6-1-2017

Swarm robotics: Cooperative navigation in unknown environments

Abdallah Galal Khalil

The American University In Cairo

Master's Thesis Document

Swarm Robotics: Cooperative Navigation in Unknown Environments

Abdallah Galal

Supervisor
Dr. Khaled El-Ayat

A thesis document submitted in fulfillment of the requirements for the degree of Master's of Science

in the

Department of Computer Science and Engineering

May 2017

AMERICAN UNIVERSITY IN CAIRO
Department of Computer Science and Engineering

Abstract

Swarm Robotics:
Cooperative Navigation in Unknown Environments

Swarm Robotics is garnering attention in the robotics field due to its substantial benefits. It has been proven to outperform most other robotic approaches in many applications such as military, space exploration and disaster search and rescue missions. It is inspired by the behavior of swarms of social insects such as ants and bees. It consists of a number of robots with limited capabilities and restricted local sensing. When deployed, individual robots behave according to local sensing until the emergence of a global behavior where they, as a swarm, can accomplish missions individuals cannot. In this research, we propose a novel exploration and navigation method based on a combination of Probabilistic Finite Sate Machine (PFSM), Robotic Darwinian Particle Swarm Optimization (RDPSO) and Depth First Search (DFS). We use V-REP Simulator to test our approach. We are also implementing our own cost effective swarm robot platform, AntBOT, as a proof of concept for future experimentation. We prove that our proposed method will yield excellent navigation solution in optimal time when compared to methods using either PFSM only or RDPSO only. In fact, our method is proved to produce 40% more success rate along with an exploration speed of 1.4x other methods. After exploration, robots can navigate the environment forming a Mobile Ad-hoc Network (MANET) and using the graph of robots as network nodes.

Acknowledgements

I would like to thank my role model and supervisor, Dr. Khaled El-Ayat, for his patient guidance, encouragement and continuous support ...

I would also like to thank my parents for always being by my side in every possible way. You showed me the right path ...

I would also like to thank my amazing wife, Yomn, for the sleepless nights and for always pushing me forward to get this done ...

A very special thank you goes to Mr Nikolaus Starzacher, Discovergy CEO, for his amazing help and review to this document ...

Last but not least, I would like to thank all my awesome friends who helped me finish this life milestone ...

Contents

List of Figures

List of Tables

Abbreviations

SRS	**S**warm **R**obotics **S**ystem
PFSM	**P**robabilistic **F**inite **S**tate **M**achine
RL	**R**einforcement **L**earning
ER	**E**volutionary **R**obotics
PSO	**P**article **S**warm **O**ptimization
WSN	**W**ireless **S**ensors **N**etwork
MANET	**M**obile **A**d-hoc **Net**work
GPS	**G**lobal **P**ositioning **S**ystem
V-REP	**V**irtual **R**obot **E**xperimentation **P**latform
RDPSO	**R**obotic **D**arwinian **P**article **S**warm **O**ptimization
DFS	**D**epth **F**irst **S**earch
BFS	**B**readth **F**irst **S**earch
AntBOT	A F**ant**astic Swarm Ro**bot**

To my Super Mum who has always been there for me ...

Chapter 1

Introduction

In this research, we present a new exploration approach to help heterogeneous robots navigate safely in unknown and hazardous environments cooperatively using concepts of swarm robotics. Swarm Robotics is inspired by the biological swarms of social insects such as ants and bees. In ant colonies for example, hundreds of thousands of ants collaboratively communicate to bring food to their colony and ensure its survival without any master commander, relying only on local information. This collective ant behavior results in huge tasks being achieved that a small number of ants cannot achieve on their own. In Swarm Robotics, behavior emerges by deployment of small, affordable and heterogeneous robots. Robots rely on their collective behavior to accomplish the task. A powerful feature of swarm robotics is the absence of a leader thus eliminating single point of failure problems. This is one of the main reasons for the popularity of Swarm Robotics.

In Swarm Robotics system design, the behavior of the system heavily depends on the emergent global collective behavior of all robots and not on a single behavior from a single robot. Studying this global emergent behavior is the main issue of Swarm Robotics because there is no clear formula that would produce (x) global behavior based on certain (y) local behavior of a single robot and vice versa. In this research, we survey different methods to evaluate this global behavior published in the swarm robotics literature. We also propose a new method of environment exploration and navigation based on Robotic Probabilistic Finite State Machine (PFSM), Darwinian Particle Swarm Optimization (RDPSO) and Depth First Search (DFS).

An example of swarm robotics application in disaster search and rescue missions is the retrieval of trapped earthquake survivors where rescuers are not able to safely lift debris. Swarm robots

Robot dependent variables	**Environment dependent variables**
Number of Robots	Size Area of the environment
Size of a single robot	Density of obstacles in the environment
Robots compatibility (Heterogeneity)	Geometry of the environment

Table 1.1: Robot vs. Environment Variables

collaboratively evaluate the situation and report their findings. In our experiment, we focus on similar search and rescue missions namely, search for a bomb and diffusing it. The swarm of robots are deployed to explore and navigate an environment while avoiding potential obstacles. Note that optimum initial deployment of the swarm to explore and make local decisions is still unknown. Our robots form a moving Mobile Ad-Hoc Network (MANET) to maintain stable network communication between members of the swarm by addressing the variables summarized in table 1.1. The table lists the main variables affecting the swarm mission; some are related to the robots themselves and others are related to the environment they are exploring.

After sufficient environment exploration, robots must be able to navigate the environment avoiding obstacles even in case of robot failures. Since Swarm Robotics research is still in its infancy, there is no available comparison between different deployment or exploration approaches. This research aims to provide such comparison. Different from all previous researches which present optimal exploration results, our approach is designed to produce optimal results with excellent performance in optimum time. Subsequently, our swarm of robots must be able to guide other robots through the environment they already explored. Swarm robotics is all about having small and affordable robots accomplishing big tasks. As such, we decided to design and build our own swarm robot platform (AntBOT) to avoid purchasing expensive commercial off the shelf robotic kits. It is based on the Pololu 3pi robot controlled by ARM mbed microcontroller. More details on the new proposed platform are available in appendix A.2.

The problem we are solving in this work is discussed further in the following section. In-depth literature review of current approaches to Swarm Robotics is described in chapter 2, while our proposed solution is explained in chapter 3. Chapter 4 contains details of our experimentation methodology and how we evaluated the performance of our system followed by the results of our experimentation in chapter 5. The final chapter 6 lists future work along with the conclusion. Our swarming platform and all its technical specifications together with the simulator model are described in detail in appendix A.

1.1 Problem Definition

Swarm Robotics research has been growing for the past decade exploring many applications and theories. In this research, we focus on a model of real world applications namely search and rescue (SaR) models where robots are required to navigate an unknown environment searching for certain targets. One such example is exploring and searching for potential bombs. Once found, a bomb diffuser robot navigates the shortest calculated path using other swarm nodes as network communication nodes.

The main problem with the current available methods in the area of swarm exploration and navigation is that they discuss the solution on either the individual behavior of a single robot or the collective behavior of all robots. This actually limits the capabilities of the developer when designing a swarm robotic system. Prior Swarm Robotics research uses either microscopic or macroscopic techniques as discussed in section 2.2. The microscopic approach implements the system on a single robot, then maps its behavior to other swarm robots until the desired behavior emerges. For macroscopic level design, desired swarm behavior is designed, then it is mapped to each robot. The current issue with Swarm Robotics design is that researchers approach the problem either on the microscopic or the macroscopic level. This limits their ability to verify their system design. A great deal of effort is then expended in trial and error looking for convergence to the desired behavior. In this work, we discuss a novel solution considering both individual and collective behaviors at the same time as found in chapter 4.

The advantage of Swarm Robotics is that it can provide good solutions where other traditional search techniques fall short. In most hazardous scenarios, speed and efficiency are required since human life may be at stake. Swarm robotics has major advantages in SaR missions:

1. *Robust*: Execution of the mission is distributed among several nodes where failure of some would not affect the completeness of the mission. Nevertheless, it might affect the performance or the time taken to complete the mission.
2. *Scalable*: Insertion or deletion of swarm nodes are allowed. The swarm can adopt a variable swarm size without major effects on the performance of the swarm.
3. *Simple*: Like in most of biological swarm systems: simple, local sensing nodes are able to collectively accomplish great missions.

4. *Parallelizable*: Although the mission can be done using a single complicated huge robot, a swarm can finish it with the same efficiency in less time due to the parallel nature of the swarm.
5. *Economical*: Swarm systems cost much less than conventional huge complicated systems. Also, losing a few swarm nodes is more cost effective than losing a well equipped expensive robot.

This work capitalizes on the above advantages to help provide a scalable, robust and a rapid solution to efficiently explore a hazardous unknown environment using robots. The swarm would be able to easily navigate the environment guiding other robots without any network infrastructure.

Chapter 2

Literature Review

In this chapter, previous work done in the field of Swarm Robotics is reviewed. To get a comprehensive understanding of the current advancements in Swarm Robotics research, we cover research done in this field including work not directly related to our proposed work. Work related to our method is discussed in more details than others. Most research in the field of Swarm Robotics fall into two categories, swarm robotic applications or design methods to build Robotic Swarm systems. In a similar review, authors analyzed the available swarm approaches from an engineering perspective where they focused on ideas and concepts that are important for real-life swarm applications [1]. They defined swarm engineering as "*An emerging discipline that aims at defining systematic and real founded procedure for modeling, designing, realizing, verifying, validating, operating and maintaining a swarm robotics system*". Following the above definition of swarm robotics, robots must be: autonomous, able to do local sensing, able to communicate, suitable for the environment, with no centralized control and able to collaborate to achieve a certain task. The behavior of a swarm robotics system is inspired by social animals where they exhibit swarm intelligence where behavior appears to be robust, scalable and flexible.

Robustness comes from the fact that there is no central node and any loss of any individual node will not affect the systems mission. Scalability means that the performance will not be dramatically affected if the number of robots in the system is decreased or increased. Flexibility is the ability to adopt different scenarios and environments without a major effect on performance. As defined, swarm engineering is the mechanism where scientific and technical knowledge is used to model and design swarm intelligent systems. Swarm engineering was first defined in 2000 as follows "to the swarm engineer, the important points in the design of a swarm are that the

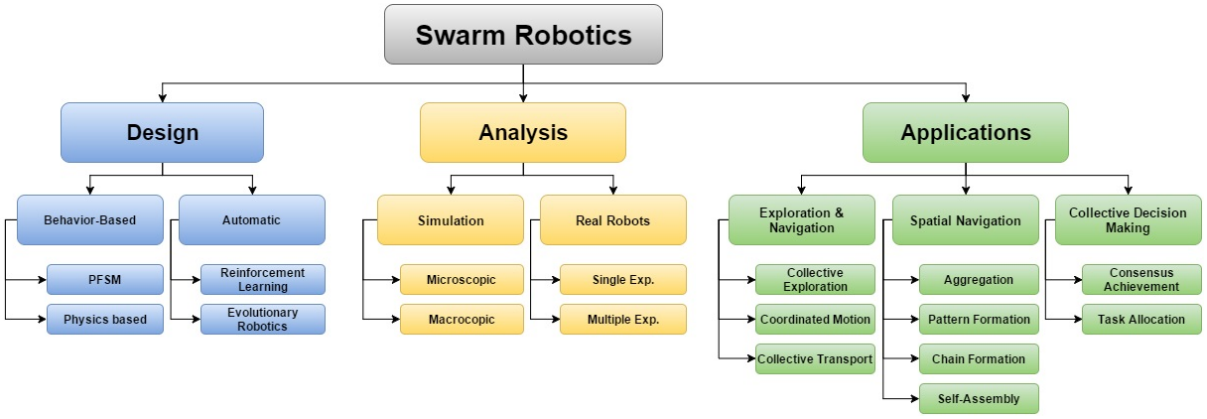


Figure 2.1: Literature Review Tree

swarm will do precisely what it is designed to do and that it will do so reliably and on time" [2, 3]. It is worth mentioning at this stage that Swarm Robotics is still on its infancy and most of the currently available research only addresses the design and analysis of a swarm system while other aspects such as requirements analysis, maintenance and performance measurement are not adequately explored yet.

In other reviews, researchers adopted different comparison schemes to review and evaluate different swarming approaches. One review chose swarm size, communication range, communication topology, communication bandwidth, swarm reconfigurability and swarm unit processing ability to evaluate the available literature [4]. Using group architecture, resource conflicts, origins of cooperation, learning and geometric problems was adopted in another review [5]. In another approach, authors grouped the available literature into aware versus unaware cooperation between robots [6]. Some other researchers divided available work into three main parts: mathematical models, swarm coordination and control, and design approaches [7]. Some other researchers categorized the literature into five approaches: modeling, behavior design, communication, analytical studies, and problems [8]. A tree view of our literature review is shown in figure 2.1 and is divided into three main categories:

1. Design of Swarm Robotics Systems.
2. Analysis of Swarm Robotics Systems.
3. Swarm Robotics Applications.

2.1 Design Methods

We discuss design and analysis methods to evaluate the performance of swarm robotics systems and to assess their real emergent properties against the desired properties. There are two types of design methods: 1) Behavior based Design Methods, and 2) Automatic Design Methods.

2.1.1 Behavior based Design Methods

In swarm robotics applications, there is no clear formula for designers that will produce a certain global behavior X based on local behavior Y or vice versa. Hence, design of swarm systems tend to be a trial and error process. Designers keep tuning their system design on the robots until the desired emergent behavior is reached. Although most of the available literature rely on this bottom-up approach, there exists some recent top-down approaches [9]. Most of the Behavior Based Design methods use either: a) Probabilistic Finite State Machine or b) Virtual-Physics based design methods.

2.1.1.1 Probabilistic finite state machine (PFSM)

In a robot swarm system, its really hard to predict the future of the swarm and the completeness of the mission because the global behavior of a swarm depends solely on the behavior of individuals in the swarm. No individual in the swarm can expect its next move because they act based on dynamic sensory data [10]. Probabilistic finite state machine is one of the design approaches used in swarm robotics systems to study their behavior and was first introduced by Minsky in 1967 [11].

There are two different models to PFSMs, one where probability is fixed and is applied all over the system until convergence to a solution [12]; the other one is a variable probability based on a mathematical model changing based on input from other robots and the environment. Convergence to a solution in this case is evaluated using a model for the varying probability called response threshold [13].

In a study of collective decision making and task allocation, response threshold function shown in figure 2.2 has been introduced in swarm robotics and was used to analyze the behavior of a swarm of social insects [14, 15] where the threshold is the likelihood for an agent to perform a

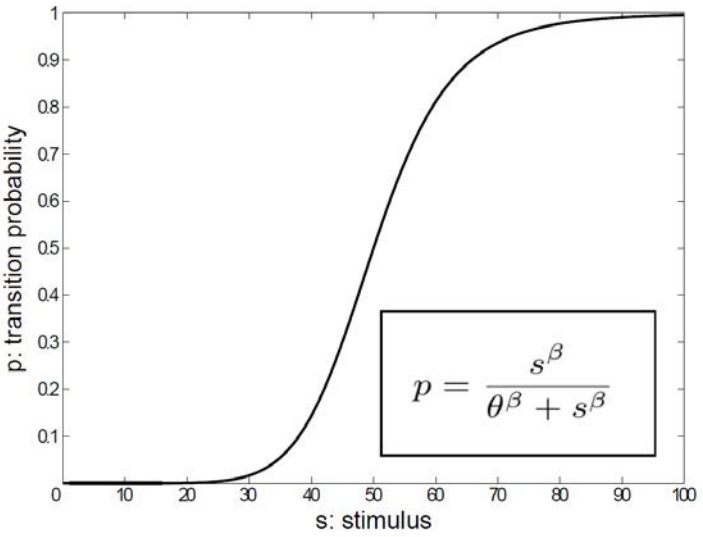


Figure 2.2: Response Threshold Model

task based on perceived stimuli. PFSMs were used to develop major collective behaviors such as: Aggregation [12], Chain Formation [16] and Task Allocation [17, 18].

2.1.1.2 Virtual physics-based design

Taking inspiration from the law of physics, virtual physics-based design were developed in a way where each agent in the swarm is considered as a virtual particle that exerts forces on other agents in the environment. Artificial potential field concept is adopted by many researchers where robots are considered as virtual forces, obstacles as virtual repulsive forces that repel with robots and the goal as a virtual attractive force that attracts other robots towards it [19, 20]. Physicomimetics which is derived from the words physics and imitation is a framework which assumes robots are aware of the environment itself and other robots in it and can easily communicate with all detected robots [21] and is used to model physics-based design approaches. A virtual force vector is computed using the following formula:

$$f = \sum_{i=1}^{k} f_i(d_i) e^{j\theta_i} \tag{2.1}$$

Where θ_i is the direction and d_i is the distance to the i_{th} robot or obstacle and the function $f_i(d_i)$ is derived from an artificial potential function, where the most commonly used one is the Lennard-Jones potential function shown in figure 2.3. The potential v depends on the current distance d between two robots. σ is the desired distance between the robots and ϵ corresponds to

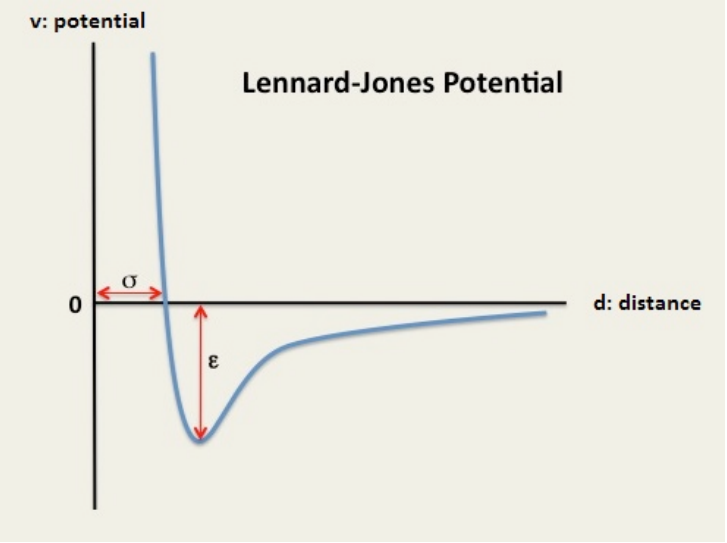


Figure 2.3: Lennard-Jones Potential Function

the depth of the potential function where the deeper the ϵ, the stronger the interaction between the two particles or robots .

Virtual physics-based design models are frequently used in applications that require robot formation due to the following advantages:

- The entire sensory input space is translated easily to the actuators output space using a mathematical rule.
- Vectorial operations can be used to combine obtained behaviors.
- Theoretical rules from physics and control logic can be used to prove system properties such as stability and reliability.

2.1.2 Automatic Design Methods

Automatic design methods formally consists of two main sections; a) Reinforcement Learning and b) Evolutionary Robotics.

2.1.2.1 Reinforcement Learning (RL)

Reinforcement learning in the field of swarm robotics was introduced in 1996 [22, 23]. It was first applied to single robot systems and then helped in the design of swarm robotics systems. A

review about reinforcement learning applied in swarm robotics is available [24]. To begin with, RL is usually defined as a learning mechanism for the agent through trial and error. The agent is rewarded upon behaving in the right way and punished otherwise. Reaching the optimal model is the goal of the robots where they get maximum rewards. The issue with RL being applied with Swarm Robotics is that designers tend to tackle the problem on the collective level although the award system works only on the individual level; designers cant reward the whole team together.

So, translating collective level behaviors into individual level learning or rewarding is the main problem associated with RL when applied to robotic swarms and is called spatial credit assignment; more on this area can be found in [25–27] where authors used communication between robots to share the reward between all robots. Other problems with RL being used to design swarm systems are the huge size of the state space, incomplete environment awareness and non-stationary environment. Neural networks [28] and fast-learning algorithms [29] were used to reduce the huge size of the state space. Research proved that incomplete awareness of the surrounding environment will only make the problem harder [30]. Unfortunately, no one addressed the issue of non-stationary environment in swarm robotics using RL design methods.

2.1.2.2 Evolutionary Robotics (ER)

Evolutionary Robotics takes its inspiration from the Darwinian principle of evolution (survival of the fittest). It was first introduced in 2000 by Nolfi and Floreano as *"an automatic design method that applies evolutionary computation techniques [31, 32] to single or multi-robot systems"* [33]. ER has been used in the study of swarm robotics systems in two scenarios; to test the effectiveness of design methods [34–36] and to provide scientific proofs [37–40].

ER algorithms in Swarm Robotics are executed in five main steps:

1. Generation of random population.
2. Experimentation to generate individual behaviors that are used across all the robots within the system.
3. Evaluation of the collective emergent behavior based on the individual behaviors using the fitness function.
4. Selection of the fittest; i.e. the best scoring individual behavior.

5. Execution of genetic operations such as cross-over and mutation on the selected individuals.

This algorithm is repeated several times until convergence; i.e. the performance is constant through different trials. Although there are applications where systems are heterogeneous and ER mechanisms are used, most of the available works with ER applied in the area of Swarm Robotics are addressing homogeneous systems where all the robots are the same to be able to apply the same fitness function throughout the whole system. Usually, ER algorithms in Swarm Robotics are classified into two categories; the first is based on the composition of the system "*homogeneous vs. heterogeneous*" while the second is based on fitness computation method "*individual level vs. swarm level*" [41]. Different ways to represent individual behavior in ER were proposed such as virtual force functions, finite state machines and neural networks [42]. Several neural network types can be found in the literature such as feed-forward NN [43] and recurrent NN [44, 45]. In applications where individuals do not need memory, feed-forward NN is used while in other applications, recurrent NN is used [37]. It is important to note that evolution is a process that does not guarantee convergence to a solution. Most results acquired by evolution are pretty simple and can be designed by hand.

Calculation of adaptive coordination behavior within a swarm of robots was designed to calculate its cost over time [46]. Authors proposed a method to compare different coordination costs and expect future estimated results. Particle Swarm Optimization (PSO) Algorithms were used and compared against Evolutionary Robotics Algorithms for automatic generation of collective global behavior from local behavior [47]. Authors used a swarm of robots to avoid obstacles and they concluded that PSO was able to achieve better results when compared to ER.

Virtual physics-based design methods were combined with ER to learn the parameters of the artificial potential functions for robots trying to avoid obstacles [42]. Furthermore, a solution was proposed for the stick pulling task using a tool of on-line learning trying to achieve diversity and specialization in a swarm of robots [48]. Others conducted research on a branch from the virtual physics-based design methods called *Learning Momentum* where the behavior of robots is learned according to the current situation in the environment [49].

2.2 Analysis Methods

In the previous section, we analyzed different design methods available in the literature for the design of a swarm system. In this section we focus more on analysis of these methods in Swarm Robotics systems. We believe that this is one of the most important phases in the design process. In this phase, the system designer should be able to validate specific properties of the system that hold true when applied to real systems involving real world scenarios. To study their behaviors, swarm systems are categorized into two different levels; the individual level, microscopic, where the behavior of a single robot is studied. Another level is the collective level, macroscopic, where the collective emergent behavior is studied. As mentioned before, due to the fact that swarm robotics research is still in its infancy, there is no clear formula that produces a certain behavior on one level based on a designed behavior on the other level. Thus, it is hardly found in swarm systems that an engineer approaches the system from both sides at the same time; either the individual level behavior or the collective level behavior. Swarm systems are based on self-organization [50] which makes it very difficult to model both levels at the same time. Though, we provide a solution based on a combination of both levels at the same time as shown in chapter 4. This section is divided as follows: 1) Behavioral Analysis, 2) Real Physical Robot Analysis.

2.2.1 Behavioral Analysis

Behavioral analysis is divided into a) microscopic models in which the individual behavior of a single robot is studied and b) macroscopic models in which the collective behavior of all robots is studied.

2.2.1.1 Microscopic Models

Studying the individual behavior of a single robot in a swarm of robots is called the microscopic model. There are two types of interactions in that model; robot-to-robot and robot-to-environment. Different levels of abstraction have been proposed [51] and the simplest of which is the one which models the environment in 2D the robots as points of mass in it. Mapping the robots with the environment to a 3D model is a more complex approach although it adds flexibility to accurately map properties of sensors and actuators. Individual behavior models are mainly used to help in initial phases of system design. The most frequently used tools in

swarm robotics systems are the simulators. Most of the work available on swarm robotics, if not all, include results from simulations. Typically, Swarm Robotics systems are based on a big number of robots performing a mission where a small number of them can not perform, thus, the need for simulators. They are mainly used to validate the behavior of the design on the collective level before moving to real robots. In our design, we use V-REP simulator described in appendix A to validate our approach. While researching different design models of swarm robotics; its not always affordable to test on such a huge number of real robots. The biggest number of robots known in a swarm application was done at Harvard University where they used 1024 very small and basic robots to do pattern formation [52]. There are many simulators available in the market for the development of swarm systems [53] such as: Virtual Robot Experimentation Platform (V-REP) and Webots. In our research, we conduct all experimentations on V-REP as it provides more flexibility in design. Its also open source; more on our simulation can be found in section A.1.

Different simulators have different characteristics when it comes to design complexity, robot modeling and environment modeling. Unfortunately, most of the available simulators do not fully support scalability when it comes to number of robots. Several benchmarks for studying scalability have been proposed [54]. Another important work in that field is the development of a simulator that can simulate up to 100,000 robots in real time was also studied [55].

2.2.1.2 Macroscopic Models

Macroscopic models study the swarm system on the collective level; i.e. the emergent behavior of all robots together. Works focusing on the macroscopic level in swarm robotics can be classified into two main categories: a) Rate and Differential Equations and b) Classical Control and Stability Theory.

Rate and Differential Equations Rate equations model is defined as "the time taken by some of the robots to be in a particular state over the total number of robots" [56]. The collective behavior can be described from the individual behavior using Rate equations through two steps:

1. Defining the variables for each individual state.
2. Defining the rate equation for each variable which contains all input and output parameters.

There are many works in the literature that use rate equations method to design swarm robotics systems. For example, an experiment was conducted for a clustering task where robots need to gather objects from the environment collaboratively [56]. Rate equations were also used to model the behavior of stick pulling where two robots have to collaborate to pull a stick [48, 57]. They were also used to model the individual behavior of foraging using each other's inference [57]. Authors found that the quality of individual behavior is a decreasing function of group size. Moreover, modeling aggregation and chain formation behavior was done using Probabilistic Finite State Machine (PFSM) as discussed in section 2.1. The behavior of foraging where robots are looking for multiple food sources within the same task was studied to test multi-goal applications [58]. Rate equations were also used to model the behavior where robots are asked to keep intact in hazardous environments [59]. These robots must be able to avoid collisions while navigating in the environment. Researchers also modeled the behavior of the collecting energy units using rate equations [60]. Their use was extended to model the behavior of aggregation where a flying robot is responsible for managing the number of robots doing the aggregation on the ground [61]. Another group provided a swarm task assignment capability to Takayama's enclosure model to achieve a highly scalable target enclosure model about the number of target to enclose [62].

Although rate equations have been used in many applications due to the huge advantage that it provides a way to generate collective behavior from single robot behavior, they still have some limitations such as the inability to model current location or time. Both of these are either assumed or estimated because each robot can easily change its location abruptly in the environment. There are some advancements to using rate equations to overcome these limitations. For example, adding location awareness to improve the performance of rate equations [63]. Furthermore, Langevin equation and the Fokker-Plank equation were used along rate equations approach to increase the performance [64]. They took their inspiration from statistical physics and differential equations.

The Langevin equation was originally used to describe the motion of particles in a fluid. This is neither a microscopic nor a macroscopic level, instead it is called "*mesoscopic model*" where particles motion is divided into two parts; deterministic to describe the microscopic aspect and stochastic to describe the macroscopic aspect. Mapping those two parts to robots, the deterministic part is the robot action depending solely on its individual behavior while the stochastic part is the action based on other robots in the environment. Furthermore, Fokker-Plank equation was used to model the evolution of the swarm with time as a probability density

function describing the location of each robot at a given time. Analyzing different applications of swarm robotics such as coordinated motion, aggregation and foraging was also done through applying Fokker-Plank equation [64]. Authors also compared their aggregation approach against another model called Stock & Flow [65]. Although Fokker-Plank equation has the advantage of modeling most of the swarm robotics collective behaviors, it also has some disadvantages such as difficulty to model communication between robots. Also, derivation of the equation is analytically difficult.

In a similar experiment, modeling the behavior of a swarm performing task allocation, authors used partial differential equations to derive the individual behavior of robots [66]. Same was used for area coverage problem [67] and was later on compared with simulation results [68]. Fokker-Plank equation was used to model the behavior of a swarm performing area coverage [69]. They tried to compare different models and note their behavior based on two aspects; "*microscopic vs. macroscopic*" and "*spatial vs. non-spatial*". They proved that accuracy of the spatial model is higher given short periods of time.

Classical Control and Stability Theory Most researchers consider classical control and stability theory models as the best available method to model the behavior of the swarm of robots because they are based on strong mathematical equations. Unfortunately, in swarm robotics, some of the variables within these models are really hard to assume due to the absence of global information. Discrete-time discrete-event dynamical systems were used to model swarms of robots in 1D [70, 71]. Lyapunov stability theory was used to demonstrate that the presence of noise in a swarm environment will not hinder coherent foraging tasks [72, 73]. Additionally, a linear discrete system was used to model the behavior of a swarm [74]. In addition, task allocation was modeled using delay differential equations which are a type of differential equation in which the derivative of the unknown function at a certain time is given in terms of the values of the function at previous times [75].

2.2.2 Real-Robot Analysis

In the swarm robotics field, two platform approaches are used for experimentation, real robots and simulation. While the usage of simulation might look more attractive due to the fact that it supports extendibility, the use of real robots is equally important. In fact, sometimes it is even more important than simulation due to the following facts:

1. Some aspects of reality are just impossible to be simulated.
2. Testing the robustness of the system with enough noise in the environment.
3. Differentiating between realistic and unrealistic collective behaviors.

