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

December 2013

Publication data:

Jade Zoe 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

Your thesis abstract goes here. This should be a single paragraph. Try to keep it as brief as possible (less than 200 words — if this abstract page runs onto a second page, it needs shortening), while keeping in mind that it should touch on all the important aspects of your research — consider whether someone unfamiliar with your research area would be able to determine whether your research is relevant to them, or not. Keep in mind that the abstract may be the only thing someone reads before choosing to either discard your work, or keep reading. Also, make sure that there are no references in the abstract. The keywords list should include no more than ten keywords. Keywords may be single words, or multi-word terms (such as "neural networks" or "particle swarm optimisers"). When choosing keywords, consider terms that are descriptive of your research, and are likely to be used in search queries that should find your work.

Keywords: Swarm Robotics, Foraging, Prioritized Foraging

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 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 as 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 located 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 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 can't locate the humans, they'll 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 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 exist 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 algorithm's are developed in 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:

- Conduct a survey of the swarm robotics field.
- Conduct a survey of foraging in social insects and swarm robotics.
- 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.
- 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

- A novel variation of the foraging problem - prioritized foraging, as well as performance measures for the prioritized foraging problem.
- A novel Desert ant inspired foraging algorithm for robot swarms.
- A novel Honey bee inspired foraging algorithm for robot swarms.
- Methods for generating various types of foraging environments in a simulated environments.
- An analysis of the efficiency, flexibility, scalability, and robustness of each algorithm on the prioritized foraging problem.

1.4 Thesis Outline

This thesis consists of 8 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 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. This chapter presents the relevant performance measures to be used to evaluate the proposed algorithms on the prioritized foraging problem.
- **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 2 appendices, which are organised as follows:

- **Appendix A** provides a list of important acronyms that are used in the course of this work, along with their definitions
- **Appendix B** lists and defines the mathematical symbols used in this thesis, divided according to the chapter in which they appear.

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] who 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] and communication, propulsion and sensing in bacteria [11, 12].

2.2 Motivations for Swarm Robotics

Swarm robots draws inspiration from insect swarms due to the characteristics of robustness, flexibility, scalability and to a lesser extent cost-effectiveness. The following sections describe the characteristics of insect 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 it's 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.

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 the sensor data of the robot swarm is adequately aggregated by the swarm, the overall effect of noise can be decreased or eliminated.

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 [16–18]. 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 have been performed in [19–21] 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. Erol 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 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, even if only to a lesser efficiency, by the remaining types of robots. 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 (due to malfunction or destruction) and then to re-elect a new leader without human intervention, otherwise the swarm can no longer continue it's function at the failure of a single individual.

4. **Localization**: Assume a group of robots that are dependant on global sensors and communication - for example, all inter-robot communications have to pass through a centralized server, or an overhead camera that all individuals of the swarm connect to and use 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 sensor 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 a 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. Early swarm robotics research explored the use of robots as a tool with which to model and validate entomology research on social insects [30–32].

This section aims to briefly highlight the history of swarm robotics and it's 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 Rodney A. 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: i.e. obstacle avoidance layer, a random wandering layer, and a search for the furthest point layer. These 3 separate behaviours cause 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 search 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. Ronald 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 the a change in focus of robotics research, from complex indviduals 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 [30–32]. 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 explores the design of the structure of multi-robot systems based on nonlinear control approaches. Freund demonstrates 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 cellubar robotics would need to be solved, in order for for cellular robotics to become feasible in real-world application.

The various ideas for the use of multiple robots to a 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 full 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 have been used to model swarms as well as factors such as noise, stochasticity and spatiality. Unfortunately, differential equations are often computationally expensive and complex to solve [63, 64]

- **Classical control and stability theory** has been used to prove swarm properties [65–67]. 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 continuously violated.

- **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.

- **Subsumption Architecture** - 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 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 representing 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 [73–75]

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 locate partners [85]. A variety of approaches to swarm robot aggregation have been investigated [86–88]. 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. 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 be used to stabilize robots on difficult terrains, and to combine forces to transporten 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 [97].

- **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, such as [98], wall creation [99] and 2D and 3D structure creation [100, 101].

Navigation behaviours

Robot navigation is difficult task and navigational complexity will increase 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[102, 103]. Swarm-guided navigation is simply to navigate an environment using swarms. Swarm navigation is usually achieved using probabilistic finite state machines [104, 105].

- **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 [106]. 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 [78, 107, 108].

- **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 [109]. 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 [110]. The SWARM-BOTS project involved substantial work with the 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 [111–113].

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 traditional robotic systems. This section discusses and provides examples of types of collective decision making behaviours.

- **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 per-

formance. This is observed in ant colonies to determine the shortest path [85] and is used by bees decide on the next best site for a nest [114].

- **Division of Labour** occurs naturally in insect colonies [115], most notably in bees and ants. Division of labour is a key primitive 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 [116, 117].

2.6 Challenges

Sahin *et al.* suggest a number of challenges faced by swarm robotics [118]. 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.* attempt to devise a mathematical model to assist in modelling swarm behaviour and communication, 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 [**bredeche2013roborobo**, 56, 57, 119]. Some are only 2D grid-world models while others attempt to mimic real life as accurately as possible [120]. 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**: Swarm robotic systems are stochastic non-linear 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 to make assumptions about the swarm, or the environment, to simplify the swarm robotics system so that it is easier to prove specific properties of the swarm or to optimize swarm parameters. The risk of the assumptions is that the conclusions achieved from analysis of the models, may not reflect reality and little work in verification of .

- **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 requires 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 defines 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. The definition of foraging in swarm robotics is discussed in Section 4.1, while Section foraginginnature describes foraging behaviours observed in nature, specifically in bees and ants. Section 3.2.1 provides a taxonomy of the types of swarm robotic foraging algorithms, and the challenges that foraging algorithms must address are discussed in Section 3.3.3. Existing swarm robotics foraging algorithms are outlined in Section 3.3 , and prioritized foraging is defined and discussed in Section 5.1. Lastly, the chapter is summarized in Section 3.4.

3.1 Definition

Foraging was initially studied by biologists, particularly the foraging behaviour of ants [121, 122], bees [32] and bacteria [123]. Foraging has a variety of real-life applications such as search and rescue [124, 125], 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) 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.

The foraging sub-tasks involve 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.

3.2.1 Ants

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. Ants use a form of communication known as stigmergy [126]. 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 colony optimization and ant systems [127, 128].

Although algorithms based on ant foraging behaviour are used to solve optimization

problems, the algorithms are often difficult to replicate in a real-life robotic 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, as pheromone deposited on desert sand would be blown away by the wind [129, 130]. Instead, desert ants use a technique called path integration for navigation to relocate food sources that have already been found by random exploration [131, 132]. 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 heat stress. The shortcut back to the nest is known as the home vector [133]. 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 [131].

Due to the lack of pheromone, the desert seed-harvester ant is thus simpler to model for real-robot interaction. Desert seed-harvester ant foraging has been modelled in [134, 135]. Hecker et al [135] performed experiments to compare the performance of two foraging algorithms algorithms: The one algorithm was based on the desert ant foraging which has no pheromone-like communication and the other used including pheromone-like communication. It was shown that communication improved performance; however, the desert ant based algorithm still performed comparatively well.

3.2.2 Bees

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 [136], and in terms of navigation, they are able to learn local features and routes due to well-developed learning and memorizing capacities [137]. Honey-bees are even time aware [138].

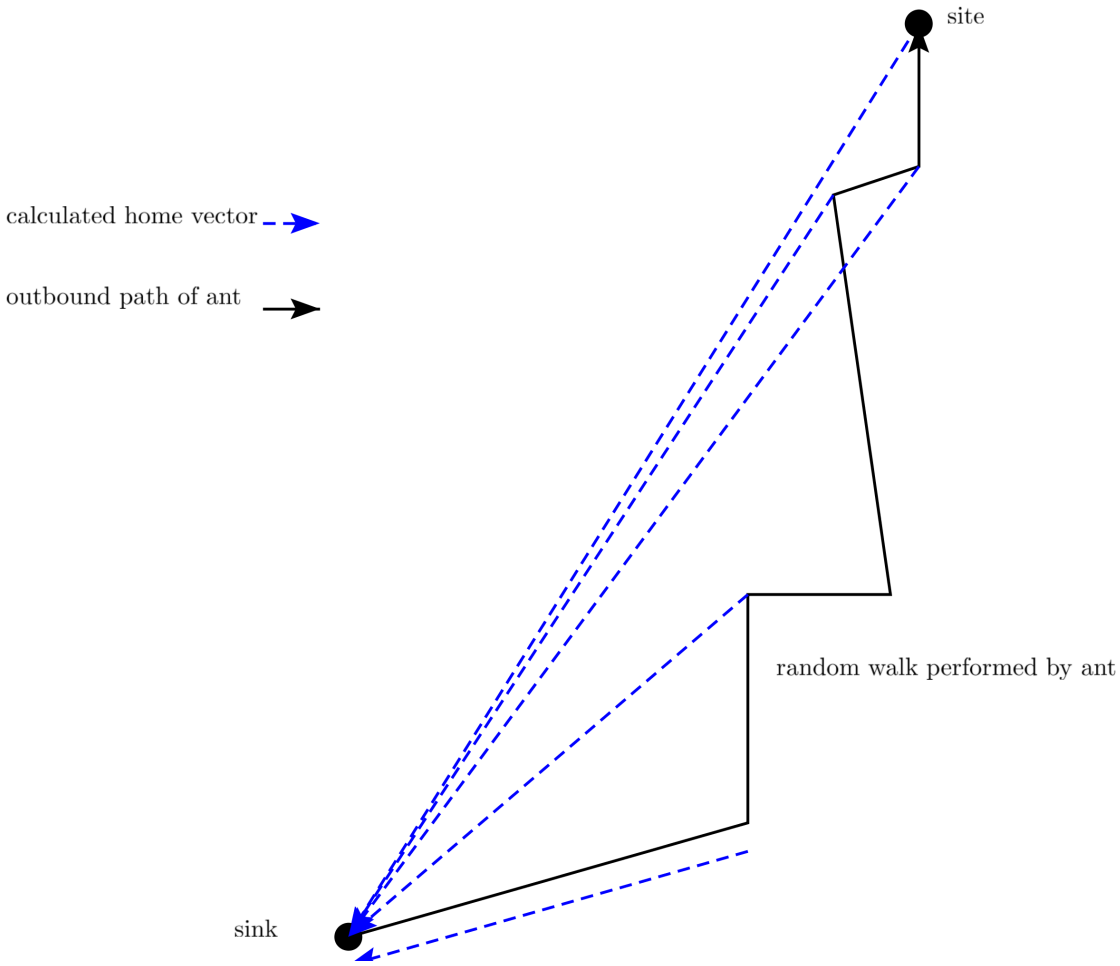


Figure 3.1: Path Integration

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. The communication of a good foraging site is known as recruitment [32].

Unemployed foragers wait on the dance floor, and evaluate the dances of the scout bees. An unemployed bee select 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 Bacteria

The *E.coli* bacteria forage for carbohydrates in the human gut. The 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"). The bacteria exhibit the behaviour continuing to move in a direction, if the bacteria are moving up a nutrient gradient. Bacteria stop moving in a direction, if they're moving down a nutrient gradient 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 is used for inspiration for a computational intelligence algorithm [140] and particle swarm optimization is, for instance based on the flocking behaviour of birds - a flocking behaviour exhibited when finding food, amongst other things [141].

3.2.4 Predator Prey

Predator prey scenarios are foraging problems where the items to be foraged actively avoid capture (known as prey) and the animals pursuing the prey are known as predators [4]. Predator prey foraging has been extensively studied in the field of Optimal Foraging Theory, which aims to predict the behaviour of an animal searching for food [142].

Predator-prey models have been used in other swarm intelligence techniques in multi-objective optimization problems [143].

3.3 Foraging in Swarm Robotics

The following section addresses foraging in a swarm robotics context. Firstly, a taxonomy of robot foraging is presented and then existing nature-inspired swarm robotics foraging algorithms are discussed.

3.3.1 Taxonomy of Robot Foraging

Foraging is a popular problem due to the vast potential opportunity of foraging for application to a variety of real-world problems from search and rescue [124] 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 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 foragi ng 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 have a number of minor axes which are outlined in Table 3.1 and Table 3.2. The following sections discuss each of the major and minor axes in more detail

Robot Axis

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

- 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 a swarm robotics.
- 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 there does exist foraging research using a heterogenous swarm. In particular, the Swarmanoid project uses a heterogenous swarm to forage an item

Table 3.1: Winfield's Robot Foraging Taxonomy [4]

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

[146]. 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.

- 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, thus foraging algorithms using global sensors are not addressed in this literature study.

- Localization refers to the robots ability to position themselves in the environment. Localization could be non-existent, relative or absolute (a global positioning system (GPS)). Absolute localization is not considered swarm robotics as then the swarm is 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, and is suited to swarm robotics foraging problems. Relative localization techniques such as path integration are used in this thesis.

- Communication capabilities of the robots to each other can have infinite range, 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[147] or local direct communication methods such as Bluetooth. The communication occurs in differing topologies such as broadcast, direct, tree or graph topologies [148]

- 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 [149].

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 time taken to forage each

item. Each minor axis described as follows:

- Performance of a foraging algorithm can be classified in terms of time taken to forage. A foraging experiment can be allowed infinite time to foraged, a fixed time for foraging or is attempting 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 [149].

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 algorithm is the search for items. There are various techniques for searching for items, such as: random search, trail following [150], following other robots [95, 146], groups of robots looking for items together [151], where robots search in geometrical patterns and where search can only occur in a limited range of the sink.
- Once an item is found it must be picked up. The techniques used by an algorithm 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 [152].
- In order to get to the sink, a number of homing techniques can be used such as homing on a beacon whereby the robot first changes their direction to face a beacon at the sink, and then move towards the beacon, following a trail or odometry [153].

- Recruitment has been defined as activities that bring individuals to a location where work is required [121]. Robots can recruit other robots to forage certain areas. Direct recruitment is where a robot explicitly communicates to another robot to come forage an item source [17, 154]. Indirect recruitment occurs when a robot is recruited to forage by inexplicit 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 [155].

- 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 robots. Centralized control is not be considered 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 swarm robotics since the swarm is then robust to individual failures [156, 157]. A swarm with 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 it's own activities and the self-organised coordination is an emergent property of the swarm.

Environment Axis

The environment axis can be divided into minor axes as follows:

- **The type of search space** which can be either unbounded or constrained by a boundary wall. Goldberg and Mataric [158] implement a foraging method for an unbounded environment. Schneider [159] developed a foraging algorithm which splits a constrained environment where the environments dimensions are known into bounded areas. Each robot is in charge of their own area.

- **The types and number of item sources** that exist in the environment. A single limited source denotes that all items come from the same source and the source as a limited capacity for items, such as in [147]. A foraging environment

with a single unlimited source is where one source exists but an unlimited number of items can be foraged from the source. A foraging environment can also have multiple sources, such as in [158].

- **The number of sinks**: The environment can have a single or multiple sinks.
- **Item distribution** in the environment varies. Items can be placed in known locations, in a uniform distribution or in clusters in the environment.
- **The type of items** in terms of whether they move (active) or not (static) and the quantity of items (single or multiple) and the number of types of items - either foraging homogenous items or heterogenous items which is known as multi-foraging. Multi-foraging is defined as a type of foraging problem where instead multiple object types are required to be foraged. [160]. The types of objects can have different characteristics and can be collected together or separately.

  Jones et al [116] describe a version of the multi-foraging problem where there exist two types of items and items are consumed instead of transported to the sink. As items are consumed, the same type of item is placed at a random spot in the arena. Robots signal what type of item they are foraging by turning on a coloured light. Other robots use the coloured light to build a local history of the number of robots they have seen allocated to each item type. A history of the number of observed items of each type is also collected. The local observations are used to estimate the current division of labour of the swarm between foraging the two item types. The estimations are used to determine the probability of changing from foraging the current item type to the other type of item.

  Campo [62] identifies characteristics of efficient multi-foraging behaviour by developing and validating a mathematical model to predict optimal foraging behaviour. The proposed foraging decision algorithm uses the amount of other robots encountered as well as the amount of prey encountered to only allocate enough robots foraging so that remaining robots can spare energy.

  Balch [161] develops a technique for multi-foraging and determining the impact of robot diversity on multi-foraging performance. This thesis's does not consider

robot diversity since robots are homogeneous.

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 addresses 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 presents a swarm robotic algorithm allowing for robust transportation of items from a single source to a single sink in an environment with spatial constraints [162]. 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 a technique called path integration utilized by both ants and bees in order to maintain position and heading estimate. Using these techniques the robots communicate multi-segment paths. These communications occur globally. This technique suffers from accumulation of localization errors over time due to the use of global communication. Local communication is preferred for the algorithms developed in this study.

Hoff [6] classifies different types of ant-inspired pheromone-based foraging algorithms by how the pheromones are represented: physical marks [**alcoholfromants2012**], use of existing communication channels such as sharing over wireless networks as well as virtual pheromone.

Labella et al [17] propose a foraging model inspired by the foraging of ants, where individual robots adapt to the environment using only locally available information. The adaptation allows for effective division of labour to occur. The probability $p1$ that a robot will changed from rest to searching is adapted upon return to the sink and from deposit to rest. If a robot fails to forage any prey after searching for a specified maximum time, then the robot returns and rests. The adaptation rule increases or decreases $p1$ by a constant, that is multiplied by the number of consecutive successes or failures. The

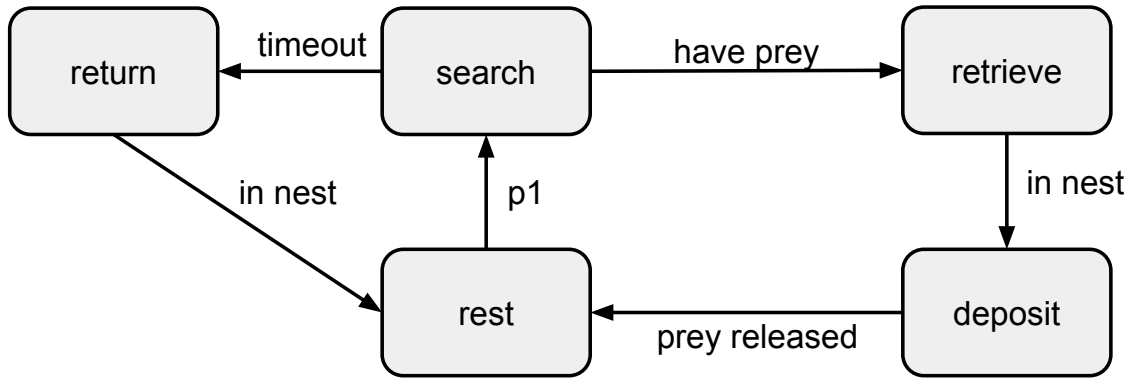


Figure 3.2: Simplified finite state machine for Labella division of labour algorithm.

study showed that adaptation improved the amount of time spent not at rest. The advantage to this type of algorithm is its simplicity as no communication is required to regulate the number of robots. As pointed out, there is a global negative effect on overall efficiency at large group sizes. This is to be expected, but is not a serious problem since less energy is used, and this would be an advantage when energy is a scarce resource.

