by the honey bee algorithm have to be examined. In particular, the honey bee algorithm employs a division of labour mechanism that allows robots to switch from foraging prioritized items to non-prioritized items (and vice versa). The honey bee algorithm's division of labour mechanism suggests the following hypothesis: The honey bee algorithm's superior flexibility is due to the fact that the robots can switch their item type specialization. In that way, the honey bee algorithm was able to adapt the swarm specialization ratio to more efficiently forage a given environment item type ratio.

To provide evidence for the above hypothesis the average swarm specialization ratio over time, $\tau(t)$, for the honey bee algorithm on a uniformly distributed environment with a high ratio of prioritized items (i.e. $r = 0.75$) is examined. The initial swarm

specialization ratio, $\tau(0)$, was set to 0, which means that all robots are set to forage non-prioritized items. The swarm specialization ratio over time, $\tau(t)$, was averaged over 30 independant runs. The purpose behind examining the described scenario is to determine if the honey bee algorithm is able to adapt the swarm's specialization ratio (which initially can only forage non-prioritized items), to more efficiently forage the high ratio of prioritized items. The above scenario is illustrated in Figure 7.1.

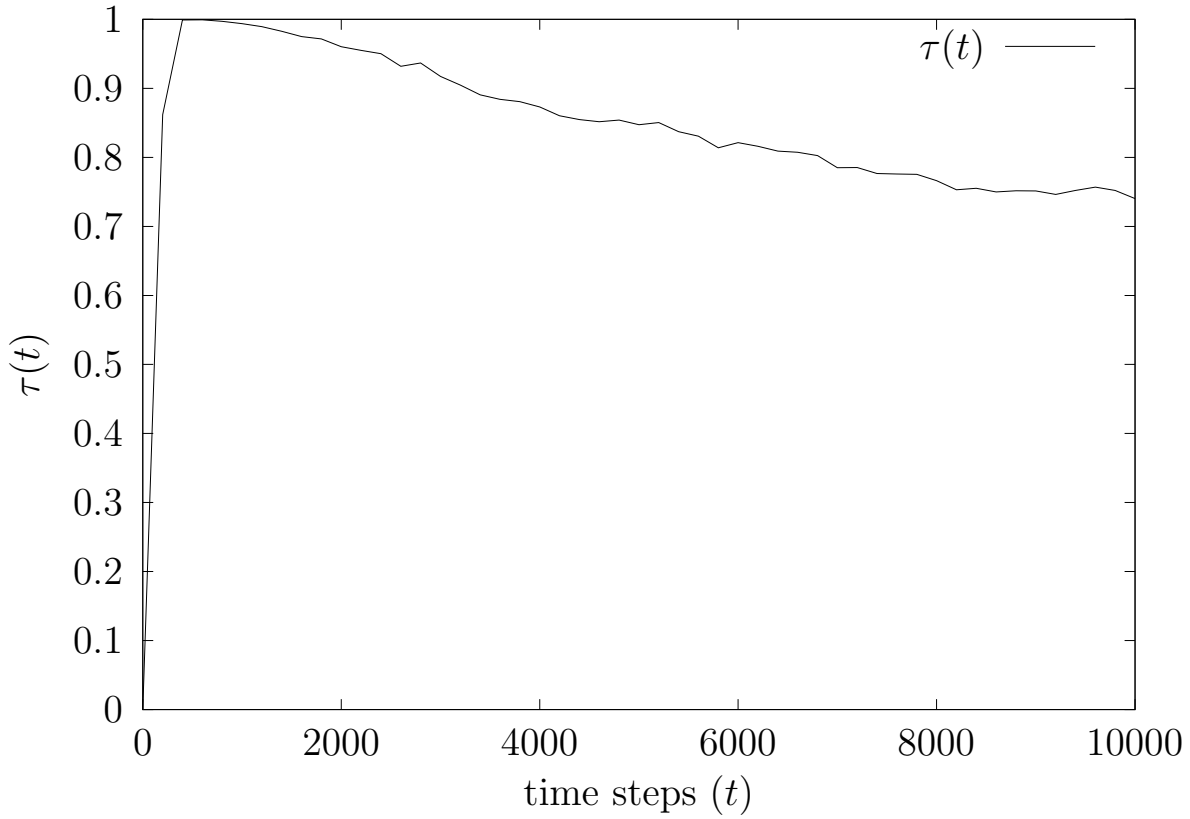


Figure 7.1: Specialization ratio over time, $\tau(t)$, of the honey bee algorithm, on a uniform environment, with environment item type ratio, $r = 0.75$, and initial swarm specialization ratio, $\tau(0) = 0$

Figure 7.1 shows that τ increased from 0 to 1 in 200 iterations, which suggests that the honey bee scout robots switched their specializations to search for prioritized items when the honey bee scout robots could not find any non-prioritized items. After 200 iterations, the specialization ratio steadily declined from 1 to 0.73. The final swarm specialization

ratio, $\tau(10000) = 0.73$, is nearly the same as the environment ratio, $r = 0.75$, suggesting that the honey bee algorithm eventually adapted the specialization ratio to effectively forage the environment item type ratio of the given environment.

The above analysis of the behaviour of the honey bee algorithm supports the hypothesis that the honey bee algorithm's superior flexibility can be attributed to the algorithm's ability to adapt the swarm's specialization ratio (via the division of labour mechanism described in Section 5.4.4) to more efficiently forage an environment of a given environment item ratio.

Both the naïve algorithm and desert ant algorithm do not have the ability to adapt the initial swarm specialization to forage any environment ratio, which means that both algorithms are less flexible than the honey bee algorithm.

The desert ant algorithm is the least flexible, which indicates that the algorithm is sensitive to the choice in $\tau(0)$ for each environment type ratio r. Unlike the naïve algorithm, which has the worst efficiency over all experiments, a good choice of τ will allow the algorithm to forage the environment significantly better for a specific environment item ratio. Despite the desert ant algorithm outperforming the naïve algorithm in efficiency (refer to Section 7.1), the naïve algorithm is substantially more flexible than the desert ant algorithm. F_r reveals that the naïve algorithm's performance is invariant under the variation of $\tau(0)$ on different environment item type ratios, resulting in a high level of flexibility. However, the naïve algorithm's flexibility is practically irrelevant given how poorly the algorithm performs on all environments.

7.2.2 Flexibility in terms of environment distributions

Similar to F_r, the performance measure, F_ϵ (defined in Section **??**), measures how well the swarm parameters yielding the best efficiency for environments with a specific environment distribution generalize across environments with different environment distributions, for a specific algorithm. Similar to F_r, the smaller F_ϵ is, the more flexible an algorithm is considered to be, and F_ϵ is also calculated based on the results of all experiments. The results summarized in Table 7.2 indicate that the honey bee algorithm was the most flexible in terms of environment distribution, followed by the desert ant algorithm. The naïve algorithm was the least flexible.

The honey bee algorithm's superior flexibility suggests the following hypothesis: The honey bee algorithm's ability to adapt τ (described in Section 5.4) enables the swarm to more efficiently forage the differing environment ratios of the accessible environment (refer to Section 6.3.3) at any point during the foraging process.

To provide evidence for this hypothesis, an in-depth examination of the honey bee algorithm is performed on a Gaussian environment with a high density of items. The environment has a low ratio of prioritized items and a swarm which has a high initial swarm specialization ratio (i.e. most robots are initially set to forage prioritized items). Since the non-prioritized items surround the prioritized items, the swarm should first forage the non-prioritized items in order to get access to the prioritized items. Once the swarm can reach the prioritized items, the swarm should switch to foraging the prioritized items. The example Gaussian environment is illustrated in Figure 7.2. The white cells represent an empty cell, the dark cells represent the cells containing a non-prioritized item and the shaded cells represent the cells containing a prioritized item.

The environment in Figure 7.2 has the following properties:

- A Gaussian environment item distribution,
- a low environmental item ratio of $r = 0.2$,
- a high density of items with $p = 90$, and
- an environment size, $\Lambda = 100$.

Also, consider a swarm with the initial specialization ratio set to mostly forage prioritized items (i.e. $\tau(0) = 0.8$) and with swarm density set to 0.5. The items nearest to the sink are all non-prioritized items. Robots will have to forage a large portion of the non-prioritized items between the sink and the centre of the environment in order to reach the high density of prioritized items at the centre of the environment.

Figure 7.3 shows the efficiency of foraging prioritized items, E_P, the efficiency of foraging non-prioritized items, E_{NP}, and specialization ratio over time, $\tau(t)$, for the honey bee algorithm on the environment described above. The performance measures were averaged over 30 independant runs.

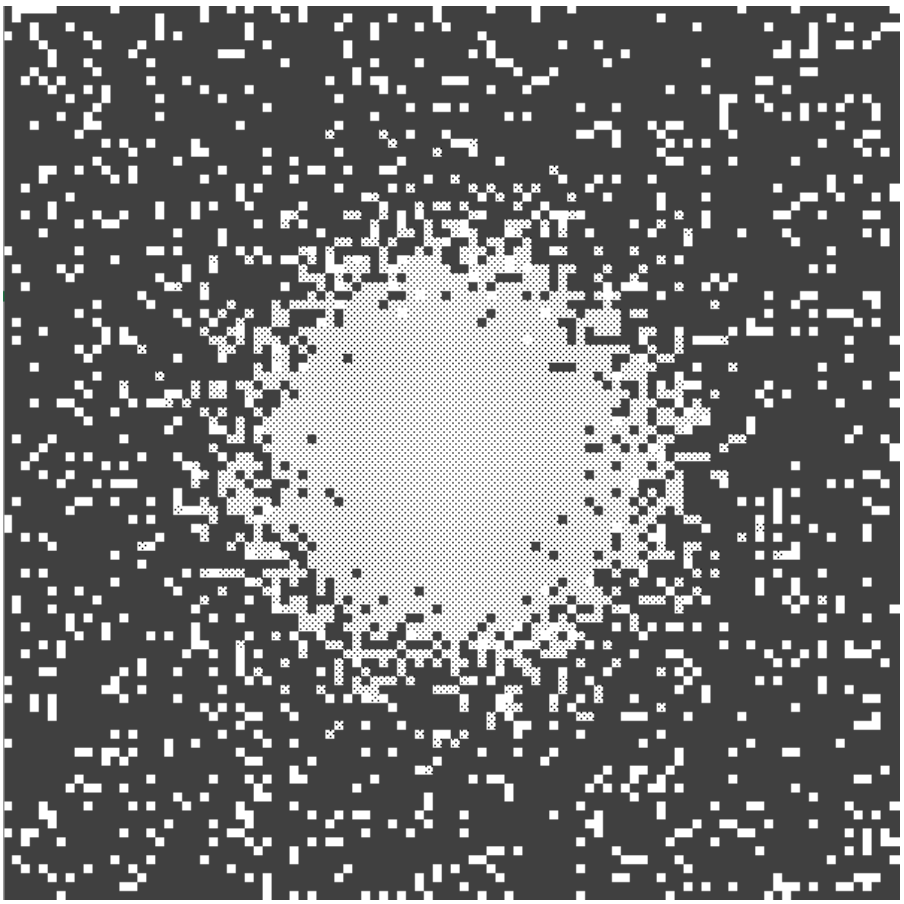


Figure 7.2: Gaussian environment with environment item type ratio, $r = 0.2$, environment item density of $p = 0.9$, and environment size $\Lambda = 100$

In the first 200 time steps, the efficiency of foraging nonprioritized items, E_{NP}, increased very slowly. This was because there were very few robots that could forage non-prioritized items because most of the robots were initialized to forage prioritized items. The efficiency of foraging prioritized items, E_P, did not increase in the first 200 time steps, because the robots had no access to the prioritized items as they were blocked by the non-prioritized items. The initial specialization ratio, $\tau(0)$, that was initialized at 0.8, decreased very quickly to close to 0 in the first 200 time steps. The decrease in specialization ratio is good, because then most robots switched to forage non-prioritized items in the accessible environment which consisted entirely of non-prioritized items. Between 200 and 800 time steps, the E_{NP} increased very quickly, due to the decrease in