The problem with real robot experiments is that some researchers use it to validate their prototype. However, this is not always accurate due to the fact that experiments are run in research environments and may differ from actual environments where robots are deployed. In most of the available literature on swarm robotics, more than 50% depend solely on the use of simulators because they are safer and faster.

It is important to note that within the available work using real robots, there are two different categories; basic and extensive experiments. Most of the available work on swarm robotics using real robots only present basic experimentation [76]. Their main goal is to show the feasibility of the system applied in real world experiments. The rest of the works provide extensive experimentation where multiple runs are done and the average is studied to truly validate properties of the system [77, 78].

2.3 Swarm Robotics Applications

To this point, we only listed available methods and approaches to design and analyze swarm robotic systems. In this section, we discuss some of the available swarm robotic applications and provide a brief explanation of each. It is divided into three application categories:

1. Environment Exploration and Navigation.
2. Spatial Organization of Robots in Space.
3. Collective Decision Making.

After each section, we discuss its relation to our work.

2.3.1 Exploration & Navigation Applications

In swarm applications, robots must explore and discover the environment before starting the mission. In this section, we focus on applications involving unknown environments where robots have no prior knowledge of the surroundings. In critical search and rescue mission applications

such as ours, environments are not usually known to robots beforehand. Environment exploration must be done before the navigation. In fact, it needs to be fast, accurate and efficient to minimize losses and maximize system performance. To understand these kinds of applications, we study two joint behaviors: area coverage and swarm-guided navigation. Area coverage studies the efficiency of deployment and how scattered robots are in the environment. Swarm-guided navigation studies how robots navigate towards their goal and the efficiency of their navigation. Both behaviors are inspired by ants where they navigate in the environment while sensing pheromone trails placed by others ants in the environment. Similarly, bees guide other bees in the environment with dancing where each dance means a different piece of information [79]. In this research, we propose a new model to help better explore the environment before navigation as well as a model to increase the efficiency of the navigation. Our model is based on Probabilistic Finite State Machine (PFSM), Robotic Darwinian Particle Swarm Optimization (RDPSO) and Depth First Search (DFS) and is further discussed in chapter 4. Three categories of Swarm Exploration and Navigation are discussed:

1. Collective Exploration.
2. Coordinated Motion.
3. Collaborative Transport.

2.3.1.1 Collective Exploration

Most of the available applications in the area of exploration and navigation use virtual-physics based design methods mentioned in section 2.1 to maximize environment coverage while maintaining connection with other robots. Other works focus on communication between robots and use probabilistic finite state machines (PFSM) models mentioned in the same section. The main issue with navigation applications is how to model pheromone trails used by ants to navigate on real physical robots. Most of the researchers use a portion of the population as "*virtual pheromones*" to guide other nearby robots in the environment [76]. Connection between robots in the environment has to be maintained in order for the swarm to succeed. Robots should be connected together, to the source and to the destination. Once this network is established, other robots in the population use it for navigation purposes. Following the model of virtual physics-based design, a behavior where robots are attracted to the goal and repelled by obstacles and other robots in the environment was developed to map the visible space of the environment as shown in figure 2.4. This helps forming a reliable network between all nodes while maintaining

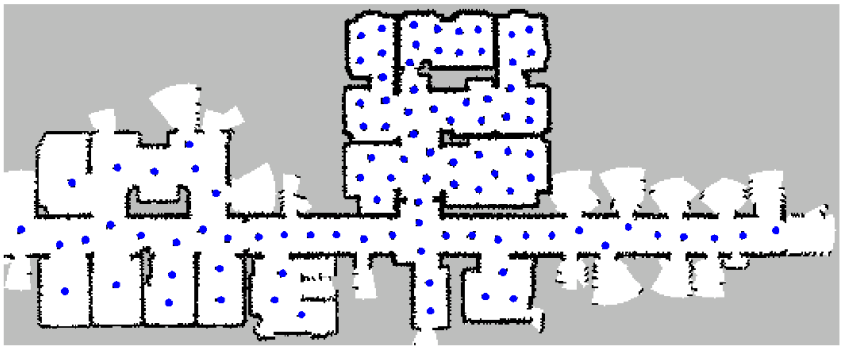


Figure 2.4: Occupancy grid; visible space is marked in black (occupied) or white (free); unseen space is marked in gray [80].

maximum area coverage [80]. Researchers conducted a survey on movement strategies to maintain area coverage using wireless sensor networks [81]. Another group of researchers developed two localization algorithms based on Particle Swarm Optimization (PSO) and Backtracking Search Algorithm (BSA) [82]. They used existing reference nodes in the swarm to eliminate the use of a external reference such as Global Positioning System (GPS).

Moreover, foraging behavior was studied using a distributed Bellman-Ford algorithm along with sensors deployed in a changing environment to search a specific route to the goal for robots to follow [83]. Chain formation was used for robot navigation where they broadcast the direction of movement for other robots to follow [16]. Different from static robots used as virtual pheromones, researchers developed a method where robots are called "passive robots" which means they can be used as a point of reference for navigation purposes although they are actually busy doing other behaviors in the environment [84]. In our model, robots are used for navigation while moving in the environment forming a dynamic moving connected Mobile Ad-hoc Network (MANET). In a recent study on swarm robotics systems, researchers developed a method that mimics the behavior of ants from many perspectives where they claim that their system can be easily scaled to a complex real-world environment [85] due to the following solutions presented in their system:

1. Increased communication when sensed information is reliable and resources to be collected are highly clustered.
2. Less communication and more individual memory when cluster sizes are variable.
3. Greater dispersal with increasing swarm size.

A swarm of flying robots called Swarmanoid were used and deployed where they navigate the environment and attach to ceiling [86]. They determine their location based on other robots

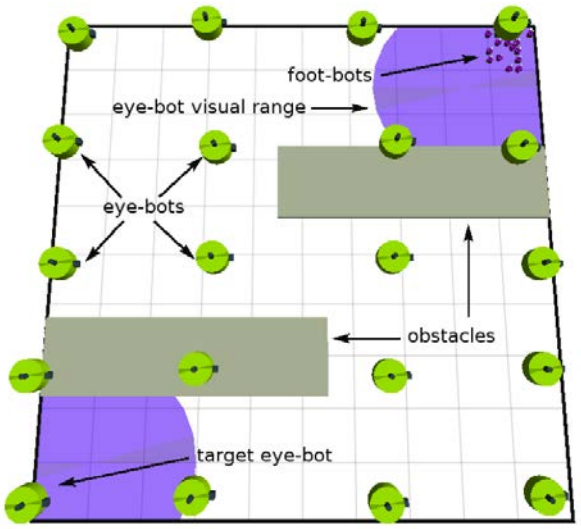


Figure 2.5: Foot-bots are deployed in the start location at the top right of the arena. The target location is at the bottom left. The eye-bots take positions against the ceiling in the area between source and target [88].

in the swarm ensuring maximum area coverage. They are divided into two main categories; robots responsible for communication in the swarm, these are attached to the ceiling and robots responsible for exploring the environment, these are continuously flying. The advantage of this system is the ability to cover large areas of the environment using a small number of exploration robots moving freely within the environment. A continuously connected network is maintained through fixed robots [87]. Static robots attached to the ceiling are responsible for ensuring network reliability and maintainability. Another approach to maintaining the network connection in a moving swarm is to handle communication in the network similar to communication in real world by means of packet routing. Robots navigate the environment using a table containing the distance between other robots and the target location. Furthermore, in an interesting work, authors studied an indoor navigation problem where a heterogeneous set of robots as shown in figure 2.5 in the top-right are required to keep navigating between the source and the target [88]. Guidance in the navigation is done through information retrieved from flying robots marked in green in the same figure responsible for covering the environment and broadcasting this information to the network. This system was developed using probabilistic finite state machine model discussed in section 2.1.

Another study proposed a decentralized control algorithm for swarm robots for target search and trapping inspired by bacteria chemotaxis [89]. Robots initially start performing target search and trapping tasks driven by their bacteria chemotaxis algorithm until they locate their target. The study proved that their results are less vulnerable to local optimum in which most

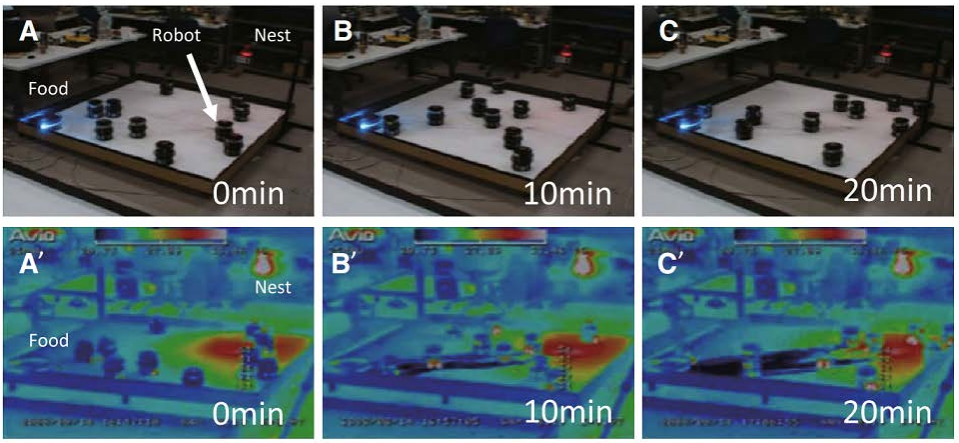


Figure 2.6: Robots performing foraging using real pheromone of Alcohol

other commonly used approaches fail to deliver. Another study of cooperative navigation based on general event-servicing was conducted [90]. Authors focused on how robots should inform each other about the current event and proposed a solution based on delay-tolerant wireless communications. Another study provided a task abstraction module for swarm robotics in navigation called TAM [91]. Their approach is based on a physical device called TAM which abstracts tasks for a single robot to be performed by an e-puck. These single behaviors can be mapped to more complex tasks using a group of TAMs. In a recent study, a group of researchers used pheromone trails from an actual chemical substance such as alcohol as shown in the heat map in figure 2.6 to guide robots doing group foraging behavior [92]. Their results showed that communication through pheromone trails can increase the system performance in a non-linear way depending on the size of the robot swarm.

2.3.1.2 Coordinated Motion

Coordinated motion applications are studied in the literature where robots move together in the same direction forming a swarm. This behavior is called flocking and is inspired by the movement of flocks of birds and schools of fish [93]. It is essential when robots are required to navigate in the environment keeping minimal distance between each other while avoiding obstacles [94]. There are many advantages to coordinated motion [95] such as:

1. Higher survival rate.
2. Precise navigation.
3. Less energy consumption.

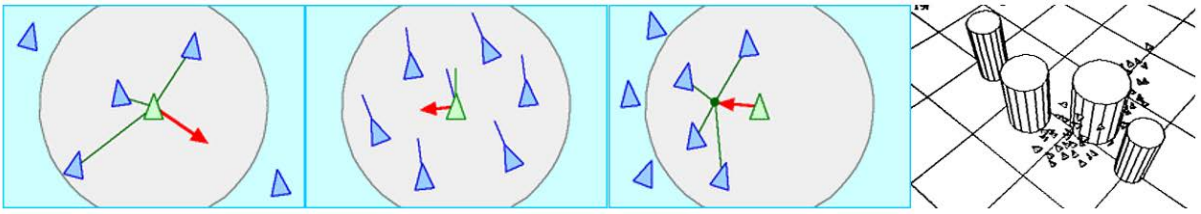


Figure 2.7: a) steer to avoid local mates, b) steer towards the average of local mates, c) steer to move toward the average position of local mates, d) simulated flock avoiding cylindrical obstacles [96]

Most of the works studying coordinated motion depends on virtual physics-based design approaches where robots are required to keep minimal distance between each other using uniform alignment. Other researchers use means of artificial evolution to design coordinated motion within swarm systems. A swarm of virtual birds were used in computer graphics where robots are supposed to measure the velocity and distance of their neighbors and perform different behaviors as shown in figure 2.7. They follow some basic rules including velocity matching and collision avoidance [96]. Both techniques ensure that the swarm is moving together while minimizing collisions. Another research based on social potentials was conducted where each robot is aware of other robots in its range and using information stored on each robot such as distance and orientation to other robots, the swarm is able to calculate the best route to the destination. Authors also created a swarm which is able to do pattern formation while navigating in the environment such as lines, circles and squares [97].

Artificial Evolution techniques were used to tune the parameters of neural networks to coordinate the motion of robots [98]. Authors were able to generate this behavior through three different models; a) robots rotating around the center of the swarm, b) robots moving in a constant speed and c) robots following one moving robot. In another research, robots were able to measure the direction of movement of other robots through a sensor called *heading sensor* while an *infrared sensor* was used to measure the distance. Given information retrieved from the two sensors, robots were able to develop a coordinated motion behavior with no mutual goal between all robots. Another work divided robots into two main categories: a) informed robots and b) non-informed robots. Informed robots are aware of the direction of the target and are using it to inform other non-informed robots in the environment [77, 99]. Tuning the ratio of informed to non-informed robots in the environment results in varying performance of the whole swarm system. Another research developed a similar approach but with only a portion of robots reporting their current direction to other robots [100]. In another work, authors developed a model of coordinated motion where robots are not required to be aware of the orientation of

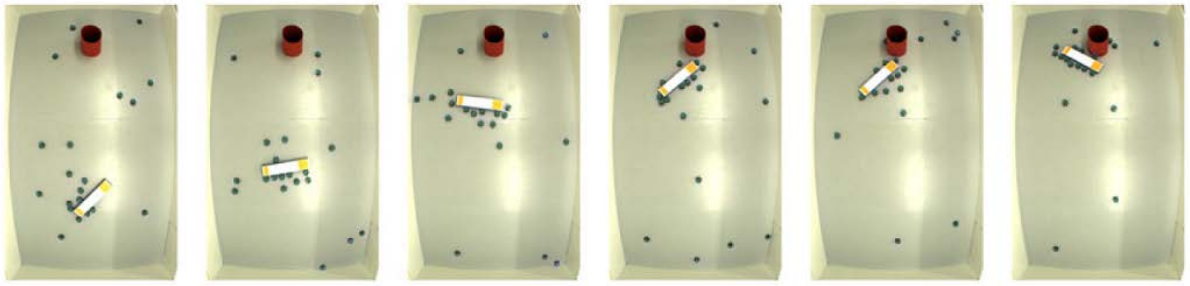


Figure 2.8: 20 e-pucks transporting a rectangular object to the goal

other robots in the environment. They depend on virtual physics-based design where robots attract and repel each other in the environment. Authors used established repulsion or attraction forces to measure the desired movement of robots [101]. In this work, authors showed that a swarm can successfully develop a coordinated motion without the use of informed robots that are aware of the desired direction [102].

2.3.1.3 Collaborative Transport

In this section, we focus on exploration and navigation tasks where robots are required to transfer an object between two points. The object is relatively heavy and cannot be transferred using only a small set of the robots. To do this, first, they have to inform each other about the object to be transferred and to where it should be transferred, then, they all agree on a common direction of movement so that they can collaboratively move together. This behavior is inspired by ants collaboratively moving food to their nest [103]. Ants look for food and when they find a food source, they evaluate the weight of each prey by pulling and pushing the prey until they agree on a common direction. In case of failure to agree on a common direction, they de-attach and re-attach again in different positions until they successfully start moving towards the nest. Another research developed a mathematical model measuring the magnitude of collaboration between ants [67]. They tested their model by observing ants collaborating to move fabricated elastic structures. In most of the collaborative transport applications, collaboration is done by means of one of two models: a) direct communication where robots inform each other of the desired direction or b) indirect communication where robots measure the force applied on the object by other robots. Another group of researchers proposed a strategy for transporting large objects to a mobile goal where robots have to keep evaluating the location of the goal [104]. They tested their approach using a group of 20 e-pucks [105] as shown in figure 2.8 where they were successfully able to transport the object 43 times out of 45.

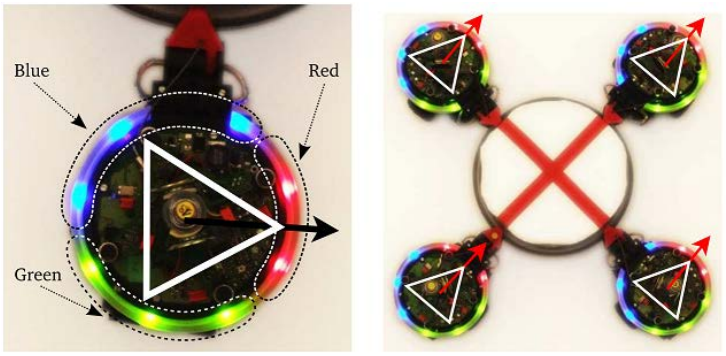


Figure 2.9: a) S-bot displaying a direction using a triangular LED pattern. b) Star-like formation of four s-bots around the prey [106].

In one of the early works on collaborative transport, authors studied the movement of objects using three different kinds of sensing: position sensing, orientation sensing and force sensing. Another research proposed a method where robots agree on a common direction based on colored LEDs as shown in figure 2.9 and start broadcasting it until all of them are moving in the same direction [106]. Similar to their other work, authors developed a method to tune the parameters of a neural network responsible for collective transport [107]. In their experimentations, they tried different weights and sizes of objects to be transferred. They also tried varying the swarm size while adopting three different techniques to move objects; a) robots are directly attached to the object, b) robots are attached to each other and then to the object and c) robots are surrounding the object. Artificial evolution along with neural networks were used to move objects in a space where robots use sensors to measure the applied force on the object and move accordingly while avoiding obstacles [98]. In another work, a different collective behavior was developed using information from each robot [86]. Each robot has its desired direction where the final direction is the average of all desired directions from all the robots.

2.3.2 Spatial Navigation Applications

Organization of robots in a swarm plays a vital role to the swarm success. Efficient organization increase the performance of the system as well as the efficiency of resource usage within the swarm. This section lists four different ways to organize a swarm in space:

1. Aggregation.
2. Pattern Formation.
3. Chain Formation.
4. Self-assembly.

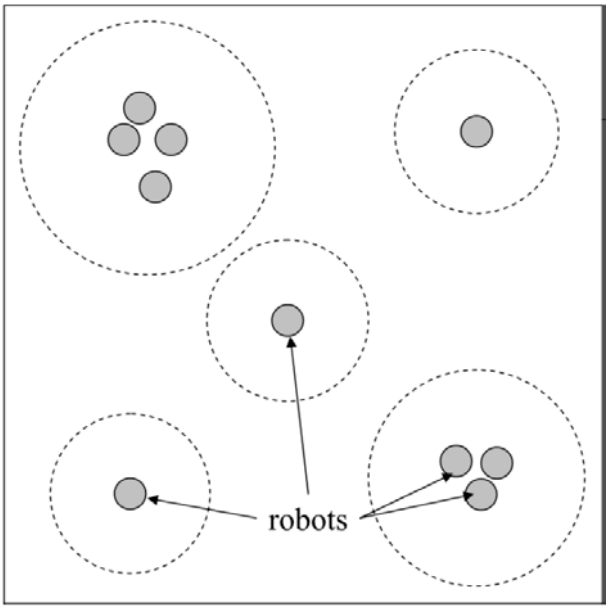


Figure 2.10: A sketch of the environment. The square frame represents the arena. The gray circles represent the robots and dashed circles represent the part of environment where the robot aggregate can be perceived by another robot [113]

2.3.2.1 Aggregation

Aggregation is a simple behavior where robots are required to form aggregates in a certain area of the environment. This behavior is mainly used in applications where robots are to navigate keeping small distances between each other. Many examples of aggregation in natural swarms are found in the literature such as schools of fish and flocks of bees [79, 108–111]. Similar to most navigation applications, aggregation applications use approaches such as probabilistic finite state machine (PFSM) and artificial evolution. In the first approach, robots are not aware of any aggregates in the environment and they start exploration. Once they locate an aggregate, robots calculate the probability of joining the aggregate or not while ensuring that only one aggregate exists in the system at any given time. For the second approach, tuning the parameters of neural networks is used to develop this behavior. Inspired by the behavior of cockroaches, researchers developed a similar system where robots form aggregates in a circular environment similar to those of cockroaches [111, 112]. In another work, authors used probabilistic finite state machines to develop a similar aggregation behavior as shown in figure 2.10. Robots are either waiting for other robots to join the aggregate or moving around looking for the aggregate in the environment [12, 113].

Artificial evolution was used to develop aggregation behavior where authors were able to maintain both static and moving aggregates through tuning of neural network parameters [114].

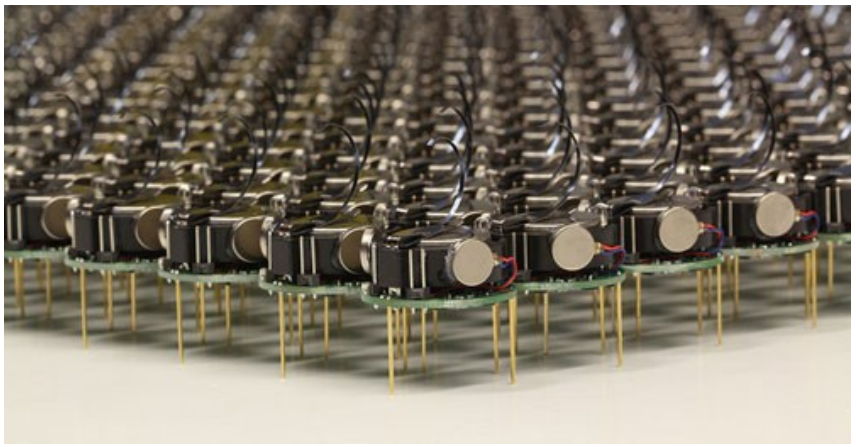


Figure 2.11: Thousand Kilobots

Comparison between probabilistic finite state machine approaches and artificial evolution approaches is also available in the literature [12, 113, 115].

2.3.2.2 Pattern Formation

Pattern formation involves applications where robots are deployed in a certain way while keeping a distance between each other, thus, the formation of a new pattern or shape. Inspiration to such kinds of applications came from biological forms of bacteria where different shapes result in different behaviors [116]. Similar patterns are found in physical molecules and crystals [117, 118]. Most applications doing pattern formation depend on the use of virtual physics-based design where robots use repulsion and attraction forces to form different patterns.

There are two reviews in the area of pattern formation where authors discussed related applications from different perspectives; a) *regular vs. irregular* shapes and b) *centralized vs decentralized models* [119, 120]. One of the most famous applications in the literature in the pattern formation area is the self-organizing thousand robot swarm *Kilo-Bots* shown in figure 2.11 developed at Harvard University [52]. A shape is broad-casted between all robots when they start moving around the biggest formed aggregate until this shape is formed as shown in figure 2.12 where the thousand robots formed an English letter and a star.

Another example of pattern formation was developed using virtual physics-based design where robots use repulsion and attraction forces to measure the distance between each other and form patterns [21, 121]. Another group of researchers developed a method where robots as shown in figure 2.13 are connected via virtual medium where they form a fully connected network to be used to exchange information about the mission such as desired pattern and progress

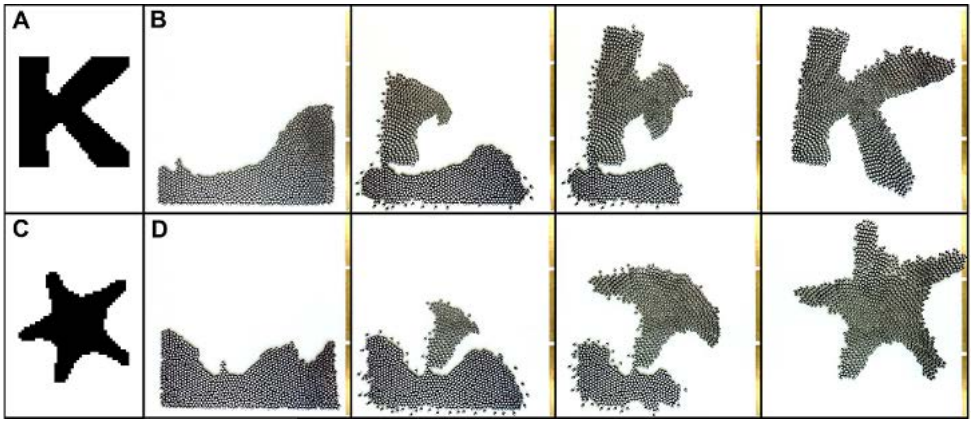


Figure 2.12: a) Kilobots forming the letter K after being broad-casted between them and b) Kilobots forming a star

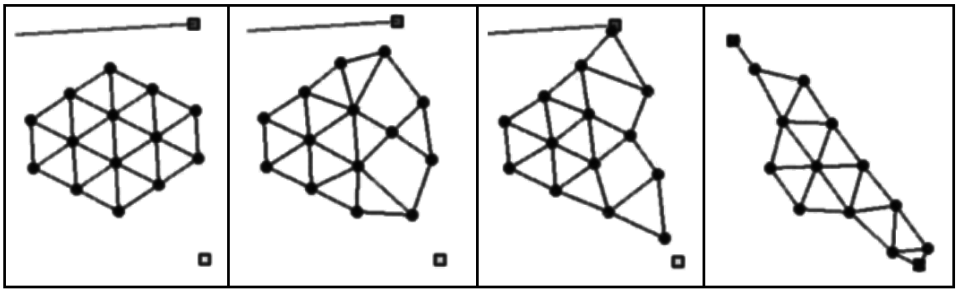


Figure 2.13: Robots are tracking their target while keeping a fully connected network between the source and the destination [21]

of the mission [122, 123]. Another group of researchers proved theoretically that a group of asynchronous robots can't form all possible patterns. Some patterns are only achieved in case of availability of a global knowledge between robots in the swarm such as a single global direction reference [124].

2.3.2.3 Chain Formation

Chain formation is similar to that of pattern formation except that robots are required to form a chain between two points in the environment. A possible scenario could be robots forming a chain between the source and the destination to be used for navigation purposes. It takes inspiration from ants going back and forth between the nest and the food source. A study was done on how ants keep a formed chain between these two points in a foraging application [125]. Similar to previous applications, robots use methods from probabilistic finite state machine models, virtual physics-based design models and artificial evolution models. One of the famous works on chain formation was developed where robots have one of two labels; chain robot or explorer robot and are forming chains between the nest and the prey as shown in figure 2.14

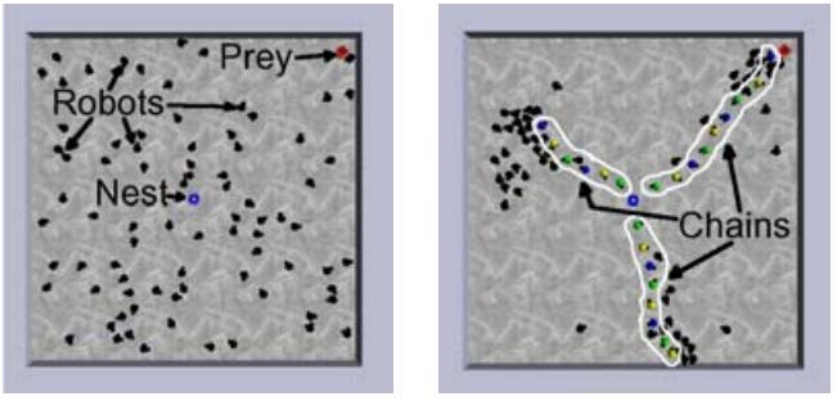


Figure 2.14: a) snapshot from the initial positions and b) typical outcome when employing the chain formation models [16]

[16, 126]. The chain robot is responsible for ensuring the chain is fully connected and a reliable link between the two points exists. The explorer robot uses the chain to navigate between the two points. At the beginning, all robots are explorers and trying to form a chain, new explorers introduced to the system look for existing chains and connect to them. Once connected to a chain, title is changed to chain robot.

Another work using approaches from virtual physics-based design to model the behavior of pattern formation was developed where authors used repulsion and attraction forces to inform robots about their target location in the chain [127]. In this work, results showed that using this model will result in chains that are similar in shape to the environment due to the effect of virtual forces from the environment. Moreover, work involving chains of moving robots was introduced using means of artificial evolution such as cross-over and mutation [36]. In fact, robots use colored LEDs for communication and exchanging of information. In their work, they managed to form a double chain of moving robots as shown in figure 2.15. Another work on chain formation was proposed using probabilistic finite state machines and network routing which obtained another chain of moving robots [87].

2.3.2.4 Self-assembly

Self-assembly applications are found in the literature where robots are required to physically attach to each other. This can be useful in many scenarios where a single robot is not able to successfully navigate in the environment. For example, forming chain of physically connected robots will allow them to navigate in rough terrains. This behavior can be seen in nature where ants physically connect together when there are winds or in the water [129]. Thus, self-assembly

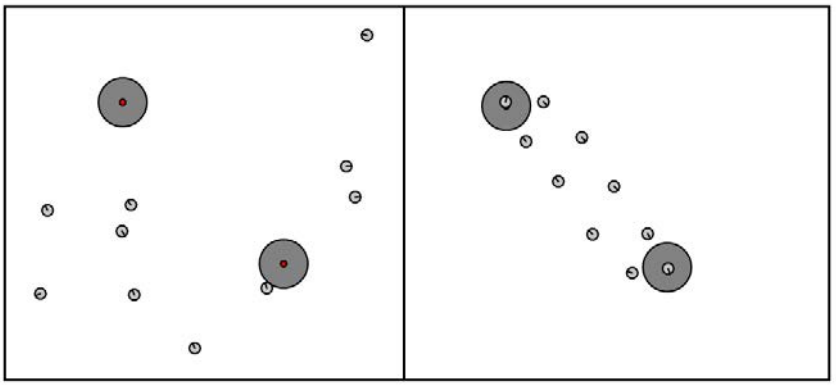


Figure 2.15: a) robots are scattered in the environment and distance between the two points is $1.5m$, b) the final configuration where robots were able to form a double moving chain between the two points [128]

Figure 2.16: Swarm-bots robots [130] attaching to each other and navigating in the environment using methods described in [131]

can increase robot stability as well as the robots pulling force. There are many challenges when it comes to self-assembly in a swarm of robots such as a) how robots are going to attach to each other and b) how to coordinate navigation after attachment to maximize the performance. Most of the works discussing the first issue is developed using probabilistic finite state machine while works discussing the second issue is either using artificial evolution or probabilistic finite state machines. A self-assembly behavior was proposed through the usage of colored LEDs [61]. Robots having the same LED color on will attach to each other forming a predefined shape [130]. This shape is defined through defining the attachment points on the robots [131]. So every robot knows where to attach on the other robot until they form aggregates that can move together as shown in figure 2.16. A scripting language was developed to ease the development of similar behaviors [132].