Hoff [6] proposes a two ant-based foraging algorithm which do not require marking the physical environment. Instead, robots themselves become pheromone markers. The algorithms differ in the manner in which the beacon-to-pheromone role is chosen.

Virtual pheromone algorithm functions as follows: Two pheromone trails are created - one trail from the source to the food and another trail from the source to the nest. The trail is created by robots: During execution, robots stop exploration and become pheromone beacons. The pheromone beacon robots simply broadcast a floating point number known as virtual pheromone. Local direct communications are used to transmit the virtual pheromone value.

The second technique uses cardinality where if a robot can hear 2 or more beacons then the robot stays a walker else the robot becomes a beacon. The advantage of using this technique over the other mentioned pheromone techniques is that robots are simpler due to simpler communication mechanisms as well as the lack of need for complex beacon deployment systems. Experiments are conducted with and without obstacles and a real-world simulator is used. The result show that congestion effects performance the most

at high robot density.

Honey Bee Inspired Foraging Algorithms

Bee swarms have been used in swarm robotics for problems such as path planning [163], aggregation [164], collective perception [165]. Foraging algorithms have been developed that simulate a simplified honey bee recruitment in [166], which focuses on an actual robot demonstration of the technique using a turtle robot. Prior work focuses on implementing the same algorithm on e-pucks.

The algorithm consists of two phases: an exploratory phase followed by an exploitive phase. Initially, all robots are located around the hive. In the exploratory phase, all robots perform a random search with a levy distribution until a food source is found by a robot. The exploitation phase begins where the robot that located the food returns to the hive and communicates the location to other members of the robot swarm. The other robots are recruited and begin to commute between the hive and the food source, transporting virtual food. This algorithm claims to exhibit three behaviours of bees: waiting around the hive, exploring the environment and foraging. Unlike bees, the algorithm used does not consider the division of labour between foragers, explorers and waiting robots, but rather uses an extremely simple model for bees which can only exploit a single food source. The robots use path integration in order to remember the location of the food source, and use a north-star type navigation in order to locate the hive. The path integration vector gets communicated to the other robots which use the vector to guide them to the food 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 a 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 their position in the environment. In swarm robotics, global knowledge about location (such as GPS) cannot be used by a robot to determine their position and where they are going. Non-trivial algorithms must be designed in order for a robot to position itself in the environment as ins [155, 172, 173].

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 the 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 focused on reviewing foraging in nature and in swarm robotics as well as presenting a novel form of the foraging algorithm called prioritized foraging.

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 and thus use a technique called path integration to relocate a food source. Honey bees employ more complex foraging behaviour including division of labour and communication. Winfield's foraging taxonomy is presented and discussed so that research can be contextualized in the field.

Existing swarm robotics foraging algorithms are addressed and discussed under three sections: nature-inspired algorithm, standard foraging algorithms and multi foraging algorithms. Lastly, the novel idea of prioritized foraging is introduced. In prioritized foraging a particular item type has higher priority than other items.

Table 3.2: Winfield's Robot Foraging Taxonomy Part 2 [4]

Major Axis	Minor Axis	Value
Performance	Time	Fixed Minimum Unlimited
	Energy	Fixed Minimum Unlimited
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

Chapter 4

Division of Labour

As explained in Section 2.2.2, division of labour forms part of foraging activities of social insects. This chapter defines division of labour as well as analyses 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] defines division of labour as a "stable pattern of variation" of the repertoire of tasks that individuals perform. Each individual specializes on a subset of the complete repertoire of tasks. The subset of the complete repertoire of tasks is called the specialization of an individual and the specialization of individuals varies across the swarm.

Robinson et al. [174], more simply, defines 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 correlate 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 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.

4.2.1 Response Threshold Model

The response threshold model is where individuals each have a response threshold for each specific tasks that individuals in the swarm can perform. The task-specific thresholds vary across individuals in 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, $r_{\gamma,\nu}$, for task ν, for an individual γ. 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 been used to explain temporal polyethism in honey bees where the response thresholds of the honey bees would change as the honey bees age [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. This means that tasks further away from the nest are performed by older individuals and the younger individuals perform the tasks that are closer 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 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 response threshold model generate temporal polyethism by differences in internal thresholds. Unlike response 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 of the probability of doing the task again [182, 183]. Self reinforcement has been attributed for division of labour in many systems in nature [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 swarm, juvenile bees work in the hive while an older bee forages. When a juvenile bee develops, it becomes a forager bee. The activator hormone motivates growth of a juvenile bee into a forager bee [175]. An inhibitor hormone is released by the forager bees which is transferred to the juvenile worker bees. The inhibitor hormone inhibits the development of a juvenile bee into a forager bee [185].

The existence of forager bees in the hive slows the development process of the juvenile bees and thus the juvenile bees remain workers. If forager bees are destroyed, then less inhibitor hormone is released and more worker bees become forager bees, replenishing the number of foragers.

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 of the number of individuals required per task, between individuals of the swarm [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 in swarm robotics. This section provides an overview of the use of division of labour in swarm robotics.

Agassounon et al. [188] design and demonstrate three response threshold-based methods for division of labour. Response threshold model is described in Section 4.2.1. The robots had to perform a clustering task with multiple sites. The task choice was to select which site to cluster. Division of labour techniques were employed to avoid interference

between robots that occurs more often when too many robots attempt to cluster the same site.

The experiment determined that the swarm benefited from threshold division of labour since the algorithms with division of labour performed similarly or better than those with a fixed task group size. 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 re-enforcement 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 delta when a robot successfully returned to the nest with an item and decremented by the same delta when a robot returns to the nest without an item. Labella et al. classified the foragers into different types, 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] introduce three mechanisms for adapting the number of resting robots to foraging robots in a simple foraging problem. The mechanisms consist of variations on how one perceives the worker demand. 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 amount of collisions with other robots. This technique is a social inhibition model.

The mechanisms are 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.

Four combinations of these factors are evaluated as well as compared to a naïve foraging approach. The performance of the swarm is measured as the energy spent by a robot. The efficiency is 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] notes that scalability was not tested in the experiments.

By examining the existing swarm robot division of labour literature, one can conclude that the majority of research in division of labour for swarm robotics, is concerned with regulating the number of active robots in a swarm. The existing research shows that the ability of a swarm to adjust the amount of robots actively participating in a task with the amount of robots at rest, enables the swarm to adapt to changes in the environment. Existing research also concluded that division of labour improved swarm robustness to destruction of robots, by replenishing the number of active robots when active robots had been removed. There exists scope to investigate division of labour between multiple different active tasks. This thesis addresses division of labour between two active tasks, contrasting the most division of labour research.

The existing research did not test scalability of the division of labour techniques, since the maximum swarm size was usually 10 or less. This thesis will evaluate the scalability of division of labour techniques used. There exists scope for investigating other types of division of labour in swarm robotics, such as network models, but is not 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 and 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 amount 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 was 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 re-enforcement and social inhibition models.

The author notes that the reviewed research did not adequately address the scalability of the 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 prioritized foraging problem. The three algorithms are based based upon phenomenon 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, shown 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 building collapsing, 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 priority trapped survivors. Prioritized foraging could be applied to the gold mining problem where the gold needs to be foraged as a priority and the waste needs to be moved out 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.

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.

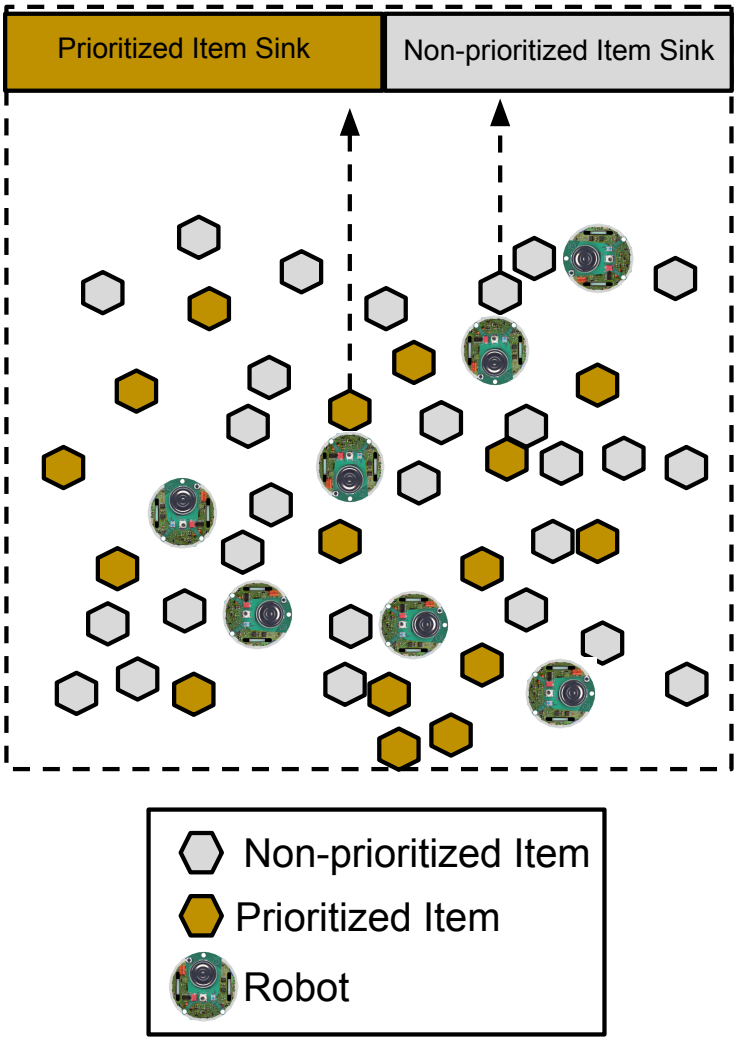


Figure 5.1: Prioritized Foraging Problem

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 [6, 145].

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. The robot walks in direction σ for distance m. The robot then chooses new values for σ and m.

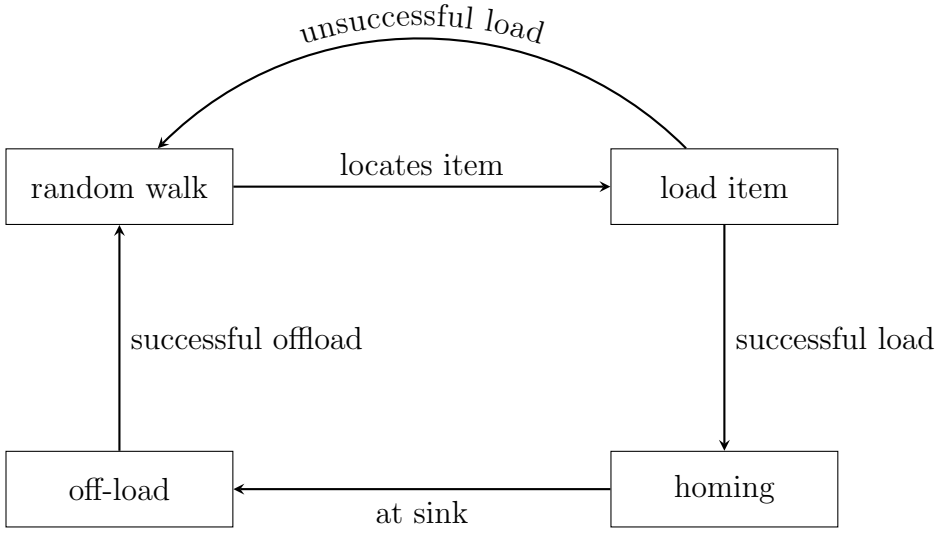


Figure 5.2: Naiïve Foraging State Diagram

5.3 Desert Ant Foraging

As discussed in Section 3.2.1, due to the lack of pheromone usage to foraging, 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 path integration (PI) to memorize the location of an existing food source and later to return to the memorized source to find more food, as addressed 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 other many swarm robotics foraging algorithms. 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 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 the it to return to the sink

and then later to return the site where the item was found. If loading the item fails (perhaps due to another robot loading the item), then the robot returns to the exploration state; else 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 prioritized 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 three roles, namely, scout robots, unemployed forager robots and employed forager robots. The roles of the robots and the transitions to and from those roles are described in this section.

Figure 5.4 provides a simplified diagram for the Honey bee prioritized foraging algorithm showing the roles and the transitions between each role and the states within those 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.

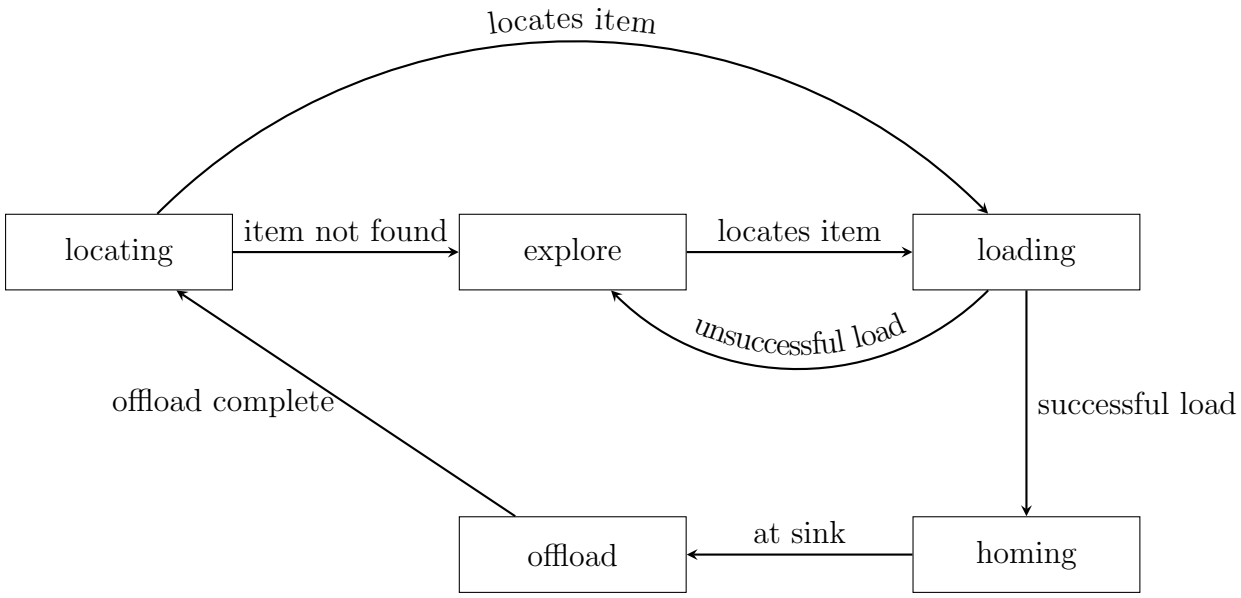


Figure 5.3: Desert Ant Foraging State Diagram

To more succinctly define the Honey bee prioritized foraging algorithm, the algorithms for each behavioural state, within each role, have been provided. For each algorithm, one should note that the current state of the robot is denoted *state*, and *state* is gets 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.

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 robot is to locate sites of items. If the discovered site is of a high enough quality, then the scout will broadcast 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 root. Variable ς saves the robot's item specialization. The random walk performed is the same random walk discussed in Section 5.2. As the scout robot moves, the robot

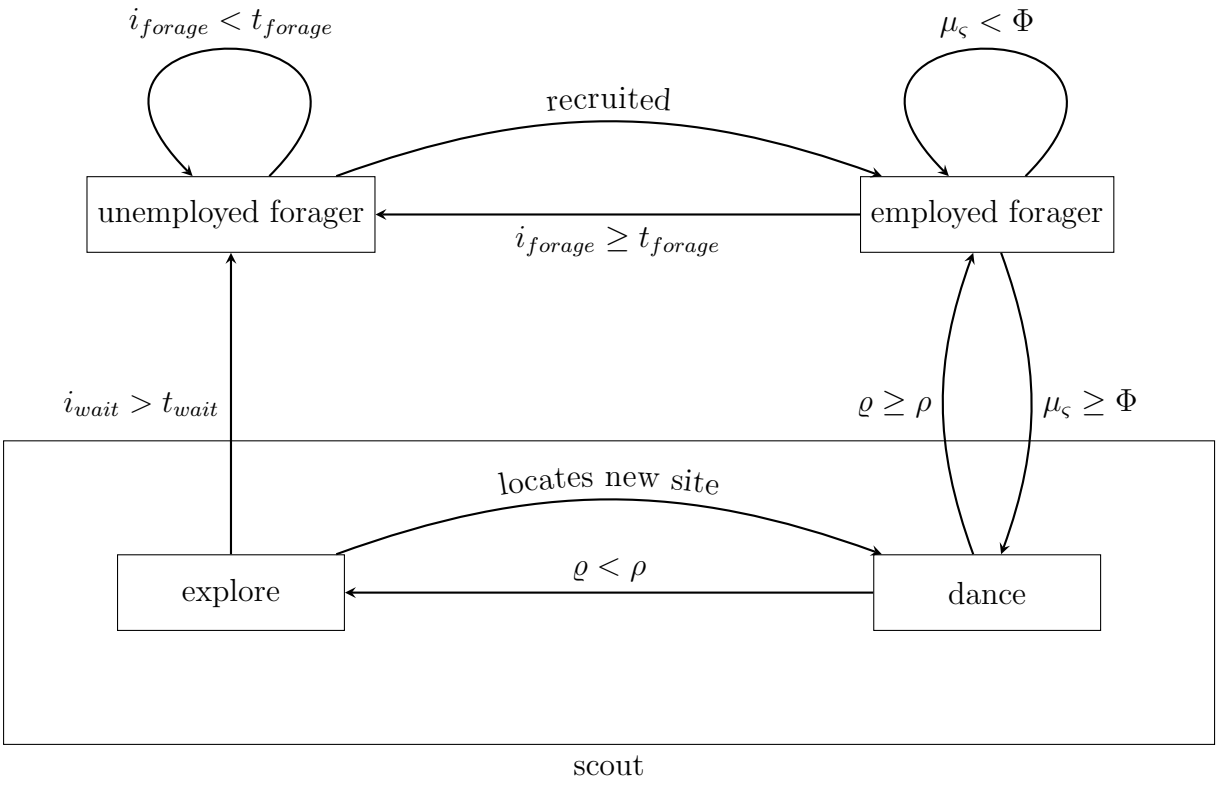


Figure 5.4: Honey Bee Foraging State Diagram

maintains a PI vector v as explained in Section 3.2.1. Upon finding an item ϑ of type ς at site ξ, the robot loads the item and then performs an evaluation of the quality, mu_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. The sensor value k_j, for item type ς, denoted k_{j_ς}, is calculated as

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

Algorithm 1 Explore State of Scout Robot

```
1: function EXPLORE(role, state, v, ς, i)
2:     perform a random walk step from location
3:     update path integration vector ¯v
4:     if item ϑ of priority ς is detected in vicinity then
5:         calculate quality μ_ς of site ξ
6:         ω ← v
7:         load item ϑ
8:     else if i_explore > f_max and i_explore ≤ t_explore then
9:         ς ← N
10:    else if i_explore > t_explore then
11:        state ← homing
12:    end if
13:    i = i + 1
14: end function
```

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

$$\mu_\sigma = \frac{1}{n}\sum_{i=1}^{n} k_{i_\varsigma} \tag{5.2}$$

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 as shown 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 threshold Φ, the scout robot switches into the dance state, which is depicted in Algorithm 3. If the quality μ_ς of the visited site ξ is less than threshold Φ, the robot takes on the role of an employed forager and switches into the locate state which is outlined in Algorithm 4.