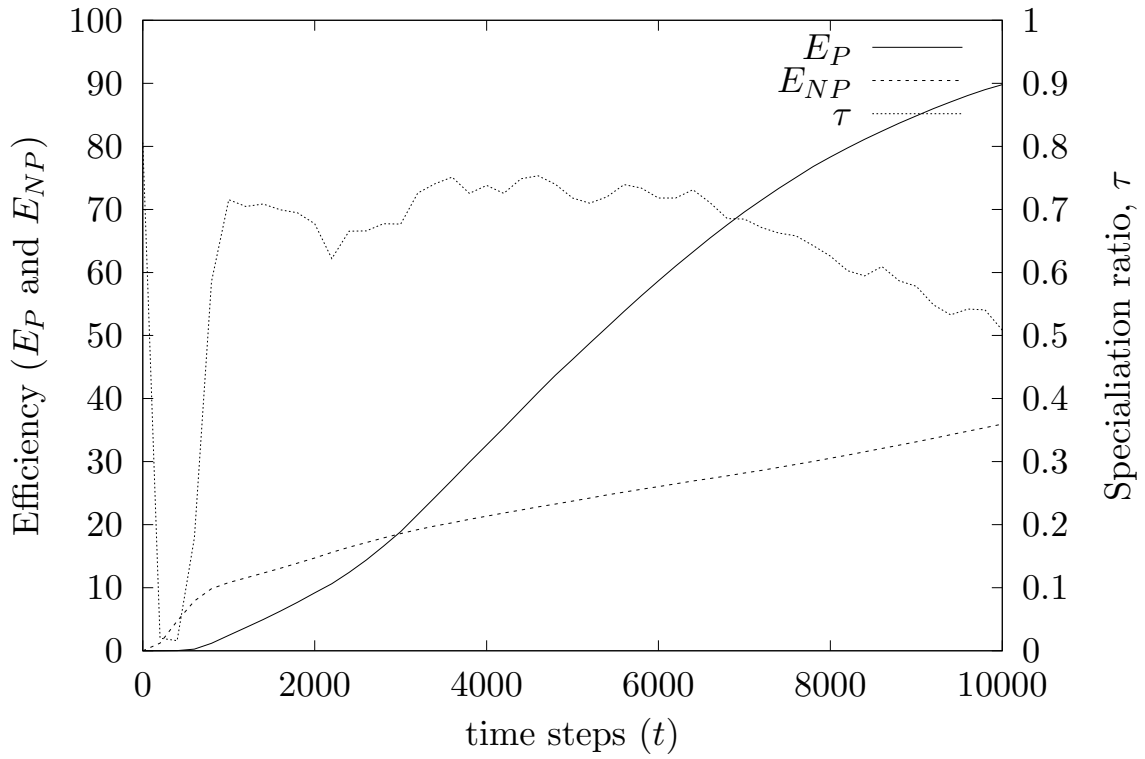


Figure 7.3: Efficiency of prioritized item foraging, E_P, efficiency of non-prioritized item foraging, E_{NP}, and specialization ratio over time, $\tau(t)$, for the honey bee algorithm on an example Gaussian environment, averaged over 30 independant runs

the swarm specialization ratio. The E_P remained low between 200 and 800 time steps, because the prioritized items were not yet accessible to the swarm. After 800 time steps, τ increased from near 0 to just above 0.7, suggesting that the non-prioritized items have been cleared and the concentration of prioritized items in the centre have been reached. Thus, most of the robots in the swarm switched from foraging non-prioritized items to foraging prioritized items. The increase in swarm specialization ratio after 800 time steps was followed by a sharp increase in E_P. This demonstrated that the honey bee algorithm could adapt τ in order to initially forage the large number of non-prioritized items, and then switched to foraging mostly prioritized items when the prioritized items became accessible.

In constrast, the naïve algorithm's performance suffered because it lacks the ability to adapt swarm specialization. Figure 7.4 illustrates the same performance measures, on the same Gaussian environment and swarm parameters for the naïve algorithm.

Because the naïve algorithm does not enable robots to switch their item specialization, the swarm specialization ratio, τ, remains constant. Because so few robots could clear non-prioritized items, due to the large initial swarm specialization ratio, E_{NP} increased slowly throughout the experiment. Because the swarm was slow to clear the non-prioritized items, E_P also remained small throughout the experiment, since the swarm struggled to make the prioritized items easily accessible. This suggests that prioritized foraging robots experienced environmental interference while attempting to navigate around the unforaged non-prioritized items. The rate of increase in E_P increased over the duration of the experiment, as the swarm cleared more of the non-prioritized items. The behaviour for the desert ant algorithm was similar to that of the naïve algorithm, except that the final efficiency was greater for the desert ant algorithm.

Figure 7.3 shows that the final E_P for the honey bee algorithm, averaged over 30 independant runs, was 0.9, while Figure 7.4 shows that the final E_P for the naïve algorithm was only 0.3. Since the honey bee algorithm adapts τ, the honey bee algorithm shows an increased ability to forage obstacles (the non-prioritized items) in order to more easily access hard to reach prioritized items. As a result, the honey bee algorithm is more flexible to different environment distributions than the naïve or desert ant algorithms.

To provide further evidence for the above hypothesis, consider other environment distributions to show that the honey bee algorithm can adapt the swarm's swarm specialization ratio appropriately for different distributions. Thus, consider a uniform environment with the same environment parameters as above. Because the environment distribution is uniform, the environment ratio of the accessible environment will fluctuate erratically as the swarm forages the environment.

Figure 7.5 shows the same performance measures for an experiment with the same swarm parameters as above on the uniform environment for the honey bee algorithm. The specialization ratio has a downward erratic trend, from over 0.9 to approximately 0.4. The erratic fluctuation of τ is due to minor and short-lived erratic differences in the environment ratio of the accessible environment which would feature in uniform envi-

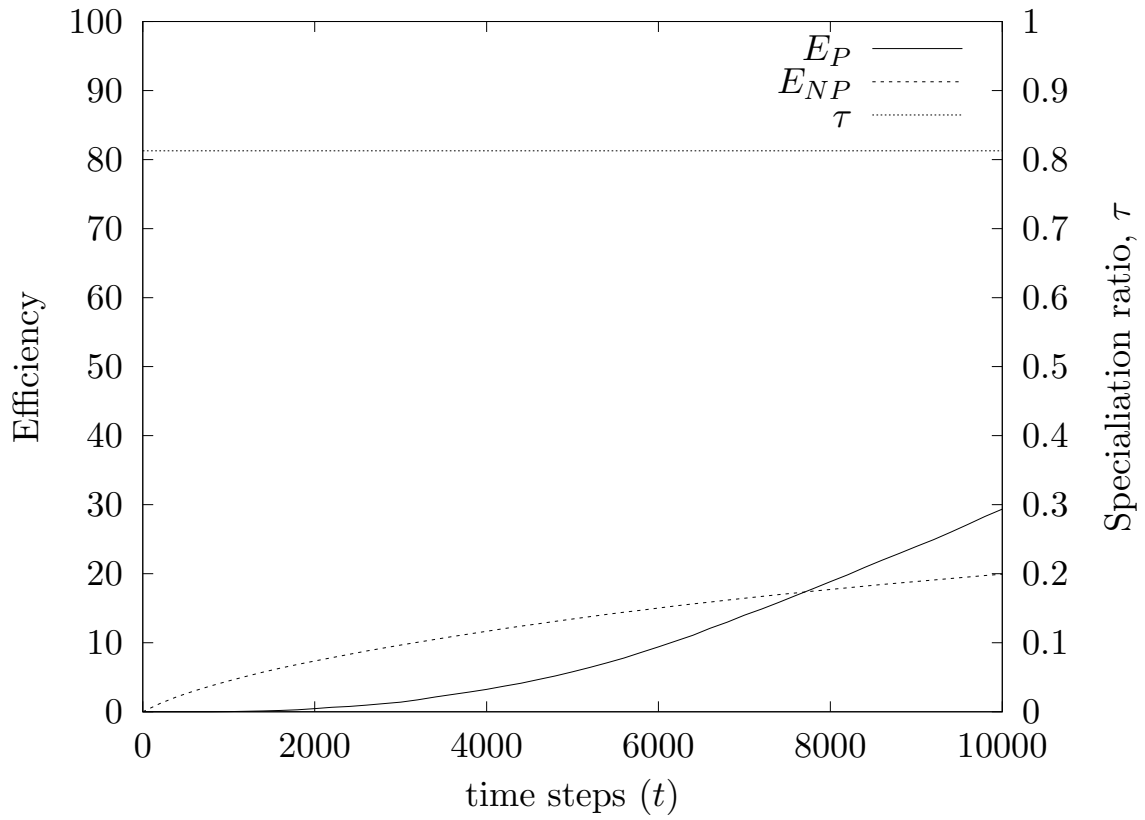


Figure 7.4: Efficiency of prioritized item foraging, E_P, efficiency of non-prioritized item foraging, E_{NP}, and the specialization ratio over time, $\tau(t)$, for the naïve algorithm on a Gaussian evironment, averaged over 30 independant runs

ronments. The rates of foraging efficiency, E_P and E_{NP}, increase constantly throughout the experiment.

The above analysis provides evidence for the hypothesis that suggests that the honey bee algorithm is capable of adapting the swarm to forage the accessible environment. Thus the honey bee algorithm has higher flexibility on different environment distributions compared to the desert ant and naïve algorithms. The desert ant and naïve algorithms have more difficulty foraging certain distributions as they cannot adapt τ to more effectively forage the accessible environment.

Table 7.2 indicates that the desert ant algorithm is more flexible over environment distributions than the the naïve algorithm. Considering the fact that the only difference

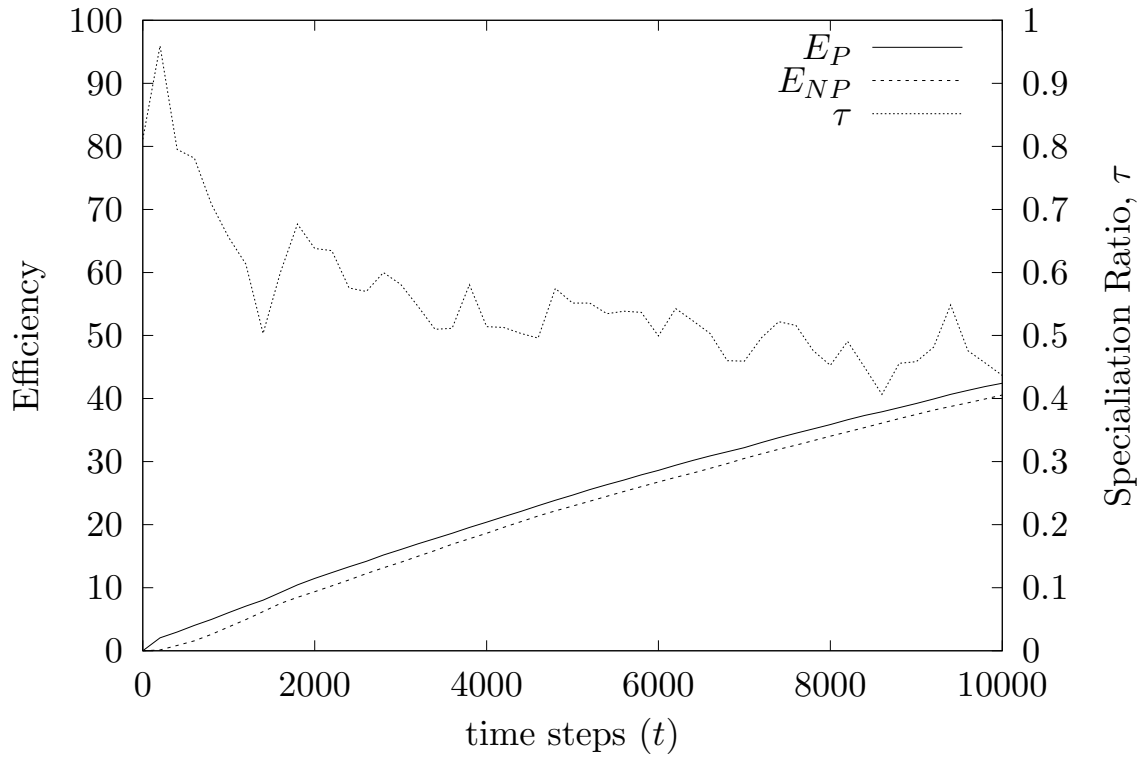


Figure 7.5: Efficiency of prioritized item foraging, E_P, over time t, efficiency of non-prioritized item foraging, E_{NP}, over time t, and specialization ratio, $\tau(t)$, for the honey bee algorithm on a uniform environment, averaged over 30 independant runs

between the naïve algorithm and the desert ant algorithm is the fact that the desert ant algorithm uses site fidelity while the naïve algorithm does not, the difference in performance must be attributed to the desert ant's use of site fidelity.

7.3 Scalability

This section discusses the scalability of each algorithm in terms of the macro performance measures: swarm density and problem complexity. Section 7.3.1 discusses the scalability of each algorithm in terms of swarm density, while Section 7.3.2 evaluates the scalability of each algorithm in terms of the complexity of the problem.

7.3.1 Swarm scalability

Table 7.3 presents the swarm density scalability results, S_c. Figure 7.6 shows the swarm density scalability for each density, S_{c_i}, for each algorithm (as defined Section 6.6.3). The naïve algorithm was the most scalable while the desert ant and the honey bee algorithms show similar scalability. The honey bee algorithm is more scalable than the desert ant algorithm for $c < 0.9$, but at $c \geq 0.9$, the swarm scalability, S_{c_i}, of the desert ant algorithm was greater than the honey bee algorithm.

Table 7.3: Swarm scalability, S_{c_i}, for each swarm density in $\bar{c}$, for each algorithm.

c	**0.1**	**0.3**	**0.5**	**0.7**	**1**
naïve	1	2.148	2.951	3.568	4.255
desert ant	1	1.973	2.605	3.058	3.556
honey bee	1	2.067	2.706	3.119	3.532

Figure 7.6 also plots the ideal scalability. Ideal scalability is when the increase in swarm density is directly proportional to the increase in efficiency. The ideal values for S_{c_i} were calculated and plotted. All algorithms fall below the ideal scalability line and thus have sublinear (logarithmic) scalability. Section 6.6.3 highlighted that increased inter-robot interference is a major factor that impacts on swarm scalability. It can be concluded that the reason for sub-linear performance in all algorithms was because all algorithms suffer from increased inter-robot interference as swarm size increases.