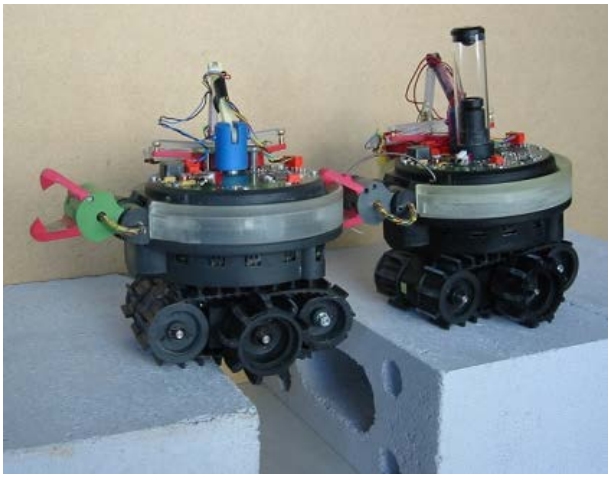


Figure 2.17: S-bots passing a gap in swarm-bot configuration while attached together [133]

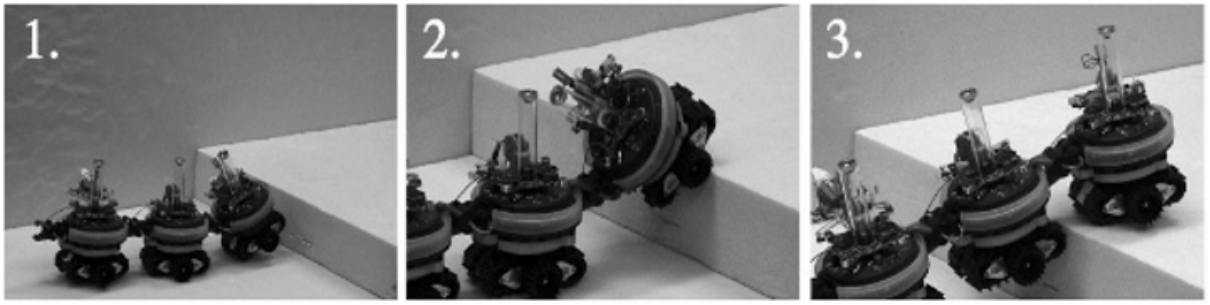


Figure 2.18: Sequence of actions a swarm of three s-bots must execute to pass a step of 10 cm [133]

Some other methods discuss movement coordination issue and how robots should move together while attached. Researchers concluded that this behavior depends heavily on the mission of the swarm. They also stated that its much easier for robots to navigate in rough terrains while connected than to navigate separately [78]. In their work, robots will start exploring the environment first measuring the slopes of the terrain and they only initiate attachment process if environment is steeper than a certain threshold based on robots' capabilities. In our research environment, we developed a proof of concept swarming platform called AntBOTs described in appendix A, which are able to navigate easily so we do not consider rough terrains in our work. Another work proposed a method for robots to pass over a channel that is too large for a single robot to pass as shown in figure 2.17 using mechanical stability [133]. Another usage is when robots are going up the stairs as shown in figure 2.18.

Another work showed that attached robots are able to increase the performance of the system as in the case of pulling a heavy object such as the kid shown in figure 2.19. Eighteen s-bots were successfully able to pull her to the door [134]. This figure shows another example of robots connecting based on colored LEDs and are able to collaboratively move an object that a

Figure 2.19: Eighteen s-bots self-assemble into four swarm-bots to pull a kid on the ground [134]

small set of them can't move. Another model where robots forming 3D structures was studied from the self-assembly and control aspect where robots are able to share their resources once attached together [135]. Another solution based on artificial evolution was proposed where robots assemble together without pre-knowledge of which robot will initiate the attachment process [37]. In another work, authors developed a self-assembly behavior where flying robots are commanding ground robots on how to attach together and is guiding their collective movement afterwards [136]. For a deeper review of available works related to self-assembly behavior, we suggest further reading of the work done by Gross and Dorigo [35]. Another group of researchers in thesis work used Brooks' Subsumption Architecture to achieve self-assembly to enhance robots performance [137, 138]. This architecture shown in figure 2.20 consists of a stack of parallel behavior where the higher levels don't need the lower ones to complete the mission. Lower layers are responsible for the survival of the swarm while the higher ones are responsible for goal achieving behaviors.

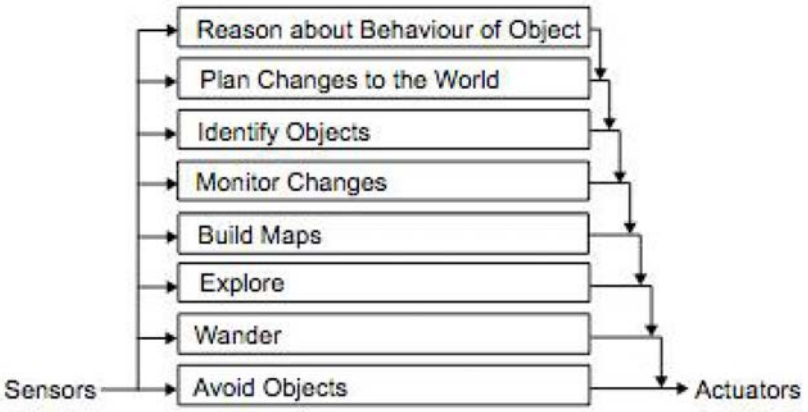


Figure 2.20: Brooks subsumption architecture

2.3.3 Collective Decision-Making Applications

In a swarm robotics system, its important for robots to agree to the same decision. Otherwise, they might waste their resources being busy with different goals. The main advantage of swarm robotics is that all robots focus on the same goal and go for it. In this section, we study the influence of robots on each other and how they make decisions collectively. It is divided into two main parts:

1. *Consensus Achievement*: where robots eventually agree on one decision.
2. *Task Allocation*: where tasks are fairly distributed among the robots based on their capabilities.

2.3.3.1 Consensus Achievement

In swarm robotics applications, it is often the case when robots are required to choose between different options. They have to collaboratively decide on which path they will follow. This decision should be the one maximizing systems performance. Although, achieving consensus between robots is not an easy task, this behavior can be seen in most of the swarm robotics applications. The dynamic environment along with the lack of memory in most of the swarm robotics applications are the reasons why a consensus is not always easy to achieve. Applications involving the consensus achievement are inspired by social insects such as ants for example when they reach a consensus overtime on which path they should be following using pheromone trails of other ants [79]. Achieving a consensus can be found in other social insects such as bees when deciding nest location [108, 139]. Works available on consensus achievement can be divided into two main parts: a) *direct communication* where robots communicate their decisions until

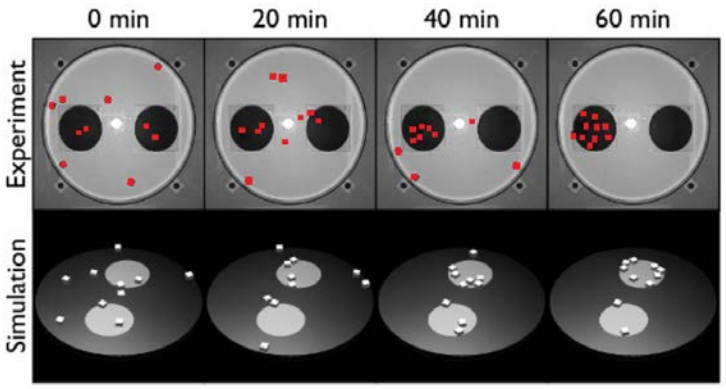
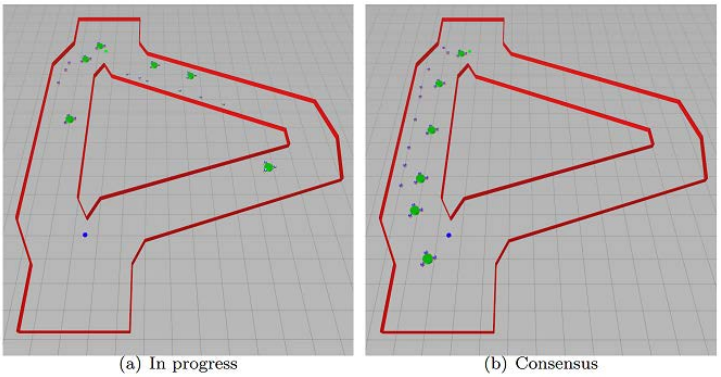


Figure 2.21: Robots are choosing between two shelters both in simulation and real robots [112]

a decision is made and b) *indirect communication* where robots dont broadcast their decisions, instead they depend on other measurements such as number of robots in the environment. One interesting application to this behavior was conducted where robots are chasing two targets and they have to decide on which one to chase first [140]. Two approaches were presented: a) robots are either achieving consensus based on their location distribution in the environment where they choose the target with closer robots or b) they vote and the majority wins. Researchers showed that a consensus can be achieved via indirect communication such as in the case of cockroaches choosing between two different shelters [112, 141]. An experiment was conducted where robots had to choose between two circular spots (shelters) in the environment as shown in figure 2.21.

A similar experiment was conducted where robots are required to look for the smallest shelter that can hold all robots in the swarm [142]. Another work using evolutionary robotics was proposed and was then compared against results from previous experiments where robots used indirect communication to reach a consensus [143]. Another experiment based on direct communication was proposed where a swarm of robots is looking for the nearest nest to the food source [144]. They keep exchanging information on the distance between the food source and the current explored nests until they reach a consensus on which of the nests they will choose. Most of the time, it tends to be the closest one to the food source. Another algorithm was proposed using probabilistic finite state machines and based on how social insects such as bees choose their nest location [139, 145]. Robots keep exploring the environment and once a potential nest location is found, they exchange information as recruiting messages for other robots to come and evaluate their findings until they agree on the best location for the nest. Researchers developed a mathematical model for robots trying to choose between two paths



(a) In progress (b) Consensus

Figure 2.22: a) a swarm of robots in the process of transporting objects from source to destination. At this stage, the swarm has not reached consensus yet and thus robots still use both branches of the environment, b) shows the state of the environment when the swarm of robots has reached consensus. The path selected by the swarm of robots is the shortest one [146]

where they adopted both direct and indirect communication models [146]. Robots usually prefer the shorter path as shown in figure 2.22. At the beginning, each robot has a preferred nest and they keep moving until a certain number of robots in one of the nests is exceeded, this nest is chosen as the preferred one for all robots. Of course, the shorter path will result in having more robots in the corresponding nest faster than the other path. This results in the shorter one being chosen by all robots [147].

2.3.3.2 Task Allocation

In most swarm robotics applications, robots are assigned different tasks. Although the mechanism of task allocation differs from one application to another, all applications aim at maximizing system performance through decent task allocation behavior where each robot is allocated a task matching its capabilities. Behaviors like these are observed between social insects such as in ant colonies where ants are divided into three main roles: a) queen ant responsible for laying eggs, b) soldier ants responsible for guarding the colony and c) worker ants responsible for bringing food and housekeeping of the nest [14, 148]. Most of the available work in the literature discussing task allocation are foraging applications and are mainly using probabilistic finite state machines. Researchers developed an early study on task allocation using a simple threshold based mechanism where they have to maintain a certain energy level [149]. They consume energy while in the environment and they regain this energy from an energy bank located at the nest. They decide whether to stay in the nest or leave according to a stochastic component that is a function of the energy of the nest. If it is above a certain threshold, robots

can leave. Otherwise they have to stay. A similar study was conducted where robots take the same decision but on an individual basis [150]. The decision is a function of the success of the last foraging task. Authors developed a mathematical model for this stochastic component [18].

Another task allocation approach was discussed where robots are sent to a construction site and tasks are allocated equally between all of them. This is done through broadcasting the current load of each robot and maintaining the same load until the mission is done [151]. Another approach was discussed in the context of foraging where robots are to move between three different areas in the environment: a) the nest, b) the food source and c) exchanging area where robots coming from food source hand the food to robots going to the nest [152]. After several iterations, robots were successfully able to divide tasks between them and bring food to the nest. An additional study on task allocation was conducted where robots collaborate to pull a stick from the ground [153]. In this application, each robot holds the stick from one side so it must be done collaboratively and efficiently or else, the stick will fall. One of the most interesting works on task allocation is the research conducted on a swarm of heterogeneous robots retrieving a missing book. In this mission, each robot evaluates its capabilities and its need for other robots. Once each robot is allocated a task, they start collaborating until the book is retrieved [134]. This is part of the swarm-bot research and is one of the perfect examples for task allocation. Other researchers used Elisa-III robot to test the distributed algorithm called *Local Dynamic Task Allocation (LDTA)* for dynamic task assignment [154].

2.4 Literature Review Conclusion

To conclude, this chapter listed most of the available literature on Swarm Robotics. We divided the literature into three categories as shown in figure 2.1:

1. Design of Swarm Robotics Systems.
2. Analysis of Swarm Robotics Systems.
3. Swarm Robotics Applications.

In the first part, we discussed methods to design and analyze Swarm Robotics Systems including ones we are using in this work such as Robotic Darwinian Particle Swarm Optimization and Probabilistic Finite State Machine. In the second part, we listed many examples to applications using Swarm Robotics techniques. We focused on similar applications to ours in the area of Environment Exploration and Navigation.

Chapter 3

Proposed Solution

In this chapter, we discuss our solution for exploration and navigation problems. Prior Swarm Robotics research uses either microscopic or macroscopic techniques as discussed in section 2.2. The microscopic approach implements the system on a single robot, then maps its behavior to other swarm robots until the desired behavior emerges. For macroscopic level design, desired swarm behavior is designed, then it is mapped to each robot. The current issue with Swarm Robotics design is that researchers approach the problem either on the microscopic or the macroscopic level. This limits their ability to verify their system design. A great deal of effort is then expended in trial and error looking for convergence to the desired behavior.

One of the most famous techniques used when designing a swarm system from the microscopic level is the Probabilistic Finite State Machine (PFSM) while Robotic Darwinian Particle Swarm Optimization (RDPSO) is the most used when modeling systems on the macroscopic level. The solution proposed in this work for exploration and navigation problems uses a combination of both microscopic as well as macroscopic techniques and is based on both PFSM which was first introduced by Minsky in 1967 [11] & RDPSO which was first introduced by Couceiro in 2011 [155]. We believe this approach is novel and used for the first time in Swarm Robotics.

PSO showed success in many applications due to implementation simplicity and reduced computational and memory consumption of its design. A key problem with PSO is the possibility that it might get stuck at local optima and robots will never be aware that other solutions might exist. Therefore, we decided to use a modified version of PSO called RDPSO. It has been proven to outperform traditional PSO along with other variants from PSO such as Extended Particle Swarm Optimization (EPSO), Area Extension Particle Swarm Optimization (AEPSO)

and Physically-embedded Particle Swarm Optimization (PPSO) [156]. It's important to note that distribution of robots in the environment highly depends on the deployment of the robots which is in our case random. Detailed explanation of RDPSO can be found in section 4.1.1.

PFSM will be used to implement the microscopic design level while RDPSO will be used for macroscopic design. RDPSO divides the swarm in a set of a smaller swarms where each swarm is looking for a solution and all solutions are then compared together. Different swarms are then allowed to change their best solution based on comparison with each other until all robots converge to the best solution found in the environment.

Additionally, PFSM is used to model our solution on the microscopic level. We use Probabilistic Finite State Machine technique rather than Traditional Finite State machines for reasons that are explained in section 4.1.2. Depth First Search (DFS) is used to ensure all nodes in the swarm form a dynamic semi-connected graph as shown in figure 3.1. This allows graph traversal which aids in robot navigation once our swarm reached an adequate level of environment exploration. This adequate level of environment exploration is usually set to exploring 90% or more of the environment although in some work, this value might be changed. Robots navigate the environment forming a moving Mobile Ad-hoc Network (MANET) to maintain connection between all robots in the swarm.

We decided to build our own swarming platform (AntBOT) as a much lower cost platform. A single AntBOT costs around $250 while a single robot from other famous platforms such as e-puck costs around $1000. The complete physical model for the AntBOT will be prepared as a proof of concept for future experimentation. A limited number of AntBOTs - below 50 - will not produce accurate results and will be limited in testing. Thus, for the purpose of this research, we will build a virtual model for the AntBOT on the V-REP Simulator to be used for our testing purposes.

3.1 Testing Metrics

Single robot behavior is modeled using Probabilistic Finite State Machine (PFSM). We believe integration of Robotic Darwinian Particle Swarm Optimization (RDPSO) to model the global behavior of the swarm should yield faster exploration and enhanced stable navigation. We prove that our method provides higher success rate by around 40% than methods using a single method. We test our approach in two ways:

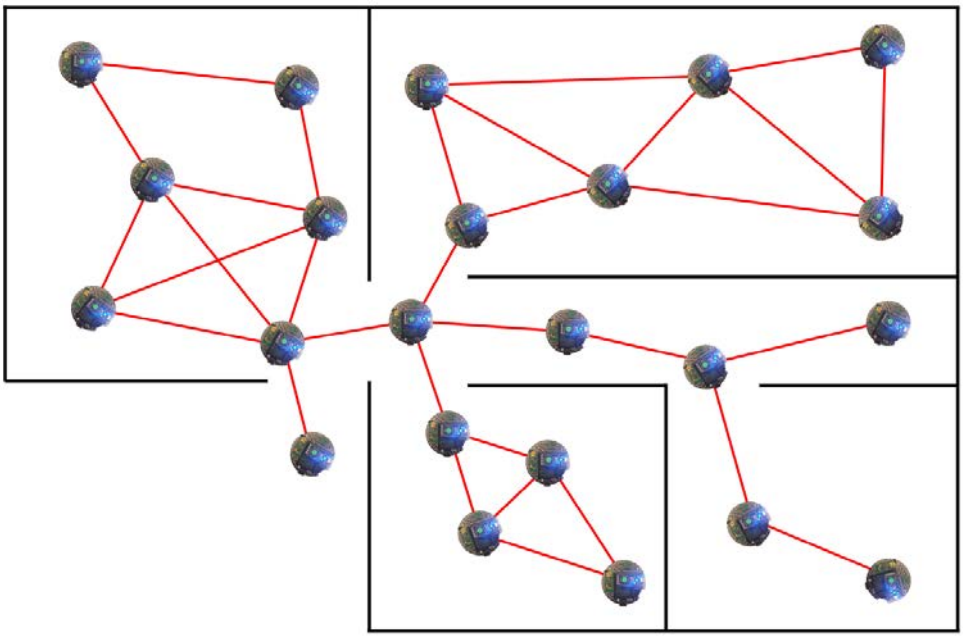


Figure 3.1: AntBOTs forming a connected graph

1. Using PFSM only.
2. Using RDPSO along with PFSM and compare against one.

We will then compare both results to test the efficiency of our proposed approach both in terms of accuracy and performance, specifically speed of discovery of best solution.

From each experiment, we intend to collect the following:

- Experiment Runtime
- Environment Area
- Swarm Size
- Path Size
- Success or failure

The path size is the number of lead robots contributing to the optimal path between the start and the target location. Success or failure is decided based on the propagation of solution in the swarm network. If a big portion of robots are aware of the solution but there is no path between the start and the target location, this is considered failure. With all experiments, we will vary the size of the environment between $25m^2$, $100m^2$ & $400m^2$ area for reasons mentioned below. Further, we found that most of the literature starts with a small number of robots and

increments that number gradually so we decided to increment the number of robots used for each environment area as follows:

- $25m^2$ Area: 10, 20, 30, 40, 50 robots
- $100m^2$ Area: 20, 40, 60, 80, 100 robots
- $400m^2$ Area: 40, 80, 120, 160, 200 robots

The choice of the above three environment areas is to provide meaningful data for realistic scenarios where robots are required to explore different area sizes. We aim to conclude the relationship between the number of robots and the size of the environment. For each environment size, we start with a relatively small number barely sufficient to cover the environment and end with a larger number where the environment is fully covered. Furthermore, we will be using different environment layouts; one with walls and one with obstacle. This will help us verify our approach under different environment designs.

Chapter 4

Experimental Methodology

As seen in the literature review chapter, there is no clear formula that would produce (x) global behavior based on certain (y) local behavior of a single robot and vice versa. The current issue with Swarm Robotics design is that researchers approach the problem either on the microscopic or the macroscopic level. This limits their ability to verify their system design. A great deal of effort is then expended in trial and error looking for convergence to the desired behavior.

We use a combination of both microscopic as well as macroscopic techniques to solve the problem of exploring and navigating unknown environments. It is based on Probabilistic Finite State Machine (PFSM), Robotic Darwinian Particle Swarm Optimization (RDPSO) and Depth First Search (DFS). A robot is in one of several possible states (i.e. Random Movement, Obstacle Avoidance, Reporting Target Location, Path Calculation towards a received target location, etc). While in one of these states, RDPSO guides the formation of the swarm where it is divided into smaller ones where each one is aware of its own location. Location in our experiments is given by the x and the y coordinates of our environment simulation. In real scenarios, a location can be defined by the starting position of the robot along with its starting orientation. Taking these into consideration and saving the movement with the angle of the robot. The current location of the robot can be given relative to the starting location. Further details on the implementation of each algorithm and its usage is discussed later in this chapter.

We use RDPSO to model our solution on the macroscopic level. Additionally, PFSM is used to model our solution on the microscopic level. Depth First Search is used to keep all nodes in the swarm forming a dynamic semi-connected graph. This allows graph traversal using Depth First Search algorithm which aids in robot navigation once our swarm reached an adequate

level of environment exploration. Robots navigate the environment forming a moving Mobile Ad-hoc Network (MANET) to maintain connection between all robots in the swarm. We test our solution by simulation using V-REP simulator later explained is section A.1. A physical robot model of the AntBOT is provided as a proof of concept for future experimentation.

4.1 Techniques Used

In this section, we will discuss each technique and present its implementation details along with its usage. This should give further understanding of the application as a whole consisting of three main techniques:

- Robotic Darwinian Particle Swarm Optimization
- Probabilistic Finite State Machine
- Depth First Search Graph Theory

4.1.1 Robotic Darwinian Particle Swarm Optimization

In this section, we will focus on the Particle Swarm Optimization (PSO) Technique used. PSO showed success in many applications due to implementation simplicity and reduced computational and memory consumption of its design. A key problem with PSO is the possibility that it might get stuck at local optima and robots will never be aware that other solutions might exist. Therefore, RDPSO is used where it divides the swarm in a set of a smaller swarms where each swarm is looking for a solution and all solutions are then compared together [155]. Different swarms are then allowed to change their best solution based on comparison with each other until all robots converge to the best solution found in the environment. It's important to note that distribution of robots in the environment highly depends on the deployment of the robots which is random in our case.

In our case, we use RDPSO to guide the evolution of the smaller swarms. Our robots move around the environment in groups (i.e. smaller swarms) and whenever a member of the group either finds or receives a solution from a nearby swarm, it broadcasts this solution to all other members of its own swarm where they update their solution accordingly. Due to the random movement behavior of robots while in the exploration phase, we expect to see robots moving

alone in the environment looking for a solution. If a robot arrives at a solution on its own, other members of the swarm will not be able to know about this finding unless both their wireless signals overlap. If the numbers of robots in the environment is low for the whole environment coverage, most of the swarm might be aware of the solution and they still fail to accomplish the mission due failure in linking between the start and the target locations. A pseudo code of RDPSO algorithm is shown in Algorithm 1.

Algorithm 1 RDPSO Algorithm

```
1:  procedure EXPLORE_ENVIRONMENT
2:      num_of_swarms ← deploy_robots()        // Deploy all robots in the environment
3:      for i ← 1, num_of_swarms do            // Loop over all swarms i
4:          for j ← 1, num_of_robots do        // Loop over all robots j in swarm i
5:              S ← current_solution()
6:              if S > S_best then
7:                  S_best ← S
8:              end if
9:              build array X for all S_best for swarm i
10:             X_max ← max(X)
11:         end for
12:         build array B for all X_max
13:     end for

14:     for i ← 1, num_of_swarms do
15:         if B_i ≥ threshold then
16:             reward_swarm()                 // call new robot or create new swarm
17:         else
18:             punish_swarm()                 // exclude robot or exclude swarm
19:         end if
20:     end for
21: end procedure
```

As seen in algorithm 1, and using the Darwinian evolutionary theory which states that survival is to the fittest, swarms arriving at better solutions are rewarded with more robots which will allow the increase of knowledge of target locations in the environment among deployed robots. On the other hand, swarms having local optima solutions are punished by excluding one robot from the swarm. Robots that do not belong to a swarm are by nature dangling in the environment looking for solutions.

Once a target location is found and confirmed with the majority of robots, robots are divided into two main categories:

- Lead robot
- Helper robot

Lead Robots are the ones forming the path between the start location and the target location. Robots which don't contribute to the main path are called helper robots. One of the main goals of this work is to guarantee network connectivity even in the case of robotic failure. Therefore, helper robots start aligning themselves with lead robots to assure continuous network connectivity in case of any failure within the lead robots.

To conclude, RDPSO is used in this work to define the macroscopic behavior of our system where distribution of robots follow the Darwinian evolutionary theory to guarantee arriving at the best solution while maintaining network connectivity.

4.1.2 Probabilistic Finite State Machine

In this section, we will discuss in details a Finite State Machine technique which is used to aid the movement of robots in the environment. A finite state machine is a mathematical model of computation where robots are in one of many possible states. Transitions between different states happen frequently based on inputs from either:

- The robot itself
- Other robots
- The environment

Key inputs to robots while in a specific state triggers a transition to another state. Robots can be in one of two state sets:

- Prior to agreement on solution
  - Searching state
  - Obstacle avoiding state
  - Stopping state
  - Broadcasting state
  - Receiving state
  - Path finding state
- Post agreement on solution

  - Reporting state
  - Helping state

In traditional FSM, transitions between different states are binary. That is, based on inputs, robots either stay in the current state or make a transition to the next state. Usually, there are two types of FSM:

- Deterministic FSM
- Non-deterministic FSM

This model is limited if applied directly to our system as robots can go only from a state to a specific state based on a certain input. Considering the situation where a robot is in a state where it received a signal from the environment about an obstacle and at the same time step, it received another signal from a nearby robot containing a broadcast for a target location. Since a robot cannot be in two states at the same time, the robot will choose the next state discarding one of the received signals which can be dangerous especially if it hits the obstacle. So, for the purpose of our system, we will be using an advanced version of FSM called Probabilistic Finite State Machine (PFSM).

PFSM is somewhere between the Deterministic FSM and the non-deterministic FSM. In PFSM, robots can go from one state to another state based on a certain probability. This probability is calculated based on the current situation and importance of the signal received. Figure 4.1 shows a simple PFSM where robots can go from one state to another based on a probability. **S** is the searching state where a robot is randomly dangling in the environment looking for potential targets. **R** is the receiving state where a robot is listening to broadcasts from surrounding robots. **B** is the broadcasting state where a robot either arrived at the target or received a target location from a neighbor robot. **O** is the obstacle avoiding state where the proximity sensor of the robot signals a nearby obstacle. **F** is the finding state where a robot has already received a target and is calculating the path towards the sender robot.