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

Algorithm 2 Homing State of Scout Robot

function SCOUT HOMING($role, state, v, \varsigma, i, \omega$)
  if robot is at sink S_ς and robot is loaded **then**
    robot offloads item ϑ
    if $\mu_\varsigma > \phi$and$\varsigma = P$ **then**
      $state \leftarrow dance$
    else
      $role \leftarrow$ employed forager
      $state \leftarrow locate$
    end if
  else
    calculate direction d to sink S_ς from current location
    if robot can move in direction d **then**
      robot moves in direction d
    end if
  end if
  $i = i + 1$
end function

the scout and the unemployed forager robots near the sinks. The scout robot will communicate 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 become an employed forager robot and begin foraging the previous site. The scout robot becomes an employed forager robot with the probability of ρ. If random number $\varrho \in 0, 1$ is selected 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 ρ will make the scout robots more selective about the sites they broadcast. Decreasing ρ will result in scout robots being less selective about the sites they broadcast.

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
```

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.

When a scout robot dances at the sink after locating an item ϑ, each unemployed forager chooses whether to listen to the details communicated by the scout robot, of the site ξ with a probability α. An unemployed forager's acceptance of the site details, that are communicated by a scout robot, is known as recruitment. If an unemployed forager robot is recruited by the scout robot, the unemployed forager robot 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 after the employed forager bees as discussed in Section 3.2.2. The purpose of the employed forager robot is to forage the sites

Algorithm 4 Locate State of Employed Forager

1: **function** FORAGE($role, state, v, \varsigma, i, \omega, \mu_\varsigma$)
2: direction $d = \omega - l$
3: robot moves in direction d
4: **if** $\omega - l == 0$ OR robot can see item of type ς **then**
5: $state \leftarrow load$
6: **end if**
7: $i = i + 1$
8: **end function**

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 has been 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 can be loaded. If an item 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 ς, but it is not close enough to be loaded, then the robot moves in the direction of the item ς. If the employed forager can't see an item in the vicinity, then the robot moves in a random direction d. Local search state is shown 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

Algorithm 5 Wait State of Unemployed Forager

```
1: function WAIT(state, v, ς, i, ω, μ_ς)
2:     if i_wait ≥ t_wait then
3:         role ← scout
4:         ς ← P
5:         state ← explore
6:     else if scout robot is broadcasting at sink then
7:         receive site location ω and site quality μ_ς
8:         role ← employed forager
9:         state ← locate
10:    end if
11:    i = i + 1
12: end function
```

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 thus the site has likely been depleted so 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, since they did not fit into Figure 5.4.

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 collisions between robots, which would mean the employed foragers take longer to forage items. Also, if there too many employed foragers in a environment which is sparse in items, then the employed foragers are not only causing collisions but they're 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

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
```

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 amount of unemployed forager robots waiting to be recruited. The greater quantity of unemployed foragers can form a large work force for a recruiting scout, however too many unemployed foragers results in a smaller workforce 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. One should note that robots can't be initialized as employed forager robots since unemployed forager require a scout robot to recruit them to learn the location of the sites and become employed foragers and at initialization, the scouts do not yet have site location details available.

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, unemployed forager and unemployed forager as described in Section 5.4.

Algorithm 7 Local Search State of Employed Forager

```
1: function LOCAL_SEARCH(role, state, v, ς, i, ω, μ_ς)
2:     if i_ls < t_ls then
3:         if item ϑ of priority ς is nearby and can be loaded then
4:             LOAD(ϑ)
5:             state ← homing
6:         else if item ϑ of priority ς can be seen but is not close enough then
7:             select direction d to move towards item ϑ
8:             robot moves in direction d
9:         else
10:            select a random direction d
11:            robot moves in direction d
12:        end if
13:    else
14:        state ← homing
15:    end if
16:    i = i + 1
17: end function
```

However, 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, they 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 division of labour are defined as follows:

1. An scout robot that is set to search for the prioritized type $\varsigma = P$ will only forage a non-prioritized type only if a prioritized item can not be located for the

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 d to sink S_ς from current location
12:        if robot can move in direction d then
13:            robot moves in direction d
14:        end if
15:    end if
16:    i = i + 1
17: end function
```

maximum time period, f_{max}. If $i_{explore} > f_{max}$ then ς will switch to foraging the non-prioritized type $\varsigma = N$. The described rule is shown 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. In both of these cases, the robot will switch to foraging the prioritized item type $\varsigma = N$.

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

For the purpose of this study, each robot of each algorithm is assigned an initial item type to forage. The robots of 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

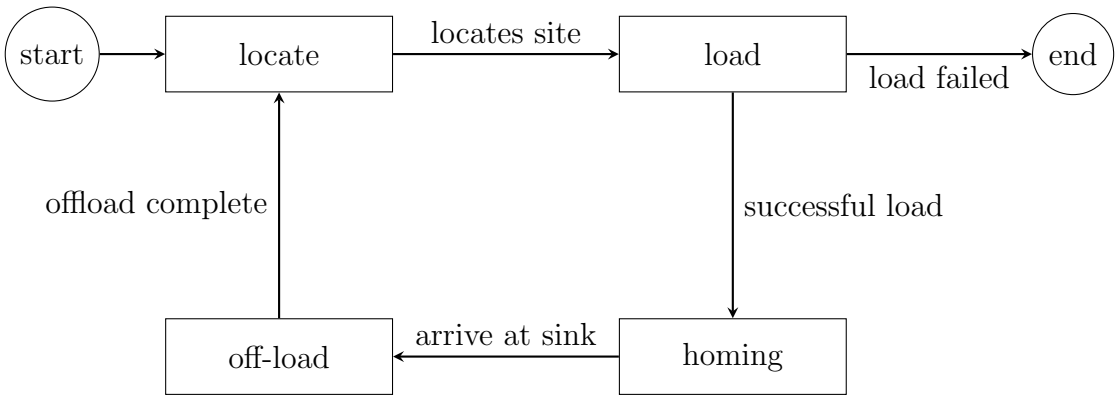


Figure 5.5: State Diagram of an Employed Forager Robot

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, as well as a Honey bee inspired foraging algorithm. 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 return 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 as well as 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 forager 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 purpose 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 **??** defines the parameters for the robot swarm. Section 6.5 proposes performance measures to be used to evaluate the performance of the algorithms on the prioritized foraging problem. The chapter is summarized in Section 6.6.

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. Section 6.2 discusses the simulator used.

6.1.1 Robot Description

The artificial robots modelled in this study are based on e-puck robots [190], which have been modified with grippers. A robot is equipped with a 360 degree camera to identify objects around the robot, as well as eight local distance sensors spaced equally 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. A 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 cannot see an item hidden by another item. As a result, 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 an 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 communication congestion 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 the global attractor while a local attractor pulls the message around congested areas.

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 to 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, i, 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_i, in a direction i, is a normalized reading from the proximity sensor or camera such that $\kappa_i \in [0, 1]$. Clarity indicates the distance of next nearest obstacle. If no obstacles exist in the depth of view v, then $\kappa_i = 0$. If there exists an obstacle immediately next to the robot, then $\kappa_i = 1$. The directness of a direction i, $\iota_i \in [0, 1]$ is calculated as the angular deviation from the direction of the destination, where $\iota_i = 0$ occurs when the direction i is the same as direction to the destination, dir, and a $\iota_i = 1$ occurs when the direction is at the edge of the field of view, f. Desirability d_i of direction i is defined mathematically in Equation 6.1.

$$d_i = \lambda\kappa_i + (1 - \lambda)\iota_i \tag{6.1}$$

where λ determines whether clarity, κ_i, or directness, ι_i, of direction i, has more 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.

6.2 Simulator

A spatially discrete 2-dimensional grid world simulator has been developed and used in this thesis in order to accelerate computation; a technique also used in [130, 147]. 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.

The simulation robots function as follows:

- Each robot fits into one grid block and each item takes up one grid block.

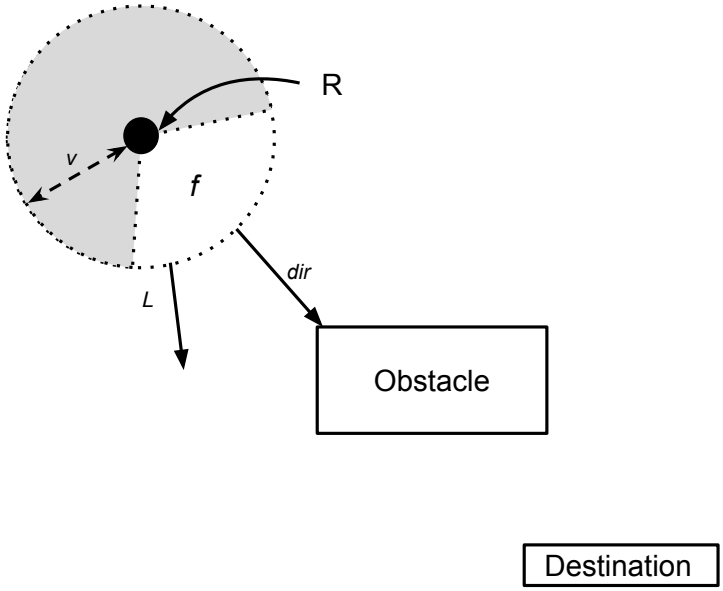


Figure 6.1: Navigation and Obstacle Avoidance, where v is depth of view, f is the field of view, R is the robot, dir is the direction of the destination and L is a possible value of local attractor

- Only a single object can occupy a grid block at a time. A 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 it's 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 light beacons that all robots can detect and navigate towards. The reason the sinks were not placed in the centre of the environment, as is commonly found in swarm robotics research [17], is because the original prioritized foraging problem was inspired the idea of using a swarm of robots to rescue trapped miners in mining tunnels. 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

The experiments were run on different environments each with different item distributions, sizes, item densities and different ratios of prioritized to non–prioritized items.

The sizes of the environment grid, S, were varied, where $S \in \{50, 100, 200, 300, 500\}$, such that S was the width and length of the grid.

Different values for the density of the items on the grid, p, were chosen where $p \in \{0.05, 0.2, 0.5, 0.7, 0.9\}$. Environments with a higher item density are more complex to forage since there exists a higher probability of an item on the unassigned type blocking the path to items of a robot's assigned type.

The ratio of prioritized to non-prioritized items r, was varied, where $r \in \{0, 0.2, 0.25, 0.33, 0.5, 0.67, 0.75, 0.8, 1\}$. When no prioritized items on the grid existed, then $r = 0$ and when only prioritized items on the grid existed, then $r = 1$.

In environments with a small r value, the abundance of non-prioritized items increases the likelihood of non-prioritized items blocking the access to prioritized items.

As r is varied over varied, the swarm specialization ratio, τ, is also varied, in order to determine of there exists a relationship between r, τ and the efficiency of the algorithms. The parameter r is also used to determine if the algorithms can adapt the swarm specialization ratio, τ, to optimally forage an environment of a given r.

Different distributions of items over the environment were chosen to examine different characteristics of the algorithms. The item distributions are shown 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. Four different classes of environments were generated as follows:

1. The position of each item in a uniformly distributed environments 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 environment 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 function is the centre of the grid, and the distribution of the Gaussian function varies (refer to Figure 6.2d). Pseudo-code for generation of Gaussian environments is provided in Algorithm 10.

   The positions of the non-prioritized items are selected after placing the prioritized items. The position for the non-prioritized items are selected from a uniform distribution.

   In Gaussian distributed environments, prioritized items occur in high concentration in 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 were 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 clustered item distribution have clusters of items of the same type (refer to Figure 6.2b). After clusters have been generated, each cluster is labelled randomly as a 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 algorithms ability to exploit areas which are rich in prioritized items. Clustered environments were also be 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.

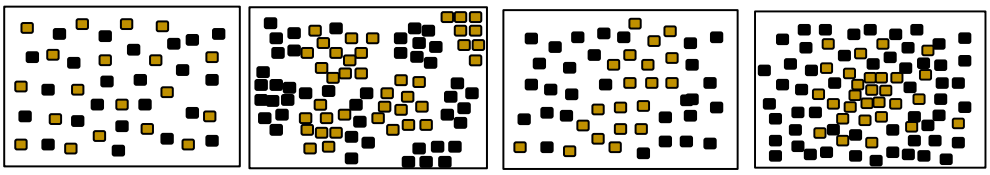
4. Environments with a vein distribution resemble the patterns observed in naturally occurring gold reefs [193] (refer to Figure 6.2c). In a gold reef, molten gold fills

planar fractures between rock resulting in a vein of gold. Inspired by the gold reef, 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 non-prioritized items are similar to the rock surrounding the gold in gold reefs.

The vein environments aimed to test whether a swarm of robots, of each algorithm, could forage the continuous length of the vein of 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 than one that cannot detect and remember the location of the vein. Pseudo-code for generation of vein environments is provided in Algorithm 11.

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, forage obstacles efficiently.
- Return to high quality areas to forage them.



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

Figure 6.2: Environment Classes

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 ratio of robots foraging prioritized items to non-prioritized items. When no robots are set to

Algorithm 9 Uniform Distributed Environments

```
1: function UNIFORM(numberitems, σ, environmentsize)
2:     nonprioritizeditemsleft = floor((1-E_p)*numberitems)
3:     prioritizeditemsleft = floor( E_p*numberitems)
4:     while prioritizeditemsleft > 0 do
5:         x ← uniform(0, environmentsize)
6:         y ← uniform(0, environmentsize)
7:         if position (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, environmentsize)
14:        y ← uniform(0, environmentsize)
15:        if position (x,y) is empty then
16:            Place prioritized item at (x,y)
17:            Decrement nonprioritizeditemsleft
18:        end if
19:    end while
20: end function
```

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 an algorithm.

The swarm size, c, is defined as the density of cells, of the grid size S (the length of the side of an environment) that are occupied by robots, where $c \in \{0.1, 0.3, 0.5, 0.7, 1\}$. The swarm size is varied in order to test the scalability of the algorithm. The swarm size is also varied to test each algorithms' ability to adjust the amount of actively foraging robots to the density of the items in the environment.

Algorithm 10 Gaussian Distributed Environments

```
1: function GAUSSIAN(numberitems, σ, environmentsize)
2:     Calculate environment centre point (center_x, center_y)
3:     prioritizeditemsleft = floor(E_p*numberitems)
4:     nonprioritizeditemsleft = floor((1-E_p)*numberitems)
5:     deviation = environmentsize*E_p/2
6:     while prioritizeditemsleft > 0 do
7:         x ← floor(gaussian(center_x, deviation))
8:         y ← floor(gaussian(center_y, deviation))
9:         if position (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, environmentsize)
16:        y ← uniform(0, environmentsize)
17:        if position (x,y) is empty then
18:            Place prioritized item at (x,y)
19:            Decrement nonprioritizeditemsleft
20:        end if
21:    end while
22: end function
```

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 honey bee algorithm specific parameters was not in the scope of this thesis.

The initial position of each robot was randomly selected, adjacent to the sink. All robots began in the exploration state – state that is shared by all algorithms. For each set, environmental and swarm configurations, each simulation was run for 10000 time steps. An experiment consists of 30 repeated samples, for a set of environment and

Algorithm 11 Vein Distributed Environments

```
1: function VEIN(numberitems, σ, environmentsize)
2:     Select 2 random sides from the grid
3:     Select a random points on each sides, (x_0, y_0) and (x_1, y_1)
4:     calculate gradient of vein, m = (y_0 − y_1)/(x_0 − x_1)
5:     calculate c of equation for line vein, c = (y_0 − m ∗ x_0)
6:     prioritizeditemsleft = floor(E_p*numberitems)
7:     nonprioritizeditemsleft = floor((1-E_p)*numberitems)
8:     while prioritizeditemsleft > 0 do
9:         x ← uniform(min(x_0, x_1), max(x_0, x_1))
10:        y ← m ∗ x + c
11:        if position (x,y) is valid and position (x,y) is empty then
12:            Place prioritized item at (x,y)
13:            Decrement prioritizeditemsleft
14:        end if
15:    end while
16:    while nonprioritizeditemsleft > 0 do
17:        x ← uniform(0, environmentsize)
18:        y ← uniform(0, environmentsize)
19:        if position (x,y) is empty then
20:            Place nonprioritized item at (x,y)
21:            Decrement nonprioritizeditemsleft
22:        end if
23:    end while
24: end function
```

swarm parameters.

Algorithm 12 Clustered Distributed Environments (Part 1)

1: **function** CLUSTERED($numitems, \sigma, environmentsize$)
2: Generate number of clusters, $total = uniform(3, 15)$
3: Calculate number of prioritized clusters, $clusters_p = floor(\sigma * total)$
4: Calculate number of non-prioritized clusters, $clusters_{np} = floor((1 - \sigma) * total)$
5: Calculate number of prioritized items, $total_p = floor(\sigma * numitems)$
6: Calculate number of non-prioritized items, $total_{np} = floor((1 - \sigma) * numitems)$
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: $num_p = uniform(ave_p/2, 2ave_p)$
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)