To understand why the desert ant and the honey bee algorithms were less scalable than the naïve algorithm, consider the following: The naïve algorithm, being the most simple foraging algorithm, has no mechanism to enable the swarm to exploit high quality sites of prioritized items and can only locate items by randomly exploring the environment. Both the desert ant algorithm and the honey bee algorithm have mechanisms to exploit high quality areas: The desert ant algorithm uses site fidelity and the honey bee algorithm uses recruitment to exploit high quality sites.

Suppose that there exists a single high quality site in an environment. Using site-fidelity and recruitment, the desert ant and the honey bee algorithms can exploit that high quality area. In exploiting that high quality site, the path between the sink and

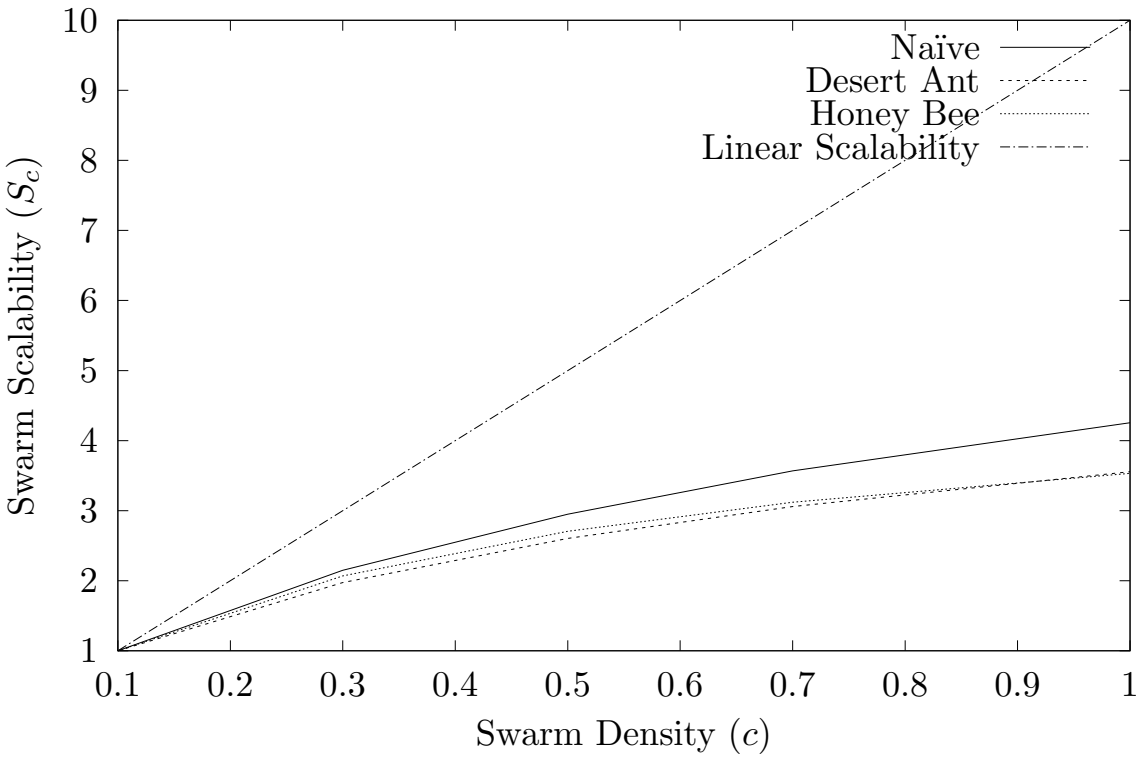


Figure 7.6: Swarm scalability, S_c, for each swarm density c_i in $\overline{c}$ for each algorithm.

the high-quality site becomes more congested as more robots share the path between the sink and the high quality area. The increased congestion on that path will cause more inter-robot interference. However, due to the focused efforts of the swarm on a high quality area of the environment, the exploitation will still improve foraging efficiency overall, compared to the naïve algorithm at low swarm densities. However, as swarm density increases, the congestion on the route between the high quality site and the sink will increase, hindering foraging efficiency due to increased inter-robot interference. It follows that the desert ant algorithm and the honey bee algorithm are less scalable in terms of swarm density than the naïve algorithm, due to the increased inter-robot interference as a result of exploitation of high quality sites.

The recruitment mechanism used by the honey bee algorithm is a more extreme form of exploitation than site fidelity used by the desert ant algorithm, because scouts recruit

groups of robots to forage single areas. On the other hand, when using site fidelity, the high quality site is only foraged by the robot that found it. Considering that the recruitment mechanism is more exploitative than the site fidelity mechanism, one would expect the honey bee algorithm to be much less scalable than the desert ant algorithm, but that is not demonstrated in Table 7.3. Instead, the honey bee algorithm performed slightly better than the desert ant algorithm for all $c < 0.8$.

To explain the honey bee algorithm's ability to outperform the desert ant algorithm at most swarm densities, consider the honey bee algorithm's division of labour mechanism as described in Section 5.4. The division of labour mechanism will cause an active forager robot to return to the sink if that robot can not find an item for a maximum amount of time. The inactive forager robot waits at the sink, until the inactive forager robot is recruited by a scout robot. Section 4.3 discussed that a mechanism of division of labour which could adjust the number of robots actively foraging to an appropriate number, would result in decreased levels of inter-robot interference and increased swarm scalability.

In order to determine whether the slight improvement in scalability of the honey bee algorithm over the desert ant algorithm can be attributed to the discussed division of labour mechanism, the average time spent by robots in the waiting state is plotted for different values of c in Figure 7.7. Figure 7.7 illustrates that the robots spend a significant portion of time in the waiting state for $c < 0.5$, which provides evidence that the honey bee algorithm's swarm scalability is greater than the desert ant algorithm's swarm scalability for low values of c.

However, the time spent by honey bee robots in the waiting state decreases as swarm density increases. This result is counter-intuitive, because an appropriately functioning division of labour mechanism would lead to an increase in average time in the waiting state as swarm density increases, due to increases in inter-robot interference. Further analysis is required in order to understand why the division of labour mechanism of the honey bee algorithm did not behave as expected. That said, the problem in the honey bee algorithm's division of labour mechanism does explain why the desert ant algorithm overtakes the honey bee algorithm for higher values of c: The honey bee algorithm is unable to regulate the number of active foragers for high values of c. The honey bee

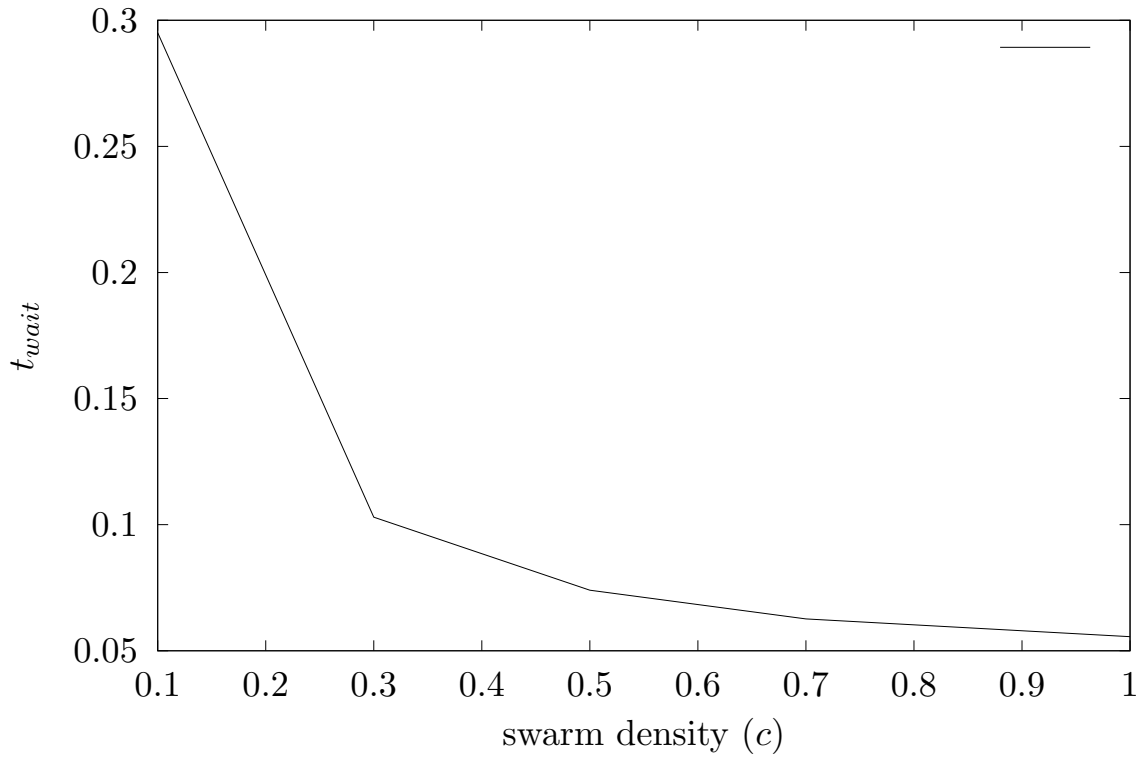


Figure 7.7: Average time in waiting state, t_{wait}, for each swarm density c, for the honey bee algorithm

algorithm's recruitment mechanism is more exploitative than the site fidelity mechanism of the desert ant algorithm, which results in increased inter-robot interference when the honey bee algorithm fails to regulate the number of active foragers. The increased inter-robot interference for the honey bee algorithm for high values of c results in a lower scalability than that of the desert ant algorithm.

7.3.2 Problem scalability

Figure 7.8 plots the problem scalability performance measure, S_p (described in Section 6.6.3), for each algorithm using the values given in Table 7.4. The results present the macro performance measure, S_p, which is calculated over all experiments, for each value of p. The figure includes a plot of "Expected scalability". The values for "Ex-

pected scalability" were calculated based on the assumption that foraging efficiency would degrade directly proportional to the increase in problem complexity. The expected values for foraging efficiency at each problem complexity were used to calculate S_p for "Expected scalability". All algorithms outperformed the expected scalability and thus problem scalability for all algorithms is better than the expected scalability which is good.

Table 7.4: Problem scalability, S_{p_i}, for each environment density, p_i, for each algorithm

p	**0.05**	**0.2**	**0.5**	**0.7**	**0.9**
desert ant	1	0.488	0.231	0.182	0.169
honey bee	1	0.602	0.283	0.217	0.196
naïve	1	0.502	0.257	0.195	0.175
Expected	1	0.25	0.1	0.071	0.056

According to Figure 7.8, the honey bee algorithm is the most scalable in terms of problem scalability, followed by the naïve algorithm, with the desert ant algorithm exhibiting the worst scalability.

In order to explain the reason for this, consider the following rational argument: The desert ant algorithm differs from the naïve algorithm in only one aspect: The desert ant algorithm employs site fidelity and the naïve algorithm does not. The site fidelity (discussed in Section 5.3) enables a robot to return to a previously foraged site. The desert ant algorithm's use of site fidelity must, directly or indirectly, be the reason that the naïve algorithm is more scalable than the desert ant algorithm. To understand why the site fidelity mechanism is negatively influencing scalability, consider the following scenario: A desert ant robot employs a random walk through a complex environment, as shown by the dashed line in Figure 7.9. The random walk leads the desert ant robot to a hard-to-reach source of prioritized items. The desert ant robot stores the PI vector, shown by the solid line on the figure, which is needed to return back to the site after offloading the loaded item at the sink. The next time the desert ant wants to return to the previous site, the robot will still need to perform the same obstacle avoidance in order to navigate around the obstacles to return to the previously foraged site, since the

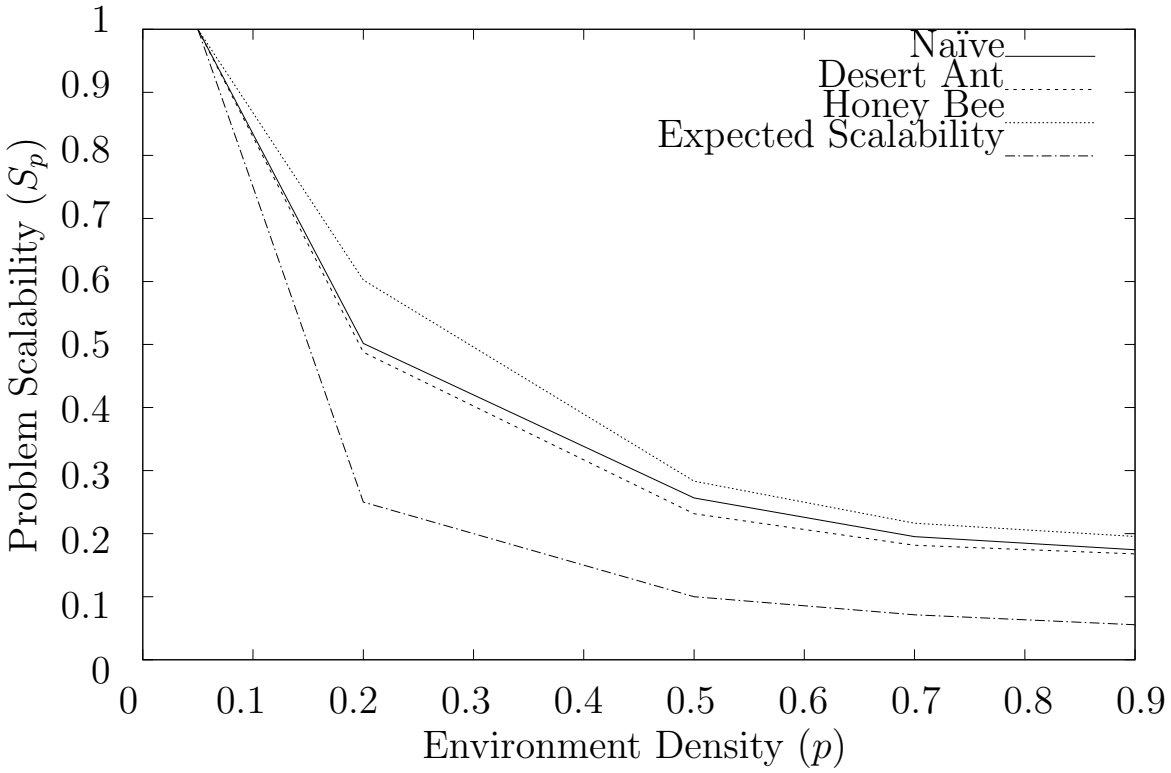


Figure 7.8: Problem Scalability, S_p, for each environment density p_i, for each algorithm

PI vector only consists of an overall heading and distance to the site. As is evident in Figure 7.9, there may exist an easier to reach site for prioritized items, but the desert ant robot will waste time by continuing to forage the hard to locate source of items. The naïve algorithm will not try to return to the hard to find site and is more likely to randomly find the prioritized resource site that is nearer to the sink (indicated by the dotted line), and thus the time taken to forage a single item will be faster, and so foraging efficiency will be increased. The time spent on relocating hard to access sites in more complex environments can slow the desert ant algorithm down. The naïve algorithm benefits from a more complex environment, because there will be lots of items to find nearer to the sink that do not require any complex search to relocate.