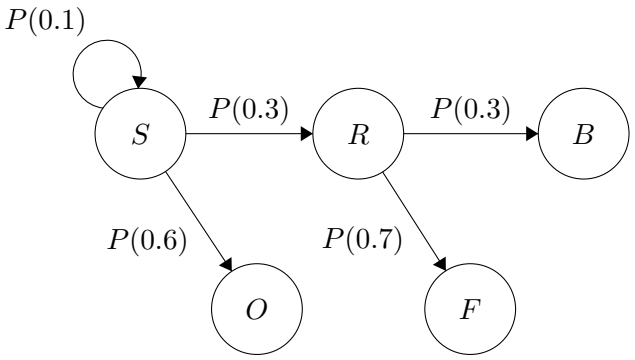


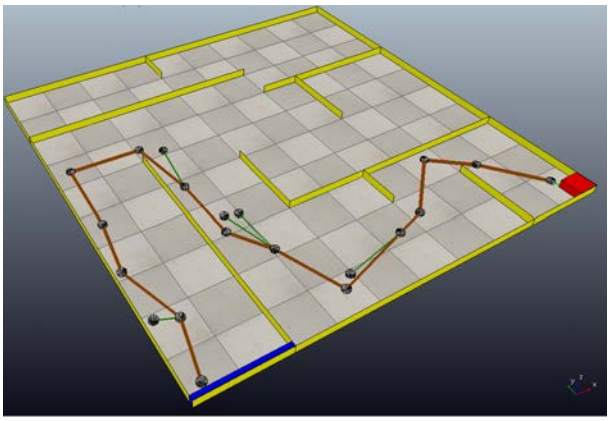
Figure 4.1: A probabilistic finite state machine example

4.1.3 Depth First Search

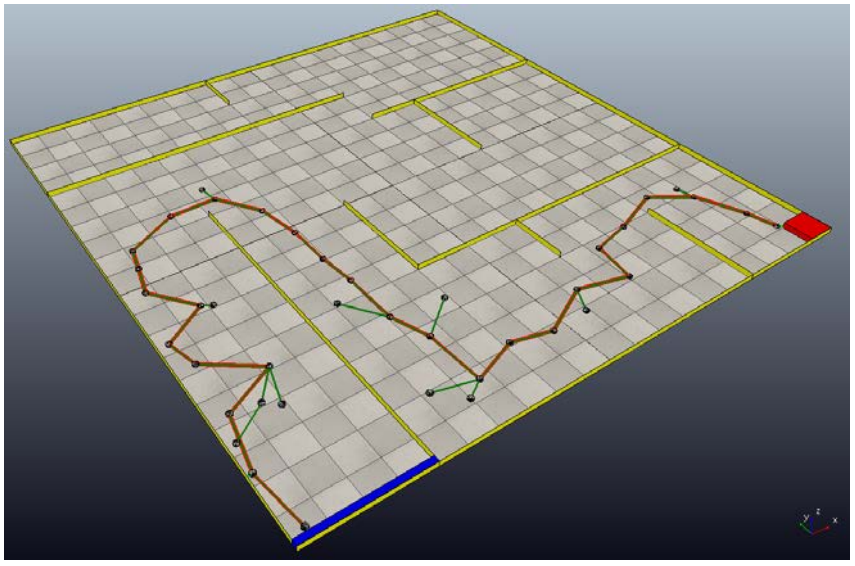
Once the majority of robots are aware of the target location and in case of success scenario, a semi connected graph is created and then traversed for the shortest path between the start and the target locations. The graph is traversed using Depth First Search (DFS) algorithm to retrieve all links between the start point and the target location. The resultant subgraph is then passed to the diffuser robot which will use it for direct navigation to the target. Note that due to the distributed nature of the swarm, the graph is traversed on board of all robots. Then, each robot evaluate its contribution to the optimal path if any. This introduces repetitions in the calculation but it insures redundancy and it eliminates single point of failure scenarios. Figure 4.2 shows the resultant path between the start point and the target location in all environment areas.

For graph traversal, the use of Depth First Search (DFS) vs Breadth First Search (BFS) depends merely on the structure of the search tree. If the solution is not far from the root of the tree, a breadth first search (BFS) might be better. If the tree is very wide, a BFS might need too much memory, so it might be completely impractical. If a solution is located deep in the tree, BFS could be impractical. In our experiments, it's most probable that the solution resides deep in the graph and the structure of the tree is not wide because we placed the solution at the farthest point from the start location. Thus, we decided to use DFS to obtain faster graph traversal. A pseudo code for the DFS we used is shown in algorithm 2.

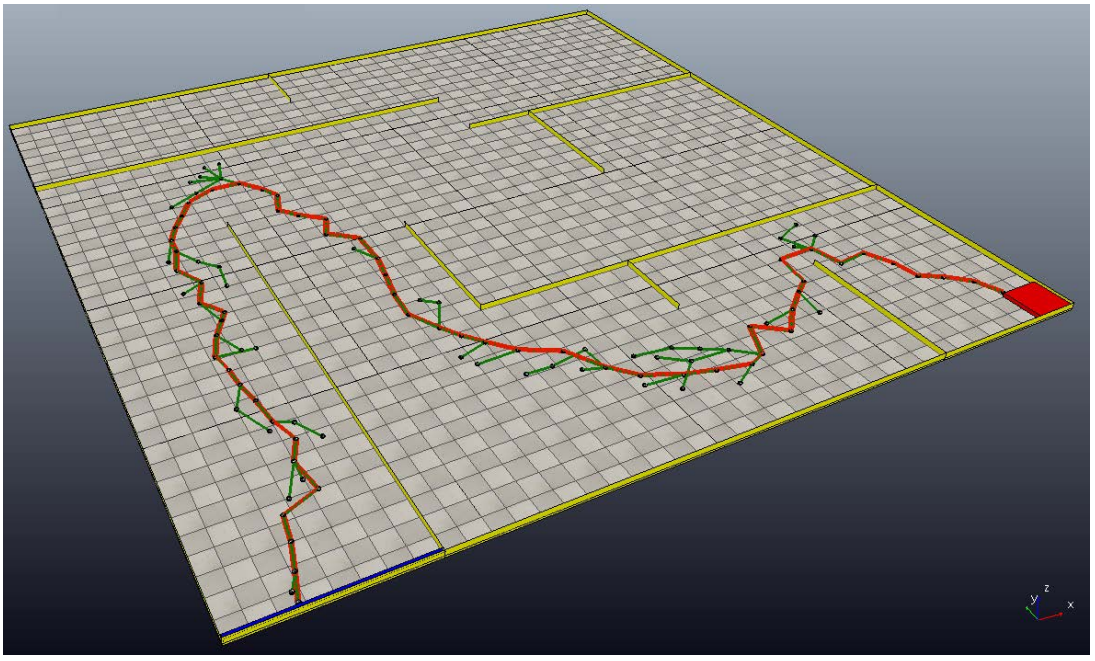
In conclusion, we list a detailed description of each component used in our approach from the Robotic Darwinian Particle Swarm Optimization to the Probabilistic Finite State Machine and the depth first search graph algorithm.



(a) Small Environment Area: $25m^2$



(b) Big Environment Area: $100m^2$



(c) Big Environment Area: $400m^2$

Figure 4.2: A graph connecting start and target location for all environment areas

Algorithm 2 DFS Algorithm

```
1: procedure DFS(G, v)
2:     Stack S ← {}                          // Start with an empty stack
3:     for u ← each_graph_vertex do
4:         visited[u] ← false
5:     end for
6:     push S, v                             // v is the vertex where the search starts
7:     while S is not empty do
8:         u ← pop S
9:         if u is not visited then
10:            visited[u] ← true
11:            for w ← unvisited_neighbour_of u do
12:                push S, w
13:            end for
14:        end if
15:    end while
16: end procedure
```

Our approach is tested using the V-REP simulator where from each experiment we collect the following data:

- Experiment Runtime
- Environment Area
- Swarm Size
- Path Size
- Success or failure

A pseudo code for our swarming platform model AntBOT further discussed in appendix A is shown in Algorithm 3.

4.2 Experimentation Strategy

In this section, we present the rationale behind our experimental choices:

- Environment Area
- Swarm Size

Algorithm 3 AntBOT Main Code

```
1: procedure AntbotMain
2:     while simulation_is_running do
3:         if targetRecieved ≠ true and targetFound ≠ true then
4:             randMove()                    // Random Movement looking for target
5:             wixelReceive()                // Listening to any broadcasts
6:         else
7:             stopRobot()
8:             wixelSend()                   // broadcast received or found target location
9:             if targetRecieved ← true then
10:                findPath()                // Path between sender & reciever robot
11:            end if
12:        end if
13:    end while
14: end procedure
```

4.2.1 Environment Area

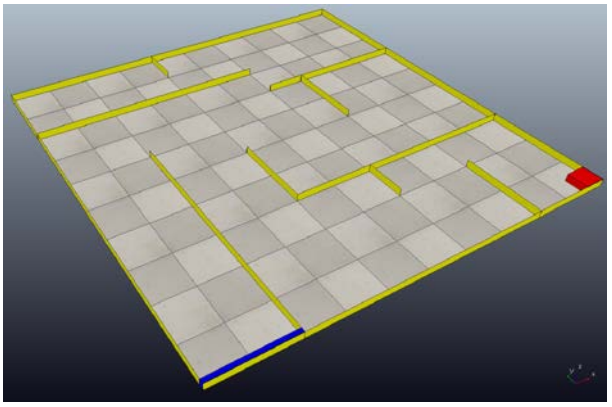
In our experiments, we use three different environment areas: $25m^2$, $100m^2$ & $400m^2$. We chose these environment areas to explore a wide range of environments where our robots can be deployed. We also use two different layouts: Walls and Obstacles layouts. Different environment areas shown in figures 4.3a and 4.3b and 4.3c can be explained as follows:

1. Small Environment Area of $25m^2$: A square environment of 5m x 5m that presents a map for a small apartment with different rooms and corridors.
2. Medium Environment Area of $100m^2$: The same environment layout but of 10m x 10m that presents a map for a bigger apartment.
3. Big Environment Area of $400m^2$: The same layout but of 20m x 20m that presents a map for a very big apartment.

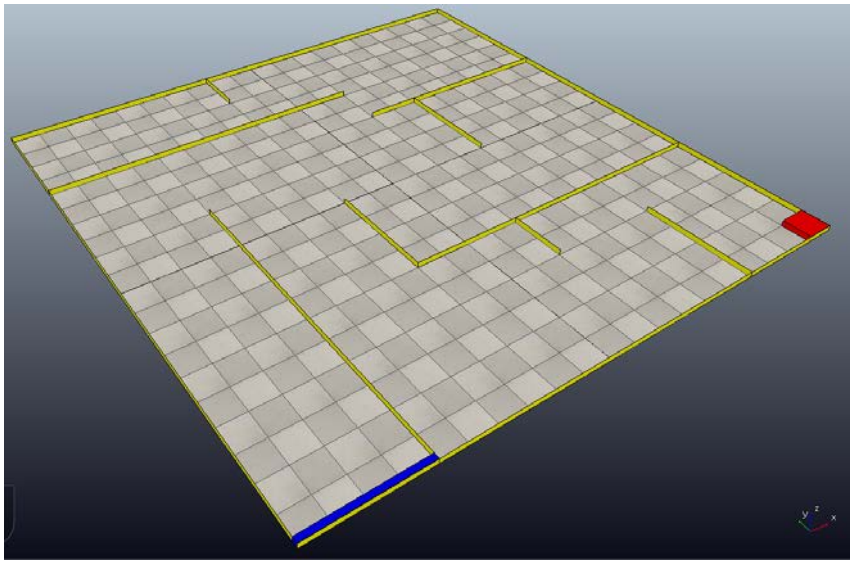
As shown in figures 4.4a and 4.4b, we use two different environment layouts for the environment area of $100m^2$:

1. Walls (Apartment): In this environment, we model an apartment or a university building consisting of different walls and corridors.
2. Obstacles (Factory): In this layout, we model a factory where there are multiple poles and machines. We model these with randomly deployed obstacles of different sizes, shapes and orientations.

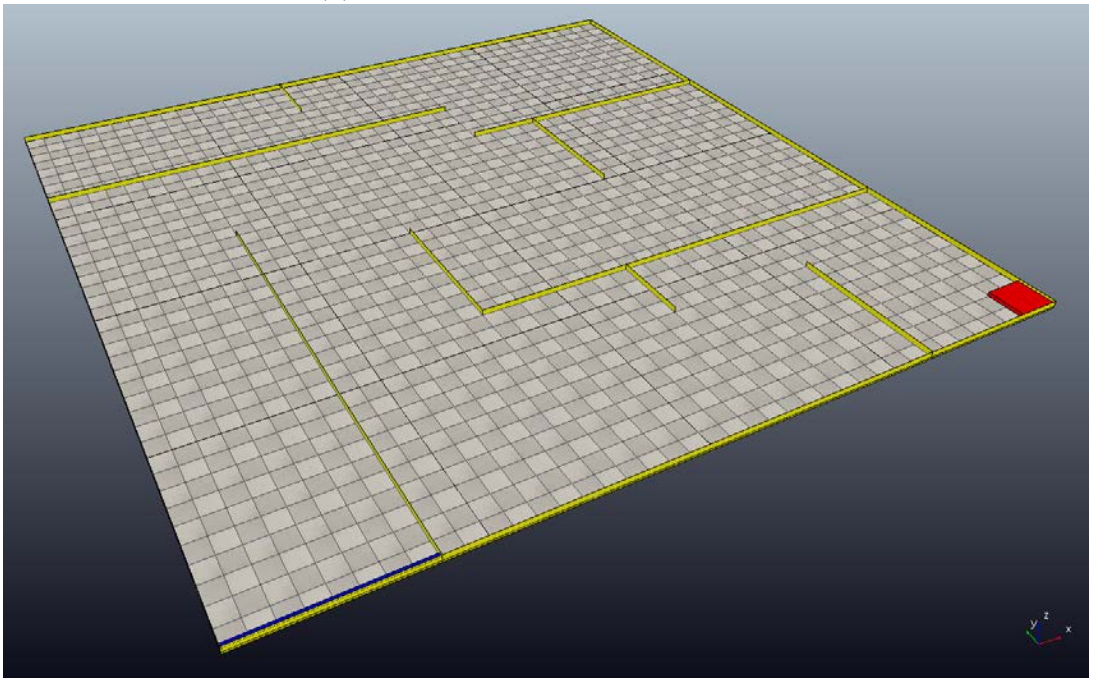
It is worth mentioning that most of the research in the field of swarm robotics use either randomly deployed obstacles or no obstacles at all. Although we understand that the introduction



(a) Small Environment Area: $25m^2$

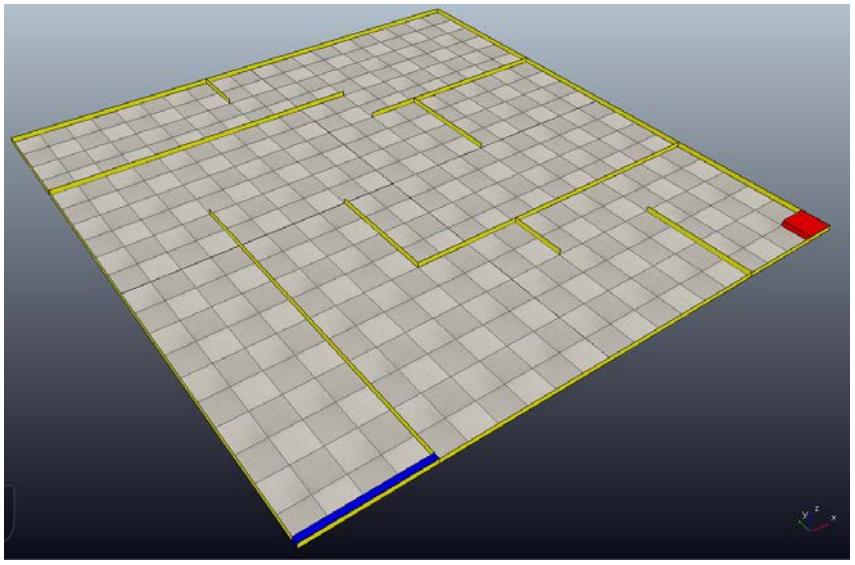


(b) Medium Environment Area: $100m^2$

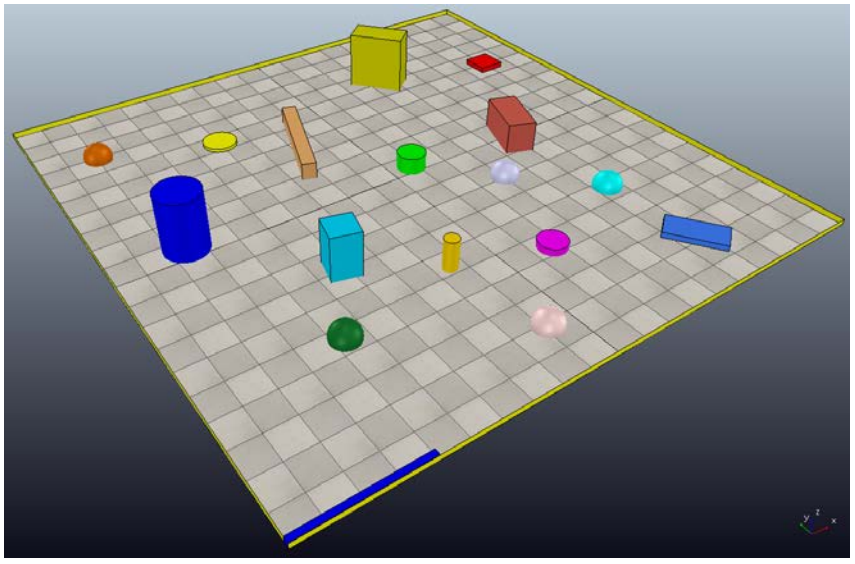


(c) Big Environment Area: $400m^2$

Figure 4.3: Different Environment Areas used for experimentation



(a) Environment Area $100m^2$ with Walls



(b) Environment of Area $100m^2$ with Obstacles

Figure 4.4: Different Environment layouts used for experimentation

of walls in our experiment design will increase the time, we need to make sure of the validity of our proposed system in different environment layouts.

4.2.2 Swarm Size

We vary the swarm size in each experiment between two values:

- The size that is barely enough to explore the whole environment.
- The size that is enough to explore the whole environment.

We ran multiple experiments with increasing swarm size from a single robot. From the collected results and given that each robot can only communicate within a radius of $1m$, we concluded the following number of robots per environment area:

- $25m^2$ Area: 10, 20, 30, 40, 50 robots
- $100m^2$ Area: 20, 40, 60, 80, 100 robots
- $400m^2$ Area: 40, 80, 120, 160, 200 robots

For the small environment, 10 robots are barely enough to cover the optimal path between the start and the target locations in the best case scenario while 50 robots are enough for full environment coverage. For the medium environment, 20 robots are also barely enough for start and target connection while 100 robots are enough for full environment coverage. Same goes for the big environment with robots between 40 and 200.

In our experiments, we use random deployment for robots in the start area. The randomness ensures various start positions and orientations for robots and thus various results. Furthermore, our target is located in the farthest place from the start location to ensure the mission is not finished before the whole environment is explored.

4.3 Limitations

During the course of our work, there were some unfortunate obstructions that led to many decision being made. In this section, we will discuss many of the challenges we faced and how we overcame them. We will discuss these challenges on the development of both the simulator and physical robots.

4.3.1 Simulation (V-REP)

As for the V-REP simulator, a complete model of the AntBOT has been built into the V-REP simulator discussed in section A.1. In this thesis, we provide a novel technique for robot exploration and navigation using Robotic Darwinian Particle Swarm Optimization (RDPSO), Probabilistic Finite State Machine (PFSM) and Depth First Search (DFS). A complete algorithm combining normal navigation behavior with RDPSO algorithm has been developed in

	Static	Non-Static
Non-Respondable	☒	☐
Respondable	☒	☐

Table 4.1: Dynamic simulation main shape types

simulation. Also, the remote API supported with V-REP has been used for environment setup and the random robot deployment. Therefore, building the AntBOT model on the simulator has gone through two stages:

- Robot Design
- Robot Code Development

4.3.1.1 Robot Design

As for the robot design, several versions of the model have been developed to ensure close to 100% similarity between the simulator and the physical AntBOTs in the future. This should guarantee similar results to physical AntBOTs in the future. Building the AntBOT model itself was an extremely challenging task because while building a model from scratch, there were many aspects other than technical which needed attention such as the following:

Robot Dimensions Mapping the dimensions of the robot includes accurate translation of robot parts such as body, motors, sensors, wheels and communication modules.

Robot Weights The weights of each of the components mentioned above is critical to the stability of the movement of the robot. There has to be an accurate center of mass which is exactly located at at the center point of the robot. If for example, a sensor is added at the front of the robot with major weight, this will move the center mass of the robot a bit to the front which will result in instability of the robot's movements.

Robot Material The material of each component has to be correctly mapped. This contributes a lot to the friction of the robot with the environment. The choice of shapes used when building the robots is extremely important. According to V-REP, shapes can be classified into 4 main groups as shown in table 4.1.

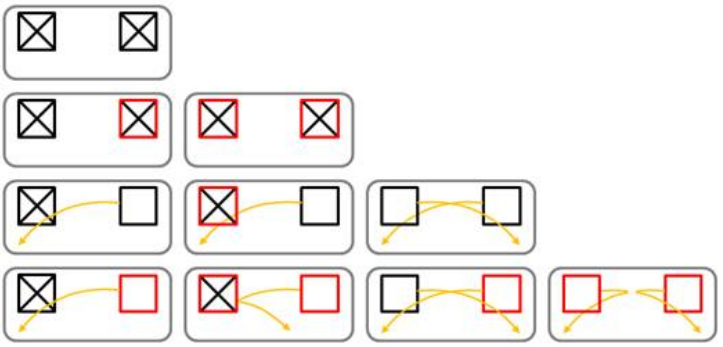


Figure 4.5: Static/non-static, respondable/non-respondable shape behaviors and interactions

During dynamic simulation, static shapes will not be affected (i.e. their position relative to their parent object is fixed), whereas non-static shapes will be directly influenced by gravity or other constraints. Respondable shapes influence each other during dynamic collision (i.e. they produce a mutual collision reaction, they will bounce off each other). Figure 4.5 illustrates the static/non-static, respondable/non-respondable behaviors.

Two respondable shapes will always produce a collision reaction unless their respective collision masks don't overlap. V-REP uses triangular meshes to describe and display shapes. While the creation of the shapes used in our model, we had to be careful of the number of triangles in each component especially those who will be dynamically enabled - those which will be moving while simulation is running - (i.e the wheels of robot rotating most of the time).

Robot Movement After arriving at the final appearance of the robot, we had to worry about its static and kinetic control which is responsible for how the robot should react to forces applied to it such as the rotation of the motor. Moreover, we had to set variant moments of inertia to guide the movement of the robot from the resting state till the full momentum state.

Robot interactions AntBOTs are designed to be part of a swarming platform. Therefore, there should be some kind of interaction with the environment and other robots. AntBOTs interact with obstacles in the environment using its 120° proximity sensor and interact with other AntBOTs using its Wixel Wireless module simulated using signal sender-receiver module within V-REP. Adjustments to the range and the direction of these sensors had to be done to insure accurate readings and accordingly accurate behavior based on external inputs.

Thus, the design of the robot itself on simulation was not an easy task. V-REP offers a variant set of physics engines which are responsible for all interaction between shapes in the environment. Our challenge was to select the one that closely simulate a real world scenario.

4.3.1.2 Robot Code Development

The main language of the V-REP Simulator for internal development is *Lua*. V-REP also supports remote communication via a very well established remote-API supporting many other development languages such as Python, C/C++ and many others. Authors decided to use the *Python* remote API for robot control. V-REP communicates with remote-API over a TCP channel, so as the swarm size increases, and considering the number of signals per a single robot, the communication is dramatically affected. Thus, the movement of the robots along with path planning algorithms were highly lowered in speed. So, we added the main code for the AntBOT movement to a child script of each robot in the simulator written in *Lua*. This child script contains all the logic for a single robot to be able to survive in the environment. Moving the main robot code from the python remote API to a child script within the simulator itself reduced the amount of data transmitted between the simulator and the remote-API and increased the speed of our system.

Remote-API written in *Python* is responsible for the following:

- Environment Setup
- Number of robots
- Robot Deployment
  - Location
  - Algorithm (i.e. random, exponential, etc)
- Simulation start
- Shortest Path retrieval
- Results collection
- Simulation end

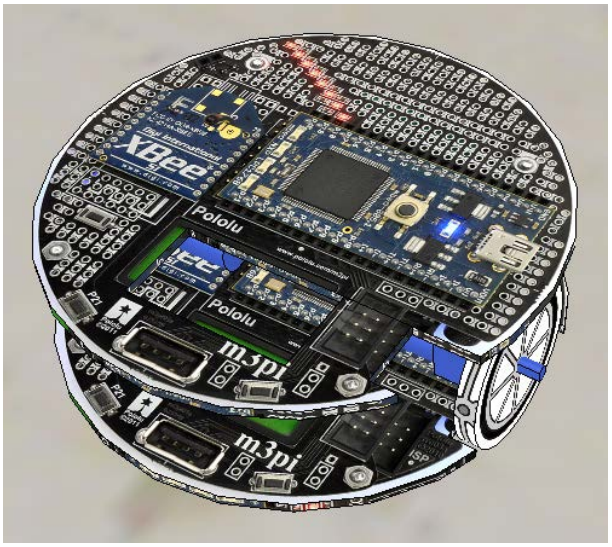


Figure 4.6: AntBOT model in V-REP Simulator

4.3.2 Real Robots

Inspiration for building our swarming platform came from the fact that the cheapest option for a swarming robot is the e-puck [105] which costs around $1000 at the time of writing this document. Authors used the Pololu 3pi robot [157] in other experiments and decided to take it from a line following robot to be part of a complete swarming platform. For this, we had to add a better processor along with a wireless communication module. More on this can be found in section A.2.

The performance of the communication module *Wixel Wireless* still needs some further improvements as it was built to work as a one-to-one communication channel. We used a library called *multiradio* available for the wixel module and developed by Geoff [158] to make communication possible between all robots simultaneously. The library is still at early stages so we had to fix many bugs while merging it with our code. It was stated by the author that some packets might reach its destination successfully but never get acknowledged (ACK'd). This might affect our communication channel credibility.

Thus, some work to ensure message delivery has to be done as the current version of the library does not provide a 100% reliable communication channel. We also worked on improving the parts responsible for message acknowledgments especially the module *radioMultiTxDataBlocking*. This module seems to have problems with message IDs and this might be the root cause why some packets are not acknowledged.

Chapter 5

Experimental Results

In this chapter, all of the experimentation results acquired from the V-REP simulator are listed. From each experiment, we collected the following:

- Experiment runtime
- Environment Area
- Swarm Size
- Path Size
- Success or failure

Discussion of each metric in details can be found in section 3.1. In our experiments, we prove that combining the RDPSO and PFSM algorithms results in a much faster exploration and a much more stable navigation in unknown environments. We also guarantee solid network connectivity if the swarm size is suitable for the size of the environment being explored.

All experiments were run twice with initial random deployment:

1. PFSM only is used.
2. RDPSO is used along with PFSM.

After running multiple simulation experiments with big swarm sizes, we concluded that the limited number of real physical robots would not give us much of a meaningful data as we were expecting. Thus, we decided to present the real model of the AntBOT as a proof of concept for

future experimentation. In the future, more real robots should be added to be able to collect more meaningful data from varying the swarm size in each experiment.

5.1 Metrics Comparison

In this section, we study the effect of varying the metrics mentioned above along with a discussion on each one. The comparison is presented as follows:

1. Experiment Runtime **vs** Swarm Size
2. Experiment Runtime **vs** Environment Area
3. Experiment Runtime **vs** Environment Layout
4. Swarm Size **vs** Reporting Time
5. Swarm Size **vs** Path Size
6. Swarm Size **vs** Success Rate

In each section, we discuss the above comparisons using two different techniques:

- Using PFSM only
- Using RDPSO along with PFSM

5.1.1 Experiment Runtime vs Swarm Size

This section demonstrates the relationship between the swarm size used in an environment and the amount of time needed to fully explore and navigate this environment and arrive at the target location. As shown in figure 5.1, the time required to explore and navigate an environment decreases as we deploy more robots in the environment. The same happened for the three environment areas as shown in figures 5.1a and 5.1b and 5.1c with the only difference in range of time needed to explore and navigate the environment for each different area. Figure 5.2 shows that the gap between the minimum and the maximum exploration time has decreased with the increase in the swarm size in all environment areas.

Table 5.1 shows the different minimum and maximum times required for each swarm size for the three environment areas. We conclude that as the swarm size increases, the amount of time required to explore and navigate the environment decreases. We also noted that as the

environment area increases, the swarm size required to explore has to increase to cover more area. For the small environment, we ran 500 experiments in total while we ran 200 for the medium environment and 50 for the big environment.

We also notice that the gap between the time needed to explore the small environment with 40 & 50 robots is not negligible while the success rate further explained in section 5.1.6 is not highly affected. This concludes that 40 robots should be enough for the small environment size. For the medium environment area with 80 & 100 robots and the big one with 160 & 200 robots, they all produced similar results for exploration which leads to the same conclusion.

As for the big environment of $400m^2$ area, for 160 & 200 robots, the time difference is a bit more significant due to the large area being explored. Though, the success rate significantly increased in this environment area when using RDPSO along with PFSM. We can also conclude that using both RDPSO & PFSM has helped the robots navigate the environment in a more organized manner and at a lesser time. We notice that the performance increase starts to be significant as we increase the swarm size required to explore the environment.

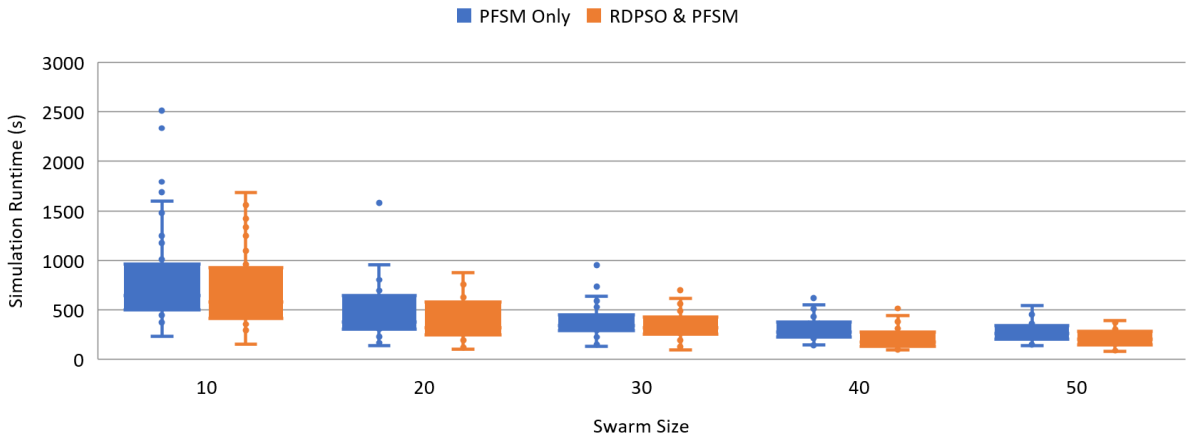
5.1.2 Experiment Runtime vs Environment Area

In this section, we present the relationship between the amount of time taken to fully explore an environment and the area of this environment. We present three different areas: $25m^2$, $100m2$ & $100m^2$ with the same design as shown in figures 4.3a, 4.3a and 4.3c. We use the same design so the only variable we have is the environment size and not the obstacles inside.