```
    while clusterstogenerate_np > 0 do
        sample centroid for cluster C (x,y) uniformly from grid
        num_np = uniform(ave_np/2, 2ave_np)
        Calculus radius, r, of cluster C as r = sqrt(num_np/π)
        if cluster C does not collide with existing clusters then
            Save centroid and radius of cluster C to list
            Decrement clusterstogenerate_np
            itemstogenerate_np = num_np
            while itemstogenerate_np > 0 do
                x_p = gaussian(x, deviation = r)
                y_p = gaussian(y, deviation = r)
                if position (x_np, y_np) is valid then
                    Place non-prioritized item at (x_np, y_np)
                    Decrement itemstogenerate_np
                end if
            end while
        end if
    end while
end function
```

6.5 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.

6.5.1 Foraging Efficiency

The foraging efficiency of a particular algorithm is defined as the total number of prioritized items, collected by all robots in a fixed time period, on a specific environment E_P. This foraging efficiency metric is similar to the efficiency metric used by Hecker et al

[130]. 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 metric has been adapted to only look at the prioritized items, for the purpose of examining efficiency in the prioritized foraging problem described in Section 5.1.

6.5.2 Flexibility

As stated in Section 2.2.2, to measure flexibility is to measure of how variations in environments and tasks affect the co-ordination mechanisms of swarm robotics algorithms. An algorithm that is highly flexible, is one that has been optimized for a specific distribution, but is equally efficient on a different distribution [130].

The flexibility performance measures used in this study are based on the flexibility performance measures used in [130], which have been adapted for the prioritized foraging problem, by only taking into account the prioritized item. The flexibility study addresses at two aspects: (i) Flexibility over the prioritized item ratios, r_i, and (ii) flexibility over environment distributions.

Flexibility over prioritized item ratio

This study evaluates flexibility by looking at how swarm efficiency is effected by different environment item ratio r_i, as described in Section 6.3 and swarm specialization ratio, τ_i, as described in Section 6.4. The value for τ, which results in the best average efficiency, E_p, for a specific r_i is denoted as $\hat{\tau}_{r_i}$ and the average efficiency for a specific swarm specialization ratio, τ over specific environment item ratio, r is denoted $E_p(\tau, r)$.

Flexibility over prioritized item ratio, F_r is defined as:

$$F_r = \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{|E_p(\hat{\tau}_{r_i}, r_i) - E_p(\hat{\tau}_{r_i}, r_j)|}{E_p(\hat{\tau}_{r_i}, r_i)} \tag{6.2}$$

The differences are normalized by $E_p(\hat{\tau}_{r_i}, r_i)$ in order to remove the differences in average efficiencies for each algorithm, so that the algorithms can be compared, purely on flexibility. If F_r of one algorithm is large, then that algorithm cannot generalize well over other environment item ratios, and is thus less flexible.

Flexibility over environment distribution types

This study evaluates flexibility, by analysing how swarm efficiency on various environment distributions (ED), described in Section 6.3, is affected by swarm specialization ratio, τ_i (described in Section 6.4).

The value for τ, which results in the best average efficiency, E_P, for a specific ED is denoted as $\hat{\tau}_{ED}$ and the average efficiency for a specific swarm specialization ratio, τ with specific ED is denoted $E_P(\tau, ED)$.

Flexibility over a specific environment distribution ED_i is defined as:

$$F_{ED_i} = \sum_{j=1}^{n} \frac{|E_(\hat{\tau}_{ED_i}, ED_i) - E_p(\hat{\tau}_{ED_i}, ED_j)|}{E_p(\hat{\tau}_{ED_i}, ED_i)} \tag{6.3}$$

Flexibility over all environment distributions ED is defined as:

$$F_{ED} = \sum_{i=1}^{n} F_{ED_i} \tag{6.4}$$

A larger value for F_{ED} of one algorithm, compared to another algorithm, means that the algorithm cannot generalize as well over other environment item distributions. It follows that an algorithm with a larger value for F_{ED} is less flexible.

6.5.3 Scalability

Scalability is described in Section 2.2.3. The study of scalability can be separated into two types: (i) Swarm size scalability and (ii) problem size scalability.

Swarm size Scalability

The efficiency of a robot swarm should be relatively undisturbed by changes in group sizes. This property is known as swarm size scalability. When the efficiency of an algorithm improves linearly (or superlinearly) as swarm size increases, an algorithm is considered scalable. However, due to increased inter-robot interference in larger swarm sizes, scalability is often sub-linear [60]. Many swarm robotics algorithms focus on the use of division of labour, navigation or communication in order to decrease inter-robot interference [60, 159]. The scalability performance measures are based on the scalability

performance measures used in [130], which have been adapted for the prioritized foraging problem, by only taking into account the prioritized items.

In order to evaluate the swarm size scalability of each of the algorithms presented, efficiency of each algorithm is examined over a variety of swarm sizes, c_i, as defined in Section 6.4. In order to compare scalability of the algorithms, the efficiency E_P at each swarm size c is normalized by the efficiency of each algorithm, at the smallest swarm size ($c_0 = 0.1$). Swarm size scalability, SS, for a specific algorithm is calculated as follows:

$$SS(c_i) = \frac{E_P(c_i) - E_P(c_0)}{E_P(c_0)} \tag{6.5}$$

$$SS = \sum_{i=1}^{n} SS(c_i) \tag{6.6}$$

where $E_P(c_i)$ is the average efficiency E_P over all environments with swarm density c_i, and $E_P(c_0)$ is the average efficiency E_P over all experiments where swarm density is c_0. By plotting $SS(c_i)$ for all values of $i \in (0, N)$, one will be able to determine how linear the scalability of each algorithm is. Normalizing each value for $E_P(c_i)$ by $E_P(c_0)$ for each algorithm, removes the specific overall efficiencies of each algorithm, and allows the algorithms to be compared purely on scalability. The total swarm scalability SS, provides a single figure, per algorithm, which one can use to compare overall swarm scalability. The larger SS is, the better the swarm size scalability.

The swarm size scalability study will specifically look at large environments ($S = 500$) to avoid the possibility of an environment running out of items to forage.

Problem Scalability

If the efficiency, E_P, of an algorithm decreases linearly (or super-linearly) as problem complexity increases, then an algorithm can be seen as scalable. Similarly to swarm size scalability, described in Section 6.5.3, problem scalability is often sub-linear, due to increased environmental interference.

In order to evaluate the problem scalability of each of the algorithms presented, efficiency of each algorithm is examined over a variety of environment item densities, p, defined in Section 6.3. In order to compare scalability of the algorithms to each other,

the efficiency E_P at each item density p_i is normalized by the efficiency of each algorithm, at the smallest item density ($p_0 = 0.1$).

Problem scalability (PS) for an algorithm, for at an item density p_i is defined as follows:

$$PS(p_i) = \frac{E_P(p_i) - E_P(p_0)}{E_P(p_0)} \tag{6.7}$$

$$PS = \sum_{i=1}^{n} PS(p_i) \tag{6.8}$$

, where $E_P(p_i)$ is the average efficiency E_P over all environments with environment density p_i, and $E_P(p_0)$ is the average efficiency E_P over all experiments where environment density is p_0. By plotting $PS(c_i)$ for all values of $i \in (0, N)$, one will be able to determine how linear the problem scalability of each algorithm is. Normalizing each value for $E_P(p_i)$ by $E_P(p_0)$, for each algorithm, removes the specific overall efficiencies of each algorithm, and allows the algorithms to be compared purely on scalability.

Similarly to total swarm scalability SS, defined in Section 6.5.3, the total problem scalability, PS, provides a single figure, per algorithm, which one can use to compare overall problem scalability. The larger PS is, the better the problem scalability.

The problem scalability study will specifically look at large environments ($S = 500$) to avoid the possibility of an environment running out of items to forage.

6.5.4 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 why the scalability, flexibility and efficiency metrics were observed.

Items foraged over time

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

- The total prioritized items foraged over time, E_P^t. The total prioritized items foraged are recorded every 200 time steps and are normalized with the total prioritized items that exist in the specific environment.
- The total non-prioritized items foraged over time, E_{NP}^t. The total non-prioritized items foraged are recorded every 200 time steps and are normalized with the total 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 more understanding of how the division of labour mechanism is performing, the swarm specialization ratio over time, τ_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 each robot in a swarm spent on each division of labour activity is recorded as follows:

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

6.6 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 occupy a single grid cell at any time point.

Different types of environment distributions were generated for experimentation. The environment types defined have the four following distributions: uniform distribution, Gaussian distribution, clustered distribution and vein distribution. 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 (i) Items foraged over time (ii) Swarm specialization ratio over time and (iii) the time spent performing recruitment activities.

Chapter 7

Results

In order to evaluate the algorithms, the discussion of the results of the experiments is organised around three major characteristics of swarm robotics algorithms, namely flexibility, robustness and scalability. Section 7.1 addresses foraging efficiency, while flexibility of the algorithms over different environment distributions and item type ratios is analysed in Section 7.2. Section 7.3 evaluates of 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 of each algorithm. Rather than optimizing each swarm parameter, for each algorithm, for each environment, the algorithms are compared using results of experiments on a range of all swarm parameters. If the study was focused on finding the most efficient algorithm for each environment, then the swarm parameters (in particular, initial item type ratio τ_0), for each of the algorithms, would need to be optimized for each environment. As this thesis focuses on the behaviours of each algorithm, and how efficiency of each algorithm is sensitive to environment and swarm parameters, the analysis of the foraging efficiency of each environment, looks at experiments with all swarm and environmental parameters.

In order determine whether samples from each experiment came from the same distri-

Table 7.1: Pairwise one-tailed Mann Whitney U wins and losses of E_p, for each algorithm, over all environments, parameter value choices

Algorithms	Wins	Losses
Naïve	1168	59042
Desert ant	42120	14557
Honey bee	45490	15179

butions, a Wilcoxon test was performed with $\alpha = 0.05$. If the Wilcoxon test determines there is a significant difference between the distributions of samples in two experiments (with the same swarm parameters and environment), then two one-tailed Mann-Whitney U tests are performed to determine which experiment samples came from a greater distribution, with $\alpha = 0.05$. If samples of experiment A is from a greater distribution than the samples of experiment B, then a win is counted for the group of experiments from which experiment A is from and a loss is counted for the group of experiments from which experiment B comes from.

To understand an algorithm's foraging efficiency against all other algorithms, the wins and losses per each experiment performed for each algorithm, are counted. This statistical analysis approach has been used in [194].

Table 7.1 shows the total wins and losses for each algorithm, against all other algorithms. The Desert ant foraging performed better than the Naïve algorithm. The Honey bee algorithm out-performed the Naïve algorithm and Desert ant algorithm. The following chapters address the swarm robotic properties of each algorithm, and explain why the foraging efficiency of the algorithms is as shown in Table7.1.

7.2 Flexibility

This section analyses the flexibility of each of the considered algorthms. Section 7.2.1 analyses flexibility of each algorithm, in terms of the prioritized item ratio of the environment, F_r, as defined Section 6.5.2. Section 7.2.2 discusses the flexibility of each algorithm, in terms of environment distribution types, F_{ED}, as defined in Section 6.5.2.

Table 7.2: Flexibility in terms of prioritized item ratio, F_r, and flexibility in terms of environment distribution, F_{ED}, for each algorithm

	Naïve	Desert ant	Honey bee
F_r	1.18580054603	3.12400224646	0.827991739287
F_{ED}	1.11160672365	0.507371308759	0.458295680454

7.2.1 Flexibility in terms of prioritized item ratio

The performance measure F_r is a measure of how performance chances between different types of specialization ratios. An algorithm that does not benefit from any values of r will be seen as more flexible by the performance measure F_r, as efficiency is eliminated from F_r.

Table 7.2 shows 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 and finally the Desert ant algorithm.