Additionally, in environments that are more dense, all desert ant robots are more likely to locate the same high density area. The more dense the area, the more robots

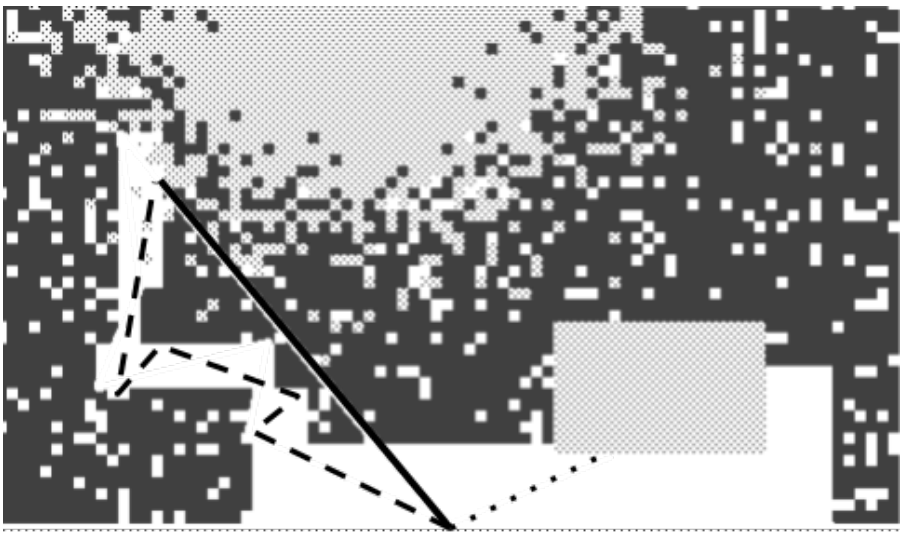


Figure 7.9: Illustration of the inefficiencies of the desert ant algorithm's site fidelity in dense environments. Solid areas are non-prioritized items, white areas are free cells, and shaded areas are prioritized items. The dashed path is a hypothetical random walk performed by a robot, the solid path is the PI vector for the dashed random walk, and the dotted path is an alternative random walk.

will try to forage the same area and the more inter-robot and environmental interference will occur. The desert ant robots try to exploit the same high density areas. On the other hand, the naïve algorithm will explore more. Thus the naïve algorithm will experience less inter-robot and environmental interference as a result.

The honey bee algorithm uses site fidelity, and also communicates those sites to others. Despite the use of a site fidelity mechanism, similar to the desert ant algorithm, the honey bee algorithm is the most scalable. The reason why the site fidelity of the honey bee algorithm does not impact on the scalability can be attributed the ability of the honey bee algorithm to adapt the swarm specialization ratio, τ, to focus on foraging dense areas of obstacles, rather than having to expensively navigate around the obstacles. The ability of the honey bee algorithm to concentrate on clearing obstacles in highly dense environments will increase overall ease of access to the high quality sites, as discovered in Section **??**. Thus efficiency of the honey bee algorithm in highly dense environments is improved. Therefore, the honey bee algorithm is the most scalable in

terms of the problem complexity.

7.4 Robustness

As discussed in Section 2.2.1, robot swarms achieve robustness by exhibiting the properties of redundancy, multiplicity of sensing, and decentralized coordination. This study does not perform an empirical robustness study to specifically address how fault tolerant each algorithm is. Instead, this section provides rational arguments supported by empirical evidence to discuss each algorithm's robustness. Section 7.4.1 discusses robustness in terms of the resulting redundancy of each of the algorithms, and Section 7.4.2 discusses robustness in terms of decentralized coordination.

7.4.1 Redundancy

As discussed in Section 2.2.1, redundancy in a swarm is achieved by giving all, or a portion of the robots in the swarm the same capabilities. In this way, if a percentage of the swarm malfunctions, the other robots have the capability to take the malfunctioning robots' place. A swarm demonstrates the highest level of redundancy when any robot can take on the role of any other robot that malfunctions in a swarm.

For each of our experiments, each swarm is configured with an initial swarm specialization ratio (as discussed in Chapter 6) to enable a portion of robots to forage prioritized items and the another portion to forage non-prioritized items.

If a portion of robots in a swarm malfunctions or are destroyed, then the swarm specialization ratio will be affected. As shown in Section 7.2, the foraging efficiency of the naïve and desert ant algorithms were effected by the initial swarm specialization ratio τ. It follows that, if the swarm specialization ratio is unexpectedly changed during the swarm experiment, a change in the foraging efficiency of the swarm would persist for the experiment. In particular, consider the following extreme scenarios: Suppose that all the robots which were foraging for prioritized items (for some $\tau > 0$) are destroyed or malfunction (thus $\tau = 0$). Foraging efficiency would drop to 0, because no robots exist to forage prioritized items. Similarly, consider an environment where all prioritized items are blocked by non-prioritized items and that $\tau < 1$. If all non-prioritized robots suffer

a malfunction or are destroyed, then $\tau = 1$ and no robots will be available to forage non-prioritized items. Foraging efficiency, E_P, will then drop to 0.

As shown in Section 7.2, the honey bee algorithm experiences little change to efficiency when $\tau(0)$ is varied. The honey bee algorithm adapts the swarm specialization ratio over time. It follows that, for the above scenarios, the honey bee algorithm will be able to re-adapt the swarm specialization ratio in order to replenish the robots that were destroyed.

The robots controlled by the honey bee algorithm are homogeneous, in that every robot in the swarm has the ability to take on any of the roles required for the algorithm to function (i.e. scout, unemployed forager, employed forager), as well as to adapt to forage either prioritized or non-prioritized items. It follows that the swarms running the honey bee algorithm are more redundant than swarms running the desert ant algorithm and the naïve algorithm.

7.4.2 Decentralized Coordination

Decentralized coordination can be attained by creating algorithms that do not depend on decisions, actions, or sensor readings of any single individual or a few individuals of the swarm. This section provides rational arguments to compare the decentralized coordination mechanisms for each algorithm, supported by evidence.

The robots of both the desert ant algorithm and naïve algorithm do not depend on each other and there is no explicit coordination mechanism. Despite no explicit coordination mechanism existing between robots in the desert ant and naïve swarms, the robots coordinate implicitly by using obstacle avoidance to avoid obstructing each other's movement. In that way, the robots in the swarm attempt to avoid inter-robot interference. The mechanism for obstacle avoidance is entirely decentralized, because each robot only depends on its own internal sensors and thus coordination is entirely decentralized.

On the other hand, the honey bee algorithm uses explicit communication as a coordination mechanism in order to recruit foragers to areas with a high density of prioritized items (as described in Section 5.4). If the scout robots experience a fault in evaluating the quality of a site (for example, a robot detects that a site is of high quality when it is

actually a site of poor quality), then foragers will be incorrectly recruited to forage sites of low quality. Only a few individuals (the scouts) can coordinate the swarm and therefore any faults in the scout robots may negatively influence the efficiency of the entire swarm. The coordination mechanism of the honey bee algorithm is not entirely decentralized and the honey bee algorithm is less robust than both the desert ant algorithm and the naïve algorithm in terms of decentralized coordination.

To provide further evidence that the scout robots can mislead the swarm to incorrectly forage areas that have low quality resources, an examination of the average time spent in the recruitment state, per item foraged, per robot has been done. Table 7.5 and Figure 7.10 illustrate the average time steps, $t_{recruitment}$, spent performing recruitment per robot, per (both prioritized and non-prioritized) item foraged for the honey bee algorithm, for each environment distribution. The results were averaged over each environment item type ratio, r.

Table 7.5: Average time steps, per robot, per prioritized item foraged, that were spent performing recruitment for the honey bee algorithm, in each environment distribution, for each environment item type ratio r.

r	**Clustered**	**Gaussian**	**Uniform**	**Vein**
0	0	0	0	0
0.2	0.119	0.152	0.120	0.161
0.25	0.127	0.144	0.128	0.159
0.33	0.140	0.135	0.139	0.152
0.5	0.162	0.157	0.163	0.169
0.67	0.182	0.175	0.183	0.152
0.75	0.179	0.177	0.184	0.172
0.8	0.183	0.183	0.186	0.191
1	0.167	0.173	0.163	0.219

Table 7.5 illustrates that, on average, each robot spent at least 11.96% of its total foraging time recruiting other robots to forage higher quality areas for all values of $r > 0$. This is particularly relevant for uniformly distributed environments with a low ratio of prioritized items ($r = 0.2$), because it is unlikely that sites of high quality do not exist,

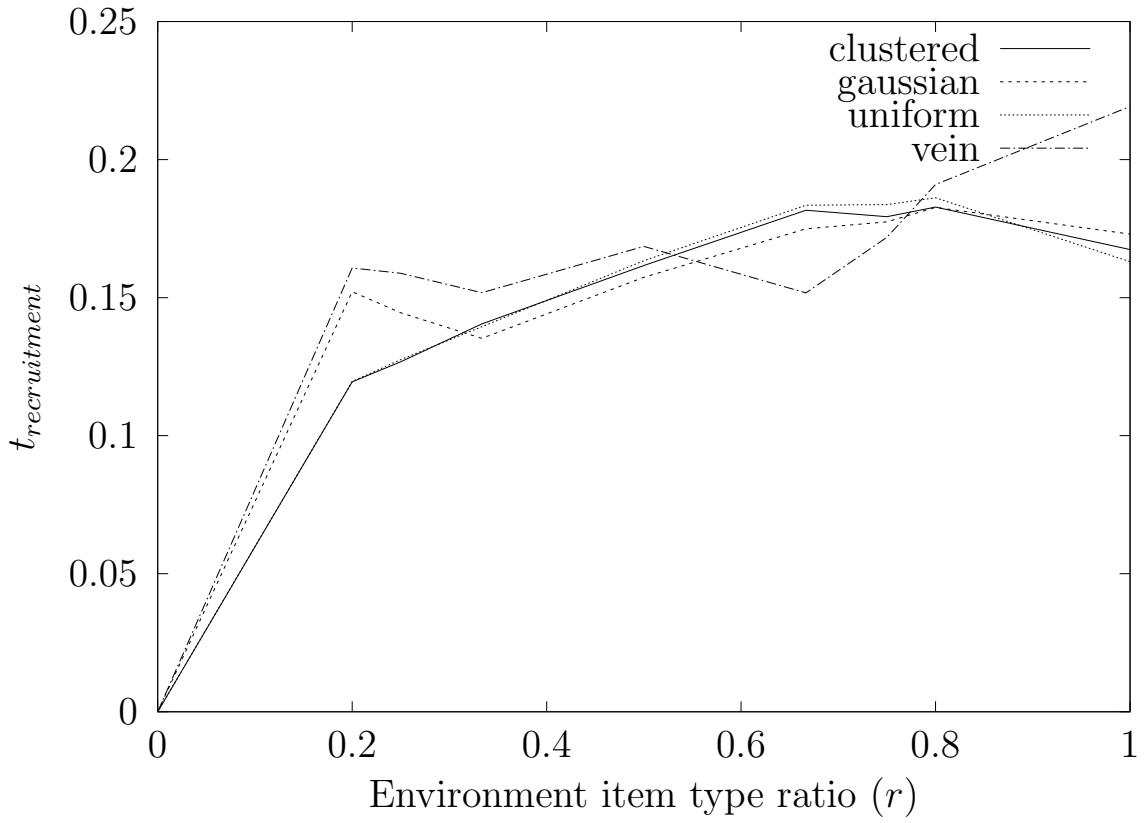


Figure 7.10: Average time steps spent performing recruitment activities by robots of the swarm for each item foraged, $t_{recruitment}$, for the honey bee algorithm, per item distribution, for each environment item type ratio, r.

because the prioritized items will be scarce and scattered. This means that the scouts were recruiting other robots to forage areas that were not of a high quality. The fact that scouts can mislead the swarm to foraging sites with low quality, demonstrates that the honey bee algorithm is less robust due to less decentralized coordination.