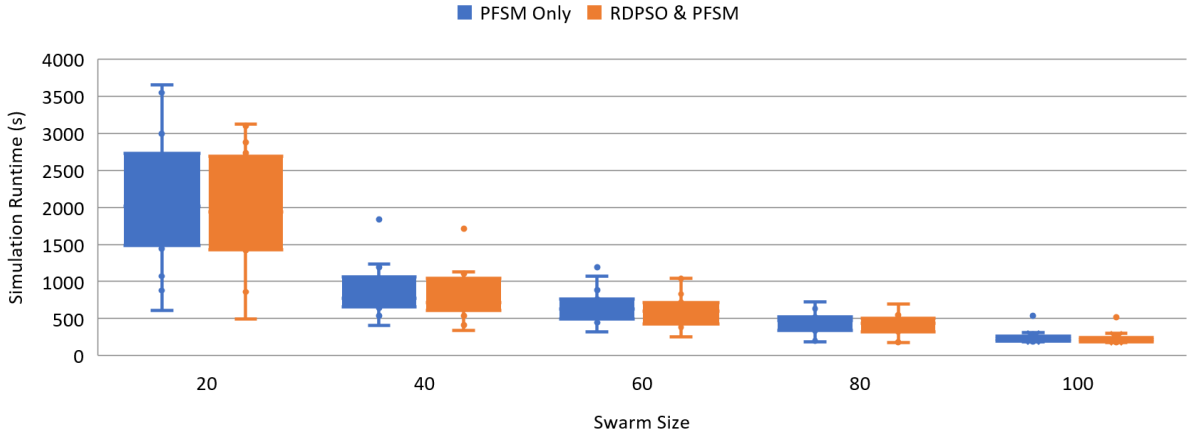
A comparison of different environment layouts is given in section 5.1.3. We conduct our experiments on three different environment sizes while varying the swarm size as follows:

- $25m^2$ Area: 10, **20**, 30, **40** & 50 robots
- $100m^2$ Area: **20**, **40**, 60, **80** & 100 robots
- $400m^2$ Area: **40**, **80**, 120, 160 & 200 robots

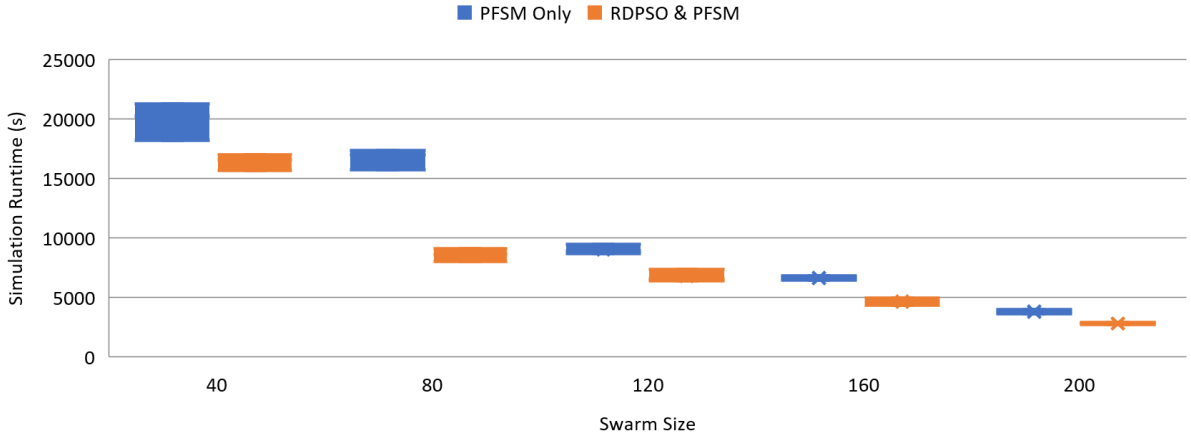
For the sake of the size comparison, we compare results of deploying only 20 & 40 robots in the small and medium environment areas. We also compare results for 40 & 80 robots in the medium and big environment areas. Fixing the swarm size ensures that results will only represent the size difference.



(a) Small Environment of area $25m^2$

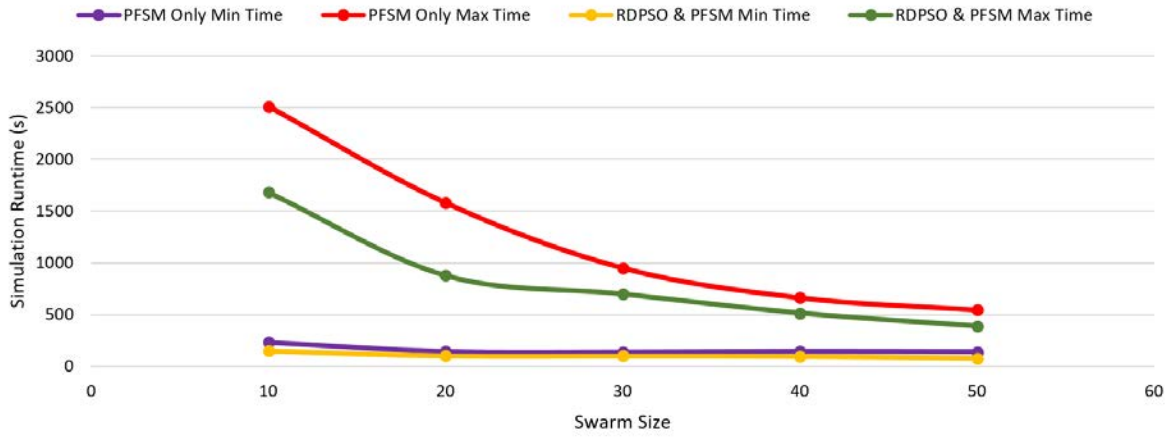


(b) Medium Environment of area $100m^2$

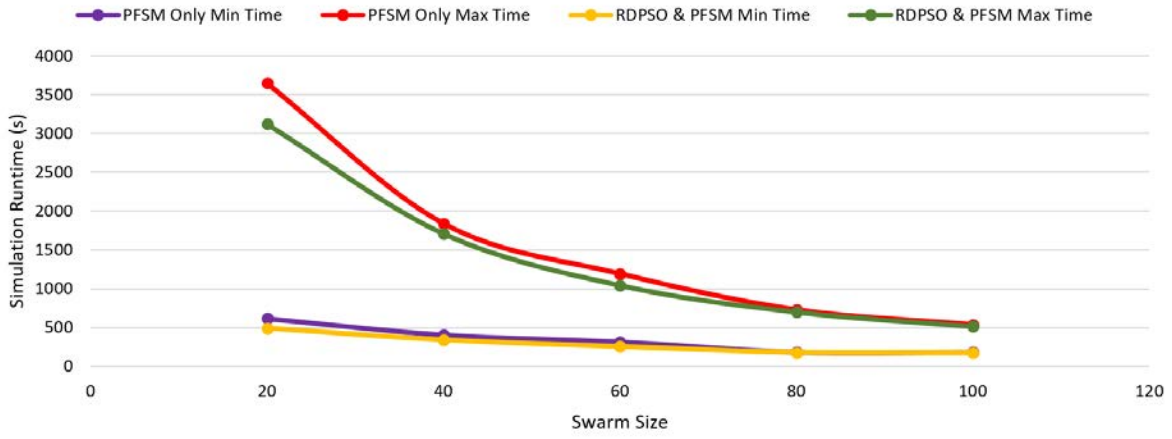


(c) Big Environment of area $400m^2$

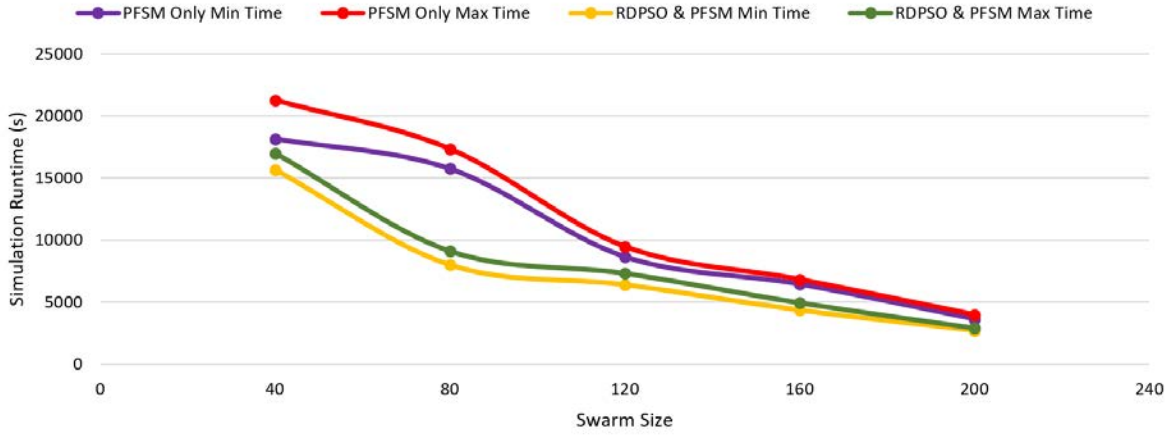
Figure 5.1: The relation between the swarm size and the amount of time taken by a swarm of AntBOTs to explore and navigate all environment areas: $25m^2$, $100m^2$, $400m^2$



(a) Small Environment Area: $25m^2$



(b) Medium Environment Area: $100m^2$



(c) Big Environment Area: $400m^2$

Figure 5.2: The minimum and maximum time taken by a swarm of AntBOTs to fully navigate all environment areas: $25m^2$, $100m^2$ and $400m^2$.

Number of Robots	Maximum Time (s)		Minimum Time (s)	
	PFSM Only	RDPSO & PFSM	PFSM Only	RDPSO & PFSM
10	2510.075	1684.355	233.252	150.501
20	1579.565	878.788	141.401	101.181
30	950.459	700.204	135.501	99.788
40	663.456	515.377	144.551	96.625
50	544.905	390.256	140.251	78.522

(a) Small Environment of $25m^2$ Area.

Number of Robots	Maximum Time (s)		Minimum Time (s)	
	PFSM Only	RDPSO & PFSM	PFSM Only	RDPSO & PFSM
20	3651.186	3120.517	613.906	490.419
40	1841.178	1710.866	405.204	339.771
60	1194.062	1044.713	316.456	253.169
80	729.007	701.008	185.901	178.275
100	537.105	514.942	185.301	174.511

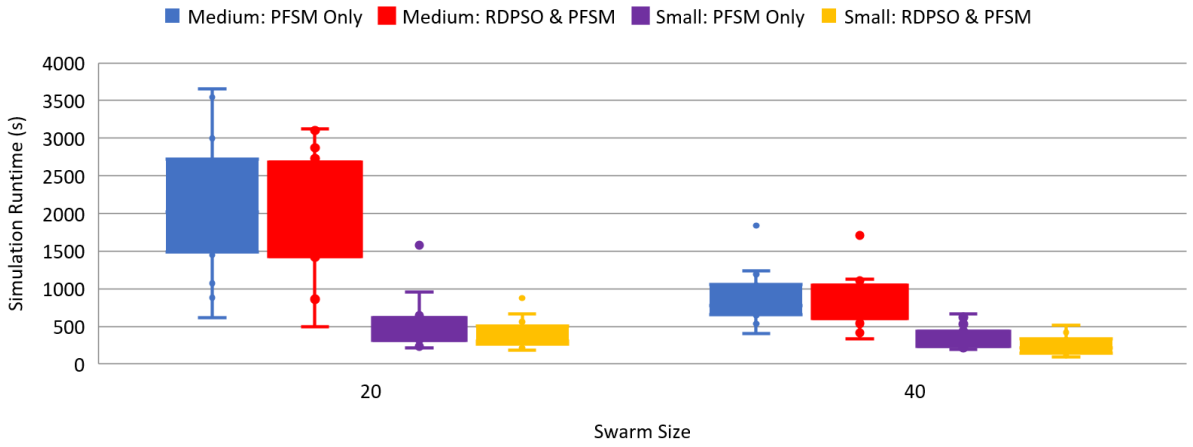
(b) Medium Environment Area: $100m^2$

Number of Robots	Maximum Time (s)		Minimum Time (s)	
	PFSM Only	RDPSO & PFSM	PFSM Only	RDPSO & PFSM
40	21268.964	17001.259	18165.963	15669.485
80	17339.353	9125.362	15748.615	7999.385
120	9486.183	7336.613	8659.756	6403.943
160	6839.193	4965.112	6456.853	4355.493
200	4002.169	2931.634	3649.843	2722.158

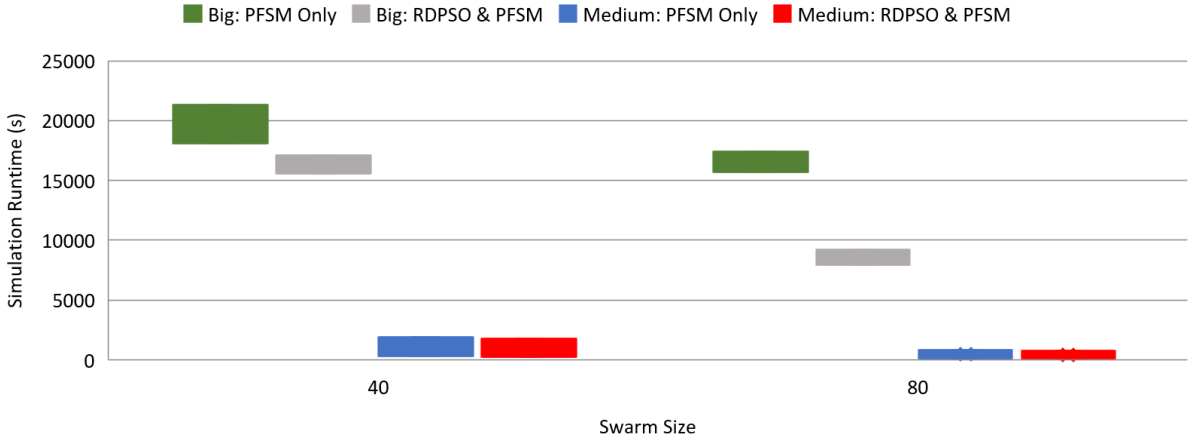
(c) Big Environment Area: $400m^2$

Table 5.1: The minimum and maximum time taken by a swarm of AntBOTs to fully navigate all environment areas: $25m^2$, $100m^2$ and $400m^2$

As can be seen in figure 5.3a, increasing the size of the environment from $25m^2$ in figure 4.3a to $100m^2$ in figure 4.3b increases the amount of time required for robots to allocate the target location and find the optimal path between the start and target location. Also as shown in figure 5.3b, increasing the size of the environment from $100m^2$ in figure 4.3b to $400m^2$ in figure 4.3c had a significant effect on the time needed for exploration and navigation. We also notice that combining RPDSO with PFSM always helps with decreasing the time required for exploration and navigation. Thus, we conclude that as the environment area increases, the time required for a mission increases at a higher rate in the case of using PFSM on its own. Although there is a proportional relation between the time and the area in Swarm Robotics, exploring such a big environment area with 200 robots will definitely be much faster than using only 1 or 2 big complex robots. This is one of the strongest points to using swarm of simple robots vs a single complex robot.



(a) Area change from $25m^2$ to $100m^2$



(b) Area change from $100m^2$ to $400m^2$

Figure 5.3: The effect of changing the environment area from $25m^2$ to $100m^2$ and from $100m^2$ to $400m^2$ on the simulation runtime

5.1.3 Experiment Runtime vs Environment Layout

In this section, we present a comparison between two different environment designs shown in figures 4.4a and 4.4b using both RDPSO and PFSM for such comparison. Both environments are $100m^2$ in area.

As shown in figure 5.4, the amount of time required to fully explore and navigate the environment in figure 4.4b with randomly deployed obstacles decreased dramatically compared to the environment of the same size in figure 4.4a. This proves that different designs can significantly affect the amount of time required to explore and navigate an environment.

As shown in figure 5.5, with 60 robots, the maximum time required to explore and navigate the environment with obstacles is even less than the minimum time required to explore and

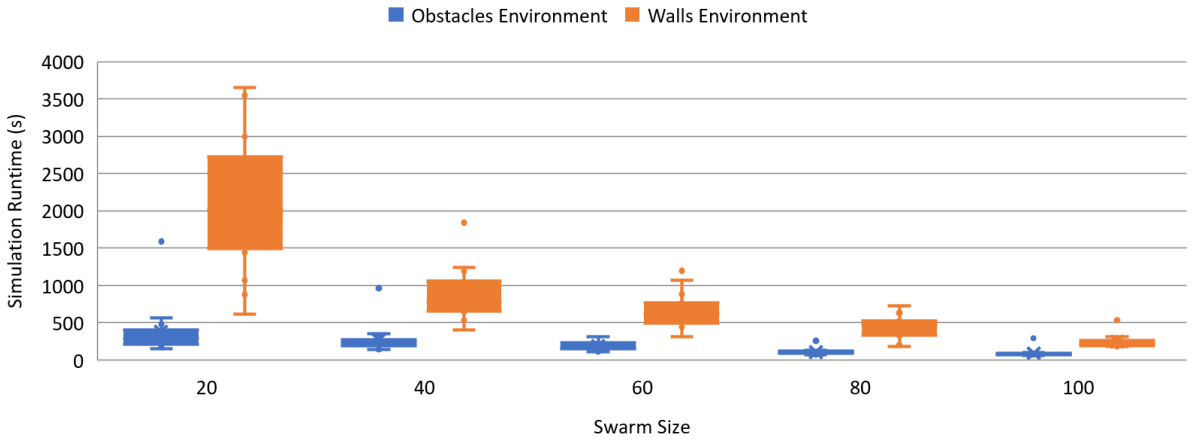


Figure 5.4: The relation between the swarm size and the amount of time taken by a swarm of AntBOTs using both RDPSO & PFSM to explore and navigate the same environment size but with different designs; Walls & Obstacles

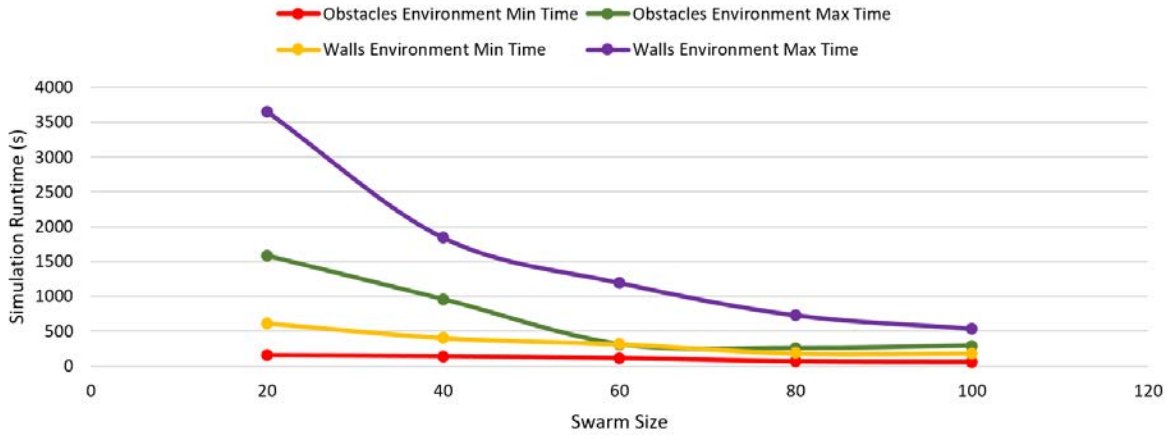


Figure 5.5: The minimum and maximum time taken by a swarm of AntBOTs to fully navigate the same environment size but with different designs; Walls & Obstacles

navigate the environment with walls. This further confirms our "walled" environment design as a more challenging and meaningful environment to use.

5.1.4 Swarm Size vs Reporting Time

The reporting time is the time taken by the whole swarm to communicate their locations and agree on the optimal path between the start and the target location. This starts right after the environment is reasonably explored and the majority of robots are aware of the target location. At this time, robots decide to stop at their locations and start broadcasting help messages to aid the movement of any robot from the start to the target location. It is worth mentioning

that for both techniques, robots took a maximum time of 1560ms as can be seen in figure 5.6 to agree on the optimal path in the big environment.

This shows the strength of Swarm Robotics where communication is super fast due the mobile ad-hoc network formed between robots. Although the time taken for the target location propagation in the network almost doubled in the case of big environment. It is still not a major increase. Thus, we conclude that time is not significantly affected by the environment area. Although there is a difference between the reporting time in the case of PFSM only and the reporting time for both RDPSO and PFSM, this difference only ranges from 100ms to 200ms which is negligible given the speed of processing on the board of each AntBOT.

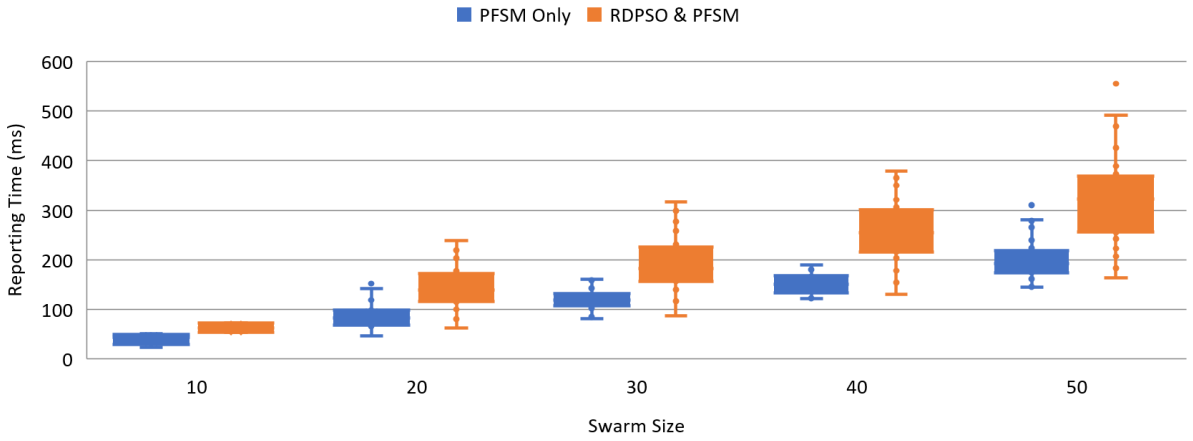
In the case of RDPSO and PFSM, for all environment areas, the reporting time increased due to the fact that robots are navigating the environment in groups. Whenever a member of the group locates or receives a target location, the rest of its swarm will also receive this signal. This guarantees continuous network connectivity because each part of the environment is covered not only with a single robot but with a number of robots forming a smaller swarm.

5.1.5 Swarm Size vs Path Size

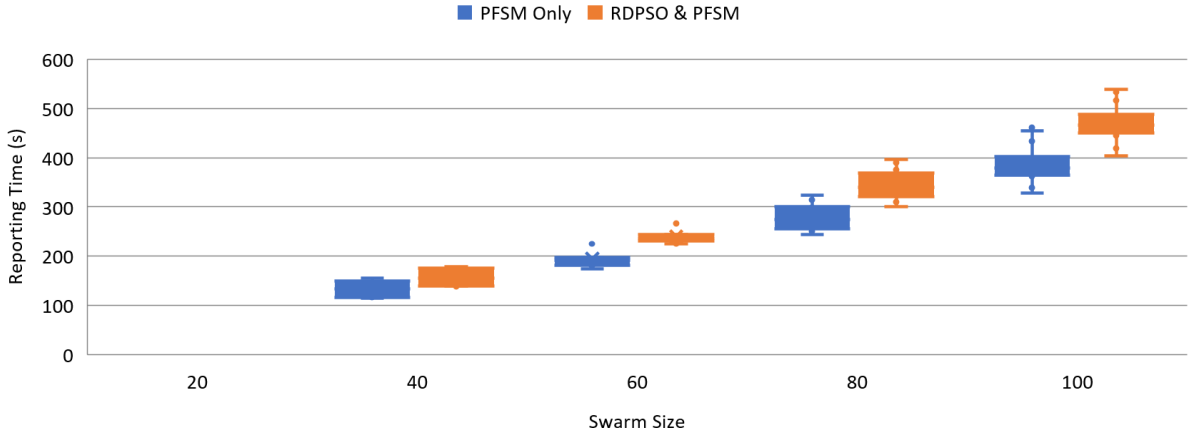
This section describes the relationship between the number of lead robots contributing to the optimal path between the start and the target locations and the swarm size. We notice that the increase in the swarm size in all environment areas has the following effect:

- PFSM Only: Path size was not significantly affected.
- RPDSO & PFSM: Path size increased significantly.

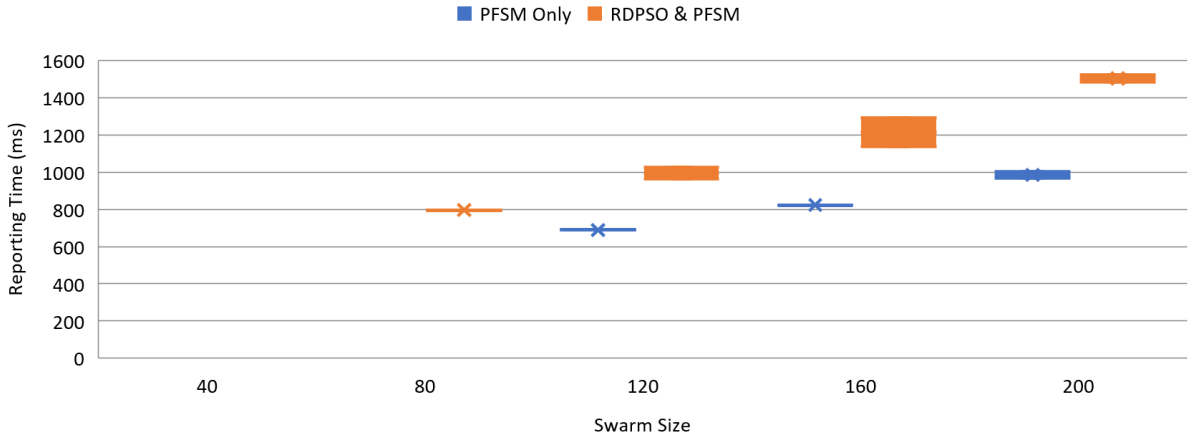
The nature of the RDPSO algorithm where it guides robots to navigate the environment in groups is the main reason for this. Figures 5.7, 5.8 and 5.9 further explain the relationship between both and prove that, in the case of RDPSO & PFSM, robots navigate the environment in groups regardless of the environment area which results in a more stable connection between robots in different places of the environment. Note that the zero points in path size indicates a failure scenario.