Figure 7.1 shows the specialization ratio, τ, over time, for the Honey bee algorithm on a particular environment. The initial swarm specialization ratio, τ, is set to 0, so that no robots are initially set to forage prioritized items. The environment, on the other hand, consists of mostly prioritized items ($r = 0.75$).

An examination of Figure 7.1 shows that τ increased from 0 to 1 in 200 iterations, which suggests that when no Honey bee scout robots could find any non-prioritized items, the Honey bee scouts switched to looking for prioritized items. After 200 iterations, the specialization ratio, τ, steadily declines throughout the experiment from 1 till 0.73, suggesting that the Honey bee algorithm eventually adapts the specialization ratio, τ, to effectively forage the environment item type ratio r of any given environment.

The above analysis suggests that Honey bee algorithm's flexibility, F_r, can be attributed the Honey bee algorithm's ability to adapt the swarm's specialization ratio, τ, via the division of labour mechanism described in Section 5.4.4, to more efficiently forage an environment of a given environment item ratio, r. The Naïve algorithm and Desert ant algorithm do not have the ability to adapt τ and are less flexible than the Honey bee algorithm.

Despite the Desert ant algorithm outperforming the Naïve algorithm in efficiency, as shown in Section 7.1, the Naïve algorithm is more flexible, according to Table 7.2. Rationally, since the Naïve algorithm does not have high efficiency on any environment, foraging efficiency was not affected very much by any specialization ratio, unable to take advantage of any value for r of τ_0 and thus, will always be equally bad.

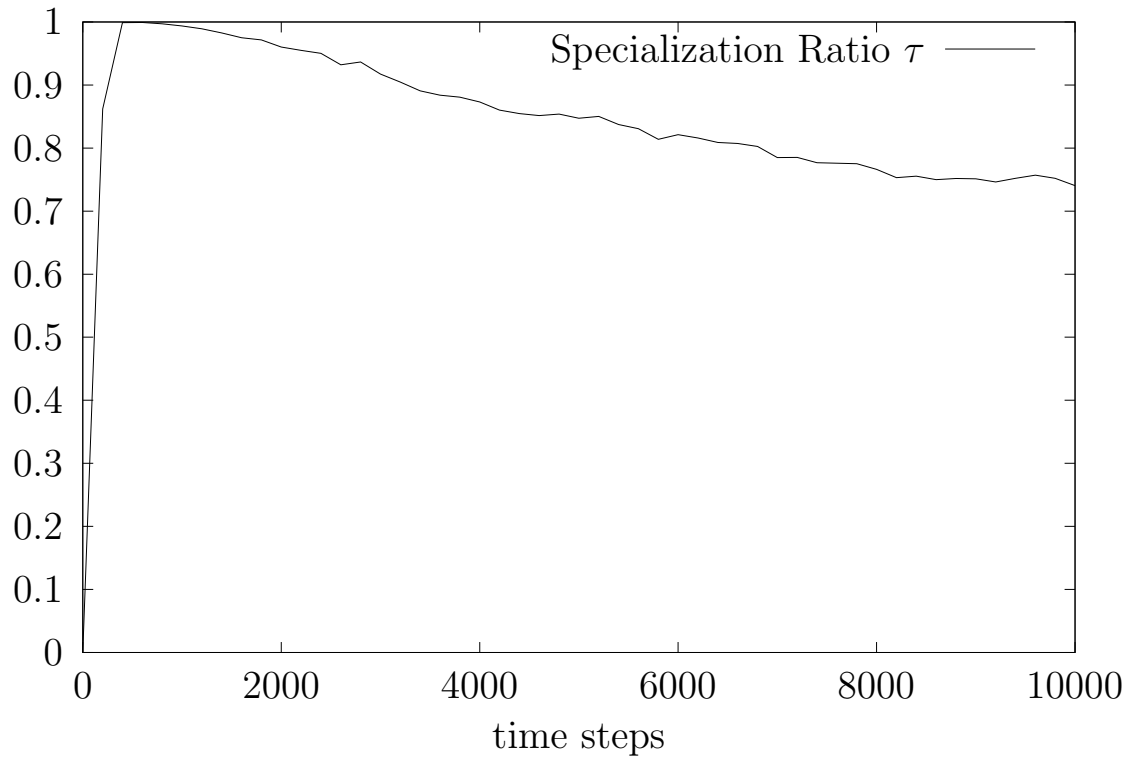


Figure 7.1: Specialization Ratio, τ, of the Honey bee algorithm, over time, when $\tau = 0$ and $r = 0.75$

7.2.2 Flexibility in terms of Environment Distributions

An examination of F_{ED} in Table 7.2 indicates that the Honey bee algorithm was the most flexible, followed by the Desert ant algorithm. The Naïve algorithm was the least flexible.

Each environment distribution was selected to test whether each algorithm has the ability to overcome specific challenges and complexities that result from each environment distributions, which were described in Section 6.3.

As described in Section 5.4, the Honey bee algorithm allows each robot to switch their item type specialization. Section **??** showed that the Honey bee algorithm could adapt τ to adequately forage a specific environment item type ratio r. This suggests, that the Honey bee algorithm could also adapt τ to adapt to localized changes in the environment item type ratio, during the course of foraging.

Consider the environment shown in Figure 7.2 where the concentration of prioritized items in the environment centre,are blocked by non-prioritized items. The white cells are empty, the dark cells are the non-prioritized items and the shaded cells are the prioritized items. The environment in Figure 7.2 has the following properties:

- Gaussian environment item distribution.
- A low environmental item ratio of $r = 0.2$.
- A high density of items $p = 90$.
- $S = 100$

Consider an experiment with the initial specialization ratio set to mostly forage prioritized items ($\tau = 0.8$) where swarm density is set to 0.5, on the Gaussian environment shown in Figure 7.2. The items nearest the sink are all non-prioritized items. Robots will have to forage a large portion 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}, as well as the specialization ratio, τ, for the honey bee algorithm, for each 200 time steps, on the Gaussian environment described above. The specialization ratio τ starts at 0.8 and then very quickly decreases to near 0 in about 200 time steps. The E_{NP}, increases very quickly after 200 time steps, since most of the robots are foraging non-prioritized items (since τ decreases to near 0). After around 800 time steps, τ increases from near 0 to just above 0.7, followed by a sharp increase in

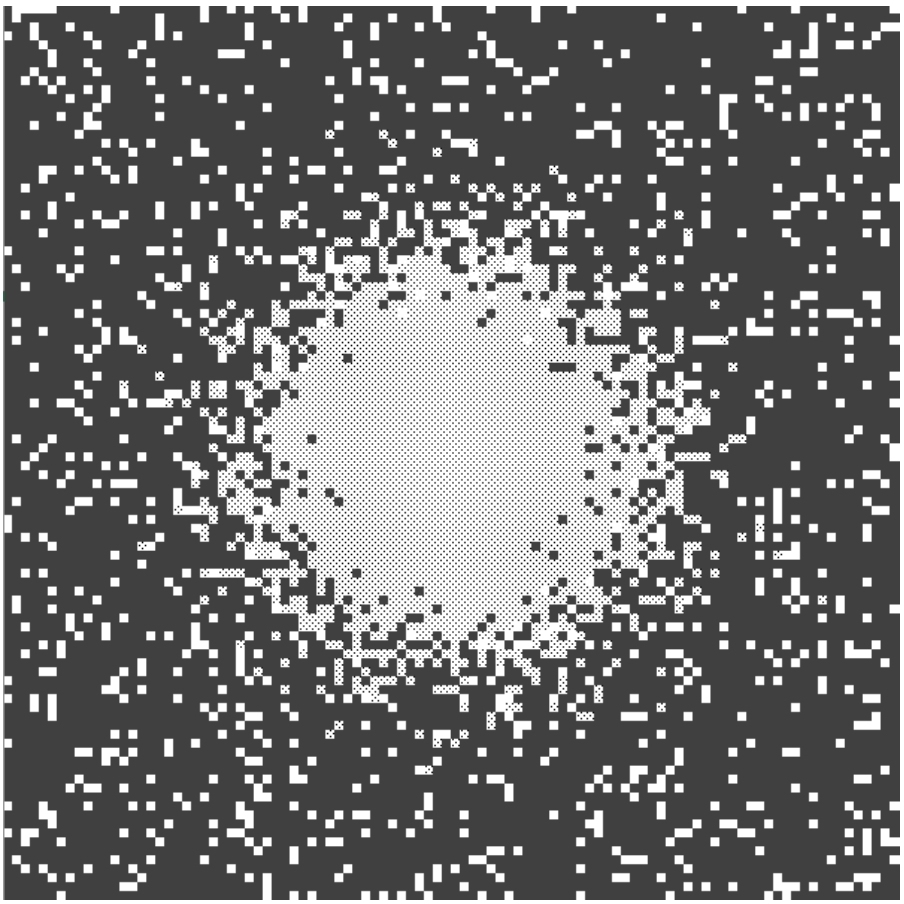


Figure 7.2: Gaussian environment, $r = 0.2$, $p = 90$, $s = 100$

E_P, suggesting that the non-prioritized items have been cleared and the concentration of prioritized items in the centre have been reached, , which resulted in more robots switching to forage prioritized items and a greater efficiency of foraging the prioritized items.

Figure 7.4 shows the efficiency of foraging prioritized items E_P and the efficiency of foraging non-prioritized items E_{NP}, as well as the specialization ratio, τ, for the Naïve algorithm, for each 200 time steps, on the Gaussian environment described above. The swarm specialization, τ, stays constant, since the Naïve algorithm does not enable robots to switch their specialization. As a result, the increase of E_{NP} was slow because only a few robots could forage the non-prioritized items, since $\tau = 0.8$. The non-prioritized were obstacles in the way of reaching the prioritized items in the environment centre. As a

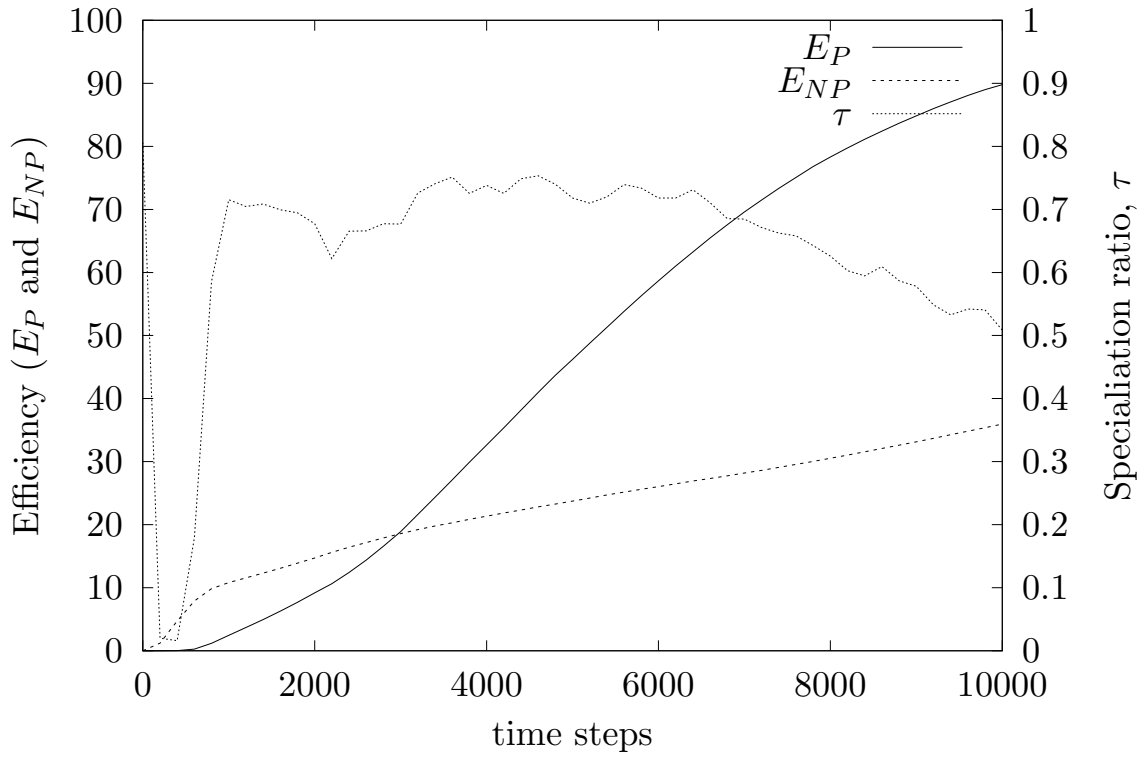


Figure 7.3: Efficiency of prioritized item foraging E_P, efficiency of non-prioritized item foraging E_{NP}, specialization ratio τ over time, for the Honey bee algorithm on a Gaussian environment

result of the swarm's slowness in clearing the non-prioritized items, E_P also remains low since foraging of prioritized items was slowed by the need to navigate around unforaged non-prioritized items. The behaviour for the Desert ant algorithm was similar to that of the Naïve algorithm, except final efficiency was greater for the Desert ant algorithm.

Figure 7.3 shows that final E_P for the Honey bee algorithm 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 too different environment distributions, than the Naïve algorithm.

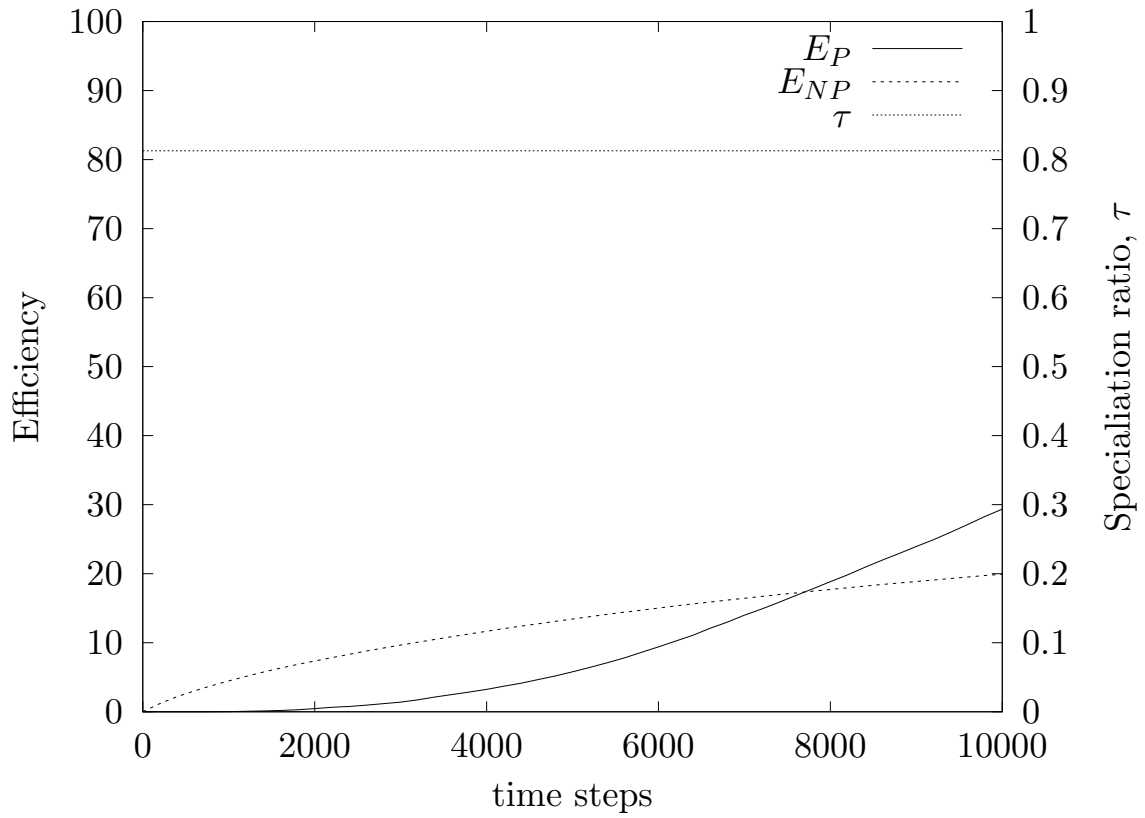


Figure 7.4: Efficiency of prioritized item foraging E_P, efficiency of non-prioritized item foraging E_{NP}, specialization ratio τ over time, for the Naïve algorithm on a Gaussian Environment

Figure 7.5 shows the efficiency of foraging prioritized items E_P, the efficiency of foraging non-prioritized items E_{NP}, as well as the specialization ratio, τ, for the honey bee algorithm, for each 200 time steps, on a Uniform environment, with a similar configuration to the environment in Figure 7.2. The specialization ratio fluctuates randomly between 0.4 and 0.7, while the rate of foraging efficiency, E_P and E_{NP} remains constant throughout the experiment. The random fluctuation of τ seems reasonable, given that the environment distribution was also random.

The Desert ant algorithm outperforms the Naïve algorithm in terms of flexibility over environment distributions, since the Desert ant algorithm uses site fidelity, as discussed in Section 5.3, enabling the Desert ant to more efficiently return to the high concentrations of prioritized items.

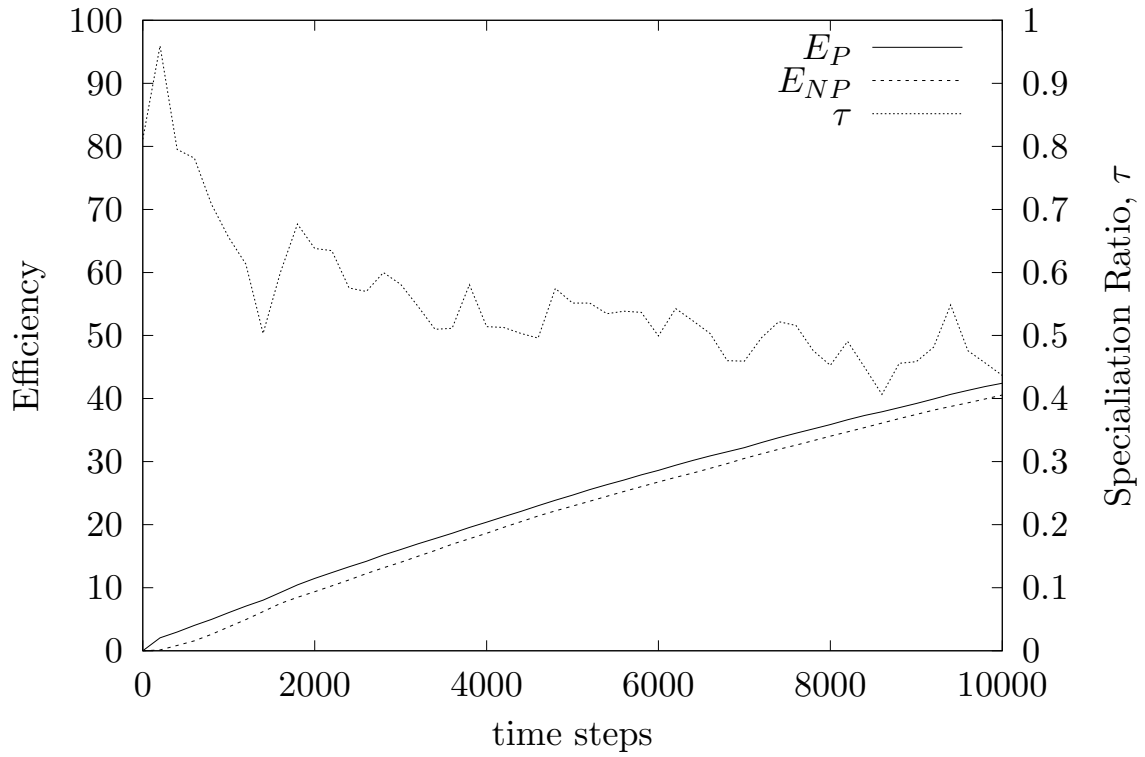


Figure 7.5: Efficiency of prioritized item foraging E_P, efficiency of non-prioritized item foraging E_{NP}, specialization ratio τ over time, for the Honey bee algorithm on a Uniform environment

7.3 Scalability

This section discusses the scalability of each algorithm. Section 7.3.1 discusses scalability of each algorithm, in terms of swarm density, SS, as defined Section 6.5.3 while Section 7.3.2 discusses the scalability of each algorithm, in terms of the complexity of the problem, PS, as defined in Section 6.5.3.

7.3.1 Swarm scalability

According to Table 7.3 and Figure 7.6, that present the swarm density scalability SS of each algorithm, the Naïve algorithm was the most scalable. The Desert ant and Honey

bee algorithms show similar scalability. The Honey bee algorithm at lower densities appears to be more scalable than the Desert ant algorithm, but at the higher densities, SS of the Desert ant algorithm was greater than the Honey bee algorithm.

Figure 7.6 also plots ideal linear scalability, where the increase in swarm density is directly proportional to the increase in efficiency. The performance of all algorithms was sub-linear. In Section 6.5.3, increased inter-robot interference was highlighted as a major factor impacting swarm scalability and thus one can conclude that the reason for sub-linear performance was because all algorithms suffer from increased inter-robot interference as swarm size increases

To understand why the Desert ant and Honey bee algorithm 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 Honey bee algorithms can exploit that high quality area. In exploiting that high quality site, the path between the sink and 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, but due to the focused efforts, the exploitation will still improve performance 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 the overall efficiency due to increased inter-robot interference. It follows that the Desert ant algorithm and 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, since 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 evidential in Table 7.3. Instead, the Honey bee algorithm performed slightly better than the Desert ant algorithm, according to Table 7.3, for everything except the extremely high values of c.

To explain the Honey bee algorithm's ability to outperform the Desert ant algorithm at most swarm densities, one can look to the Honey bee algorithm's the division of labour mechanism, 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't 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 amount, 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 **??**. It is clear from Figure **??** that the robots spend a significant portion of time in the waiting state. However the time spent 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 of average time in the waiting as swarm density increases, due to increases in inter-robot interference.

7.3.2 Problem scalability

Figure 7.9 plots the problem scalability performance measure PS (described in Section 6.5.3) for each algorithm, with values given in Table 7.4. The figure includes a plot of "Expected scalability". "Expected scalability" is the expected values for PS, using the assumption that foraging efficiency would degrade as the problem size increases.

Table 7.3: Swarm scalability, SS, for each swarm density, c, for each algorithm.

c	**0.1**	**0.3**	**0.5**	**0.7**	**1**
Naïve	1	2.148309055	2.95067493	3.567754813	4.25529855
Desert ant	1	1.972768821	2.604500744	3.05833543	3.555516648
Honey bee	1	2.067364168	2.706034502	3.119174449	3.532352867

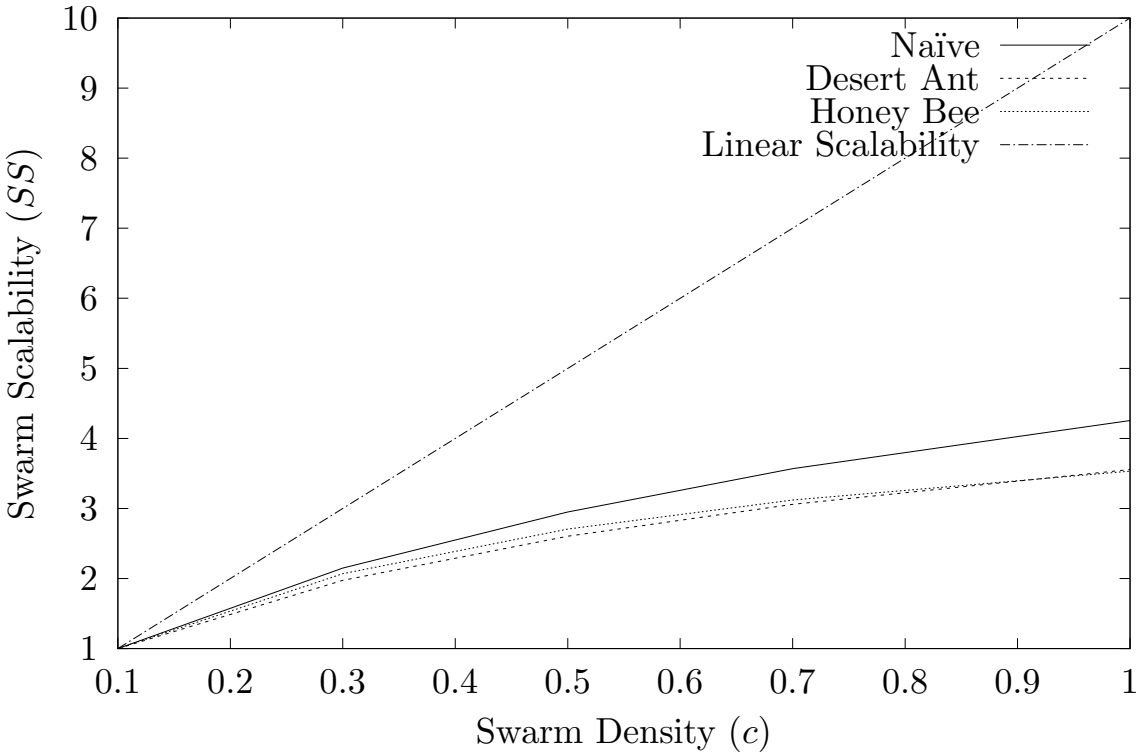


Figure 7.6: Swarm Scalability, SS, for each swarm density c, for each algorithm, as well as linear scalability as swarm density increases.

All algorithm's outperform the expected scalability and thus problem scalability for all algorithms is super-linear.

According to Figure 7.9, the Honey bee algorithm is the most scalable in terms of PS, followed by the Naïve algorithm, with the Desert ant algorithm having the worst scalability.

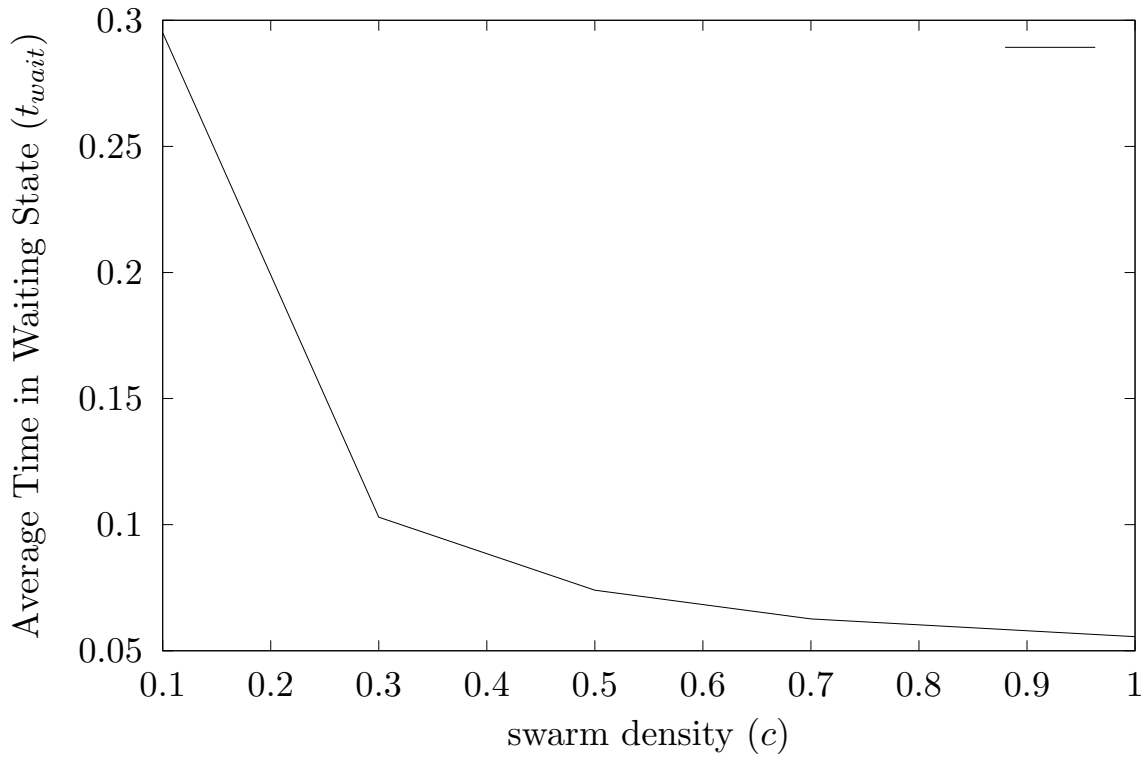


Figure 7.7: Average time in waiting state, t_{wait}, for each swarm density c, for the Honey bee algorithm

As stated in Section 6.5.3, the inability for an algorithm to scale with regards to the size of the problem, is due to ineffectively handling the increased environmental interference that occurs in problems of greater size and complexity.

The Naïve algorithm is more scalable than the Desert ant algorithm. The Desert ant algorithm is the same as the Naïve algorithm, except that the Desert ant algorithm employs site fidelity, described in Section 5.3, to enable a robot to return to a previously foraged site. Rationally, the site fidelity mechanism, directly or indirectly, must be the reason for the decreased scalability of 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 in red in Figure 7.8. 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 in blue 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 PI vector only consists of an overall heading and distance to the site. As is evident on Figure 7.8, there may exist an easier to reach site for prioritized items, but the Desert ant robot will wait time, continuing to forage the hard to locate source of items. On the other hand the Naïve algorithm, due to the fact it can't exploit previously location areas using site fidelity, will not return to the hard to find site and is more likely to randomly find the prioritized resource site that is nearer the sink (indicated by the green path). The time spent on relocating hard to access sites, in more complex environments, can slow the Desert ant algorithm down.

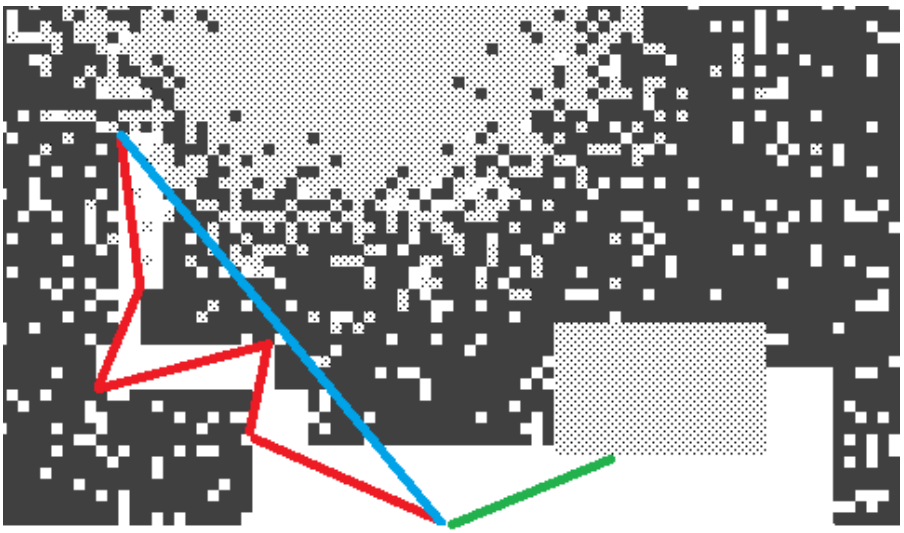


Figure 7.8: Illustration of the inefficiencies of Desert ant algorithm's site fidelity in dense environments. Solid areas are non-prioritized items, white areas are free cells and shaded areas are prioritized items. The red path is a hypothetical random walk performed by a robot, the blue path is the PI vector for the red random walk, and the green path is an alternative random walk.

The Honey bee algorithm uses site fidelity, and also communicates those sites to others. Despite the use of similar site fidelity mechanism to the Desert ant algorithm,

Table 7.4: Problem Scalability, PS, for each environment density p, for each algorithm

p	**0.05**	**0.2**	**0.5**	**0.7**	**0.9**
Desert ant	1	0.487816741	0.231581619	0.181686679	0.167953071
Honey bee	1	0.602283728	0.283264856	0.216749784	0.195686487
Naïve	1	0.501722909	0.256758663	0.195206738	0.174522486
Expected	1	0.25	0.1	0.071428571	0.055555556

the Honey bee algorithm is the most scalable. The reason why the site fidelity of the Honey bee algorithm does not impact 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 7.2.2. Thus efficiency of the Honey bee algorithm in highly dense environments s improved. Therefore, the Honey bee algorithm is the most scalable in terms of the problem size and 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 capability for robustness. Section 7.4.1 discusses robustness in terms of redundancy of each algorithm, and Section 7.4.2 discusses robustness in terms of decentralized coordination.