7.5 Summary

This chapter presented the results of the experiments and performed analysis of the major properties of swarm robotics, for each algorithm. The results were discussed with respect to four performance axes: efficiency, flexibility, robustness, and scalability.

Section 7.1 presented an overview of foraging efficiency of each algorithm. The results showed that the honey bee algorithm was the most efficient over all environments and swarm configurations, while the desert ant was the next most efficient and the naïve algorithm was the least efficient.

Section 7.2 analysed the flexibility of each algorithm. Flexibility was analysed in terms of flexibility over environments prioritized item ratio, F_r, and flexibility over environment distribution type, F_ϵ.

Section 7.2.1 concluded that the honey bee algorithm was the most flexible in terms of prioritized item ratio. The honey bee algorithm's flexibility was demonstrated to be a result of its ability to adapt the specialization ratio, τ. The division of labour mechanism allowed the honey bee robots to more efficiently forage an environment for any given environment item ratio, r. The naïve algorithm was more flexible than the desert ant algorithm in terms of environment item distribution. The naïve algorithm's flexibility is attributed to the fact that the naïve algorithm is less efficient across all environment item ratios. This suggests that the naïve algorithm appears flexible only because it performs equally bad across all values of r, unable to exploit any differences between the environments.

Section **??** concluded that the honey bee algorithm is the most flexible in terms of environmental distribution, followed by the desert ant algorithm, and lastly the naïve algorithm. The honey bee algorithm was determined to be more flexible, due to its ability to adapt the specialization ratio to better suit the environment ratio of the accessible environment. The adaptation of specialization ratio allowed the honey bee swarm to focus on foraging non-prioritized items when only non-prioritized items were accessible, and then adjust the ratio to focus on foraging prioritized items when prioritized items were accessible. The desert ant and naïve algorithms are less flexible than the honey bee algorithm. The naïve algorithm's perceived flexibility over environment distribution, compared to the desert ant algorithm, was attributed to the same reasons as determined by Section 7.2.1.

Section 7.3 evaluated scalability in terms of swarm scalability metrics, S_c and S_p. Section 7.3.1 determined that the scalability of all algorithms, in terms of swarm scalability, was sub-linear. More specifically, the naïve algorithm was the most scalable in terms

of swarm density, followed by the desert ant algorithm and honey bee algorithm which exhibited similar scalability. The desert ant algorithm's poor performance, compared to the naïve algorithm, was attributed the desert ant algorithm's use of site fidelity. A rational argument was presented that argued that the desert ant algorithm's site fidelity increased inter-robot interference, which resulted in decreased efficiency when swarm density was high. The honey bee algorithm was shown to be slightly more scalable than the desert ant algorithm, due to the honey bee algorithm's attempt to regulate the number of active foragers by division of labour. The difference in swarm scalability between the desert ant algorithm and the honey bee algorithm was very small, suggesting that the regulation of active foragers was not functioning very well. The honey bee algorithm's ability to regulate the number of active foragers as swarm density increases was shown to be ineffective.

Section 7.3.2 determined that the honey bee algorithm is the most scalable in terms of problem scalability, followed by the naïve algorithm, and lastly the desert ant algorithm. The discussion revealed that the desert ant algorithm performed comparatively worse than the naïve algorithm in terms of problem scalability, due to the desert ant algorithm's use of site fidelity to exploit good search areas. The section determined the naïve algorithm's ability to explore more than the desert ant algorithm, resulting in better problem scalability. The honey bee algorithm was the most scalable in terms of problem scalability. The problem scalability of the honey bee algorithm can be attributed to the ability of the honey bee algorithm to adapt τ to help clear non-prioritized items, which resulted in lower inter-robot and environmental interference, which in turn increased the foraging efficiency.

Section 7.4 addressed robustness of each algorithm in terms of redundancy and decentralized coordination. Section 7.4.1 illustrated that the honey bee algorithm is the most redundant, since the swarm is homogeneous. The honey bee swarm is homogeneous because the robots can switch specialization, rather than being set to forage only a specific item type. The desert ant algorithm and naïve algorithm robots can only forage a single item type, which is pre-configured, and thus their swarms are heterogeneous, and thus the swarms are less redundant.

Section 7.4.2 determined that the desert ant and naïve algorithms are more robust

than the honey bee algorithm in terms of decentralized coordination, because neither coordination mechanism between robots of the swarm are completely decentralized. The honey bee algorithm was determined to be the least robust in terms of decentralized coordination. The honey bee algorithm's coordination mechanism was discovered to be sensitive to faults in certain individuals where invalid information was communicated to the swarm, and the invalid information impacted efficiency.

Chapter 8

Conclusions

This chapter provides an overview of the conclusions arrived at in this work. A synopsis of the contributions and experimental findings of this thesis are presented in Section 8.1, while Section 8.2 outlines possible future work that can be performed.

8.1 Summary of Conclusions

The first contribution of this research was the introduction of a novel variation of the swarm robotics multi-foraging problem, denoted as the prioritized foraging problem. The prioritized foraging problem differs from other multi-foraging swarm robotics problems in that two types of items which have different priorities exist. The items with higher priorities should be foraged as fast as possible, while the non-prioritized items should only be foraged to enable the prioritized items to be foraged at a faster rate. The prioritized foraging problem is a novel way of modelling the real-world problem of search and rescue.

Existing naïve and desert ant algorithms were reviewed and selected for evaluation on the prioritized foraging problem. Furthermore, a novel honey bee inspired foraging algorithm was developed, which specifically modelled the recruitment mechanism and division of labour mechanism of honey bee swarms.

Emperical experiments were defined in Chapter 6, and an emperical analysis was performed in Chapter 7. The purpose of the emperical analysis was to investigate each

algorithm's performance on the prioritized foraging problem in terms of the major swarm robotics characteristics of efficiency, scalability, flexibility, and robustness as well as the behaviours that enabled those characteristics.

Each algorithm's efficiency was compared to each other algorithm's efficency using a wins and losses approach. Results showed that the honey bee algorithm was the most efficient over all environments and swarm configurations, while the desert ant was the next most efficient and the naïve algorithm was the least efficient.

The flexibility of the algorithms was analysed in terms of the flexibility over environmental item ratio, and the flexibility over different environment distribution types. Results indicated that the honey bee algorithm could adapt the swarm's specialization ratio to effectively forage a given environment item ratio, and was therefore the most flexible in terms of environmental item ratio. The naïve algorithm only appeared more flexible than the desert ant algorithm because the naïve algorithm performed equally poorly across all environment item distributions, where as the desert algorithm favoured particular item distributions.

The honey bee algorithm was the most flexible over different types of environmental distributions, followed by the desert ant algorithm, and lastly the naïve algorithm. The honey bee algorithm's high flexibility was attributed to its adaptation of the specialization ratio to better suit the environment ratio of the accessible environment. The honey bee algorithm's adaptation of specialization ratio allowed the swarm to focus on foraging non-prioritized items when only non-prioritized items were accessible, and then to adjust the ratio to focus on foraging prioritized items when more prioritized items became accessible, and vice versa. The desert ant and naïve algorithms were less flexible than the honey bee algorithm, due to their inability to adapt the initial swarm specialization ratio.

The scalability of each algorithm was analyzed in terms of swarm scalability and problem scalability. Swarm scalability was sub-linear for all algorithms, however the naïve algorithm was the most scalable. The desert ant algorithm had poor swarm scalability when compared to the naïve algorithm. The desert ant algorithm's site fidelity increased inter-robot interference, which resulted in decreased efficiency when swarm density was high. The honey bee algorithm was shown to be slightly more scalable than the desert

ant algorithm, due to the honey bee algorithm's attempt to regulate the number of active foragers by division of labour. Future investigation is necessary to determine why the honey bee algorithm did not regulate the number of foragers at high swarm densities very well.

The honey bee algorithm was the most scalable in terms of the problem density, followed by the naïve algorithm, and lastly the desert ant algorithm. The desert ant algorithm performed comparatively better than the naïve algorithm in terms of problem scalability. It was theorized that the desert ant algorithm could potentially exploit hard to reach sites in complex environments. The problem scalability of the honey bee algorithm was attributed the honey bee algorithm's ability to adapt the swarm specialization ratio to help clear non-prioritized items. By clearing the non-prioritized items, the honey bee algorithm decreased inter-robot and environmental interference, which in turn increased the foraging efficiency in problems of high complexity.

Lastly, the robustness of each algorithm was evaluated in terms of redundancy and decentralized coordination. A rational argument motivated that the honey bee algorithm was the most redundant, because the honey bee algorithm swarm is completely homogeneous. The robots of the desert ant algorithm and naïve algorithm are heterogeneous, and thus the swarm is less redudant. The coordination between robots of the desert ant and naïve algorithms was determined to be more decentralized than the coordination of robots in the honey bee algorithm. Evidence showed that invalid information was communicated to the rest of swarm by individuals with faulty information, impacting efficiency and showing that communication is less decentralized.

In summation, this dissertation primarily contributed the prioritized foraging problem and the novel honey bee inspired foraging swarm robotic algorithm. The emperical analysis provided a detailed view of how well each algorithm embodied each major characteristic of swarm robotics on the prioritized foraging problem.

8.2 Future Work

The following is a list of potential future studies that could follow from the work presented in this dissertation:

1. A thorough emperical study of the robustness of the described algorithm can be performed. A portion of the swarm can be destroyed during foraging and the impact on the swarm efficiency can be analysed.

2. An analysis of the impact of using different division of labour strategies for the honey bee algorithm can be done. The study should additionally determine why the division of labour strategy used in this thesis did not regulate the number of active foragers very well.

3. A more detailed investigation into the performance of each algorithm on each type of environment distribution can be done to provide more insight into each algorithm's behaviour on different environment types.

4. An extensive analysis of the efficiency of each algorithm, with optimal swarm parameters for each algorithm on each environment can be performed.

5. The implementation of the described problem and algorithms on a real robot swarm can be done, to determine feasability of the algorithms in real world applications.

Bibliography

[1] Robin R Murphy, Satoshi Tadokoro, Daniele Nardi, Adam Jacoff, Paolo Fiorini, Howie Choset, and Aydan M Erkmen. Search and rescue robotics. In *Springer Handbook of Robotics*, pages 1151–1173. Springer, 2008.

[2] Amir M Naghsh, Jeremi Gancet, Andry Tanoto, and Chris Roast. Analysis and design of human-robot swarm interaction in firefighting. In *Proceedings of the 17th IEEE International Symposium on Robot and Human Interactive Communication*, pages 255–260. IEEE, 2008.

[3] Marco Dorigo and Erol Şahin. Swarm robotics. *Autonomous Robots*, 17:111–113, 2004.

[4] Alan F. T. Winfield. Foraging Robots. In *Encyclopedia of Complexity and Systems Science*, pages 3682–3700. 2009.

[5] Erol Şahin. Swarm robotics: From sources of inspiration to domains of application. In *Swarm Robotics*, pages 10–20. Springer, 2005.

[6] Nicholas R Hoff, Amelia Sagoff, Robert J Wood, and Radhika Nagpal. Two foraging algorithms for robot swarms using only local communication. In *Proceedings of the 2010 IEEE International Conference on Robotics and Biomimetics*, pages 123–130. IEEE, 2010.

[7] Simon Garnier, Christian Jost, Raphaël Jeanson, Jacques Gautrais, Masoud Asadpour, Gilles Caprari, and Guy Theraulaz. Aggregation behaviour as a source of collective decision in a group of cockroach-like-robots. In *Advances in Artificial Life*, pages 169–178. Springer, 2005.

[8] Jong-Hyun Lee, Hyun-Tae Kim, and Chang Wook Ahn. Foraging swarm robots system adopting honey bee swarm for improving energy efficiency. In *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication*, page 100. ACM, 2012.

[9] Edward O Wilson. *The Insect Societies*. Cambridge, Massachusetts, USA, Harvard University Press, 1971.

[10] Thomas Schmickl and Karl Crailsheim. A navigation algorithm for swarm robotics inspired by slime mold aggregation. In *Swarm Robotics*, pages 1–13. Springer, 2007.

[11] Amit Dhariwal, Gaurav Sukhatme, and Aristides AG Requicha. Bacterium-inspired robots for environmental monitoring. In *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*, volume 2, pages 1436–1443. IEEE, 2004.

[12] Sylvain Martel and Mahmood Mohammadi. Using a swarm of self-propelled natural microrobots in the form of flagellated bacteria to perform complex micro-assembly tasks. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 500–505. IEEE, 2010.

[13] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013.

[14] Derek Wragge Morley. Division of labour in ants. *Nature*, 158:913–914, 1946.

[15] Samuel N Beshers and Jennifer H Fewell. Models of division of labor in social insects. *Annual Review of Entomology*, 46(1):413–440, 2001.

[16] Brian P Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954, 2004.

[17] Thomas H Labella, Marco Dorigo, and Jean-Louis Deneubourg. Division of labor in a group of robots inspired by ants' foraging behavior. *ACM Transactions on Autonomous and Adaptive Systems*, 1(1):4–25, 2006.