(a) Small Environment Area: $25m^2$

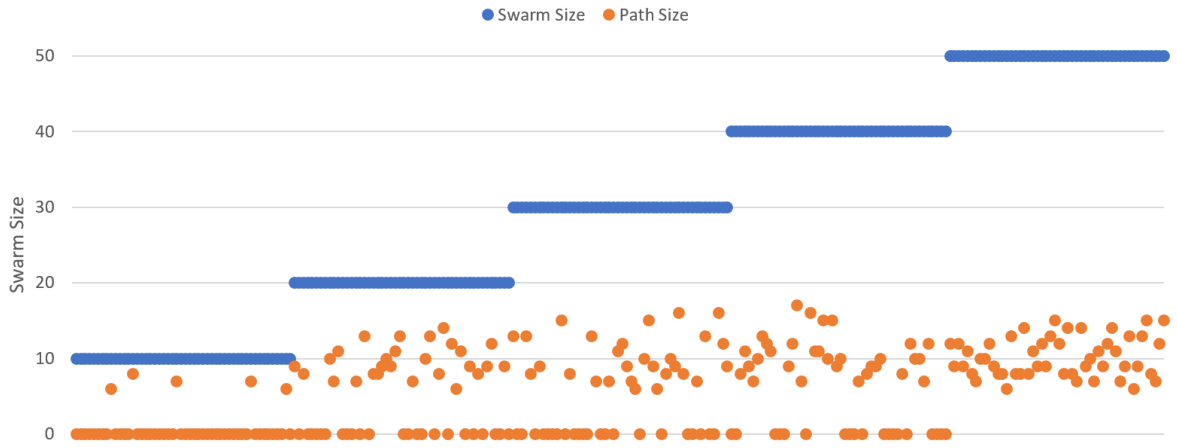


(b) Medium Environment Area: $100m^2$

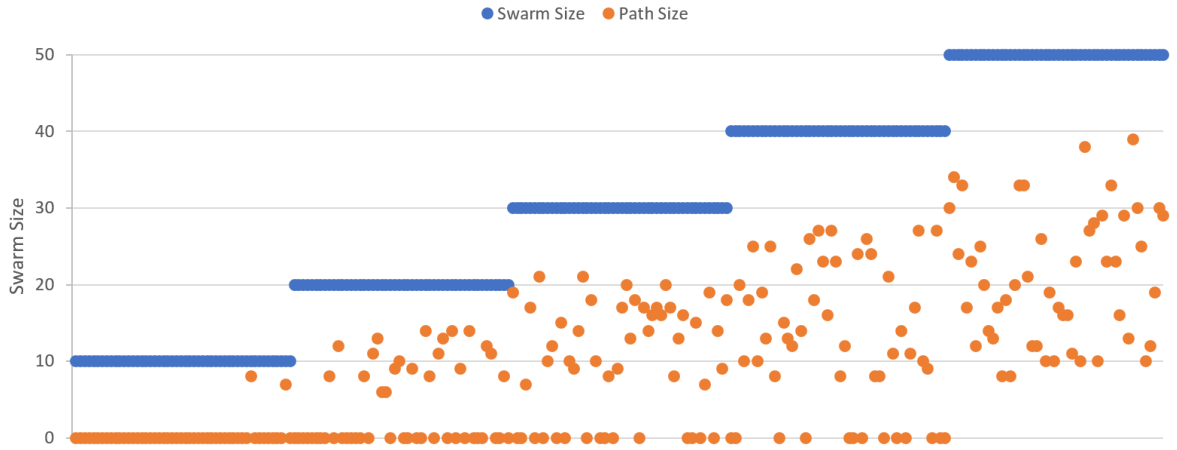


(c) Big Environment Area: $400m^2$

Figure 5.6: The relationship between the swarm size in all environment areas: $25m^2$, $100m^2$ & $400m2$ and the time taken to agree on the optimal path between the start and the target locations



(a) PFSM only

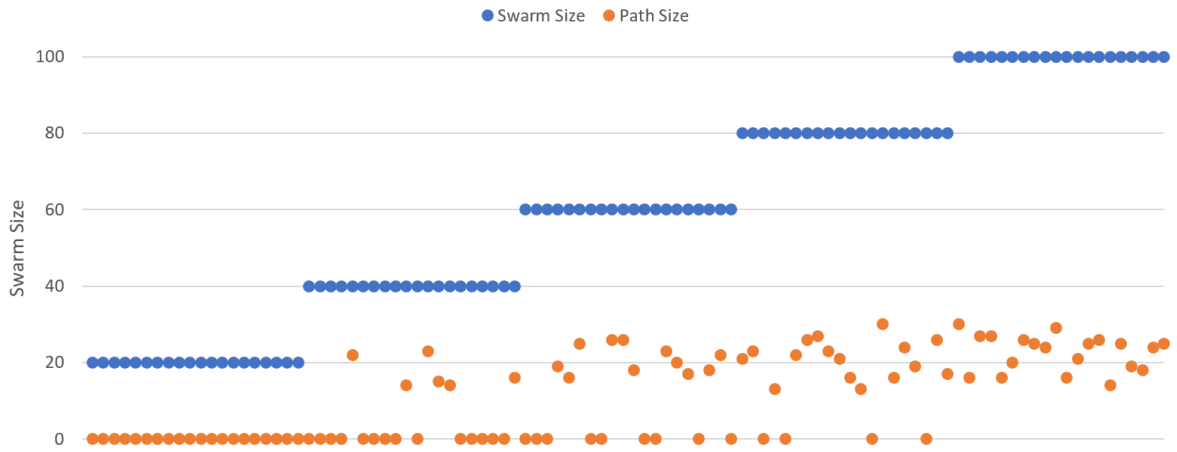


(b) RDPSO & PFSM

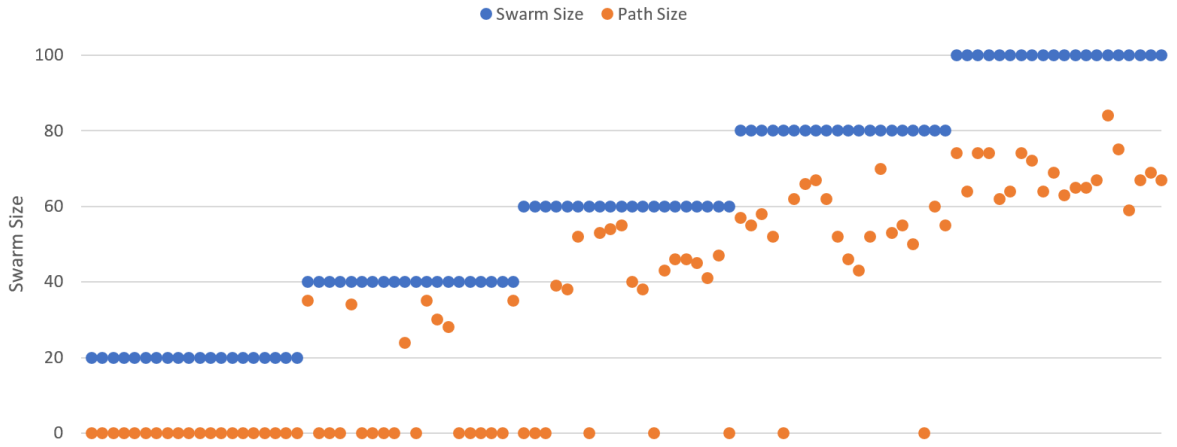
Figure 5.7: The relationship between the swarm size in the small environment of $25m^2$ area and path size

5.1.6 Swarm Size vs Success Rate

With respect to all the environment areas, we describe the relationship between the swarm size and the success rate. It's important to note that an experiment is marked as success only if robots were able to locate the target location. Tables 5.2a, 5.2b and 5.2c which illustrates the success rate comparisons in the small, medium and big environment areas respectively show that the increase in the swarm size increases the success rate in all environment areas. We also notice that while using RDPSO with PFSM, the success rate increases significantly due to the organization of the robots movement in the environment especially with bigger environment areas. Only at low swarm sizes (i.e. not enough for environment coverage), combining RDPSO with PFSM decreases the success rate. This is due to the fact that robots are not scattered in



(a) PFSM only



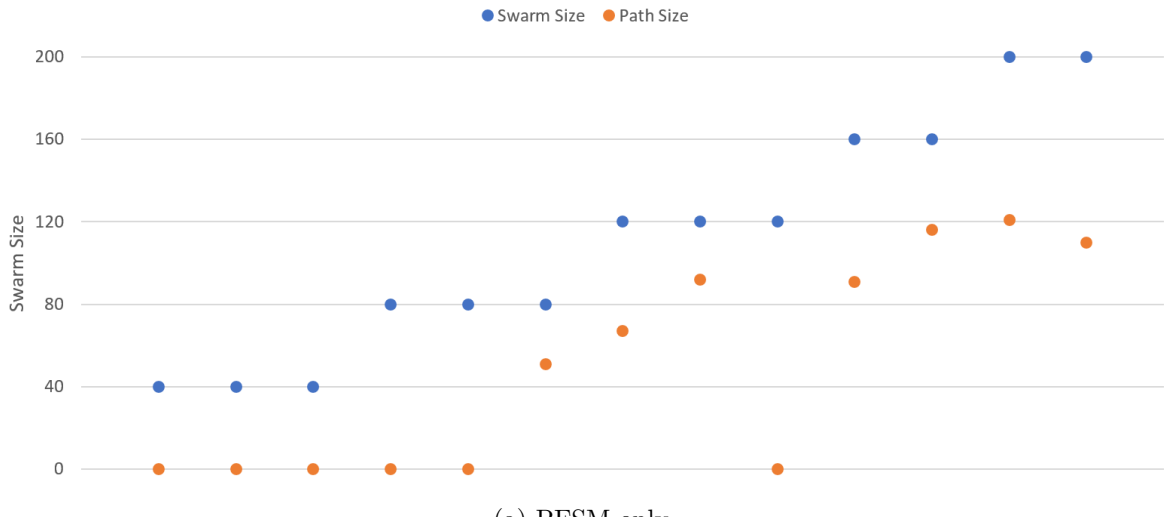
(b) RDPSO & PFSM

Figure 5.8: The relationship between the swarm size in the medium environment of $100m^2$ area and path size

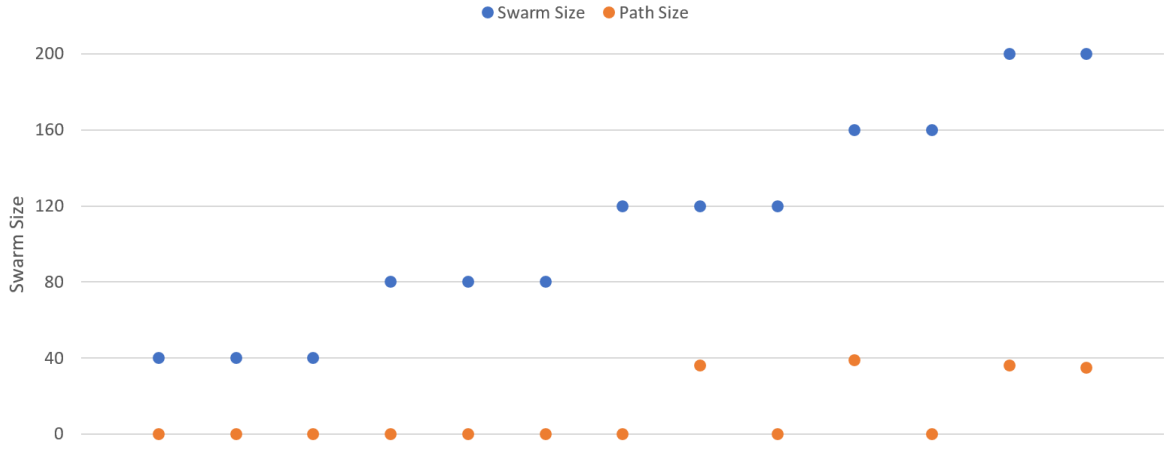
the environment individually. On the contrary, they move in groups which results in smaller areas of the environment being densely covered.

Table 5.2a which illustrates the success rate comparisons in the small environment area can be explained as follows:

- **10 & 20 Robots**: In the case of using RDPSO & PFSM, the success rate decreased due to the fact that robots are moving in groups and if the number is not sufficient to cover the whole environment area, it will be harder for the swarm to locate the target.
- **30 & 40 Robots**: The success rate increased because the swarm size is sufficient for the environment.



(a) PFSM only



(b) RDPSO & PFSM

Figure 5.9: The relationship between the swarm size in the big environment of $400m^2$ area and path size

- **50 Robots**: In the area of $25m^2$, 50 robots are enough to explore the environment. Thus, with or without RDPSO, robots were able to locate the target. The issue in this case is that when the swarm size is big and robots are scattered in the environment, they might hinder the movement of the main robot guided by the swarm.

while table 5.2b which illustrates the success rate comparisons in the medium environment area can be explained as follows:

- **20 Robots**: The success rate decreased due to the fact that robots are moving in groups and if the number is not sufficient to cover the whole environment area, it will be harder for the swarm to locate the target.

- **40, 60 & 80 Robots**: The success rate increased because the swarm size is sufficient for the environment.
- **100 Robots**: In the area of $100m^2$, 100 robots are enough to explore the environment. Thus robots were able to locate the target using both techniques.

while table 5.2c which illustrates the success rate comparisons in the big environment area can be explained as follows:

- **40 Robots**: This swarm size is not enough to explore the big environment area and thus there were no success using both approaches.
- **80, 120 & 160 Robots**: For these swarms sizes, usage of RPDSO along with PFSM helped increasing the success rate.
- **200 Robots**: In the area of $400m^2$, 200 robots are enough to explore the environment. Thus robots were able to locate the target using both techniques.

Figure 5.10 shows the relation between the increase in the swarm size and the success rate in all environment areas.

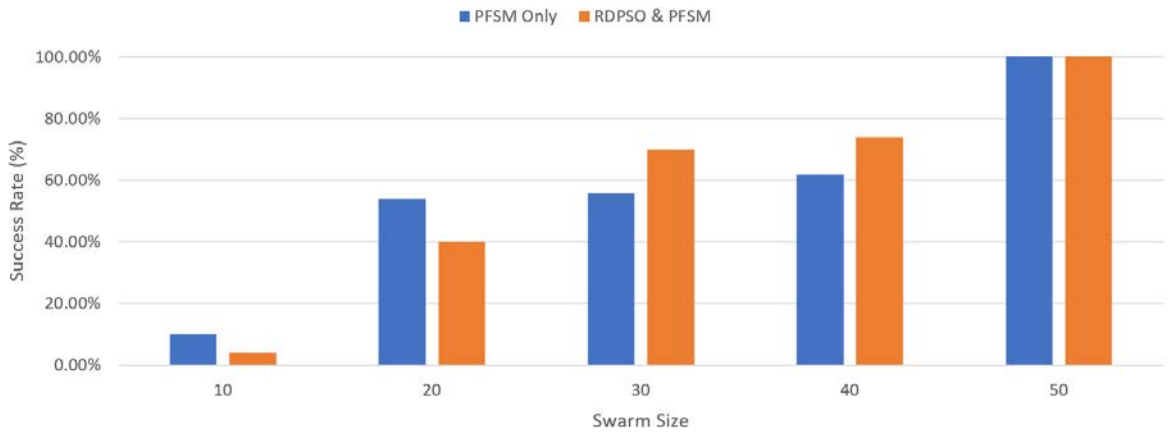
5.2 Simulation Environment & Hardware

For all the collected results, we used two DELL Alienware Aurora Desktops with the following configurations:

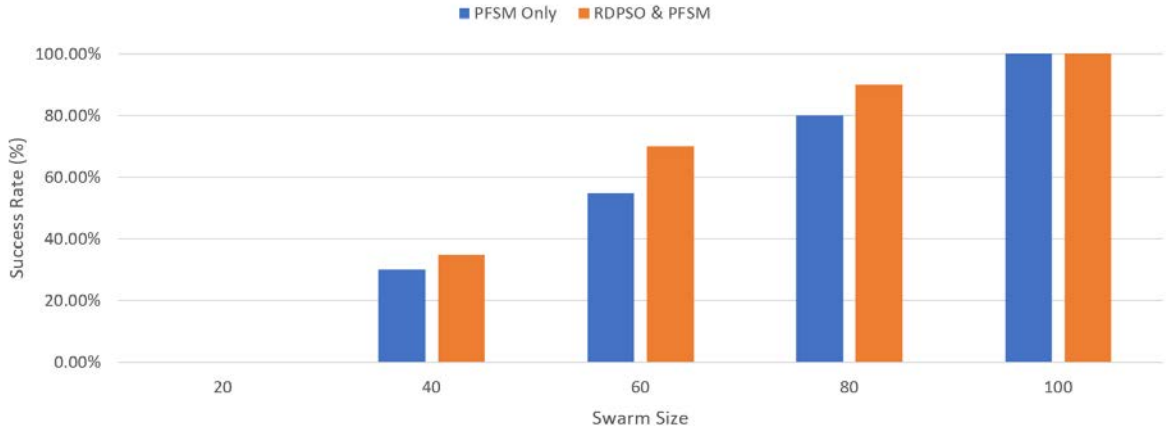
- Intel Core i7-6700K Processor (8MB Cache, Overclocked up to 4.2GHz)
- NVIDIA GeForce GTX 1080 Founders Edition with 8GB GDDR5X
- 16GB 2133MHz DDR4 Ram Memory
- 850W PSU Chassis Liquid Cooled

As for the software setup, we used the following:

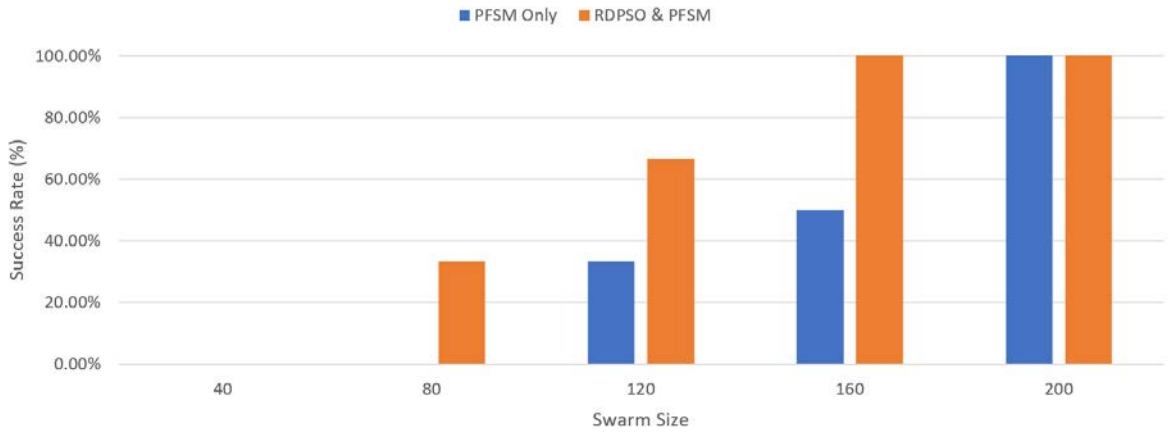
- V-REP Simulator



(a) Small Environment Area: 25^2



(b) Medium Environment Area: 100^2



(c) Big Environment Area: 400^2

Figure 5.10: Swarm Size vs the success rate given all environment areas

Number of Robots	**Success Rate**	
	PFSM Only	RDPSO & PFSM
10	10.00%	4.00%
20	54.00%	40.00%
30	56.00%	70.00%
40	62.00%	74.00%
50	100.00%	100.00%

(a) Small Environment Area: $25m^2$

Number of Robots	**Success Rate**	
	PFSM Only	RDPSO & PFSM
20	10.00%	0.00%
40	30.00%	35.00%
60	55.00%	70.00%
80	80.00%	90.00%
100	100.00%	100.00%

(b) Medium Environment Area: $100m^2$

Number of Robots	**Success Rate**	
	PFSM Only	RDPSO & PFSM
40	0.00%	0.00%
80	0.00%	33.33%
120	33.33%	66.67%
160	50.00%	100.00%
200	100.00%	100.00%

(c) Big Environment Area: $400m^2$

Table 5.2: Swarm Size and the corresponding Success Rate in all environment areas

- For remote API, we used Python(x,y) which is a free scientific and engineering development software.

Although the above setup performed reasonably good on moderate swarm sizes, as the swarm size increased beyond 100 robots, a significant lag in the simulation clock was noticed when compared to the wall clock. For example, while using 200 robots in the big environment, one second in simulation took exactly 15 minutes, 17 seconds and 23 milliseconds with dt=50ms of wall clock time. This limited us from trying even bigger environment areas with bigger swarm sizes.

5.3 Results conclusion

In conclusion, we present our approach of combining two different design levels (i.e. Microscopic and Macroscopic). The microscopic level is designed using Probabilistic Finite State Machine

(PFSM) while the macroscopic level is designed using Robotic Darwinian Particle Swarm Optimization (RDPSO). Our novel approach is proved to decrease the amount of time required by a swarm of AntBOTs to explore and navigate an environment regardless of its area. This is only guaranteed if the swarm size used in the environment is adequate for its area.

We note that increasing the swarm size above certain levels causes the effect of RDPSO algorithm to saturate. Thus, an average convenient swarm size should be used for each environment area to optimize the time and cost per a single mission. We used three different environment areas and two layouts to verify our approach on the V-REP simulator. We also present the physical model of the AntBOT presented in appendix A as a proof of concept for future experimentation.

Chapter 6

Conclusion

Swarm Robotics research has received major attention in the last decade. Swarm systems are proved to be robust, scalable, simple, parallelizable and economical. In this thesis, we proposed a novel approach discussing exploration and navigation problem on both the individual and the collective behavior. Our approach is based on Robotic Darwinian Particle Swarm Optimization (RDPSO), Probabilistic Finite State Machine (PFSM) and Depth First Search (DFS). We also provide a new cheaper swarming platform called AntBOT modeled in V-REP Simulator.

Our solution provides an innovative method to help our robots efficiently explore unknown environments in hazardous scenarios. Robots navigate the environment while maintaining swarm robot communication in the absence of a network infrastructure. Results show that combining RDPSO & PFSM increases the speed of exploration by at least 1.4x the speed on a single algorithm. Furthermore, it improves the robots movements in the environment as well as the mission success rate by a value not less than 40%. This allows the use of smaller swarm sizes for exploration and rescue. Using small swarm sizes reduces the cost per a single mission. Thus, combination of RDPSO & PFSM also helps with cost reduction.

We collected our results from different simulated environment layouts as well as different environment areas ($25m^2$, $100m^2$ & $400m^2$). We also increased the swarm size based on each environment area. The real physical model of the AntBOT is also provided as a proof of concept for future experimentation.

6.1 Future Work

Although Swarm Robotics research is garnering attention in the Multi-Robot Community, there is still a significant gap between research environments and real-world environments. As a future work, authors believe that researchers should give more attention to real-world scenarios and try to minimize the aforementioned gap by using prototypes that are closer to real models. We also believe that experiments should be conducted outside research labs to test applicability to real world applications. Also, further improvements to the real model of the AntBOT has to be conducted to start introducing an even cheaper swarming solution for those who are interested in testing their approaches in real world scenarios. As for the simulator, we believe that collecting data from an even bigger environment areas such as: $1000m^2$ and $2000m^2$ will add more credibility to the approach of combining microscopic and macroscopic system designs.

Appendix A

Experimental Platform Design

In this research, we test our exploration and navigation approach on a simulator called *Virtual Robotic Experimentation Platform (V-REP)*. We also started the development of the actual model of the AntBOT using the Pololu m3pi robot but it's still under development. Further work on this should be done as future work. Researchers conducting Swarm Robotics research can be divided in three categories: 1) researchers who depend on simulation only and they form around 60% of research, 2) researchers who depend on real robots only and they form a minority of research and 3) researchers who experiment on both real-robots and simulation and they form the rest of the available work.

A.1 Simulation - V-REP

In most of the robotics research, simulation plays a vital role in testing different approaches. For this, we decided to run our exploration and navigation approach on one of the most powerful simulators V-REP. Many surveys on different available robotic simulators were conducted [159]. At the beginning, we narrowed down all available simulators to only two of them; *Webots* from CYBERBOTICS Ltd. [160] and *Virtual Robot Experimentation Platform* from Coppelia Robotics [161]. Although Webots simulator is more frequently used in multi-robot research, we choose V-REP because its available open-source for free while the cheapest version of Webots that has enough tools and robot models costs around $3400 at the time of writing this document.

Virtual Robot Experimentation Platform (V-REP) is the Swiss army knife among robot simulators: you won't find a simulator with more functions, features, or more elaborate APIs. [161].

Robot controllers can be written in many languages such as Python, C/C++, Java and others. In our experiments, we use Python as the main development language because of its simplicity, ease of use and integration with many available complex python libraries.

A.2 Proposed Swarming Platform (AntBOTs)

In this research, we found that commercial off-the-shelf (COTS) swarming robots like e-puck [105] are quite expensive for a swarm as one piece costs around $1K at the time of writing this document. After some extensive research, we decided to design and build our own swarming platform AntBOT. The robot is a low cost platform and based on the Pololu m3pi robot [157] and Mbed Controller [162]. The cost of single AntBOT is around $250 as discussed in A.2.7. A top view of *AntBOT* is shown in figure A.1 and a bottom view is shown in figure A.2. In this section, we discuss full specifications and description of the platform.

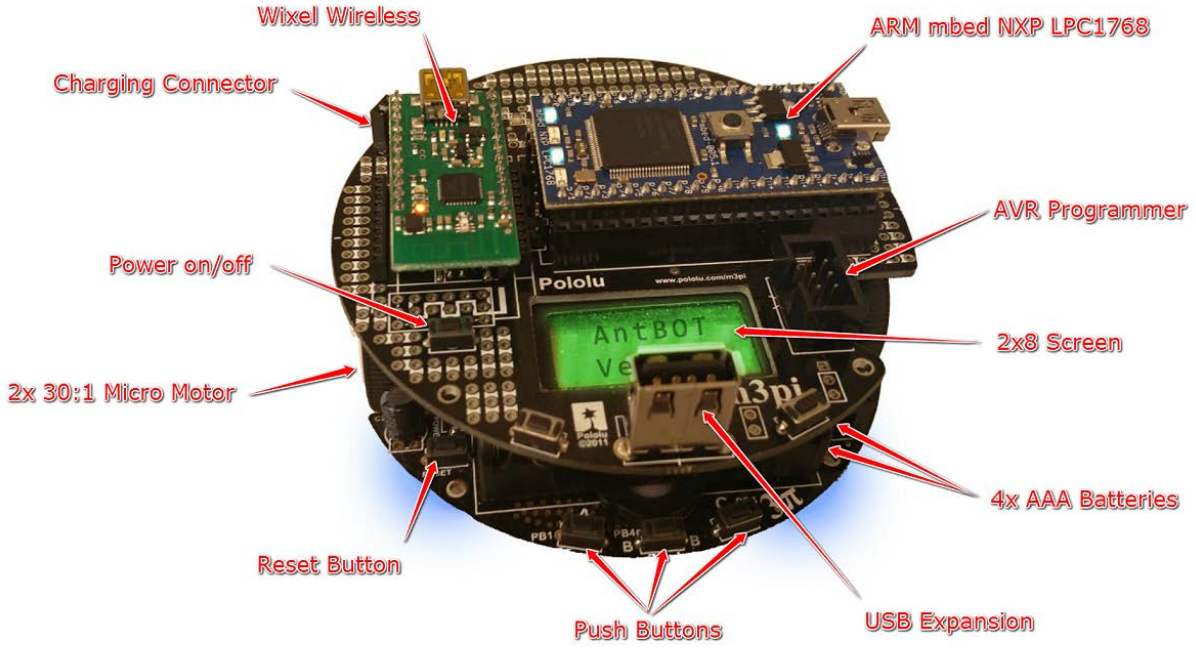


Figure A.1: AntBot Top view annotated with core features

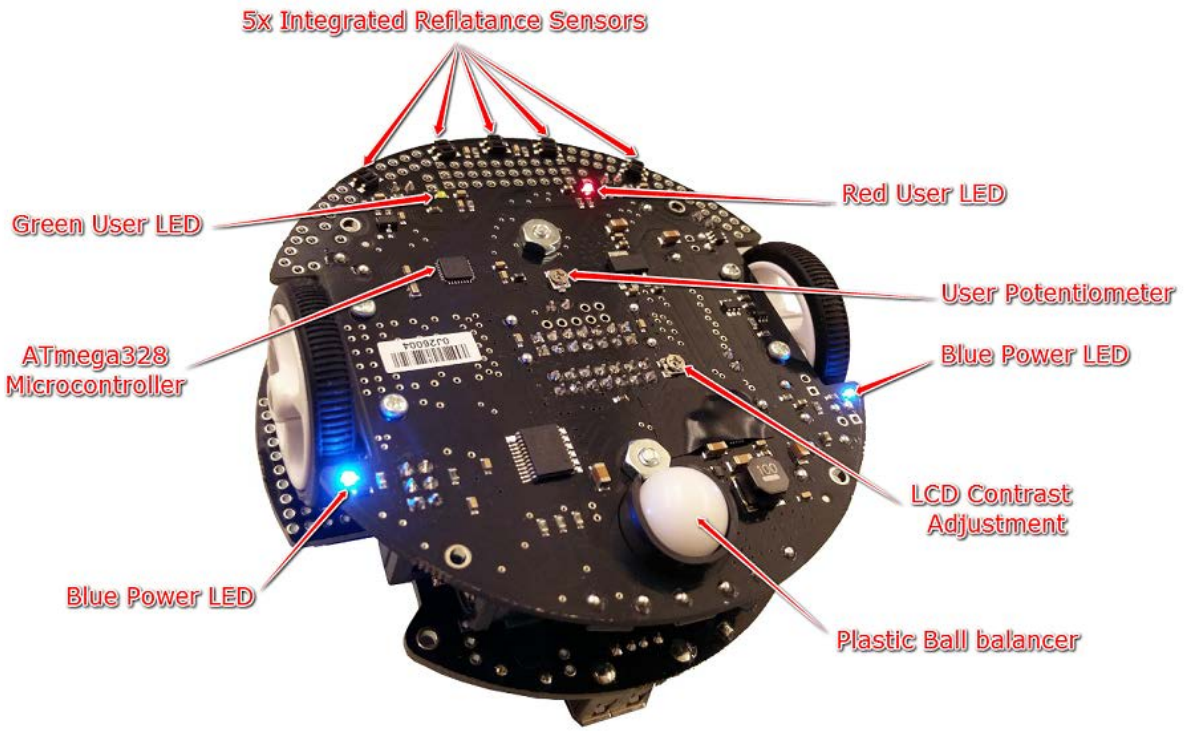


Figure A.2: AntBot Bottom view annotated with core features

We discuss several aspects of AntBOT design and some design decisions we took and why. We divide our discussion in six parts as follows: 1) mechanical design, 2) electronic design, 3) IR reflectance sensors, 4) accelerometer, 5) communication and 6) charging.

A.2.1 Mechanical Design

AntBOTs consist of two main parts:

1. Pololu 3pi robot which has a 9.5 *cm* diameter and weigh 83 *grams* without batteries. Schematics for the 3pi is shown in figure B.2.
2. The expansion kit which is the layer on top of the 3pi robot. It has the same diameter.

The 3pi works as the base of the robot with wheels. The expansion kit on top is used to upgrade the power and functionality of the robot.

A.2.2 Electronic Design

Technical specifications of Pololu 3pi robot [157] are found in table A.1.

Processor	**ATmega328P**
Motor driver	**TB6612FNG**
Motor channels	**2**
User I/O lines	**2**
Minimum operating voltage	**3V**
Maximum operating voltage	**7V**
Maximum PWM frequency	**80 kHz**
Reverse voltage protection	**Y**
External programmer required	**Y**

Table A.1: Pololu 3pi Robot Specifications

In Antbot, we use the ATmega328P microcontroller in slave mode to control the five IR reflectance sensors, two motors and other components on the 3pi robot. ARM mbed microcontroller shown in figure A.3 is used the main controller of the AntBOT. Pinouts and simplified block diagram of ARM mbed micro-controller can be found in Appendix B.

ARM mbed technical details [162]:

- NXP LPC1768 MCU
  - High performance ARM Cortex-M3 Core
  - 96MHz, 32KB RAM, 512KB FLASH
  - Ethernet, USB Host/Device, 2xSPI, 2xI2C, 3xUART, CAN, 6xPWM, 6xADC, GPIO
- Prototyping form-factor
  - 40-pin 0.1" pitch DIP package, 54x26mm
  - 5V USB or 4.5-9V supply
  - Built-in USB drag 'n' drop FLASH programmer

A.2.3 IR reflectance sensors

The 3pi was originally developed for line following applications. Thus, its equipped with 5 QTR-RC reflectance sensors. They are 5 IR emitter and receiver (phototransistor) pairs which are facing the ground as shown in figure A.2.