7.4.1 Redundancy

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

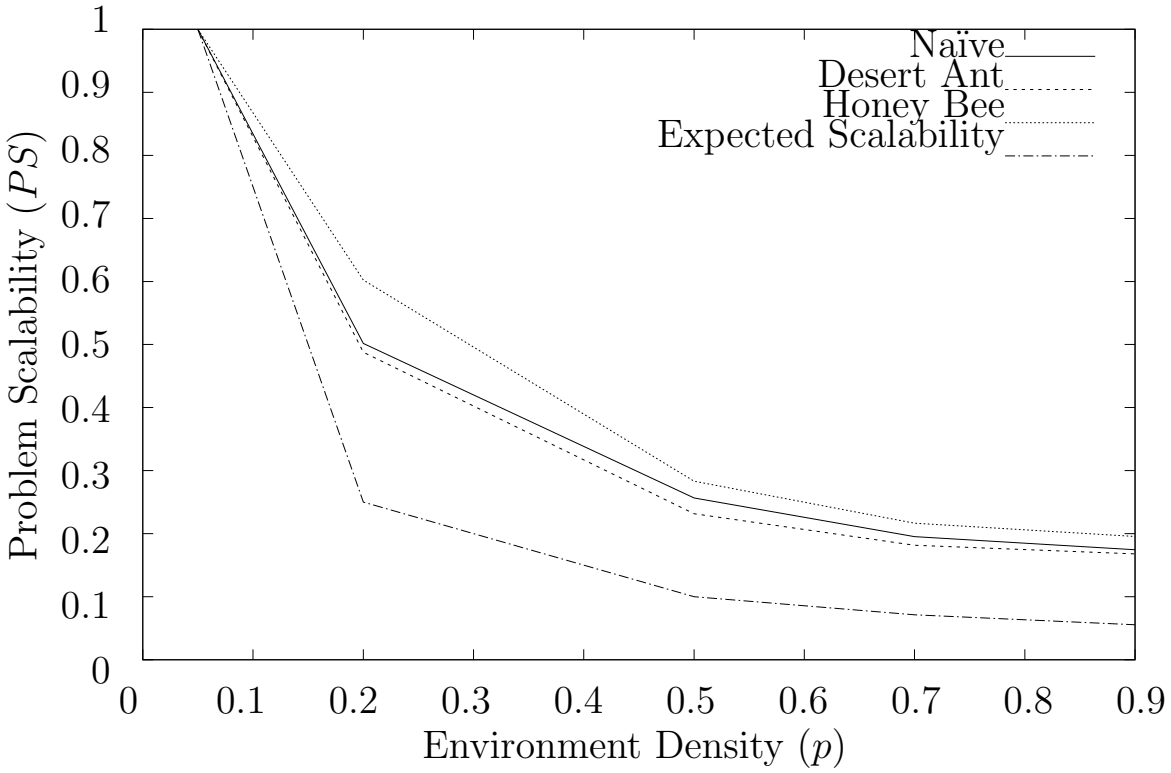


Figure 7.9: Problem Scalability, PS, for each environment density p, for each algorithm

and the another portion to forage non-prioritized items.

If a portion of robots in a swarm malfunction or are destroyed, then the swarm specialization ratio could 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, one would expect changes to foraging efficiency that would persist for the swarm's lifetime. In particular, consider the following extreme scenarios:

Suppose 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 since there do not exist any robots to forage prioritized items. Similarly, consider an environment where all prioritized items are blocked by non-prioritized items, and $\tau < 1$. If all non-prioritized robots suffer a malfunction or are destroyed, then $\tau = 1$ and there would exist

no robots to forage non-prioritized items, and foraging efficiency, E_P, would drop to 0.

As shown in Section 7.2, the Honey bee algorithm experiences little change to efficiency, when τ_0 is varied. The Honey bee algorithm adapted the swarm specialization ratio, τ, over time. It follows that, in the above scenarios, the Honey bee algorithm would be able to re-adapt the swarm specialization ratio, in order to replenish the robots that were destroyed.

The robots in 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 (scout, unemployed forager, employed forager), as well as adapt to forage either prioritized and non-prioritized items. It follows that the Honey bee algorithm is more redundant than the Desert ant and Naïve algorithms.

7.4.2 Decentralized Co-ordination

The Desert ant and Naïve algorithm do not have an explicit coordination mechanism between individuals of a swarm. Any co-ordination that could emerges would be entirely decentralized.

On the other hand, the Honey bee algorithm uses 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 experiences a fault in evaluating the quality of a site (for instance, detecting that a site is of high quality, when it is actually site of poor quality), then foragers would be recruited to forage sites of low quality, incorrectly. Only a few individuals (the scouts), can coordinate the swarm and therefore any faults in the scout robots, can negatively influence effect the efficiency of the entire swarm. The co-ordination mechanisms of the Honey bee algorithm are not entirely decentralized.

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

In a uniformly distributed environment, with a low ratio of prioritized items, sites with a high quality do not exist, as the prioritized items would be scarce and scattered. However, Table 7.5 shows that each robot spent, on average, more than a 10th of the total duration of the experiment broadcasting site quality to other robots in uniform

Table 7.5: Average time steps, per robot, 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	0.163853094	0.162636914	0.159305395	0.163353443
0.2	0.119486333	0.152139262	0.11962912	0.160737113
0.25	0.126752914	0.144484389	0.127568794	0.158813789
0.333333	0.140492087	0.135243974	0.139460537	0.151803907
0.5	0.161644793	0.15738657	0.16335568	0.168543223
0.666667	0.181659692	0.174940591	0.183477822	0.151701447
0.75	0.179285021	0.177420259	0.183721153	0.171862739
0.8	0.182807781	0.182650316	0.186144899	0.190969001
1	0.167397088	0.172990294	0.163048904	0.219223401

environments, when r was between 0 and 0.25. 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, shows that the honey bee algorithm is less robust.

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 around four 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_{ED}.

Section 7.2.1 concluded that the Honey bee algorithm was the most flexible in terms of prioritized item ratio. It was shown that the Honey bee algorithm's flexibility is a result of Honey bee algorithm's ability to adapt the specialization ratio τ, via the Honey bee algorithm's division of labour mechanism. 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 perceived flexibility is attributed to the fact that the Naïve algorithm suffers far lower efficiency across all environment item ratios, suggesting that it appears flexible since it performs equally bad across all values of r, unable to exploit any differences between the environments.

Section 7.2.2 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 it's ability to adapt the specialization ratio to better suit the types of items that are reachable by the swarm. The adaptation of specialization ratio allowed the Honey bee swarm to focus on foraging non-prioritized when only non-prioritized items were reachable, and then adjust the ratio to focus on foraging prioritized items when prioritized items were reachable. The Desert ant and Naïve algorithms, without the ability to adjust τ to more effectively forage the ratio of reachable objects, 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, SS and PS. 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, and then the Desert ant and Honey bee algorithms 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 the Desert ant algorithm's site fidelity increased inter-robot interference, which resulted in poor 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 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 densities increase, 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 showed that the Desert ant's 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, resulted 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.

Section 7.4 addressed robustness of each of the algorithms, in terms of redundancy and decentralized co-ordination. Section 7.4.1 showed that the Honey bee algorithm has the highest redundancy, since the swarm is homogeneous. The Honey bee swarm is homogeneous as 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 algorithm are the most robust in terms of decentralized coordination, since neither algorithm employ a coordination mechanism between robots of the swarm. The Honey bee algorithm was determined to be the least robust in terms of decentralized coordination. The Honey bee algorithm's coordination mechanism was sensitive to faults in certain individuals where invalid information was communicated to the swarm and the invalid information impacted efficiency.

Taking the discussion of the flexibility, scalability, and robustness of each algorithm back to foraging efficiency: The Honey bee algorithm exhibited a high level of flexibility,

problem scalability, and redundancy, resulting in greater overall efficiency than the Desert ant algorithm and Naïve algorithm. On the other hand, despite the Naïve algorithm exhibiting better scalability than the Desert ant algorithm, the Desert ants algorithm's ability to exploit high quality areas, resulted in greater average performance than the Naïve algorithm.

Chapter 8

Conclusions

8.1 Summary of Conclusions

8.2 Future Work

Bibliography

[1] R. R. Murphy et al. "Search and rescue robotics". In: *Springer Handbook of Robotics*. Springer, 2008, pp. 1151–1173.

[2] A. M. Naghsh et al. "Analysis and design of human-robot swarm interaction in firefighting". In: *The 17th IEEE international symposium on Robot and human interactive communication, 2008. RO-MAN 2008*. IEEE. 2008, pp. 255–260.

[3] M. Dorigo and E. Sahin. "Swarm Robotics". In: *Autonomous Robots* 17 (2-3 2004), pp. 111–113.

[4] A. F. T. Winfield. "Foraging Robots". In: *Encyclopedia of Complexity and Systems Science*. 2009, pp. 3682–3700.

[5] E. Sahin. "Swarm Robotics: From Sources of Inspiration to Domains of Application". In: *Swarm Robotics*. Springer, 2005, pp. 10–20.

[6] N. R. Hoff et al. "Two Foraging Algorithms for Robot Swarms Using Only Local Communication". In: *Proceedings of the 2010 IEEE International Conference on Robotics and Biomimetics*. IEEE, 2010, pp. 123–130.

[7] S. Garnier et al. "Aggregation Behaviour as a Source of Collective Decision in a Group of Cockroach-like-Robots". In: *Advances in Artificial Life*. Springer, 2005, pp. 169–178.

[8] J.-H. Lee, H.-T. Kim, and C. W. 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*. ACM, 2012, p. 100.

[9] E. O. Wilson et al. "The Insect Societies". In: *The Insect Societies* (1971).

[10] T. Schmickl and K. Crailsheim. “A Navigation Algorithm for Swarm Robotics Inspired by Slime Mold Aggregation”. In: *Swarm Robotics*. Springer, 2007, pp. 1–13.

[11] A. Dhariwal, G. Sukhatme, and A. A. Requicha. “Bacterium-Inspired Robots for Environmental Monitoring”. In: *Proceedings of the 2004 IEEE International Conference on Robotics and Automation*. Vol. 2. IEEE, 2004, pp. 1436–1443.

[12] S. Martel and M. 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*. IEEE, 2010, pp. 500–505.

[13] M. Brambilla et al. “Swarm Robotics: A Review from the Swarm Engineering Perspective”. In: *Swarm Intelligence* 7.1 (2013), pp. 1–41.

[14] D. W. Morley. “Division of Labour in Ants”. In: *Nature* 158 (1946), pp. 913–914.

[15] S. N. Beshers and J. H. Fewell. “Models of Division of Labor in Social Insects”. In: *Annual Review of Entomology* 46.1 (2001), pp. 413–440.

[16] B. P. Gerkey and M. J. Matari. “A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems”. In: *The International Journal of Robotics Research* 23.9 (2004), pp. 939–954.

[17] T. H. Labella, M. Dorigo, and J.-L. Deneubourg. “Division of Labor in a Group of Robots Inspired by Ants’ Foraging Behavior”. In: *ACM Transactions on Autonomous and Adaptive Systems* 1.1 (2006), pp. 4–25.

[18] W. Liu et al. “Towards Energy Optimization: Emergent Task Allocation in a Swarm of Foraging Robots”. In: *Adaptive Behavior* 15.3 (2007), pp. 289–305.

[19] E Bahge ci and others. “Evolving Aggregation Behaviors for Swarm Robotic Systems: A Systematic Case Study”. In: *Proceedings of the 2005 IEEE Swarm Intelligence Symposium*. IEEE, 2005, pp. 333–340.

[20] S. Nouyan, A. Campo, and M. Dorigo. “Path Formation in a Robot Swarm”. In: *Swarm Intelligence* 2.1 (2008), pp. 1–23.

[21] D. Zarzhitsky, D. F. Spears, and W. M. Spears. "Distributed Robotics Approach to Chemical Plume Tracing". In: *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 4034–4039.

[22] T. Balch. "Hierarchic Social Entropy: An Information Theoretic Measure of Robot Group Diversity". In: *Autonomous Robots* 8.3 (2000), pp. 209–238.

[23] M. Rubenstein, C. Ahler, and R. Nagpal. "Kilobot: A Low Cost Scalable Robot System for Collective Behaviors". In: *Proceedings of the 2012 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2012, pp. 3293–3298.

[24] A. Kushleyev et al. "Towards a Swarm of Agile Micro Quadrotors". In: *Autonomous Robots* 35.4 (2013), pp. 287–300.

[25] D. Mellinger et al. "Cooperative Grasping and Transport Using Multiple Quadrotors". In: *Distributed Autonomous Robotic Systems*. Springer, 2013, pp. 545–558.

[26] F. Mondada et al. *Search for Rescue: An Application for the SWARM-BOT Self-Assembling Robot Concept*. Citeseer, 2002.

[27] T. Balch et al. "Io, Ganymede, and Callisto - a Multiagent Robot Trash-Collecting Team". In: *AI Magazine* 16.2 (1995), p. 39.

[28] N. Correll and A. 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] T. Balch and R. C. Arkin. "Behavior-Based Formation Control for Multirobot Teams". In: *IEEE Transactions on Robotics and Automation* 14.6 (1998), pp. 926–939.

[30] M. Dorigo, M. Birattari, and M. Brambilla. "Swarm Robotics". In: *Scholarpedia* 9.1 (2014), p. 1463.

[31] G. Beni and J. Wang. "Swarm Intelligence in Cellular Robotic Systems". In: *Robots and Biological Systems: Towards a New Bionics?* Springer, 1993, pp. 703–712.

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

[33] R. A. Brooks. “Elephants Don’t Play Chess”. In: *Robotics and Autonomous Systems* 6.1 (1990), pp. 3–15.

[34] R. Brooks and others. “A Robust Layered Control System for a Mobile Robot”. In: *IEEE Journal of Robotics and Automation* 2.1 (1986), pp. 14–23.

[35] J. H. Connell. *A Colony Architecture for an Artificial Creature*. DTIC Document, 1989.

[36] M. J. Mataric. *A Distributed Model for Mobile Robot Environment-Learning and Navigation*. DTIC Document, 1990.

[37] A. M. Flynn et al. “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*. IEEE, 1989, pp. 98–101.

[38] R. A. Brooks. “A Robot That Walks; Emergent Behaviors from a Carefully Evolved Network”. In: *Neural Computation* 1.2 (1989), pp. 253–262.

[39] R. C. Arkin. “Integrating Behavioral, Perceptual, and World Knowledge in Reactive Navigation”. In: *Robotics and Autonomous Systems* 6.1 (1990), pp. 105–122.

[40] T. Arai, E. Pagello, and L. E. Parker. “Editorial: Advances in Multi-Robot Systems”. In: *IEEE Transactions on Robotics and Automation* 18.5 (2002), pp. 655–661.

[41] J. Sudd. “A Model of Digging Behaviour and Tunnel Production in Ants”. In: *Insectes Sociaux* 22.2 (1975), pp. 225–235.

[42] R. T. Ryti and T. J. Case. “Spatial Arrangement and Diet Overlap between Colonies of Desert Ants”. In: *Oecologia* 62.3 (1984), pp. 401–404.

[43] T. D. Seeley, S. Camazine, and J. Sneyd. “Collective Decision-Making in Honey Bees: How Colonies Choose among Nectar Sources”. In: *Behavioral Ecology and Sociobiology* 28.4 (1991), pp. 277–290.

[44] H. de Vries and J. C. Biesmeijer. "Modelling Collective Foraging by Means of Individual Behaviour Rules in Honey-Bees". In: *Behavioral Ecology and Sociobiology* 44.2 (1998), pp. 109–124.

[45] J. L. R. Lopez. "Optimal Foraging in Seed-Harvester Ants: Computer-Aided Simulation". In: *Ecology* 68 (1987), pp. 1630–1633.

[46] C. R. Kube and H. Zhang. "Collective Robotics: From Social Insects to Robots". In: *Adaptive Behavior* 2.2 (1993), pp. 189–218.