[18] Wenguo Liu, Alan F.T. Winfield, Jin Sa, Jie Chen, and Lihua Dou. Towards energy optimization: Emergent task allocation in a swarm of foraging robots. *Adaptive Behavior*, 15(3):289–305, 2007.

[19] E Bahçeci and Erol Şahin. Evolving aggregation behaviors for swarm robotic systems: A systematic case study. In *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, pages 333–340. IEEE, 2005.

[20] Shervin Nouyan, Alexandre Campo, and Marco Dorigo. Path formation in a robot swarm. *Swarm Intelligence*, 2(1):1–23, 2008.

[21] Dimitri Zarzhitsky, Diana F Spears, and William M Spears. Distributed robotics approach to chemical plume tracing. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4034–4039. IEEE, 2005.

[22] Tucker Balch. Hierarchic social entropy: An information theoretic measure of robot group diversity. *Autonomous Robots*, 8(3):209–238, 2000.

[23] Michael Rubenstein, Christian Ahler, and Radhika Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3293–3298. IEEE, 2012.

[24] Alex Kushleyev, Daniel Mellinger, Caitlin Powers, and Vijay Kumar. Towards a swarm of agile micro quadrotors. *Autonomous Robots*, 35(4):287–300, 2013.

[25] Daniel Mellinger, Michael Shomin, Nathan Michael, and Vijay Kumar. Cooperative grasping and transport using multiple quadrotors. In *Distributed Autonomous Robotic Systems*, pages 545–558. Springer, 2013.

[26] Francesco Mondada, Dario Floreano, André Guignard, Jean-Louis Deneubourg, Luca Gambardella, Stefano Nolfi, and Marco Dorigo. Search for rescue: An application for the SWARM-BOT self-assembling robot concept, 2002.

[27] Tucker Balch, Gary Boone, Thomas Collins, Harold Forbes, Doug MacKenzie, and Juan Carlos Santamar. Io, Ganymede, and Callisto - a multiagent robot trash-collecting team. *AI Magazine*, 16(2):39, 1995.

[28] Nikolaus Correll and Alcherio Martinoli. A challenging application in swarm robotics: The autonomous inspection of complex engineered structures. In *Bulletin of the Swiss Society for Automatic Control*. Citeseer, 2007.

[29] Tucker Balch and Ronald C Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.

[30] Marco Dorigo, Mauro Birattari, and Manuele Brambilla. Swarm robotics. *Scholarpedia*, 9(1):1463, 2014.

[31] Gerardo Beni and Jing Wang. Swarm intelligence in cellular robotic systems. In *Robots and Biological Systems: Towards a New Bionics?*, pages 703–712. Springer, 1993.

[32] Thomas D Seeley. *The Wisdom of the Hive: The Social Physiology of Honey Bee Colonies*. Harvard University Press, 2009.

[33] Rodney A Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6(1):3–15, 1990.

[34] Rodney Brooks and others. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

[35] Jonathan H Connell. A colony architecture for an artificial creature, 1989.

[36] Maja J Mataric. A distributed model for mobile robot environment-learning and navigation, 1990.

[37] Anita M Flynn, Rodney Brooks, William M Wells III, David S Barrett, and others. The world's largest one cubic inch robot. In *Proceedings of the 1989 International Conference of Micro Electro Mechanical Systems, An Investigation of Micro Structures, Sensors, Actuators, Machines and Robots*, pages 98–101. IEEE, 1989.

[38] Rodney A Brooks. A robot that walks; emergent behaviors from a carefully evolved network. *Neural Computation*, 1(2):253–262, 1989.

[39] Ronald C Arkin. Integrating behavioral, perceptual, and world knowledge in reactive navigation. *Robotics and Autonomous Systems*, 6(1):105–122, 1990.

[40] Tamio Arai, Enrico Pagello, and Lynne E Parker. Editorial: Advances in multi-robot systems. *IEEE Transactions on Robotics and Automation*, 18(5):655–661, 2002.

[41] JH Sudd. A model of digging behaviour and tunnel production in ants. *Insectes Sociaux*, 22(2):225–235, 1975.

[42] Randall T Ryti and Ted J Case. Spatial arrangement and diet overlap between colonies of desert ants. *Oecologia*, 62(3):401–404, 1984.

[43] Thomas D Seeley, Scott Camazine, and James Sneyd. Collective decision-making in honey bees: How colonies choose among nectar sources. *Behavioral Ecology and Sociobiology*, 28(4):277–290, 1991.

[44] Han de Vries and Jacobus C Biesmeijer. Modelling collective foraging by means of individual behaviour rules in honey-bees. *Behavioral Ecology and Sociobiology*, 44(2):109–124, 1998.

[45] Joaquin L Reyes Lopez. Optimal foraging in seed-harvester ants: Computer-aided simulation. *Ecology*, 68:1630–1633, 1987.

[46] C Ronald Kube and Hong Zhang. Collective robotics: From social insects to robots. *Adaptive Behavior*, 2(2):189–218, 1993.

[47] E Freund. On the design of multi-robot systems. In *Proceedings of the 1984 IEEE International Conference on Robotics and Automation*, volume 1, pages 477–490. IEEE, 1984.

[48] Y Uny Cao, Alex S Fukunaga, and Andrew Kahng. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4(1):7–27, 1997.

[49] Hajime Asama, Toshio Fukuda, Tamio Arai, and Isao Endo. *Distributed Autonomous Robotic Systems 2*. Springer Science & Business Media, 2013.

[50] Maja J Mataric, Martin Nilsson, and Kristian T Simsarin. Cooperative multi-robot box-pushing. In *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems 'Human Robot Interaction and Cooperative Robots'*, volume 3, pages 556–561. IEEE, 1995.

[51] E Freund and H Hoyer. Pathfinding in multi-robot systems: Solution and applications. In *Proceedings of the 1986 IEEE International Conference on Robotics and Automation*, volume 3, pages 103–111. IEEE, 1986.

[52] Hajime Asama, Akihiro Matsumoto, and Yoshiki Ishida. Design of an autonomous and distributed robot system: ACTRESS. In *Proceedings of the 1989 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 283–290, 1989.

[53] Toshio Fukuda, Yoshio Kawauchi, and Martin Buss. Communication method of cellular robotics CEBOT as a selforganizing robotic system. In *Proceedings of the 1989 IEEE/RSJ International Workshop on Intelligent Robots and Systems'. The Autonomous Mobile Robots and Its Applications*, pages 291–296. IEEE, 1989.

[54] Toshio Fukuda, Yoshio Kawauchi, and Hajime Asama. Analysis and evaluation of cellular robotics (CEBOT) as a distributed intelligent system by communication information amount. In *Proceedings of the 1990 IEEE International Workshop on Intelligent Robots and Systems' 'Towards a New Frontier of Applications'*, pages 827–834. IEEE, 1990.

[55] Gerardo Beni and Jing Wang. Theoretical problems for the realization of distributed robotic systems. In *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*, pages 1914–1920. IEEE, 1991.

[56] Richard Vaughan. Massively multi-robot simulation in stage. *Swarm Intelligence*, 2:189–208, 2008.

[57] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, and others. ARGoS: A modular, multi-engine simulator for heterogeneous swarm robotics. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5027–5034. IEEE, 2011.

[58] Alcherio Martinoli, Auke Jan Ijspeert, and Francesco Mondada. Understanding collective aggregation mechanisms: From probabilistic modelling to experiments with real robots. *Robotics and Autonomous Systems*, 29(1):51–63, 1999.

[59] Kristina Lerman, Aram Galstyan, Alcherio Martinoli, and Auke Ijspeert. A macroscopic analytical model of collaboration in distributed robotic systems. *Artificial Life*, 7(4):375–393, 2001.

[60] Kristina Lerman and Aram Galstyan. Mathematical model of foraging in a group of robots: Effect of interference. *Autonomous Robots*, 13(2):127–141, 2002.

[61] Vito Trianni, Thomas H Labella, Roderich Gross, Erol Şahin, Marco Dorigo, and Jean-Louis Deneubourg. Modeling pattern formation in a swarm of self-assembling robots. *European Community Grant 1st-2000-31010*, 2002.

[62] Alexandre Campo and Marco Dorigo. Efficient multi-foraging in swarm robotics. In *Advances in Artificial Life*, pages 696–705. Springer, 2007.

[63] Heiko Hamann and Heinz Wörn. A framework of space–time continuous models for algorithm design in swarm robotics. *Swarm Intelligence*, 2:209–239, 2008.

[64] Amanda Prorok, Nikolaus Corell, and Alcherio Martinoli. Multi-level spatial modeling for stochastic distributed robotic systems. *The International Journal of Robotics Research*, 30:574–589, 2011.

[65] Veysel Gazi and Kevin M Passino. Stability of a one-dimensional discrete-time asynchronous swarm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 35(4):834–841, 2005.

[66] Yanfei Liu and Kevin M Passino. Stable social foraging swarms in a noisy environment. *IEEE Transactions on Automatic Control*, 49(1):30–44, 2004.

[67] Mac Schwager, Nathan Michael, Vijay Kumar, and Daniela Rus. Time scales and stability in networked multi-robot systems. In *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3855–3862. IEEE, 2011.

[68] Alan F.T. Winfield, Jin Sa, Mari-Carmen Fernández-Gago, Clare Dixon, and Michael Fisher. On formal specification of emergent behaviours in swarm robotic systems. *International Journal of Advanced Robotic Systems*, 2(4):363–370, 2005.

[69] Savas Konur, Clare Dixon, and Michael Fisher. Analysing robot swarm behaviour via probabilistic model checking. *Robotics and Autonomous Systems*, 60(2):199–213, 2012.

[70] Nithin Mathews, Anders Lyhne Christensen, Eliseo Ferrante, Rehan O'Grady, and Marco Dorigo. Establishing spatially targeted communication in a heterogeneous robot swarm. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: Volume 1-Volume 1*, pages 939–946. International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[71] Thomas H Labella, Marco Dorigo, and Jean-Louis Deneubourg. Efficiency and task allocation in prey retrieval. In *Biologically Inspired Approaches to Advanced Information Technology*, pages 274–289. Springer, 2004.

[72] Onur Soysal and others. Probabilistic aggregation strategies in swarm robotic systems. In *Proceedings of the 2005 IEEE Swarm Intelligence Symposium*, pages 325–332. IEEE, 2005.

[73] Derek J Bennet and Colin R McInnes. Distributed control of multi-robot systems using bifurcating potential fields. *Robotics and Autonomous Systems*, 58(3):256–264, 2010.

[74] Laura Barnes, Mary-Anne Fields, and Kimon Valavanis. Unmanned ground vehicle swarm formation control using potential fields. In *Proceedings of the Mediterranean Conference on Control & Automation*, pages 1–8. IEEE, 2007.

[75] Dong Hun Kim, Hua Wang, and Seiichi Shin. Decentralized control of autonomous swarm systems using artificial potential functions: Analytical design guidelines. *Journal of Intelligent and Robotic Systems*, 45(4):369–394, 2006.

[76] Kazuyuki Samejima and Takashi Omori. Adaptive internal state space construction method for reinforcement learning of a real-world agent. *Neural Networks*, 12(7):1143–1155, 1999.

[77] Ron Sun and Todd Peterson. Multi-agent reinforcement learning: Weighting and partitioning. *Neural Networks*, 12(4):727–753, 1999.

[78] Gianluca Baldassarre, Stefano Nolfi, and Domenico Parisi. Evolving mobile robots able to display collective behaviors. *Artificial Life*, 9(3):255–267, 2003.

[79] Elio Tuci. Evolutionary Swarm Robotics: Genetic Diversity, Task-Allocation and Task-Switching. In *Swarm Intelligence*, pages 98–109. Springer, 2014.

[80] Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Vito Trianni, and Mauro Birattari. AutoMoDe: A novel approach to the automatic design of control software for robot swarms. *Swarm Intelligence*, 8(2):89–112, 2014.

[81] Gianpiero Francesca, Manuele Brambilla, Arne Brutschy, Lorenzo Garattoni, Roman Miletitch, Gaetan Podevijn, Andreagiovanni Reina, Touraj Soleymani, Mattia

Salvaro, Carlo Pinciroli, and others. An experiment in automatic design of robot swarms. In *Swarm Intelligence*, pages 25–37. Springer, 2014.

[82] Auke Jan Ijspeert, Alcherio Martinoli, Aude Billard, and Luca Maria Gambardella. Collaboration through the exploitation of local interactions in autonomous collective robotics: The stick pulling experiment. *Autonomous Robots*, 11(2):149–171, 2001.

[83] Ryusuke Fujisawa, Shigeto Dobata, Ken Sugawara, and Fumitoshi Matsuno. Designing pheromone communication in swarm robotics: Group foraging behavior mediated by chemical substance. *Swarm Intelligence*, 8(3):227–246, 2014.

[84] Eric J Barth. A dynamic programming approach to robotic swarm navigation using relay markers. In *Proceedings of the 2003 American Control Conference*, volume 6, pages 5264–5269. IEEE, 2003.

[85] Scott Camazine. *Self-Organization in Biological Systems*. Princeton University Press, 2003.

[86] Xinan Yan, Alei Liang, and Haibing Guan. An algorithm for self-organized aggregation of swarm robotics using timer. In *Proceedings of the 2011 IEEE Symposium on Swarm Intelligence*, pages 1–7. IEEE, 2011.