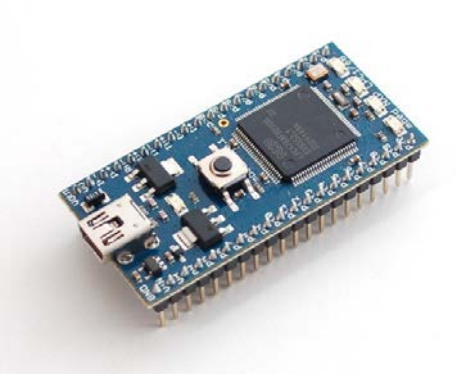


Figure A.3: ARM Mbed NXP LPC1768

Figure A.4: MMA7455 3-AXIS Accelerometer

A.2.4 Accelerometer

In our design, to be able to do collision detection, we use the MMA7455 3-AXIS Accelerometer shown in figure A.4 to detect collision detection. We only use one of the axes because AntBots move forward and backward only. We use the MMA7455 because of current availability and it can be replaced with any of the available single axis accelerometers.

A.2.5 Communication

For communication between AntBOTs or between AntBOTs and the PC, we use the Wixel Programmable USB Wireless Module shown in figure A.5 which is a general-purpose programmable module featuring a 2.4 GHz radio and USB [163]. General specifications of the Wixel Programmable USB Wireless Module are found in table A.2.

A.2.6 Charging

AntBOT is equipped with 4 AAA Ni-MH rechargeable batteries. Note that any regular alkaline cells can be used although rechargeable batteries are recommended to avoid the hassle of dissembling and reassembling the robot to change the batteries. Any NiMH off-the-shelf chargers

Processor	**CC2511F32 @ 24 MHz**
RAM size	**4096 bytes**
Program memory size	**29 Kbytes**
User I/O lines	**15**
Minimum operating voltage	**2.7 V**
Maximum operating voltage	**6.5 V**
Reverse voltage protection	**Y**
External programmer required	**N**

Table A.2: Wixel Programmable USB Wireless Module Technical Specifications

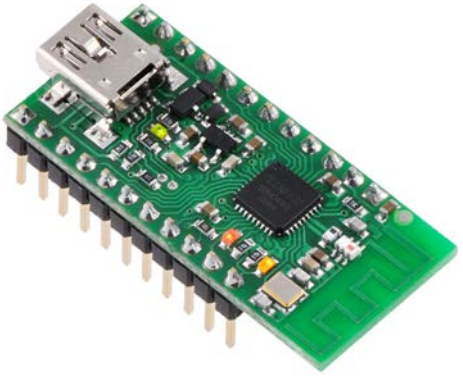


Figure A.5: Pololu Wixel Programmable USB Wireless Module

Figure A.6: iMAX B6AC Balance Charger

can be used although its recommended to use the iMAX B6AC Balance Charger shown in figure A.6.

A.2.7 Cost

The cost of a single AntBOT is around $250 and can be further optimized if the quantity increased. More details on the cost can be found in table A.3. The cost of a single charger which can be used to charge the whole swarm sequentially is around $55.

#	Item Description	Unit Pr.	Qt.	Total Pr.
1	Pololu m3pi Robot with mbed Socket	$ 149	1	$ 149
2	ARM mbed NXP LPC1768 Development Board	$ 49	1	$ 49
3	Wixel Programmable USB Wireless Module	$ 19	1	$ 19
4	Rechargeable NiMH AAA Battery: 1.2 V, 900 mAh	$ 1.59	4	$ 6.36
5	Single Axis Accelerometer	$ 25	1	$ 25
Total				**$ 248.36**

Table A.3: Single AntBOT Cost List

Appendix B

AntBOT Schematics

A pinout for the ARM mbed LPC1768 is shown in figure B.1. Schematics of the Pololu 3pi robot is shown in figure B.2. A simplified block diagram for LPC1768 is shown in figure B.3 with detailed features listed in table B.1.

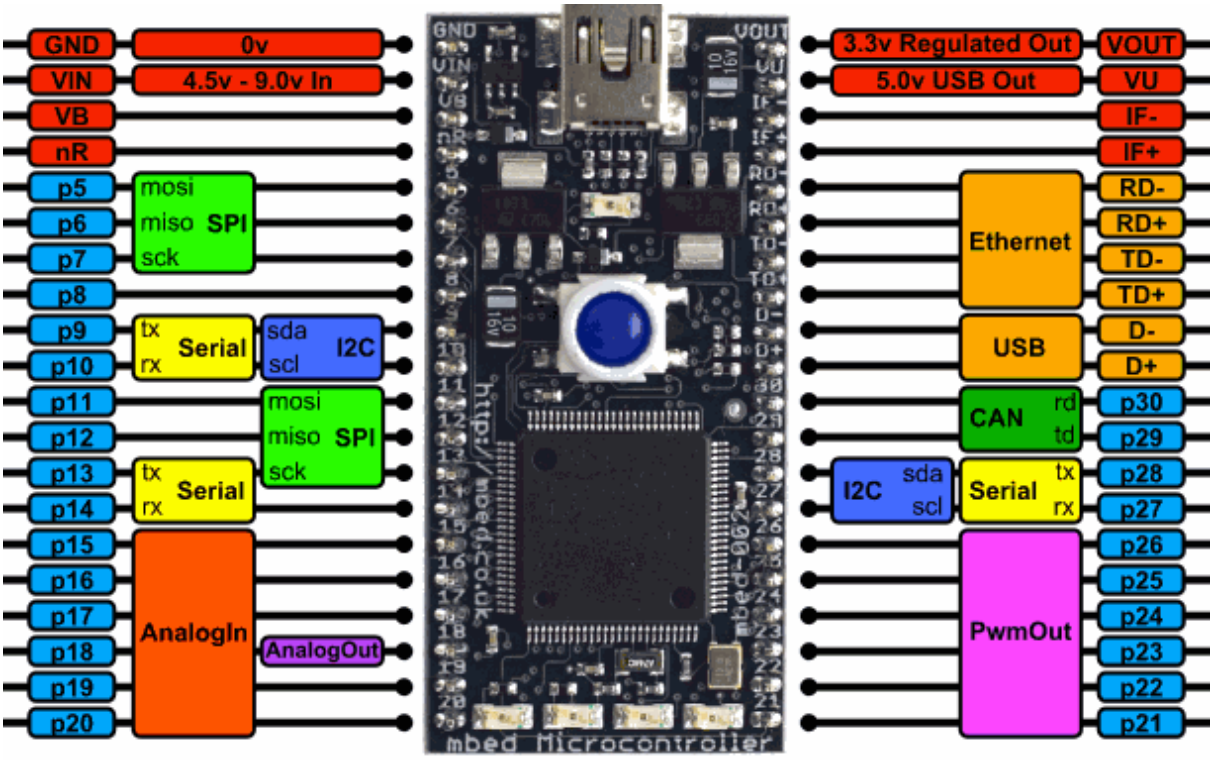


Figure B.1: ARM mbed LPC1768 Pinout

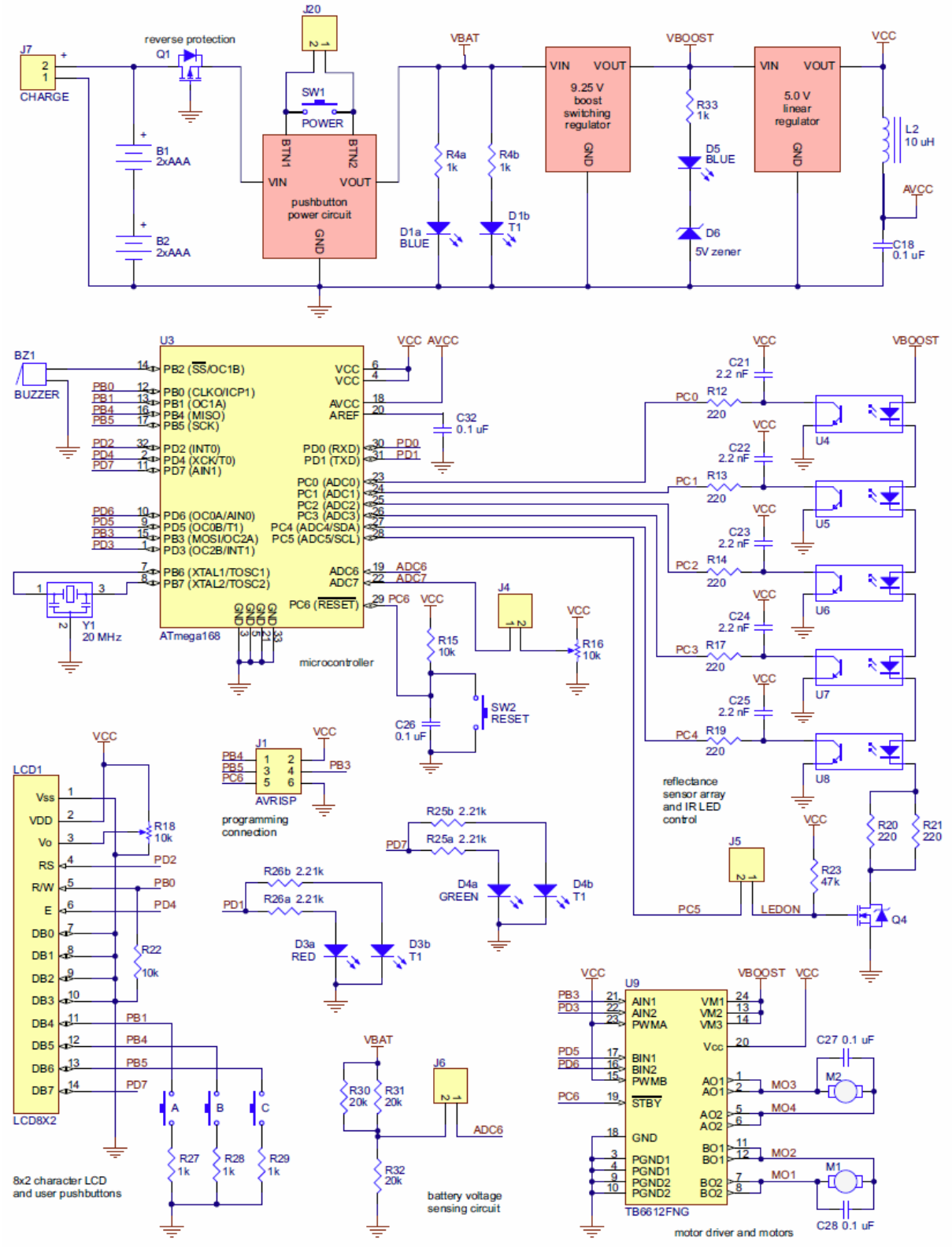


Figure B.2: Pololu 3pi Schematics

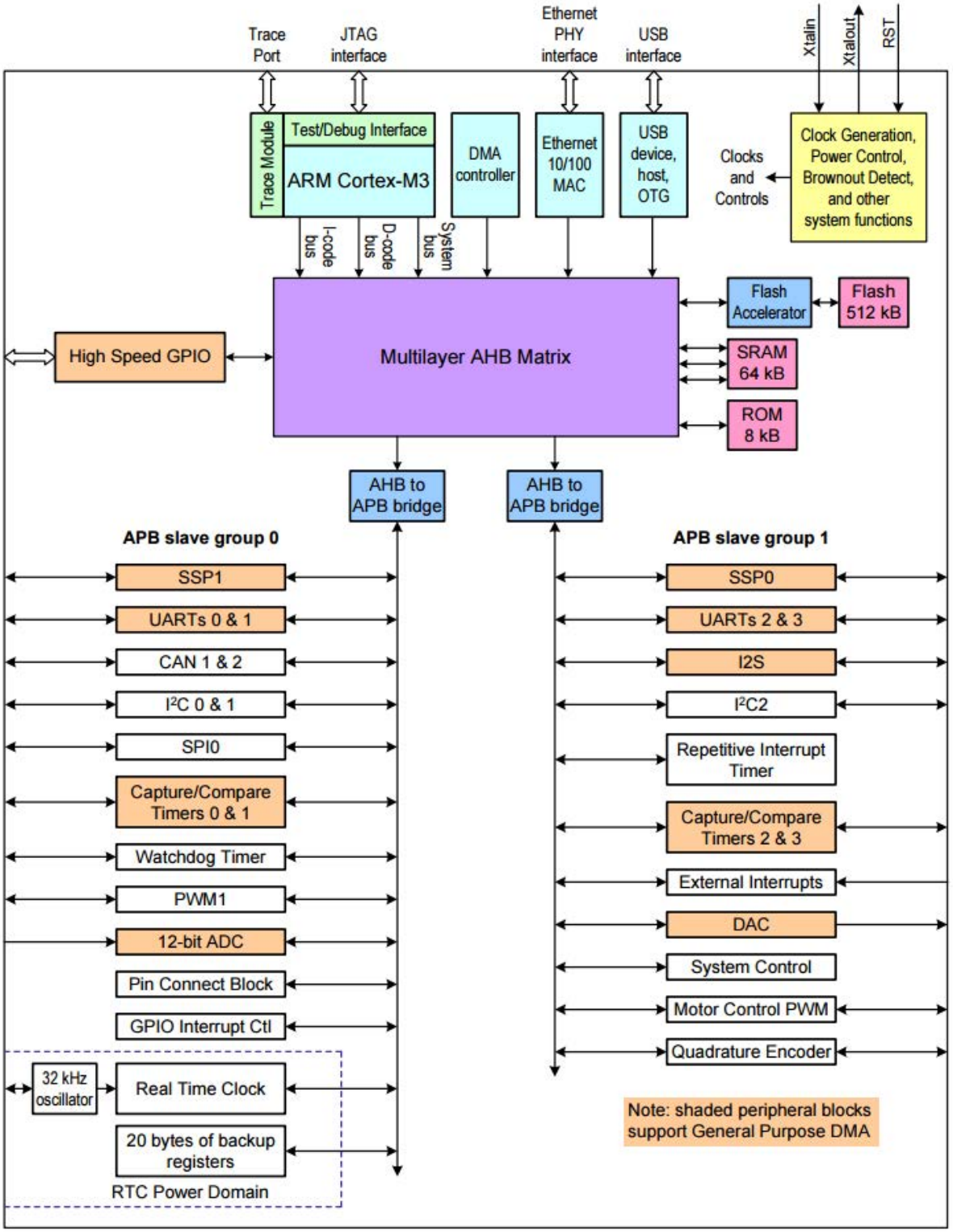


Figure B.3: LPC1768 simplified block diagram

ARM Cortex-M3 core	100 MHz operation
	Nested Vectored Interrupt Controller for fast deterministic interrupts
	Wakeup Interrupt Controller allows automatic wake from any priority interrupt
	Memory Protection Unit
	Four reduced-power modes: sleep, deep sleep, power-down and deep power-down
Memories	512 KB of Flash memory
	64 KB of SRAM
Serial peripherals	10/100 Ethernet MAC
	USB 2.0 full-speed device/Host/ OTG controller with on-chip PHY
	Four UARTs with fractional baud rate generation, RS-48, modem control, and IrDA
	Two CAN 2.0B controllers
	Three SSP/SPI controllers
	Three I2C-bus interfaces with one supporting Fast Mode Plus (1-Mbit/s data rates)
	I2S interface for digital audio
Analog peripherals	12-bit ADC with eight channels
	10-bit DAC
Other peripherals	Ultra-low-power (<1 uA) RTC
	General-purpose DMA controller with eight channels
	Up to 70 GPIO
	Motor control PWM and Quadrature Encoder Interface to support three-phase motors
	Four 32-bit general-purpose timers/counters
Package	100-pin LQFP (14 x 14 x 1.4 mm)

Table B.1: LPC1768 features

Bibliography

[1] Manuele Brambilla, Eliseo Ferrante, Mauro Birattari, and Marco Dorigo. Swarm robotics: a review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41, 2013. ISSN 1935-3812. doi: 10.1007/s11721-012-0075-2. URL http://dx.doi.org/10.1007/s11721-012-0075-2.

[2] S. Kazadi. Model independence in swarm robotics. *International Journal of Intelligent Computing and Cybernetics*, 2(4):672 – 694, 2009.

[3] Alan F. Winfield, Christopher J. Harper, and Julien Nembrini. Towards Dependable Swarms and a New Discipline of Swarm Engineering. *Swarm Robotics*, pages 126–142, 2005. URL http://www.springerlink.com/content/507nevmxlyd9vn8u.

[4] G. Dudek, M. R. M. Jenkin, and D. Wilkes. A taxonomy for swarm robots. In *Proceeding on the IEEE/RSJ International Conference on Intelligent Robotic Systems*, pages 441–447. IEEE, 1993.

[5] Y. Uny Cao, Alex S. Fukunaga, and Andrew B. Kahng. Cooperative mobile robotics: Antecedents and directions. *Auton. Robots*, 4(1):7–27, 1997. URL http://dblp.uni-trier.de/db/journals/arobots/arobots4.html#CaoFK97.

[6] Luca Iocchi, Daniele Nardi, and Massimiliano Salerno. Reactivity and deliberation: A survey on multi-robot systems. In *Balancing Reactivity and Social Deliberation in Multi-Agent Systems, From RoboCup to Real-World Applications (selected papers from the ECAI 2000 Workshop and additional contributions)*, pages 9–34, 2000. doi: 10.1007/3-540-44568-4_2. URL http://dx.doi.org/10.1007/3-540-44568-4_2.

[7] Veysel Gazi and Baris Fidan. Coordination and control of multi-agent dynamic systems: Models and approaches. In *Swarm Robotics, Second International Workshop, SAB 2006, Rome, Italy, September 30-October 1, 2006, Revised Selected Papers*, pages

71–102, 2006. doi: 10.1007/978-3-540-71541-2_6. URL http://dx.doi.org/10.1007/978-3-540-71541-2_6.

[8] L. Bayindir and E. Sahin. A review of studies in swarm robotics. *Turkish Journal of Electrical Engineering*, 15(2), 2007.

[9] Valentino Crespi, Aram Galstyan, and Kristina Lerman. Top-down vs bottom-up methodologies in multi-agent system design. *Auton. Robots*, 24(3):303–313, April 2008. ISSN 0929-5593. doi: 10.1007/s10514-007-9080-5. URL http://dx.doi.org/10.1007/s10514-007-9080-5.

[10] Rodney A. Brooks. A robust layered control system for a mobile robot. Technical report, Massachusetts Institute of Technology, Cambridge, MA, USA, 1985.

[11] Marvin L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1967. ISBN 0-13-165563-9.

[12] Onur Soysal and Erol Sahin. Probabilistic aggregation strategies in swarm robotic systems. In *2005 IEEE Swarm Intelligence Symposium, SIS 2005, Pasadena, California, USA, June 8-10, 2005*, pages 325–332, 2005. doi: 10.1109/SIS.2005.1501639. URL http://dx.doi.org/10.1109/SIS.2005.1501639.

[13] M. Granovetter. Threshold models of collective behavior. *The American Journal of Sociology*, 83(6):1420–1443, 1978.

[14] Guy Theraulaz, Eric Bonabeau, and Jean-Louis Deneubourg. Response threshold reinforcement and division of labour in insect societies, 1998.

[15] AlanF.T. Winfield. Towards an engineering science of robot foraging. In Hajime Asama, Haruhisa Kurokawa, Jun Ota, and Kosuke Sekiyama, editors, *Distributed Autonomous Robotic Systems 8*, pages 185–192. Springer Berlin Heidelberg, 2009. ISBN 978-3-642-00643-2. doi: 10.1007/978-3-642-00644-9_16. URL http://dx.doi.org/10.1007/978-3-642-00644-9_16.

[16] Shervin Nouyan, Alexandre Campo, and Marco Dorigo. Path formation in a robot swarm – self-organized strategies to find your way home, 2008.

[17] Thomas Halva Labella, Marco Dorigo, and Jean-Louis Deneubourg. Division of labor in a group of robots inspired by ants' foraging behavior. *TAAS*, 1(1):4–25, 2006. URL http://dblp.uni-trier.de/db/journals/taas/taas1.html#LabellaDD06.

[18] Wenguo Liu 0001, Alan F. T. Winfield, Jin Sa, Jie Chen 0003, and LiHua Dou. Towards energy optimization: Emergent task allocation in a swarm of foraging robots. *Adaptive Behaviour*, 15(3):289–305, 2007. URL http://dblp.uni-trier.de/db/journals/adb/adb15.html#0001WS0D07.

[19] O Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *Int. J. Rob. Res.*, 5(1):90–98, April 1986. ISSN 0278-3649. doi: 10.1177/027836498600500106. URL http://dx.doi.org/10.1177/027836498600500106.

[20] John H. Reif and Hongyan Wang. Social potential fields: A distributed behavioral control for autonomous robots. *Robotics and Autonomous Systems*, 27(3):171–194, 1999. URL http://dblp.uni-trier.de/db/journals/ras/ras27.html#ReifW99.

[21] William M. Spears, Diana F. Spears, Jerry C. Hamann, and Rodney Heil. Distributed, physics-based control of swarms of vehicles. *Auton. Robots*, 17(2-3):137–162, 2004. doi: 10.1023/B:AURO.0000033970.96785.f2. URL http://dx.doi.org/10.1023/B:AURO.0000033970.96785.f2.

[22] R.S. Sutton and A.G. Barto. *Reinforcement learning: An introduction*, volume 116. Cambridge Univ Press, 1998.

[23] Leslie Pack Kaelbling, Michael L. Littman, and Andrew P. Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996. URL http://people.csail.mit.edu/lpk/papers/rl-survey.ps.

[24] Liviu Panait and Sean Luke. Cooperative multi-agent learning: The state of the art. *Autonomous Agents and Multi-Agent Systems*, 11(3):387–434, November 2005. ISSN 1387-2532. doi: 10.1007/s10458-005-2631-2. URL http://dx.doi.org/10.1007/s10458-005-2631-2.

[25] M. J. Mataric. Reinforcement learning in the multi-robot domain. *Autonomous Robots*, 4:73–83, 1997.

[26] Maja J. Matari'c. Using communication to reduce locality in distributed multi-agent learning. *Journal of Experimental and Theoretical Artificial Intelligence*, 10:357–369, 1998. URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.40.3185.

[27] David Wolpert and Kagan Tumer. An introduction to collective intelligence. *CoRR*, cs.LG/9908014, 1999. URL http://dblp.uni-trier.de/db/journals/corr/corr9908.html#cs-LG-9908014.

[28] M. Riedmiller, T. Gabel, R. Hafner, and S. Lange. Reinforcement learning for robot soccer. *Autonomous Robots*, 27(1):55–74, 2009.

[29] Shivaram Kalyanakrishnan and Peter Stone. Batch reinforcement learning in a complex domain. In *Proceedings of the 6th International Joint Conference on Autonomous Agents and Multiagent Systems*, AAMAS '07, pages 94:1–94:8, New York, NY, USA, 2007. ACM. ISBN 978-81-904262-7-5. doi: 10.1145/1329125.1329241. URL http://doi.acm.org/10.1145/1329125.1329241.

[30] Leslie Pack Kaelbling, Michael L. Littman, and Anthony R. Cassandra. Planning and acting in partially observable stochastic domains. *Artificial Intelligence*, 101:99–134, 1998. URL http://people.csail.mit.edu/lpk/papers/aij98-pomdp.pdf.

[31] David E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc., New York, NY, USA, 1989.

[32] J. H. Holland. *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence*. The MIT Press, 1992.

[33] Stefano Nolfi and Dario Floreano. *Evolutionary Robotics: The Biology,Intelligence,and Technology*. MIT Press, Cambridge, MA, USA, 2000. ISBN 0262140705.

[34] Gianluca Baldassarre, Vito Trianni, Michael Bonani, Francesco Mondada, Marco Dorigo, and Stefano Nolfi. Self-organized coordinated motion in groups of physically connected robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(1):224–239, 2007. doi: 10.1109/TSMCB.2006.881299. URL http://doi.ieeecomputersociety.org/10.1109/TSMCB.2006.881299.

[35] Roderich Gro and Marco Dorigo. Evolution of solitary and group transport behaviors for autonomous robots capable of self-assembling. *Adaptive Behaviour*, 16(5):285–305, 2008. URL http://dblp.uni-trier.de/db/journals/adb/adb16.html#GrossD08.

[36] Valerio Sperati, Vito Trianni, and Stefano Nolfi. Evolving coordinated group behaviours through maximization of mean mutual information. *Swarm Intelligence*, 2(2-4):73–95,

December 2008. ISSN 1935-3812. doi: 10.1007/s11721-008-0017-1. URL http://dx.doi.org/10.1007/s11721-008-0017-1.

[37] Christos Ampatzis, Elio Tuci, Vito Trianni, Anders Lyhne Christensen, and Marco Dorigo. Evolving self-assembly in autonomous homogeneous robots: Experiments with two physical robots. *Artif. Life*, 15(4):465–484, October 2009. ISSN 1064-5462. doi: 10.1162/artl.2009.Ampatzis.013. URL http://dx.doi.org/10.1162/artl.2009.Ampatzis.013.

[38] Giovanni Pini and Elio Tuci. On the design of neuro-controllers for individual and social learning behaviour in autonomous robots: an evolutionary approach. *Connect. Sci.*, 20 (2&3):211–230, 2008. doi: 10.1080/09540090802092014. URL http://dx.doi.org/10.1080/09540090802092014.

[39] Vito Trianni and Marco Dorigo. Self-organisation and communication in groups of simulated and physical robots. *Biol. Cybern.*, 95(3):213–231, August 2006. ISSN 0340-1200. doi: 10.1007/s00422-006-0080-x. URL http://dx.doi.org/10.1007/s00422-006-0080-x.

[40] Elio Tuci, Vito Trianni, and Marco Dorigo. 'Feeling' the flow of time through sensorimotor co-ordination. *Connection Science*, 16(4):301–324, December 2004. ISSN 0954-0091. doi: 10.1080/09540090412331314740. URL http://dx.doi.org/10.1080/09540090412331314740.

[41] Markus Waibel, Laurent Keller, and Dario Floreano. Genetic team composition and level of selection in the evolution of cooperation. *IEEE Trans. Evolutionary Computation*, 13 (3):648–660, 2009. URL http://dblp.uni-trier.de/db/journals/tec/tec13.html#WaibelKF09.

[42] Suranga Hettiarachchi. *Distributed Evolution for Swarm Robotics*. PhD thesis, University of Wyoming, 2007. An optional note.

[43] Terrence L. Fine. *Feedforward neural network methodology*. Statistics for engineering and information science. Springer, New York, 1999. ISBN 0-387-98745-2. URL http://opac.inria.fr/record=b1095172.

[44] R. Beer and J. Gallagher. Evolving dynamic neural networks for adaptive behavior. *Adaptive Behavior*, 1:91–122, 1992.

[45] Jeffrey L. Elman. Finding structure in time. *COGNITIVE SCIENCE*, 14(2):179–211, 1990.

[46] Avi Rosenfeld, Gal A. Kaminka, Sarit Kraus, and Onn Shehory. A study of mechanisms for improving robotic group performance. *Artif. Intell.*, 172(6-7):633–655, April 2008. ISSN 0004-3702. doi: 10.1016/j.artint.2007.09.008. URL http://dx.doi.org/10.1016/j.artint.2007.09.008.

[47] J. Pugh and A. Martinoli. Parallel learning in heterogeneous multi-robot swarms. In *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, pages 3839–3846, Sept 2007. doi: 10.1109/CEC.2007.4424971.

[48] Alcherio Martinoli, K. Easton, and William Agassounon. Modeling Swarm Robotic Systems: A Case Study in Collaborative Distributed Manipulation. *International Journal of Robotics Research*, 23(4):415–436, 2004. URL http://www.ijrr.org/contents/23_04/abstract/415.html.

[49] J. Brian Lee and Ronald C. Arkin. Adaptive multi-robot behavior via learning momentum. In *IROS*, pages 2029–2036. IEEE, 2003. ISBN 0-7803-7860-1. URL http://dblp.uni-trier.de/db/conf/iros/iros2003.html#LeeA03.

[50] Russ Abbott. Emergence explained: Abstractions: Getting epiphenomena to do real work. *Complexity*, 12(1):13–26, 2006. doi: 10.1002/cplx.20146. URL http://dx.doi.org/10.1002/cplx.20146.

[51] Martin Friedmann. *Simulation of Autonomous Robot Teams With Adaptable Levels of Abstraction*. PhD thesis, Technische Universitt Darmstadt, Nov. 30 2009. URL http://tuprints.ulb.tu-darmstadt.de/2113/.

[52] Michael Rubenstein, Alejandro Cornejo, and Radhika Nagpal. Programmable self-assembly in a thousand-robot swarm. *Robotics*, 345(6198):795–799, 2014. doi: 10.1126/science.1254295.

[53] James Kramer and Matthias Scheutz. Development environments for autonomous mobile robots: A survey. *Auton. Robots*, 22(2):101–132, February 2007. ISSN 0929-5593. doi: 10.1007/s10514-006-9013-8. URL http://dx.doi.org/10.1007/s10514-006-9013-8.

[54] Richard Vaughan. Massively multi-robot simulation in stage, 2008.

[55] Carlo Pinciroli, Vito Trianni, Rehan O'Grady, Giovanni Pini, Arne Brutschy, Manuele Brambilla, Nithin Mathews, Eliseo Ferrante, Gianni Di Caro, Frederick Ducatelle, Mauro Birattari, Luca Maria Gambardella, and Marco Dorigo. ARGoS: a modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295, 2012.

[56] A. Martinoli, A.J. Ijspeert, and F. Mondada. Understanding collective aggregation mechanisms: From probabilistic modelling to experiments with real robots. *Robotics and Autonomous Systems*, 29:51–63, 1999.

[57] Kristina Lerman and Aaram Galstyan. Mathematical Model of Foraging in a Group of Robots: Effect of Interference. *Autonomous Robots*, 13:127–141, 2002.