[47] E Freund. "On the Design of Multi-Robot Systems". In: *Proceedings of the 1984 IEEE International Conference on Robotics and Automation*. Vol. 1. IEEE, 1984, pp. 477–490.

[48] Y. U. Cao, A. S. Fukunaga, and A. Kahng. "Cooperative Mobile Robotics: Antecedents and Directions". In: *Autonomous Robots* 4.1 (1997), pp. 7–27.

[49] H. Asama et al. *Distributed Autonomous Robotic Systems 2*. Springer Science & Business Media, 2013.

[50] M. J. Mataric, M. Nilsson, and K. 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'*. Vol. 3. IEEE, 1995, pp. 556–561.

[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*. Vol. 3. IEEE, 1986, pp. 103–111.

[52] H. Asama, A. Matsumoto, and Y. Ishida. "Design of an Autonomous and Distributed Robot System: ACTRESS". In: *Proceedings of the 1989 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 1989, pp. 283–290.

[53] T. Fukuda, Y. Kawauchi, and M. 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*. IEEE, 1989, pp. 291–296.

[54] T. Fukuda, Y. Kawauchi, and H. 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'*. IEEE, 1990, pp. 827–834.

[55] G. Beni and J. Wang. “Theoretical Problems for the Realization of Distributed Robotic Systems”. In: *Proceedings of the 1991 IEEE International Conference on Robotics and Automation*. IEEE, 1991, pp. 1914–1920.

[56] R. Vaughan. “Massively Multi-Robot Simulation in Stage”. In: *Swarm Intelligence* 2 (2-4 2008), pp. 189–208.

[57] C. Pinciroli et al. “ARGoS: A Modular, Multi-Engine Simulator for Heterogeneous Swarm Robotics”. In: *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2011, pp. 5027–5034.

[58] A. Martinoli, A. J. Ijspeert, and F. Mondada. “Understanding Collective Aggregation Mechanisms: From Probabilistic Modelling to Experiments with Real Robots”. In: *Robotics and Autonomous Systems* 29.1 (1999), pp. 51–63.

[59] K. Lerman et al. “A Macroscopic Analytical Model of Collaboration in Distributed Robotic Systems”. In: *Artificial Life* 7.4 (2001), pp. 375–393.

[60] K. Lerman and A. Galstyan. “Mathematical Model of Foraging in a Group of Robots: Effect of Interference”. In: *Autonomous Robots* 13.2 (2002), pp. 127–141.

[61] V. Trianni et al. “Modeling Pattern Formation in a Swarm of Self-Assembling Robots”. In: *European Community Grant 1st-2000-31010* (2002).

[62] A. Campo and M. Dorigo. “Efficient Multi-Foraging in Swarm Robotics”. In: *Advances in Artificial Life*. Springer, 2007, pp. 696–705.

[63] H. Hamann and H. Wrn. “A Framework of Space–Time Continuous Models for Algorithm Design in Swarm Robotics”. In: *Swarm Intelligence* 2 (2-4 2008), pp. 209–239.

[64] A. Prorok, N. Corell, and A. Martinoli. “Multi-Level Spatial Modeling for Stochastic Distributed Robotic Systems”. In: *The International Journal of Robotics Research* 30 (2011), pp. 574–589.

[65] V. Gazi and K. M. Passino. “Stability of a One-Dimensional Discrete-Time Asynchronous Swarm”. In: *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics* 35.4 (2005), pp. 834–841.

[66] Y. Liu and K. M. Passino. “Stable Social Foraging Swarms in a Noisy Environment”. In: *IEEE Transactions on Automatic Control* 49.1 (2004), pp. 30–44.

[67] M. Schwager et al. “Time Scales and Stability in Networked Multi-Robot Systems”. In: *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2011, pp. 3855–3862.

[68] A. F. Winfield et al. “On Formal Specification of Emergent Behaviours in Swarm Robotic Systems”. In: *International Journal of Advanced Robotic Systems* 2.4 (2005), pp. 363–370.

[69] S. Konur, C. Dixon, and M. Fisher. “Analysing Robot Swarm Behaviour via Probabilistic Model Checking”. In: *Robotics and Autonomous Systems* 60.2 (2012), pp. 199–213.

[70] N. Mathews et al. “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*. International Foundation for Autonomous Agents and Multiagent Systems, 2010, pp. 939–946.

[71] T. H. Labella, M. Dorigo, and J.-L. Deneubourg. “Efficiency and Task Allocation in Prey Retrieval”. In: *Biologically Inspired Approaches to Advanced Information Technology*. Springer, 2004, pp. 274–289.

[72] O. Soysal and others. “Probabilistic Aggregation Strategies in Swarm Robotic Systems”. In: *Proceedings of the 2005 IEEE Swarm Intelligence Symposium*. IEEE, 2005, pp. 325–332.

[73] D. J. Bennet and C. R. McInnes. “Distributed Control of Multi-Robot Systems Using Bifurcating Potential Fields”. In: *Robotics and Autonomous Systems* 58.3 (2010), pp. 256–264.

[74] L. Barnes, M.-A. Fields, and K. Valavanis. “Unmanned Ground Vehicle Swarm Formation Control Using Potential Fields”. In: *Proceedings of the Mediterranean Conference on Control & Automation*. IEEE, 2007, pp. 1–8.

[75] D. H. Kim, H. Wang, and S. Shin. “Decentralized Control of Autonomous Swarm Systems Using Artificial Potential Functions: Analytical Design Guidelines”. In: *Journal of Intelligent and Robotic Systems* 45.4 (2006), pp. 369–394.

[76] K. Samejima and T. Omori. “Adaptive Internal State Space Construction Method for Reinforcement Learning of a Real-World Agent”. In: *Neural Networks* 12.7 (1999), pp. 1143–1155.

[77] R. Sun and T. Peterson. “Multi-Agent Reinforcement Learning: Weighting and Partitioning”. In: *Neural Networks* 12.4 (1999), pp. 727–753.

[78] G. Baldassarre, S. Nolfi, and D. Parisi. “Evolving Mobile Robots Able to Display Collective Behaviors”. In: *Artificial Life* 9.3 (2003), pp. 255–267.

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

[80] G. Francesca et al. “AutoMoDe: A Novel Approach to the Automatic Design of Control Software for Robot Swarms”. In: *Swarm Intelligence* 8.2 (2014), pp. 89–112.

[81] G. Francesca et al. “An Experiment in Automatic Design of Robot Swarms”. In: *Swarm Intelligence*. Springer, 2014, pp. 25–37.

[82] A. J. Ijspeert et al. “Collaboration through the Exploitation of Local Interactions in Autonomous Collective Robotics: The Stick Pulling Experiment”. In: *Autonomous Robots* 11.2 (2001), pp. 149–171.

[83] R. Fujisawa et al. “Designing Pheromone Communication in Swarm Robotics: Group Foraging Behavior Mediated by Chemical Substance”. In: *Swarm Intelligence* 8.3 (2014), pp. 227–246.

[84] E. J. Barth. “A Dynamic Programming Approach to Robotic Swarm Navigation Using Relay Markers”. In: *Proceedings of the 2003 American Control Conference*. Vol. 6. IEEE, 2003, pp. 5264–5269.

[85] S. Camazine. *Self-Organization in Biological Systems*. Google-Books-ID: zMgyNN6Ufj0C. Princeton University Press, 2003. ISBN: 978-0-691-11624-2.

[86] X. Yan, A. Liang, and H. Guan. “An Algorithm for Self-Organized Aggregation of Swarm Robotics Using Timer”. In: *Proceedings of the 2011 IEEE Symposium on Swarm Intelligence*. IEEE, 2011, pp. 1–7.

[87] O. Soysal, E. Bah ceci, and E. Sahin. “Aggregation in Swarm Robotic Systems: Evolution and Probabilistic Control”. In: *Turkish Journal Electrical Engineering* 15.2 (2007), pp. 199–225.

[88] V. Trianni et al. “Evolving Aggregation Behaviors in a Swarm of Robots”. In: *Advances in Artificial Life*. Springer, 2003, pp. 865–874.

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

[90] T. Schmickl et al. “Two Different Approaches to a Macroscopic Model of a Bio-Inspired Robotic Swarm”. In: *Robotics and Autonomous Systems* 57.9 (2009), pp. 913–921.

[91] E. Bahceci, O. Soysal, and E. Sahin. “A Review: Pattern Formation and Adaptation in Multi-Robot Systems”. In: *Robotics Institute, Carnegie Mellon University, Pittsburgh, Technical Report, CMU-RI-TR-03-43* (2003).

[92] S. Hettiarachchi et al. “A Review and Implementation of Swarm Pattern Formation and Transformation Models”. In: *International Journal of Intelligent Computing and Cybernetics* 2.4 (2009), pp. 786–817.

[93] S. Nouyan and M. Dorigo. “Chain Based Path Formation in Swarms of Robots”. In: *Ant Colony Optimization and Swarm Intelligence*. Springer, 2006, pp. 120–131.

[94] S. Goss and J.-L. Deneubourg. “Harvesting by a Group of Robots”. In: *Proceedings of the First European Conference on Artificial Life*. 1992, pp. 195–204.

[95] B. B. Werger and M. J. Mataric. “Robotic” food” chains: Externalization of State and Program for Minimal-Agent Foraging”. In: *In (Maes et Al.* Citeseer, 1996.

[96] S. Nouyan et al. “Group Transport Along a Robot Chain in a Self-Organised Robot Colony”. In: *Proceedings of the 9th International Conference on Intelligent Autonomous Systems*. 2006, pp. 433–442.

[97] R. Gro and M. Dorigo. “Self-Assembly at the Macroscopic Scale”. In: *Proceedings of the IEEE* 96.9 (2008), pp. 1490–1508.

[98] R. Beckers, O. Holland, and J.-L. Deneubourg. “From Local Actions to Global Tasks: Stigmergy and Collective Robotics”. In: *Artificial Life IV*. Vol. 181. 1994, p. 189.

[99] J. Wawerla, G. S. Sukhatme, and M. J. Matari. “Collective Construction with Multiple Robots”. In: *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 3. IEEE, 2002, pp. 2696–2701.

[100] J. Werfel and R. Nagpal. “Extended Stigmergy in Collective Construction”. In: *Intelligent Systems, IEEE* 21.2 (2006), pp. 20–28.

[101] J. K. Werfel, K. Petersen, and R. Nagpal. “Distributed Multi-Robot Algorithms for the TERMES 3D Collective Construction System”. In: Institute of Electrical and Electronics Engineers, 2011.

[102] A. Howard, M. J. Matari, and G. S. Sukhatme. “Mobile Sensor Network Deployment Using Potential Fields: A Distributed, Scalable Solution to the Area Coverage Problem”. In: *Distributed Autonomous Robotic Systems 5*. Springer, 2002, pp. 299–308.

[103] T. Stirling and D. Floreano. “Energy Efficient Swarm Deployment for Search in Unknown Environments”. In: *Swarm Intelligence*. Springer, 2010, pp. 562–563.

[104] D. W. Payton et al. “Pheromone Robotics”. In: *Intelligent Systems and Smart Manufacturing*. International Society for Optics and Photonics, 2001, pp. 67–75.

[105] F. Ducatelle et al. “Self-Organized Cooperation between Robotic Swarms”. In: *Swarm Intelligence* 5.2 (2011), pp. 73–96.

[106] J. K. Parrish, S. V. Viscido, and D. Grnbaum. "Self-Organized Fish Schools: An Examination of Emergent Properties". In: *The Biological Bulletin* 202.3 (2002), pp. 296–305.

[107] A. E. Turgut et al. "Self-Organized Flocking in Mobile Robot Swarms". In: *Swarm Intelligence* 2 (2-4 2008), pp. 97–120.

[108] E. Ferrante et al. "Flocking in Stationary and Non-Stationary Environments: A Novel Communication Strategy for Heading Alignment". In: *Parallel Problem Solving from Nature, PPSN XI*. Springer, 2010, pp. 331–340.

[109] H. Sugie et al. "Placing Objects with Multiple Mobile Robots-Mutual Help Using Intention Inference". In: *Proceedings of the 1995 IEEE International Conference on Robotics and Automation*. Vol. 2. IEEE, 1995, pp. 2181–2186.

[110] B. P. Gerkey and M. J. Mataric. "Sold!: Auction Methods for Multirobot Coordination". In: *IEEE Transactions on Robotics and Automation* 18.5 (2002), pp. 758–768.

[111] R. Gro and M. Dorigo. "Group Transport of an Object to a Target That Only Some Group Members May Sense". In: *Parallel Problem Solving from Nature-VIII*. Springer, 2004, pp. 852–861.

[112] M. Dorigo et al. "The Swarm-Bots Project". In: *Swarm Robotics*. Springer, 2005, pp. 31–44.

[113] E. Ferrante et al. "Socially-Mediated Negotiation for Obstacle Avoidance in Collective Transport". In: *Distributed Autonomous Robotic Systems*. Springer, 2013, pp. 571–583.

[114] T. D. Seeley and S. C. Buhrman. "Nest-Site Selection in Honey Bees: How Well Do Swarms Implement The" best-of-N" decision Rule?" In: *Behavioral Ecology and Sociobiology* 49.5 (2001), pp. 416–427.

[115] J. Gautrais et al. "Emergent Polyethism as a Consequence of Increased Colony Size in Insect Societies". In: *Journal of Theoretical Biology* 215.3 (2002), pp. 363–373.

[116] C. Jones and M. J. Mataric. “Adaptive Division of Labor in Large-Scale Minimalist Multi-Robot Systems”. In: *Proceedings of Intelligent Robots and Systems, 2003*. Vol. 2. IEEE, 2003, pp. 1969–1974.

[117] M. J. Krieger and J.-B. Billeter. “The Call of Duty: Self-Organised Task Allocation in a Population of up to Twelve Mobile Robots”. In: *Robotics and Autonomous Systems* 30.1 (2000), pp. 65–84.

[118] E. Sahin and A. Winfield. “Special Issue on Swarm Robotics”. In: *Swarm Intelligence* 2.2 (2008), pp. 69–72.

[119] S. Luke et al. “Mason: A Multiagent Simulation Environment”. In: *Simulation* 81.7 (2005), pp. 517–527.

[120] O. Michel. “Webots: Symbiosis between Virtual and Real Mobile Robots”. In: *Virtual Worlds*. Springer, 1998, pp. 254–263.

[121] B. Hlldobler. *The Ants*. Harvard University Press, 1990.

[122] R. A. Bernstein. “Seasonal Food Abundance and Foraging Activity in Some Desert Ants”. In: *American Naturalist* (1974), pp. 490–498.

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

[124] J. S. Jennings, G. Whelan, and W. F. Evans. “Cooperative Search and Rescue with a Team of Mobile Robots”. In: *Proceedings of the 8th International Conference on Advanced Robotics, 1997*. IEEE, 1997, pp. 193–200.

[125] R. R. Murphy. “Biomimetic Search for Urban Search and Rescue”. In: *Proceedings of the 2000 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 3. IEEE, 2000, pp. 2073–2078.

[126] M. Dorigo, E. Bonabeau, and G. Theraulaz. “Ant Algorithms and Stigmergy”. In: *Future Generation Computer Systems* 16.8 (2000), pp. 851–871.

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

[128] M. Dorigo and M. Birattari. “Ant Colony Optimization”. In: *Encyclopedia of Machine Learning*. Springer, 2010, pp. 36–39.

[129] T. S. Collett et al. “Visual Landmarks and Route Following in Desert Ants”. In: *Journal of Comparative Physiology A* 170.4 (1992), pp. 435–442.

[130] J. P. Hecker and M. E. Moses. “Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms”. In: *Swarm Intelligence* 9.1 (2015), pp. 43–70.

[131] M. Collett et al. “Local and Global Vectors in Desert Ant Navigation”. In: *Nature* 394.6690 (1998), pp. 269–272.

[132] R. Wehner. “Desert Ant Navigation: How Miniature Brains Solve Complex Tasks”. In: *Journal of Comparative Physiology A* 189.8 (2003), pp. 579–588.