[87] Onur Soysal, Erkin Bahçeci, and Erol Şahin. Aggregation in swarm robotic systems: Evolution and probabilistic control. *Turkish Journal Electrical Engineering*, 15(2):199–225, 2007.

[88] Vito Trianni, Roderich Groß, Thomas H Labella, Erol Şahin, and Marco Dorigo. Evolving aggregation behaviors in a swarm of robots. In *Advances in Artificial Life*, pages 865–874. Springer, 2003.

[89] Thomas Schmickl and Heiko Hamann. *BEECLUST: A Swarm Algorithm Derived from Honeybees*. CRC Press, 2011.

[90] Thomas Schmickl, Heiko Hamann, Heinz Wörn, and Karl Crailsheim. Two different approaches to a macroscopic model of a bio-inspired robotic swarm. *Robotics and Autonomous Systems*, 57(9):913–921, 2009.

[91] Erkin Bahceci, Onur Soysal, and Erol Şahin. A review: Pattern formation and adaptation in multi-robot systems. *Robotics Institute, Carnegie Mellon University, Pittsburgh, Technical Report, CMU-RI-TR-03-43*, 2003.

[92] Suranga Hettiarachchi, William M Spears, Blesson Varghese, and Gerard McKee. A review and implementation of swarm pattern formation and transformation models. *International Journal of Intelligent Computing and Cybernetics*, 2(4):786–817, 2009.

[93] Shervin Nouyan and Marco Dorigo. Chain based path formation in swarms of robots. In *Ant Colony Optimization and Swarm Intelligence*, pages 120–131. Springer, 2006.

[94] Simon Goss and Jean-Louis Deneubourg. Harvesting by a group of robots. In *Proceedings of the First European Conference on Artificial Life*, pages 195–204, 1992.

[95] Barry Brian Werger and Maja J Mataric. Robotic" food" chains: Externalization of state and program for minimal-agent foraging. In *In (Maes et Al.* Citeseer, 1996.

[96] Shervin Nouyan, Roderich Groß, Marco Dorigo, Michael Bonani, and Francesco Mondada. Group Transport Along a Robot Chain in a Self-Organised Robot Colony. In *Proceedings of the 9th International Conference on Intelligent Autonomous Systems*, pages 433–442, 2006.

[97] Chris R Reid, Matthew J Lutz, Scott Powell, Albert B Kao, Iain D Couzin, and Simon Garnier. Army ants dynamically adjust living bridges in response to a cost–benefit trade-off. *Proceedings of the National Academy of Sciences*, 112(49):15113–15118, 2015.

[98] Helen F McCreery and MD Breed. Cooperative transport in ants: a review of proximate mechanisms. *Insectes Sociaux*, 61(2):99–110, 2014.

[99] Roderich Großand Marco Dorigo. Self-assembly at the macroscopic scale. *Proceedings of the IEEE*, 96(9):1490–1508, 2008.

[100] Ralph Beckers, OE Holland, and Jean-Louis Deneubourg. From local actions to global tasks: Stigmergy and collective robotics. In *Artificial Life IV*, volume 181, page 189, 1994.

[101] Andrew Howard, Maja J Matarić, and Gaurav S Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Distributed Autonomous Robotic Systems 5*, pages 299–308. Springer, 2002.

[102] Timothy Stirling and Dario Floreano. Energy efficient swarm deployment for search in unknown environments. In *Swarm Intelligence*, pages 562–563. Springer, 2010.

[103] David W Payton, Michael J Daily, Bruce Hoff, Michael D Howard, and Craig L Lee. Pheromone robotics. In *Intelligent Systems and Smart Manufacturing*, pages 67–75. International Society for Optics and Photonics, 2001.

[104] Frederick Ducatelle, Gianni A Di Caro, Carlo Pinciroli, and Luca M Gambardella. Self-organized cooperation between robotic swarms. *Swarm Intelligence*, 5(2):73–96, 2011.

[105] Julia K Parrish, Steven V Viscido, and Daniel Grünbaum. Self-organized fish schools: An examination of emergent properties. *The Biological Bulletin*, 202(3):296–305, 2002.

[106] Ali E Turgut, Hande Celikkanat, Fatih Gökçe, and Erol Şahin. Self-organized flocking in mobile robot swarms. *Swarm Intelligence*, 2:97–120, 2008.

[107] Eliseo Ferrante, Ali Emre Turgut, Nithin Mathews, Mauro Birattari, and Marco Dorigo. Flocking in stationary and non-stationary environments: A novel communication strategy for heading alignment. In *Parallel Problem Solving from Nature, PPSN XI*, pages 331–340. Springer, 2010.

[108] Hiroshi Sugie, Yasuhiro Inagaki, Shuchir Ono, Hidyuki Aisu, and Tatsuo Unemi. Placing objects with multiple mobile robots-mutual help using intention inference. In *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*, volume 2, pages 2181–2186. IEEE, 1995.

[109] Brian P Gerkey and Maja J Mataric. Sold!: Auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, 2002.

[110] Roderich Großand Marco Dorigo. Group transport of an object to a target that only some group members may sense. In *Parallel Problem Solving from Nature-VIII*, pages 852–861. Springer, 2004.

[111] Marco Dorigo, Elio Tuci, Roderich Groß, Vito Trianni, Thomas Halva Labella, Shervin Nouyan, Christos Ampatzis, Jean-Louis Deneubourg, Gianluca Baldassarre, Stefano Nolfi, and others. The swarm-bots project. In *Swarm Robotics*, pages 31–44. Springer, 2005.

[112] Eliseo Ferrante, Manuele Brambilla, Mauro Birattari, and Marco Dorigo. Socially-mediated negotiation for obstacle avoidance in collective transport. In *Distributed Autonomous Robotic Systems*, pages 571–583. Springer, 2013.

[113] Thomas D Seeley and Susannah C Buhrman. Nest-site selection in honey bees: How well do swarms implement the" best-of-N" decision rule? *Behavioral Ecology and Sociobiology*, 49(5):416–427, 2001.

[114] Jacques Gautrais, Guy Theraulaz, Jean-Louis Deneubourg, and Carl Anderson. Emergent polyethism as a consequence of increased colony size in insect societies. *Journal of Theoretical Biology*, 215(3):363–373, 2002.

[115] Chris Jones and Maja J Mataric. Adaptive division of labor in large-scale minimalist multi-robot systems. In *Proceedings of Intelligent Robots and Systems, 2003*, volume 2, pages 1969–1974. IEEE, 2003.

[116] Michael JB Krieger and Jean-Bernard Billeter. The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems*, 30(1):65–84, 2000.

[117] Erol Şahin and Alan Winfield. Special issue on swarm robotics. *Swarm Intelligence*, 2(2):69–72, 2008.

[118] Sean Luke, Claudio Cioffi-Revilla, Liviu Panait, Keith Sullivan, and Gabriel Balan. Mason: A multiagent simulation environment. *Simulation*, 81(7):517–527, 2005.

[119] Olivier Michel. Webots: Symbiosis between virtual and real mobile robots. In *Virtual Worlds*, pages 254–263. Springer, 1998.

[120] Bert Hölldobler. *The Ants*. Harvard University Press, 1990.

[121] Ruth A Bernstein. Seasonal food abundance and foraging activity in some desert ants. *American Naturalist*, pages 490–498, 1974.

[122] Mitchel Resnick. *Turtles, Termites, and Traffic Jams: Explorations in Massively Parallel Microworlds*. Mit Press, 1994.

[123] James S Jennings, Greg Whelan, and William F Evans. Cooperative search and rescue with a team of mobile robots. In *Proceedings of the 8th International Conference on Advanced Robotics, 1997*, pages 193–200. IEEE, 1997.

[124] Robin R Murphy. Biomimetic search for urban search and rescue. In *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2073–2078. IEEE, 2000.

[125] Marco Dorigo, Eric Bonabeau, and Guy Theraulaz. Ant algorithms and stigmergy. *Future Generation Computer Systems*, 16(8):851–871, 2000.

[126] Marco Dorigo. *Ant Colony Optimization and Swarm Intelligence: 5th International Workshop, ANTS 2006, Brussels, Belgium, September 4-7, 2006, Proceedings*, volume 4150. Springer-Verlag New York Incorporated, 2006.

[127] Marco Dorigo and Mauro Birattari. Ant colony optimization. In *Encyclopedia of Machine Learning*, pages 36–39. Springer, 2010.

[128] Thomas S Collett, Elisabeth Dillmann, A Giger, and Rüdiger Wehner. Visual landmarks and route following in desert ants. *Journal of Comparative Physiology A*, 170(4):435–442, 1992.

[129] Joshua P Hecker and Melanie E Moses. Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms. *Swarm Intelligence*, 9(1):43–70, 2015.

[130] Matthew Collett, Thomas S Collett, Sonja Bisch, and Rüdiger Wehner. Local and global vectors in desert ant navigation. *Nature*, 394(6690):269–272, 1998.

[131] Rudiger Wehner. Desert ant navigation: How miniature brains solve complex tasks. *Journal of Comparative Physiology A*, 189(8):579–588, 2003.

[132] Martin Müller and Rüdiger Wehner. Path integration in desert ants, Cataglyphis fortis. *Proceedings of the National Academy of Sciences*, 85(14):5287–5290, 1988.

[133] Ralf Möller, Dimitrios Lambrinos, Rolf Pfeifer, Thomas Labhart, and Rüdiger Wehner. Modeling ant navigation with an autonomous agent. *From Animals to Animats*, 5:185–194, 1998.

[134] Joshua P Hecker, Kenneth Letendre, Karl Stolleis, Daniel Washington, and Melanie E Moses. Formica ex machina: Ant swarm foraging from physical to virtual and back again. In *Swarm Intelligence*, pages 252–259. Springer, 2012.

[135] Shaowu Zhang, Sebastian Schwarz, Mario Pahl, Hong Zhu, and Juergen Tautz. Honeybee memory: A honeybee knows what to do and when. *Journal of Experimental Biology*, 209(22):4420–4428, 2006.

[136] Randolf Menzel and Martin Giurfa. Cognitive architecture of a mini-brain: The honeybee. *Trends in Cognitive Sciences*, 5(2):62–71, 2001.

[137] Darrell Moore, Dana Siegfried, Richard Wilson, and Mary Ann Rankin. The influence of time of day on the foraging behavior of the honeybee, Apis mellifera. *Journal of Biological Rhythms*, 4(3):305–325, 1989.

[138] Joe R Riley, Uwe Greggers, Alan D Smith, Don R Reynolds, and Randolf Menzel. The flight paths of honeybees recruited by the waggle dance. *Nature*, 435(7039):205–207, 2005.

[139] Stefan Janson, Martin Middendorf, and Madeleine Beekman. Searching for a new home–scouting behavior of honeybee swarms. *Behavioral Ecology*, 18(2):384–392, 2007.

[140] Kevin M Passino. Bacterial foraging optimization. *International Journal of Swarm Intelligence Research (IJSIR)*, 1(1):1–16, 2010.

[141] Eric L Charnov. Optimal foraging, the marginal value theorem. *Theoretical Population Biology*, 9(2):129–136, 1976.

[142] Stefano Nolfi and Dario Floreano. Coevolving predator and prey robots: Do "arms races" arise in artificial evolution? *Artificial Life*, 4(4):311–335, 1998.

[143] James Kennedy, Russell Eberhart, and others. Particle swarm optimization. In *Proceedings of IEEE International Conference on Neural Networks*, volume 4, pages 1942–1948. Perth, Australia, 1995.

[144] George F Oster and Edward O Wilson. *Caste and Ecology in the Social Insects*. Princeton University Press, 1978.

[145] Esben H Østergaard, Gaurav S Sukhatme, and Maja J Mataric. Emergent bucket brigading-a simple mechanism for improving performance in multi-robot constrained-space foraging tasks. In *In Autonomous Agents*. Citeseer, 2001.

[146] Dani Goldberg and Maja J. Mataric. Design and Evaluation of Robust Behavior-Based Controllers for Distributed Multi-Robot Collection Tasks. In *Robot Teams: From Diversity to Polymorphism*, pages 315–344. A K Peters Ltd, 2001.

[147] Miguel Schneider-Fontan and Maja J Mataric. Territorial multi-robot task division. *IEEE Transactions on Robotics and Automation*, 14(5):815–822, 1998.

[148] Ken Sugawara and Toshinori Watanabe. Swarming robots-foraging behavior of simple multirobot system. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 3, pages 2702–2707. IEEE, 2002.

[149] ACM. *The impact of diversity on performance in multi-robot foraging*, 1999.

[150] Marco Dorigo, Dario Floreano, Luca Maria Gambardella, Francesco Mondada, Stefano Nolfi, Tarek Baaboura, Mauro Birattari, Michael Bonani, Manuele Brambilla, Arne Brutschy, and others. Swarmanoid: A novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, 20(4):60–71, 2013.

[151] Gregory Dudek, Michael Jenkin, Evangelos Milios, and David Wilkes. A taxonomy for swarm robots. In *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems'*, volume 1, pages 441–447. IEEE, 1993.

[152] Wenguo Liu, Alan Winfield, Jin Sa, Jie Chen, and Lihua Dou. Strategies for energy optimisation in a swarm of foraging robots. In *International Workshop on Swarm Robotics*, pages 14–26. Springer, 2006.

[153] Thomas Schmickl and Karl Crailsheim. Trophallaxis among swarm-robots: A biologically inspired strategy for swarm robotics. In *Proceedings of the First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics*, pages 377–382. IEEE, 2006.

[154] Patrick Vincent and Izhak Rubin. A framework and analysis for cooperative search using UAV swarms. In *Proceedings of the 2004 ACM Symposium on Applied Computing*, pages 79–86. ACM, 2004.

[155] C Ronald Kube and Eric Bonabeau. Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, 30(1):85–101, 2000.

[156] Alan F.T. Winfield. Towards an engineering science of robot foraging. In *Distributed Autonomous Robotic Systems 8*, pages 185–192. Springer, 2009.

[157] Michael JB Krieger, Jean-Bernard Billeter, and Laurent Keller. Ant-like task allocation and recruitment in cooperative robots. *Nature*, 406(6799):992–995, 2000.

[158] Ronald C Arkin. Cooperation without communication: Multiagent schema-based robot navigation. *Journal of Robotic Systems*, 9(3):351–364, 1992.

[159] Valery Karpov and Irina Karpova. Leader election algorithms for static swarms. *Biologically Inspired Cognitive Architectures*, 12:54–64, 2015.

[160] Matthew Hoeing, Prithviraj Dasgupta, Plamen Petrov, and Stephen O'hara. Auction-based multi-robot task allocation in comstar. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, page 280. ACM, 2007.

[161] Richard T Vaughan, Kasper Støy, Gaurav S Sukhatme, and Maja J Matarić. Blazing a trail: Insect-inspired resource transportation by a robot team. In *Distributed Autonomous Robotic Systems 4*, pages 111–120. Springer, 2000.

[162] Ryusuke Fujisawa, Hikaru Imamura, Takashi Hashimoto, and Fumitoshi Matsuno. Communication using pheromone field for multiple robots. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1391–1396. IEEE, 2008.

[163] Jiann-Horng Lin, Li-Ren Huang, and others. Chaotic bee swarm optimization algorithm for path planning of mobile robots. In *Proceedings of the 10th WSEAS International Conference on Evolutionary Computing*, pages 84–89. World Scientific and Engineering Academy and Society (WSEAS), 2009.

[164] Serge Kernbach, Ronald Thenius, Olga Kernbach, and Thomas Schmickl. Reembodiment of honeybee aggregation behavior in an artificial micro-robotic system. *Adaptive Behavior*, 17(3):237–259, 2009.

[165] Thomas Schmickl, Christoph Möslinger, and Karl Crailsheim. Collective perception in a robot swarm. In *Swarm Robotics*, pages 144–157. Springer, 2007.

[166] Sjriek Alers, Daniel Claes, Karl Tuyls, and Gerhard Weiss. Biologically inspired multi-robot foraging. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*, pages 1683–1684. International Foundation for Autonomous Agents and Multiagent Systems, 2014.

[167] Ashutosh Saxena, Justin Driemeyer, and Andrew Y Ng. Robotic grasping of novel objects using vision. *The International Journal of Robotics Research*, 27(2):157–173, 2008.

[168] Francesco Mondada, Luca Maria Gambardella, Dario Floreano, Stefano Nolfi, J-L Deneuborg, and Marco Dorigo. The cooperation of swarm-bots: Physical interactions in collective robotics. *IEEE Robotics & Automation Magazine*, 12(2):21–28, 2005.

[169] Brett Browning and Manuela Veloso. Real-time, adaptive color-based robot vision. In *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3871–3876. IEEE, 2005.

[170] Matthias Jüngel, Jan Hoffmann, and Martin Lötzsch. A real-time auto-adjusting vision system for robotic soccer. In *Robot Soccer World Cup*, pages 214–225. Springer, 2003.

[171] Vito Trianni, Stefano Nolfi, and Marco Dorigo. Cooperative hole avoidance in a swarm-bot. *Robotics and Autonomous Systems*, 54(2):97–103, 2006.

[172] Xun Zhou. *Motion Induced Robot-to-Robot Extrinsic Calibration*. PhD thesis, 2012.

[173] Joseph A Rothermich, M İhsan Ecemiş, and Paolo Gaudiano. Distributed localization and mapping with a robotic swarm. In *International Workshop on Swarm Robotics*, pages 58–69. Springer, 2004.

[174] Gene E Robinson. Regulation of division of labor in insect societies. *Annual Review of Entomology*, 37(1):637–665, 1992.

[175] Gene E Robinson and Robert E Page. Genetic basis for division of labor in an insect society. *Nature Reviews Genetics*, 9:61–80, 1989.

[176] Robert E Page Jr and Sandra D Mitchell. Self organization and adaptation in insect societies. In *Proceedings of the Biennial Meeting of the Philosophy of Science Association*, pages 289–298. JSTOR, 1990.

[177] Eric Bonabeau and Guy Theraulaz. Role and variability of response thresholds in the regulation of division of labor in insect societies. In *Information Processing in Social Insects*, pages 141–163. Springer, 1999.

[178] Gene E Robinson. Regulation of honey bee age polyethism by juvenile hormone. *Behavioral Ecology and Sociobiology*, 20(5):329–338, 1987.

[179] Nigel R Franks and Chris Tofts. Foraging for work: How tasks allocate workers. *Animal Behaviour*, 48(2):470–472, 1994.

[180] Chris Tofts. Algorithms for task allocation in ants.(A study of temporal polyethism: Theory). *Bulletin of Mathematical Biology*, 55(5):891–918, 1993.

[181] Glennis E Julian and Sara Cahan. Undertaking specialization in the desert leafcutter ant Acromyrmex versicolor. *Animal Behaviour*, 58(2):437–442, 1999.

[182] Guy Theraulaz, Eric Bonabeau, and JN Denuebourg. Response threshold reinforcements and division of labour in insect societies. *Proceedings of the Royal Society of London B: Biological Sciences*, 265(1393):327–332, 1998.

[183] Jacques M Pasteels, Jean-Louis Deneubourg, and others. *From Individual to Collective Behavior in Social Insects*. Birkhäuser Verlag, 1987.

[184] Andrew J Spencer, Iain D Couzin, and Nigel R Franks. The dynamics of specialization and generalization within biological populations. *Advances in Complex Systems*, 1:115–127.

[185] Zhi-Yong Huang and Gene E Robinson. Honeybee colony integration: Worker-worker interactions mediate hormonally regulated plasticity in division of labor. *Proceedings of the National Academy of Sciences*, 89(24):11726–11729, 1992.

[186] Deborah M Gordon, Brian C Goodwin, and Lynn EH Trainor. A parallel distributed model of the behaviour of ant colonies. *Journal of Theoretical Biology*, 156(3):293–307, 1992.

[187] Stephen W Pacala, Deborah M Gordon, and HCJ Godfray. Effects of social group size on information transfer and task allocation. *Evolutionary Ecology*, 10(2):127–165, 1996.

[188] William Agassounon and Alcherio Martinoli. Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems. In *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 3*, pages 1090–1097. ACM, 2002.

[189] Paul V Switzer. Site fidelity in predictable and unpredictable habitats. *Evolutionary Ecology*, 7(6):533–555, 1993.

[190] Francesco Mondada, Michael Bonani, Xavier Raemy, James Pugh, Christopher Cianci, Adam Klaptocz, Stéphane Magnenat, Jean-Christophe Zufferey, Dario Floreano, and Alcherio Martinoli. The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, volume 1, pages 59–65, 2009.

[191] Pavlos Antoniou, Andreas Pitsillides, Tim Blackwell, Andries Engelbrecht, and Loizos Michael. Congestion control in wireless sensor networks based on bird flocking behavior. *Computer Networks*, 57(5):1167–1191, 2013.

[192] Jürgen F Brune. *Extracting the Science: A Century of Mining Research*. SME, 2010.

[193] Hartwig E. Frimmel and W.E.L. Minter. Recent developments concerning the geological history and genesis of the Witwatersrand gold deposits, South Africa. *Special Publication-Society of Economic Geologists*, 9:17–46, 2002.

[194] Mardé Helbig and Andries P Engelbrecht. Performance measures for dynamic multi-objective optimisation algorithms. *Information Sciences*, 250:61–81, 2013.

Appendix A

Symbols

This appendix provides an exhaustive list of symbols used throughout this dissertation. Symbols are listed by chapter and sorted alphabetically. Where applicable, Roman symbols are listed first, followed by Greek symbols. In cases where symbols are re-used, they are re-defined under the relevant chapters.

A.1 Chapter 4: Division Of Labour

$r_{\gamma,\nu}$	A response threshold for of robot γ for task ν.
R	The set of tasks that can be performed by individuals of the swarm.
z	The probability of a robot to switch from search behaviour to rest behaviour.
γ	An individual robot in a swarm.
ν	A task such that $\nu \in R$

A.2 Chapter 5: Nature Inspired Algorithms for Prioritized Foraging

f_{max}	Maximum time a robot will forage for prioritized items unsuccessfully.
i	The current time step in a swarm robot experiment

i_{state}	The current time step of being in consecutive state *state*.
k_j	is the value captured on the $j-$th distance sensor of a robot.
n	The number of distance sensors.
$role$	The role of a particular robot in the honey bee algorithm.
$state$	The state a particular robot is in.
t_{dance}	Maximum time a robot can spend consecutively in the recruitment state.
$t_{explore}$	Maximum time a robot can spend consecutively in the explore state
t_{forage}	Maximum time a robot can spend consecutively in the forage state.
t_{ls}	Maximum time a robot can spend consecutively in the local search state.
t_{wait}	Maximum time a robot can spend consecutively in wait state.
v	A robot's path integration vector
X	The percentage of robots initialized as scout robots.
α	The probability of listening to the details communicated by the scout robot.
μ_ς	The evaluation of the quality of a site of type ς.
ξ	The site where an item is found.
ω	A memorized path integration vector representing the location of a site.
Φ	Site quality threshold.
ϑ	An item found by a robot.
ϱ	A random number selected from a uniform distribution such that $\varrho \in (0, 1)$.
ρ	The probability that a scout robot becomes a forager robot.
ς	The type of an item which can either be prioritized or non-prioritized.
σ	The direction a robot must take to get to the sink.

A.3 Chapter 6: Experimental Setup

c	Density of robots in an environment.
$\overline{c}$	A list of all considered values for c.
c_i	The ith item in $\overline{c}$.

c_{min}	The smallest value considered value for c.
d	Desirability of a direction q.
D_{ij}	The average efficiency E_P over all experiments with τ_i and ϵ_j.
D	The matrix of the values D_{ij} where $i = 1, .., n_\tau$ and $j = 1, ..., n_\epsilon$.
E_P	The percentage of prioritized items foraged over time.
E_{NP}	The percentage of non-prioritized items foraged over time.
E_{c_i}	The average foraging efficiency, E_P, with swarm density c_i.
E_{p_i}	The average foraging efficiency, E_P, with problem complexity p_i.
E_P^t	Total prioritized items foraged over time t.
E_{NP}^t	Total non-prioritized items foraged over time t.
f	The field of view of a robot's perception.
F_{r_i}	The flexibility of a robot swarm in terms of prioritized item ratio r_i.
F_r	The flexibility of a robot swarm in terms of prioritized item ratio.
F_{ϵ_i}	The flexibility of a robot swarm in terms of environment distribution ϵ_i.
F_ϵ	The flexibility of a robot swarm in terms of environment distribution.
I	A list where entry I_j is the index yielding the maximum value for Z_{ij}.
n_r	The number of items in $\overline{r}$
n_τ	The number of items in $\overline{\tau}$
p	The density of the items on the grid, also referred to as problem complexity.
p_i	The ith item of $\overline{p}$.
$\overline{p}$	A list of all considered values for p.

p_{min}	The smallest considered value for p.
q	A direction a robot can perceive.
r	The ratio of prioritized to non-prioritized items.
$\overline{r}$	A list of all considered values for r.
r_i	The ith item in $\overline{r}$.
S_c	The scalability of a robot swarm in terms of swarm density.
S_p	The scalability of a robot swarm in terms of problem size.
t	The time step of an experiment.
t_{wait}	The average time steps spent by each robot in the waiting state.
$t_{recruitment}$	The average time steps spent by each robot in the recruitment state.
Y	A list where entry Y_j is the index yielding the maximum value for D_{ij}.
Z_{ij}	The average efficiency E_P over all experiments with τ_i and r_j.
Z	The matrix of the values Z_{ij} where $i = 1, .., n_\tau$ and $j = 1, ..., n_r$.
ϵ	The distribution of prioritized and non-prioritized items on an environment.
$\overline{\epsilon}$	A list of all considered environment distributions ϵ.
ϵ_i	The ith item of $\overline{\epsilon}$.
η	The depth of view of a robot's perception.
ι_q	The directness of a direction q.
κ_q	The clarity of a direction.
λ	A ratio between the effect of κ_q or ι_q on d.
Λ	The height and width of the environment grid.

τ	The specialization ratio of the robot swarm.
$\tau(t)$	The value of τ at time step t.
$\overline{\tau}$	A list of all considered values for $\tau(0)$.
$\tau_i(0)$	The ith item in $\overline{\tau}$.

Appendix B

Derived Publications

The following publication was derived from this thesis:

- Jade Abbott, and Andries P. Engelbrecht. Nature-inspired swarm robotics algorithms for prioritized foraging. In *Proceedings of 9th International Conference on Swarm Intelligence*, pages 246–253, 2014