[133] M. Mller and R. Wehner. “Path Integration in Desert Ants, Cataglyphis Fortis”. In: *Proceedings of the National Academy of Sciences* 85.14 (1988), pp. 5287–5290.

[134] R. Mller et al. “Modeling Ant Navigation with an Autonomous Agent”. In: *From Animals to Animats* 5 (1998), pp. 185–194.

[135] J. P. Hecker et al. “Formica Ex Machina: Ant Swarm Foraging from Physical to Virtual and Back Again”. In: *Swarm Intelligence*. Springer, 2012, pp. 252–259.

[136] S. Zhang et al. “Honeybee Memory: A Honeybee Knows What to Do and When”. In: *Journal of Experimental Biology* 209.22 (2006), pp. 4420–4428.

[137] R. Menzel and M. Giurfa. “Cognitive Architecture of a Mini-Brain: The Honeybee”. In: *Trends in Cognitive Sciences* 5.2 (2001), pp. 62–71.

[138] D. Moore et al. “The Influence of Time of Day on the Foraging Behavior of the Honeybee, Apis Mellifera”. In: *Journal of Biological Rhythms* 4.3 (1989), pp. 305–325.

[139] S. Janson, M. Middendorf, and M. Beekman. “Searching for a New Homescouting Behavior of Honeybee Swarms”. In: *Behavioral Ecology* 18.2 (2007), pp. 384–392.

[140] K. M. Passino. “Bacterial Foraging Optimization”. In: *International Journal of Swarm Intelligence Research (IJSIR)* 1.1 (2010), pp. 1–16.

[141] J. Kennedy, R. Eberhart, and others. "Particle Swarm Optimization". In: *Proceedings of IEEE International Conference on Neural Networks*. Vol. 4. Perth, Australia, 1995, pp. 1942–1948.

[142] E. L. Charnov. "Optimal Foraging, the Marginal Value Theorem". In: *Theoretical Population Biology* 9.2 (1976), pp. 129–136.

[143] S. Nolfi and D. Floreano. "Coevolving Predator and Prey Robots: Do arms Races Arise in Artificial Evolution?" In: *Artificial Life* 4.4 (1998), pp. 311–335.

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

[145] E. H. stergaard, G. S. Sukhatme, and M. 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] M. Dorigo et al. "Swarmanoid: A Novel Concept for the Study of Heterogeneous Robotic Swarms". In: *IEEE Robotics & Automation Magazine* 20.4 (2013), pp. 60–71.

[147] K. Sugawara and T. Watanabe. "Swarming Robots-Foraging Behavior of Simple Multirobot System". In: *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Vol. 3. IEEE, 2002, pp. 2702–2707.

[148] G. Dudek et al. "A Taxonomy for Swarm Robots". In: *Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems'*. Vol. 1. IEEE, 1993, pp. 441–447.

[149] W. Liu et al. "Strategies for Energy Optimisation in a Swarm of Foraging Robots". In: *International Workshop on Swarm Robotics*. Springer, 2006, pp. 14–26.

[150] T. Schmickl and K. Crailsheim. "Trophallaxis among Swarm-Robots: A Biologically Inspired Strategy for Swarm Robotics". In: *The First IEEE/RAS-EMBS International Conference on Biomedical Robotics and Biomechatronics*. IEEE, 2006, pp. 377–382.

[151] P. Vincent and I. Rubin. “A Framework and Analysis for Cooperative Search Using UAV Swarms”. In: *Proceedings of the 2004 ACM Symposium on Applied Computing*. ACM, 2004, pp. 79–86.

[152] C. R. Kube and E. Bonabeau. “Cooperative Transport by Ants and Robots”. In: *Robotics and Autonomous Systems* 30.1 (2000), pp. 85–101.

[153] A. F. Winfield. “Towards an Engineering Science of Robot Foraging”. In: *Distributed Autonomous Robotic Systems 8*. Springer, 2009, pp. 185–192.

[154] M. J. Krieger, J.-B. Billeter, and L. Keller. “Ant-like Task Allocation and Recruitment in Cooperative Robots”. In: *Nature* 406.6799 (2000), pp. 992–995.

[155] R. C. Arkin. “Cooperation without Communication: Multiagent Schema-Based Robot Navigation”. In: *Journal of Robotic Systems* 9.3 (1992), pp. 351–364.

[156] V. Karpov and I. Karpova. “Leader Election Algorithms for Static Swarms”. In: *Biologically Inspired Cognitive Architectures* 12 (2015), pp. 54–64.

[157] M. Hoeing et al. “Auction-Based Multi-Robot Task Allocation in Comstar”. In: *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*. ACM, 2007, p. 280.

[158] D. Goldberg and M. J. Mataric. “Design and Evaluation of Robust Behavior-Based Controllers for Distributed Multi-Robot Collection Tasks”. In: *Robot Teams: From Diversity to Polymorphism*. A K Peters Ltd, 2001, pp. 315–344.

[159] M. Schneider-Fontan and M. J. Mataric. “Territorial Multi-Robot Task Division”. In: *IEEE Transactions on Robotics and Automation* 14.5 (1998), pp. 815–822.

[160] *The impact of diversity on performance in multi-robot foraging*. ACM. 1999, pp. 92–99.

[161] T. Balch. “The Impact of Diversity on Performance in Multi-Robot Foraging”. In: *Proceedings of the Third Annual Conference on Autonomous Agents*. ACM, 1999, pp. 92–99.

[162] R. T. Vaughan et al. “Blazing a Trail: Insect-Inspired Resource Transportation by a Robot Team”. In: *Distributed Autonomous Robotic Systems 4*. Springer, 2000, pp. 111–120.

[163] J.-H. Lin, L.-R. 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*. World Scientific and Engineering Academy and Society (WSEAS), 2009, pp. 84–89.

[164] S. Kernbach et al. "Re-Embodiment of Honeybee Aggregation Behavior in an Artificial Micro-Robotic System". In: *Adaptive Behavior* 17.3 (2009), pp. 237–259.

[165] T. Schmickl, C. Mslinger, and K. Crailsheim. "Collective Perception in a Robot Swarm". In: *Swarm Robotics*. Springer, 2007, pp. 144–157.

[166] S. Alers et al. "Biologically Inspired Multi-Robot Foraging". In: *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-Agent Systems*. International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 1683–1684.

[167] A. Saxena, J. Driemeyer, and A. Y. Ng. "Robotic Grasping of Novel Objects Using Vision". In: *The International Journal of Robotics Research* 27.2 (2008), pp. 157–173.

[168] F. Mondada et al. "The Cooperation of Swarm-Bots: Physical Interactions in Collective Robotics". In: *IEEE Robotics & Automation Magazine* 12.2 (2005), pp. 21–28.

[169] B. Browning and M. Veloso. "Real-Time, Adaptive Color-Based Robot Vision". In: *Proceedings of the 2005 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2005, pp. 3871–3876.

[170] M. Jngel, J. Hoffmann, and M. Ltzsch. "A Real-Time Auto-Adjusting Vision System for Robotic Soccer". In: *Robot Soccer World Cup*. Springer, 2003, pp. 214–225.

[171] V. Trianni, S. Nolfi, and M. Dorigo. "Cooperative Hole Avoidance in a Swarm-Bot". In: *Robotics and Autonomous Systems* 54.2 (2006), pp. 97–103.

[172] X. Zhou. "Motion Induced Robot-to-Robot Extrinsic Calibration". PhD thesis. University Of Minnesota, 2012.

[173] J. A. Rothermich, M. h. Ecemi s, and P. Gaudiano. "Distributed Localization and Mapping with a Robotic Swarm". In: *International Workshop on Swarm Robotics*. Springer, 2004, pp. 58–69.

[174] G. E. Robinson. "Regulation of Division of Labor in Insect Societies". In: *Annual Review of Entomology* 37.1 (1992), pp. 637–665.

[175] G. E. Robinson and R. E. Page. "Genetic Basis for Division of Labor in an Insect Society". In: *Nature Reviews Genetics* 9 (1989), pp. 61–80.

[176] R. E. Page Jr and S. D. Mitchell. "Self Organization and Adaptation in Insect Societies". In: *Proceedings of the Biennial Meeting of the Philosophy of Science Association*. JSTOR, 1990, pp. 289–298.

[177] E. Bonabeau and G. Theraulaz. "Role and Variability of Response Thresholds in the Regulation of Division of Labor in Insect Societies". In: *Information Processing in Social Insects*. Springer, 1999, pp. 141–163.

[178] G. E. Robinson. "Regulation of Honey Bee Age Polyethism by Juvenile Hormone". In: *Behavioral Ecology and Sociobiology* 20.5 (1987), pp. 329–338.

[179] N. R. Franks and C. Tofts. "Foraging for Work: How Tasks Allocate Workers". In: *Animal Behaviour* 48.2 (1994), pp. 470–472.

[180] C. Tofts. "Algorithms for Task Allocation in ants.(A Study of Temporal Polyethism: Theory)". In: *Bulletin of Mathematical Biology* 55.5 (1993), pp. 891–918.

[181] G. E. Julian and S. Cahan. "Undertaking Specialization in the Desert Leaf-Cutter Ant Acromyrmex Versicolor". In: *Animal Behaviour* 58.2 (1999), pp. 437–442.

[182] G. Theraulaz, E. Bonabeau, and J. Denuebourg. "Response Threshold Reinforcements and Division of Labour in Insect Societies". In: *Proceedings of the Royal Society of London B: Biological Sciences* 265.1393 (1998), pp. 327–332.

[183] J. M. Pasteels, J.-L. Deneubourg, and others. *From Individual to Collective Behavior in Social Insects*. Birkhuser Verlag, 1987.

[184] A. J. Spencer, I. D. Couzin, and N. R. Franks. "The Dynamics of Specialization and Generalization within Biological Populations". In: *Advances in Complex Systems* 1 (01 1998), pp. 115–127.

[185] Z.-Y. Huang and G. E. Robinson. "Honeybee Colony Integration: Worker-Worker Interactions Mediate Hormonally Regulated Plasticity in Division of Labor". In: *Proceedings of the National Academy of Sciences* 89.24 (1992), pp. 11726–11729.

[186] D. M. Gordon, B. C. Goodwin, and L. E. Trainor. "A Parallel Distributed Model of the Behaviour of Ant Colonies". In: *Journal of Theoretical Biology* 156.3 (1992), pp. 293–307.

[187] S. W. Pacala, D. M. Gordon, and H. Godfray. "Effects of Social Group Size on Information Transfer and Task Allocation". In: *Evolutionary Ecology* 10.2 (1996), pp. 127–165.

[188] W. Agassounon and A. 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*. ACM, 2002, pp. 1090–1097.

[189] P. V. Switzer. "Site Fidelity in Predictable and Unpredictable Habitats". In: *Evolutionary Ecology* 7.6 (1993), pp. 533–555.

[190] F. Mondada et al. "The E-Puck, a Robot Designed for Education in Engineering". In: *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*. Vol. 1. 2009, pp. 59–65.

[191] P. Antoniou et al. "Congestion control in wireless sensor networks based on bird flocking behavior". In: *Computer Networks* 57.5 (2013), pp. 1167–1191.

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

[193] H. Frimmel and W. Minter. "Recent Developments Concerning the Geological History and Genesis of the Witwatersrand Gold Deposits, South Africa". In: *Special Publication-Society of Economic Geologists* 9 (2002), pp. 17–46.

[194] M. Helbig and A. P. Engelbrecht. "Performance Measures for Dynamic Multi-Objective Optimisation Algorithms". In: *Information Sciences* 250 (2013), pp. 61–81.

Appendix A

Acronyms

Appendix B

Symbols

The following appendix defines the symbols used in each chapter.

B.1 Chapter 3: Foraging

$p1$ The probability that a robot will changed from rest to searching is adapted upon return to the sink and from deposit to rest.

B.2 Chapter 4: Division Of Labour

R The set of tasks that can be performed by individuals of the swarm.

γ An individual robot in a swarm.

ν A task such that $\nu \in R$

$r_{\gamma,\nu}$ A response threshold for of robot γ for task ν.

z The probability of a robot to switch from search behaviour to rest behaviour.

B.3 Chapter 5: Nature Inspired Algorithms for Prioritized Foraging

i The current time step in a swarm robot experiment

$state$ The state a particular robot is in.

$role$ The role of a particular robot in the honey bee algorithm.

i_{state} The current time step of being in consecutive state $state$, where $state$ is any valid state for the robot.

t_{wait} The maximum time a robot can spend consecutively in wait state.

t_{ls} The maximum time a robot can spend consecutively in the local search state.

t_{forage} The maximum time a robot can spend consecutively in the forage state.

$t_{explore}$ The maximum time a robot can spend consecutively in the explore state

t_{dance} The maximum time a robot can spend consecutively in the recruitment state.

ς The type of an item which can either be prioritized or non-prioritized.

μ_{ς} The evaluation of the quality of a site of type ς.

Φ site quality threshold determining whether a the scout robot switches into the dance state.

v A robot's path integration vector

ω A memorized path integration vector representing the location of a site.

ϑ An item found by a robot.

ξ The site where an item is found.

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.

k_j is the value captured on the $j-$th distance sensor of a robot.

n The number of distance sensors.

d The direction a robot must take to get to the sink.

α The probability of listening to the details communicated by the scout robot.

f_{max} The maximum time a robot will forage for prioritized items, without finding any before switching to foraging a non-prioritized type.

X The percentage of robots initialized as scout robots.

B.4 Chapter 6: Experimental Setup

d Desirability of a direction i.

i A direction a robot can perceive.

κ_i The clarity which indicates the distance of next nearest obstacle.

v The depth of view of a robot's perception.

ι_i The directness of a direction i, calculated as the angular deviation from the direction of the destination

f The field of view of a robot's perception.

λ A ratio which determines whether clarity, κ_i or directness, ι_i of direction i, has more effect on desirability d.

S The size of the environment grid.

p The density of the items on the grid.

r The ratio of prioritized to non-prioritized items.

τ The ratio of robots foraging prioritized items to the ratio of robots foraging non-prioritized items.

c The density of robots.

E_P The percentage of prioritized items foraged over time.

E_{NP} The percentage of non-prioritized items foraged over time.

F_r The flexibility of a robot swarm in terms of prioritized item ratio.

SS The scalability of a robot swarm in terms of swarm density.

PS The scalability of a robot swarm in terms of problem size.

E_P^t Total prioritized items foraged over time t.

E_{NP}^t Total non-prioritized items foraged over time t.

τ_t Swarm specialization ratio τ, over time t.

t_{wait} The average percentage of total time steps spent by each robot in the waiting state.

$t_{recruitment}$ The average percentage of total time steps spent by each robot in the recruitment state.

E_P **The percentage of the total number of prioritized**