[58] Alexandre Campo and Marco Dorigo. Efficient Multi-foraging in Swarm Robotics. In Almeida, Luis M. Rocha, Ernesto Costa, Inman Harvey, and António Coutinho, editors, *Advances in Artificial Life*, volume 4648 of *Lecture Notes in Computer Science*, pages 696–705, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. ISBN 978-3-540-74912-7. doi: 10.1007/978-3-540-74913-4_70. URL http://dx.doi.org/10.1007/978-3-540-74913-4_70.

[59] Alan F. T. Winfield, Wenguo Liu 0001, Julien Nembrini, and Alcherio Martinoli. Modelling a wireless connected swarm of mobile robots. *Swarm Intelligence*, 2(2-4): 241–266, 2008. URL http://dblp.uni-trier.de/db/journals/swarm/swarm2.html#WinfieldLNM08.

[60] Wenguo Liu 0001 and Alan F. T. Winfield. Modeling and optimization of adaptive foraging in swarm robotic systems. *I. J. Robotic Res.*, 29(14):1743–1760, 2010. URL http://dblp.uni-trier.de/db/journals/ijrr/ijrr29.html#LiuW10.

[61] Rehan O'Grady, Carlo Pinciroli, Anders Christensen, and Marco Dorigo. Supervised group size regulation in a heterogeneous robotic swarm. In *9th Conference on Autonomous Robot Systems and Competitions, Robótica 2009*, pages 113–119. IPCB-Instituto Politcnico de Castelo Branco, Castelo Branco, Portugal, 2009.

[62] Masao Kubo, Hiroshi Sato, Tatsuro Yoshimura, Akihiro Yamaguchi, and Takahiro Tanaka. Multiple targets enclosure by robotic swarm. *Robotics and Autonomous Systems*, 62(9): 1294 – 1304, 2014. ISSN 0921-8890. doi: http://dx.doi.org/10.1016/j.robot.2014.03.014. URL http://www.sciencedirect.com/science/article/pii/S092188901400058X. Intelligent Autonomous Systems.

[63] Aram Galstyan, Tad Hogg, and Kristina Lerman. Modeling and mathematical analysis of swarms of microscopic robots. In *Proceedings of IEEE Swarm Intelligence Symposium (SIS-2005), Pasadena, CA*, June 2005.

[64] Heiko Hamann and Heinz Wörn. A framework of space-time continuous models for algorithm design in swarm robotics. *Swarm Intelligence*, 2(2-4):209–239, 2008. doi: 10.1007/s11721-008-0015-3. URL http://dx.doi.org/10.1007/s11721-008-0015-3.

[65] Thomas Schmickl, Heiko Hamann, Heinz Wrn, and Karl Crailsheim. Two different approaches to a macroscopic model of a bio-inspired robotic swarm. *Robotics and Autonomous Systems*, 57(9):913–921, 2009. URL http://dblp.uni-trier.de/db/journals/ras/ras57.html#SchmicklHWC09.

[66] Spring Berman, dm M. Halsz, M. Ani Hsieh, and Vijay Kumar. Optimized stochastic policies for task allocation in swarms of robots. *IEEE Transactions on Robotics*, 25 (4):927–937, 2009. URL http://dblp.uni-trier.de/db/journals/trob/trob25.html#BermanHHK09.

[67] Spring Berman, Radhika Nagpal, and dm M. Halsz. Optimization of stochastic strategies for spatially inhomogeneous robot swarms: A case study in commercial pollination. In *IROS*, pages 3923–3930. IEEE, 2011. ISBN 978-1-61284-454-1. URL http://dblp.uni-trier.de/db/conf/iros/iros2011.html#BermanNH11.

[68] Karthik Dantu, Spring Berman, Bryan Kate, and Radhika Nagpal. A comparison of deterministic and stochastic approaches for allocating spatially dependent tasks in micro-aerial vehicle collectives. In *IROS*, pages 793–800. IEEE, 2012. ISBN 978-1-4673-1737-5. URL http://dblp.uni-trier.de/db/conf/iros/iros2012.html#DantuBKN12.

[69] Amanda Prorok, Nikolaus Correll, and Alcherio Martinoli. Multi-level spatial modeling for stochastic distributed robotic systems. *The International Journal Of Robotics Research*, 30(5):574–589, 2011. doi: 10.1177/0278364911399521.

[70] Veysel Gazi and Kevin M. Passino. Stability of a one-dimensional discrete-time asynchronous swarm. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 35(4): 834–841, 2005. URL http://dblp.uni-trier.de/db/journals/tsmc/tsmcb35.html#GaziP05.

[71] Yang Liu, K. M. Passino, and M. M. Polycarpou. Stability analysis of M-dimensional asynchronous swarms with a fixed communication topology. *Automatic Control, IEEE Transactions on*, 48(1):76–95, 2003. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1166526.

[72] V. Gazi and K. M. Passino. Stability analysis of social foraging swarms. *Trans. Sys. Man Cyber. Part B*, 34(1):2843–2853, February 2002. ISSN 1083-4419. doi: 10.1109/TSMCB.2003.817077. URL http://dx.doi.org/10.1109/TSMCB.2003.817077.

[73] Yanfei Liu and K. M. Passino. Stable social foraging swarms in a noisy environment. *IEEE Transactions on Automatic Control*, 49(1):30–44, 2004. doi: 10.1109/tac.2003.821416. URL http://dx.doi.org/10.1109/tac.2003.821416.

[74] Mac Schwager, Nathan Michael, Vijay Kumar, and Daniela Rus. Time scales and stability in networked multi-robot systems. In *ICRA*, pages 3855–3862. IEEE, 2011. URL http://dblp.uni-trier.de/db/conf/icra/icra2011.html#SchwagerMKR11.

[75] M. Ani Hsieh, Ádám Halász, Spring Berman, and Vijay Kumar. Biologically inspired redistribution of a swarm of robots among multiple sites. *Swarm Intelligence,*, 2(2-4):121 – 141, December 2008.

[76] David Payton, Mike Daily, Regina Estowski, Mike Howard, and Craig Lee. Pheromone robotics. *Auton. Robots*, 11(3):319–324, November 2001. ISSN 0929-5593. doi: 10.1023/A:1012411712038. URL http://dx.doi.org/10.1023/A:1012411712038.

[77] Hande Çelikkanat and Erol Sahin. Steering self-organized robot flocks through externally guided individuals. *Neural Computing and Applications*, 19(6):849–865, 2010. doi: 10.1007/s00521-010-0355-y. URL http://dx.doi.org/10.1007/s00521-010-0355-y.

[78] Rehan OGrady, Roderich Gro, AndersLyhne Christensen, and Marco Dorigo. Self-assembly strategies in a group of autonomous mobile robots. *Autonomous Robots*, 28 (4):439–455, 2010. ISSN 0929-5593. doi: 10.1007/s10514-010-9177-0. URL http://dx.doi.org/10.1007/s10514-010-9177-0.

[79] Scott Camazine, Nigel R. Franks, James Sneyd, Eric Bonabeau, Jean-Louis Deneubourg, and Guy Theraula. *Self-Organization in Biological Systems*. Princeton University Press, Princeton, NJ, USA, 2001. ISBN 0691012113.

[80] Maja J. Matarić Andrew Howard and Gaurav S. Sukhatme. Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem. In *Proceedings of the International Symposium on Distributed Autonomous Robotic Systems*, pages 299–308, 2002. URL http://robotics.usc.edu/publications/71/.

[81] Bang Wang, Hock-Beng Lim, and Di Ma. A survey of movement strategies for improving network coverage in wireless sensor networks. *Computer Communications*, 32(13-14): 1427–1436, 2009. URL http://dblp.uni-trier.de/db/journals/comcom/comcom32.html#WangLM09.

[82] Alan Oliveira de S, Nadia Nedjah, and Luiza de Macedo Mourelle. Distributed efficient localization in swarm robotic systems using swarm intelligence algorithms. *Neurocomputing*, 172:322 – 336, 2016. ISSN 0925-2312. doi: http://dx.doi.org/10.1016/j.neucom.2015.03.099. URL http://www.sciencedirect.com/science/article/pii/S0925231215010498.

[83] KeithJ. OHara and TuckerR. Balch. Pervasive sensor-less networks for cooperative multi-robot tasks. In Rachid Alami, Raja Chatila, and Hajime Asama, editors, *Distributed Autonomous Robotic Systems 6*, pages 305–314. Springer Japan, 2007. ISBN 978-4-431-35869-5. doi: 10.1007/978-4-431-35873-2_30. URL http://dx.doi.org/10.1007/978-4-431-35873-2_30.

[84] Gianni A. Di Caro, Frederick Ducatelle, and Luca Maria Gambardella. Wireless communications for distributed navigation in robot swarms. In *Applications of Evolutionary Computing, EvoWorkshops 2009: EvoCOMNET, EvoENVIRONMENT, EvoFIN, EvoGAMES, EvoHOT, EvoIASP, EvoINTERACTION, EvoMUSART, EvoNUM, EvoSTOC, EvoTRANSLOG, Tübingen, Germany, April 15-17, 2009. Proceedings*, pages 21–30, 2009. doi: 10.1007/978-3-642-01129-0_3. URL http://dx.doi.org/10.1007/978-3-642-01129-0_3.

[85] JoshuaP. Hecker and MelanieE. Moses. Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms. *Swarm Intelligence*, 9(1):43–70, 2015. ISSN 1935-3812. doi: 10.1007/s11721-015-0104-z. URL http://dx.doi.org/10.1007/s11721-015-0104-z.

[86] M. Dorigo, D. Floreano, L. M. Gambardella, F. Mondada, S. Nolfi, T. Baaboura, M. Birattari, M. Bonani, M. Brambilla, A. Brutschy, D. Burnier, A. Campo, A. L. Christensen,

A. Decugnière, G. Di Caro, F. Ducatelle, E. Ferrante, A. Förster, J. Guzzi, V. Longchamp, S. Magnenat, J. Martinez Gonzales, N. Mathews, M. Montes de Oca, R. O'Grady, C. Pinciroli, G. Pini, P. Rétornaz, J. Roberts, V. Sperati, T. Stirling, A. Stranieri, T. Stützle, V. Trianni, E. Tuci, A. E. Turgut, and F. Vaussard. Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*, 20(4): 1–13, 2013.

[87] F. Ducatelle, G. A. Di Caro, C. Pinciroli, F. Mondada, and L. Gambardella. Communication assisted navigation in robotic swarms: self-organization and cooperation. In *Proceedings of the 24th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4981–4988, San Francisco, USA, September 25–30, 2011.

[88] F. Ducatelle, G.A. Di Caro, C. Pinciroli, and L. Gambardella. Self-organized cooperation between robotic swarms. *Swarm Intelligence Journal*, 5(2):73–96, 2011.

[89] Bin Yang, Yongsheng Ding, Yaochu Jin, and Kuangrong Hao. Self-organized swarm robot for target search and trapping inspired by bacterial chemotaxis. *Robotics and Autonomous Systems*, 72:83 – 92, 2015. ISSN 0921-8890. doi: http://dx.doi.org/10.1016/j.robot.2015.05.001. URL http://www.sciencedirect.com/science/article/pii/S0921889015000937.

[90] Frederick Ducatelle, GianniA. Di Caro, Alexander Frster, Michael Bonani, Marco Dorigo, Stphane Magnenat, Francesco Mondada, Rehan OGrady, Carlo Pinciroli, Philippe Rtornaz, Vito Trianni, and LucaM. Gambardella. Cooperative navigation in robotic swarms. *Swarm Intelligence*, 8(1):1–33, 2014. ISSN 1935-3812. doi: 10.1007/s11721-013-0089-4. URL http://dx.doi.org/10.1007/s11721-013-0089-4.

[91] Arne Brutschy, Lorenzo Garattoni, Manuele Brambilla, Gianpiero Francesca, Giovanni Pini, Marco Dorigo, and Mauro Birattari. The tam: abstracting complex tasks in swarm robotics research. *Swarm Intelligence*, 9(1):1–22, 2015. ISSN 1935-3812. doi: 10.1007/s11721-014-0102-6. URL http://dx.doi.org/10.1007/s11721-014-0102-6.

[92] Ryusuke Fujisawa, Shigeto Dobata, Ken Sugawara, and Fumitoshi Matsuno. Designing pheromone communication in swarm robotics: Group foraging behavior mediated by chemical substance. *Swarm Intelligence*, 8(3):227–246, 2014. ISSN 1935-3812. doi: 10.1007/s11721-014-0097-z. URL http://dx.doi.org/10.1007/s11721-014-0097-z.

[93] Akira Okubo. Dynamical Aspects of Animal Grouping: Swarms, Schools, Flocks, and Herds. *Advances in Biophysics*, 22:1–94, 1986.

[94] Gal A. Kaminka, Ruti Schechter-Glick, and Vladimir Sadov. Using sensor morphology for multirobot formations. *IEEE Transactions on Robotics*, 24(2):271–282, 2008. URL http://dblp.uni-trier.de/db/journals/trob/trob24.html#KaminkaSS08.

[95] J. K. Parrish, S. V. Viscido, and D. Grunbaum. Self-organized fish schools: An examination of emergent properties. *Biol. Bull.*, 202:296–305, 2002.

[96] Craig W. Reynolds. Flocks, herds and schools: A distributed behavioral model. *SIGGRAPH Comput. Graph.*, 21(4):25–34, August 1987. ISSN 0097-8930. doi: 10.1145/37402.37406. URL http://doi.acm.org/10.1145/37402.37406.

[97] Tucker R. Balch and Maria Hybinette. Social potentials for scalable multi-robot formations. In *Proceedings of the 2000 IEEE International Conference on Robotics and Automation, ICRA 2000, April 24-28, 2000, San Francisco, CA, USA*, pages 73–80, 2000. doi: 10.1109/ROBOT.2000.844042. URL http://dx.doi.org/10.1109/ROBOT.2000.844042.

[98] G. Baldassarre, S. Nolfi, and D. Parisi. Evolving mobile robots able to display collective behaviors. *Artificial Life*, 9(3):255–267, 2003.

[99] Ali Emre Turgut, Hande Çelikkanat, Fatih Gökçe, and Erol Sahin. Self-organized flocking in mobile robot swarms. *Swarm Intelligence*, 2(2-4):97–120, 2008. doi: 10.1007/s11721-008-0016-2. URL http://dx.doi.org/10.1007/s11721-008-0016-2.

[100] Eliseo Ferrante, Ali E. Turgut, Nithin Mathews, Mauro Birattari, and Marco Dorigo. Flocking in stationary and non-stationary environments: a novel communication strategy for heading alignment. In *Proceedings of the 11th international conference on Parallel problem solving from nature: Part II*, PPSN'10, pages 331–340, Berlin, Heidelberg, 2010. Springer-Verlag. ISBN 3-642-15870-6, 978-3-642-15870-4. URL http://portal.acm.org/citation.cfm?id=1887292.

[101] A. Stranieri, E. Ferrante, A. E. Turgut, V. Trianni, C. Pinciroli, M. Birattari, and M. Dorigo M. Self-Organized flocking with a heterogeneous mobile robot swarm. In T. Lenaerts, M. Giacobini, H. Bersini, P. Borgine, M. Dorigo, and R. Doursat, editors, *Proceedings of ECAL 2011*, pages 789–796, Cambridge, Massachussets, 2011. MIT Press.

[102] Eliseo Ferrante, Ali Emre Turgut, Cristin Huepe, Alessandro Stranieri, Carlo Pinciroli, and Marco Dorigo. Self-organized flocking with a mobile robot swarm: a novel motion control method. *Adaptive Behaviour*, 20(6):460–477, 2012. URL http://dblp.uni-trier.de/db/journals/adb/adb20.html#FerranteTHSPD12.

[103] C. Ronald Kube and Eric Bonabeau. Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, 30(1-2):85–101, 2000. URL http://dblp.uni-trier.de/db/journals/ras/ras30.html#KubeB00.

[104] Jianing Chen, M. Gauci, Wei Li, A. Kolling, and R. Gros. Occlusion-based cooperative transport with a swarm of miniature mobile robots. *Robotics, IEEE Transactions on*, 31 (2):307–321, April 2015. ISSN 1552-3098. doi: 10.1109/TRO.2015.2400731.

[105] E-puck. E-puck education robot, 2015. URL http://www.e-puck.org/index.php.

[106] Alexandre Campo, Shervin Nouyan, Mauro Birattari, Roderich Gro, and Marco Dorigo. Enhancing cooperative transport using negotiation of goal direction. In P.-Y. Schobbens, W. Vanhoof, and G. Schwanen, editors, *18th Belgium–Netherlands Conference on Artificial Intelligence*, pages 365–366, Namur, Belgium, 2006. University of Namur.

[107] Roderich Gross and Marco Dorigo. Towards group transport by swarms of robots. *Int. J. Bio-Inspired Comput.*, 1(1/2):1–13, January 2009. ISSN 1758-0366. doi: 10.1504/IJBIC.2009.022770. URL http://dx.doi.org/10.1504/IJBIC.2009.022770.

[108] Jean-Marc Am, Jos Halloy, Colette Rivault, Claire Detrain, and Jean Louis Deneubourg. Collegial decision making based on social amplification leads to optimal group formation. *Proceedings of the National Academy of Sciences*, 103(15):5835–5840, 2006. doi: 10.1073/pnas.0507877103. URL http://www.pnas.org/content/103/15/5835.abstract.

[109] Jr. Equations Descriptive of Fish Schools and Other Animal Aggregations. *Ecology*, 35 (3):361–370, 1954. ISSN 00129658. doi: 10.2307/1930099. URL http://dx.doi.org/10.2307/1930099.

[110] Daniel Grünbaum and Akira Okubo. Modeling social animal aggregations. *Frontiers in Theoretical Biology*, 100:296–325, 1994.

[111] R. Jeanson, C. Rivault, J. Deneubourg, S. Blanco, R. Fournier, C. Jost, and G. Theraulaz. Self-organized aggregation in cockroaches. *Animal Behaviour*, 69(1):169–180, January

2005. ISSN 00033472. doi: 10.1016/j.anbehav.2004.02.009. URL http://dx.doi.org/10.1016/j.anbehav.2004.02.009.

[112] Simon Garnier, Christian Jost, Raphal Jeanson, Jacques Gautrais, Masoud Asadpour, Gilles Caprari, and Guy Theraulaz. Aggregation behaviour as a source of collective decision in a group of cockroach-like-robots. In Mathieu S. Capcarrre, Alex Alves Freitas, Peter J. Bentley, Colin G. Johnson, and Jon Timmis, editors, *ECAL*, volume 3630 of *Lecture Notes in Computer Science*, pages 169–178. Springer, 2005. ISBN 3-540-28848-1. URL http://dblp.uni-trier.de/db/conf/ecal/ecal2005.html#GarnierJJGACT05.

[113] Onur Soysal and Erol Sahin. A macroscopic model for self-organized aggregation in swarm robotic systems. In Erol Sahin, William M. Spears, and Alan F. T. Winfield, editors, *Swarm Robotics*, volume 4433 of *Lecture Notes in Computer Science*, pages 27–42. Springer, 2006. ISBN 978-3-540-71540-5. URL http://dblp.uni-trier.de/db/conf/sab/sab2006sr.html#SoysalS06.

[114] V. Trianni, R. Groß, T.H. Labella, E. Şahin, and M. Dorigo. Evolving Aggregation Behaviors in a Swarm of Robots. In W. Banzhaf, T. Christaller, P. Dittrich, J. T. Kim, and J. Ziegler, editors, *Advances in Artificial Life - Proceedings of the 7th European Conference on Artificial Life (ECAL)*, volume 2801 of *Lecture Notes in Artificial Intelligence*, pages 865–874. Springer Verlag, Heidelberg, Germany, 2003.

[115] Erkin Bahçeci and Erol Sahin. Evolving aggregation behaviors for swarm robotic systems: a systematic case study. In *2005 IEEE Swarm Intelligence Symposium, SIS 2005, Pasadena, California, USA, June 8-10, 2005*, pages 333–340, 2005. doi: 10.1109/SIS.2005.1501640. URL http://dx.doi.org/10.1109/SIS.2005.1501640.

[116] Meinhardt H. Models of biological pattern formation: from elementary steps to the organization of embryonic axes. *Current topics in developmental biology*, 81:1–63, 2008. URL http://www.ncbi.nlm.nih.gov/pubmed/18023723.

[117] A.V. Getling. *Rayleigh-Bénard Convection: Structures and Dynamics*. Advanced series in nonlinear dynamics. World Scientific, 1998. ISBN 9789810226572. URL https://books.google.com.eg/books?id=a_43hQr33HcC.

[118] J. S. Langer. Instabilities and pattern formation in crystal growth. *Reviews of Modern Physics*, 52:1+, 1980.

[119] B. Varghese and G. T. McKee. A review and implementation of swarm pattern formation and transformation models. *International Journal of Intelligent Computing and Cybernetics*, 2(4):786 – 817, 2009. URL http://centaur.reading.ac.uk/15404/.

[120] Erkin Bahceci, Onur Soysal, and Erol Sahin. A review: Pattern formation and adaptation in multi-robot systems. Technical Report CMU-RI-TR-03-43, Robotics Institute, Pittsburgh, PA, October 2003.

[121] W.M. Spears and D.F. Spears. *Physicomimetics: Physics-Based Swarm Intelligence*. Springer Berlin Heidelberg, 2014. ISBN 9783642448638. URL https://books.google.com.eg/books?id=RJXrngEACAAJ.

[122] Brian Shucker, Todd D. Murphey, and John K. Bennett. Convergence-preserving switching for topology-dependent decentralized systems. *IEEE Transactions on Robotics*, 24(6): 1405–1415, 2008. URL http://dblp.uni-trier.de/db/journals/trob/trob24.html#ShuckerMB08.

[123] Brian Shucker and JohnK. Bennett. Scalable control of distributed robotic macrosensors. In Rachid Alami, Raja Chatila, and Hajime Asama, editors, *Distributed Autonomous Robotic Systems 6*, pages 379–388. Springer Japan, 2007. ISBN 978-4-431-35869-5. doi: 10.1007/978-4-431-35873-2_37. URL http://dx.doi.org/10.1007/978-4-431-35873-2_37.

[124] Paola Flocchini, Giuseppe Prencipe, Nicola Santoro, and Peter Widmayer. Arbitrary pattern formation by asynchronous, anonymous, oblivious robots. *Theor. Comput. Sci.*, 407(1-3):412–447, 2008. URL http://dblp.uni-trier.de/db/journals/tcs/tcs407.html#FlocchiniPSW08.

[125] Deneubourg, S. Aron, S. Goss, and J. M. Pasteels. The self-organizing exploratory pattern of the argentine ant. *Journal of Insect Behavior*, 3(2):159–168, March 1990. ISSN 0892-7553. doi: 10.1007/bf01417909. URL http://dx.doi.org/10.1007/bf01417909.

[126] Shervin Nouyan, Roderich Gross, Michael Bonani, Francesco Mondada, and Marco Dorigo. Teamwork in self-organized robot colonies. *IEEE Trans. Evolutionary Computation*, 13 (4):695–711, 2009. URL http://dblp.uni-trier.de/db/journals/tec/tec13.html#NouyanGBMD09.

[127] Paul M. Maxim, William Spears, and Diana Spears. Robotic chain formations. *Joint Ground Robotics Enterprise Program*, pages 19–24, 2009. doi: 10.3182/

20091006-3-US-4006.00004. URL http://www.ifac-papersonline.net/Detailed/40498.html.

[128] Valerio Sperati, Vito Trianni, and Stefano Nolfi. Self-organised path formation in a swarm of robots. *Swarm Intelligence*, 5(2):97–119, 2011. doi: 10.1007/s11721-011-0055-y. URL http://dx.doi.org/10.1007/s11721-011-0055-y.

[129] C. Anderson, G. Theraulaz, and J. L. Deneubourg. Self-assemblages in insect societies. *Insectes Sociaux*, 49(2):99–110, May 2002. ISSN 0020-1812. doi: 10.1007/s00040-002-8286-y. URL http://dx.doi.org/10.1007/s00040-002-8286-y.

[130] M. Dorigo, E. Tuci, V. Trianni, R. Groß, S. Nouyan, C. Ampatzis, T. H. Labella, R. O'Grady, M. Bonani, and F. Mondada. SWARM-BOT: Design and implementation of colonies of self-assembling robots. In G. Y. Yen and D. B. Fogel, editors, *Computational Intelligence: Principles and Practice*, pages 103–135. IEEE Computational Intelligence Society, NY, 2006.

[131] Anders Lyhne Christensen, Rehan O'Grady, and Marco Dorigo. Swarmorph-script: a language for arbitrary morphology generation in self-assembling robots. *Swarm Intelligence*, 2(2-4):143–165, 2008. doi: 10.1007/s11721-008-0012-6. URL http://dx.doi.org/10.1007/s11721-008-0012-6.

[132] A. L. Christensen, R. O'Grady, and M. Dorigo. From fireflies to fault-tolerant swarms of robots. *IEEE Transactionso on Evolutionary Computation*, 13:754–766, 2009.

[133] Francesco Mondada, Michael Bonani, Andr Guignard, Stphane Magnenat, Christian Studer, and Dario Floreano. Superlinear physical performances in a swarm-bot. In Mathieu S. Capcarrre, Alex Alves Freitas, Peter J. Bentley, Colin G. Johnson, and Jon Timmis, editors, *ECAL*, volume 3630 of *Lecture Notes in Computer Science*, pages 282–291. Springer, 2005. ISBN 3-540-28848-1. URL http://dblp.uni-trier.de/db/conf/ecal/ecal2005.html#MondadaBGMSF05.

[134] Marco Dorigo. Swarm-bots: Swarms of self-assembling artifacts, 2015. URL http://www.swarm-bots.org/.

[135] Paul Levi and Serge Kernbach. *Symbiotic Multi-Robot Organisms - Reliability, Adaptability, Evolution.*, volume 7 of *Cognitive Systems Monographs*. Springer, 2010. ISBN 978-3-642-11692-6.

[136] Nithin Mathews, Anders Lyhne Christensen, Rehan O'Grady, and Marco Dorigo. Spatially targeted communication and self-assembly. In *IROS*, pages 2678–2679. IEEE, 2012. ISBN 978-1-4673-1737-5. URL http://dblp.uni-trier.de/db/conf/iros/iros2012.html#MathewsCOD12.

[137] Jannik Berg and Camilla Haukenes Karud. Swarm intelligence in bio-inspired robotics. Masters thesis, Norwegian University of Science and Technology, Trondheim, Norway, 2011. URL http://brage.bibsys.no/xmlui/handle/11250/252512.

[138] Joachim Halvorsen and Hege Beate Seilen. Self-assembling to improve performance in swarm robotics. Masters thesis, Norwegian University of Science and Technology, Trondheim, Norway, 2014. URL http://brage.bibsys.no/xmlui/handle/11250/253810.

[139] ID Couzin, J Krause, NR Franks, and SA Levin. Effective leadership and decision-making in animal groups on the move. *Nature*, 433 (7025):513 – 516, 2005. doi: 10.1038/nature03236. Publisher: Nature Publishing Group.

[140] Jan Wessnitzer and Chris Melhuish. Collective decision-making and behaviour transitions in distributed ad hoc wireless networks of mobile robots: Target-hunting. In Wolfgang Banzhaf, Thomas Christaller, Peter Dittrich, Jan T. Kim, and Jens Ziegler, editors, *ECAL*, volume 2801 of *Lecture Notes in Computer Science*, pages 893–902. Springer, 2003. ISBN 3-540-20057-6. URL http://dblp.uni-trier.de/db/conf/ecal/ecal2003.html#WessnitzerM03.

[141] Simon Garnier, Jacques Gautrais, Masoud Asadpour, Christian Jost, and Guy Theraulaz. Self-Organized Aggregation Triggers Collective Decision Making in a Group of Cockroach-Like Robots. *Adaptive Behavior - Animals, Animats, Software Agents, Robots, Adaptive Systems*, 17(2):109–133, 2009.

[142] Alexandre Campo, S. Garnier, Olivier Ddriche, Mouhcine Zekkri, and Marco Dorigo. Self-organized discrimination of resources. *PLoS ONE*, 6(5), 2011. doi: http://dx.plos.org/10.1371/journal.pone.0019888.

[143] Gianpiero Francesca, Manuele Brambilla, Vito Trianni, Marco Dorigo, and Mauro Birattari. Analysing an evolved robotic behaviour using a biological model of collegial decision making. In Tom Ziemke, Christian Balkenius, and John Hallam, editors, *SAB*, volume 7426 of *Lecture Notes in Computer Science*, pages 381–390. Springer, 2012.

ISBN 978-3-642-33092-6. URL http://dblp.uni-trier.de/db/conf/sab/sab2012.html#FrancescaBTDB12.

[144] lvaro Gutirrez, Alexandre Campo, Flix Monasterio-Huelin, Luis Magdalena, and Marco Dorigo. Collective decision-making based on social odometry. *Neural Computing and Applications*, 19(6):807–823, 2010. URL http://dblp.uni-trier.de/db/journals/nca/nca19.html#GutierrezCMMD10.

[145] Chris A. C. Parker and Hong Zhang. Biologically inspired collective comparisons by robotic swarms. *I. J. Robotic Res.*, 30(5):524–535, 2011. URL http://dblp.uni-trier.de/db/journals/ijrr/ijrr30.html#ParkerZ11.

[146] Alexander Scheidler. Dynamics of majority rule with differential latencies. *Physical Review E*, 83(3):031116+, March 2011. doi: 10.1103/physreve.83.031116. URL http://dx.doi.org/10.1103/physreve.83.031116.

[147] M. A. Montes de Oca, E. Ferrante, A. Scheidler, C. Pinciroli, M. Birattari, and M. Dorigo. Majority-rule opinion dynamics with differential latency: A mechanism for self-organized collective decision-making. *Swarm Intelligence*, 5(3–4):305–327, 2011.

[148] Gene E. Robinson. Regulation of division of labor in insect societies. *Annual Review of Entomology*, 37(1):637–665, 1992. doi: 10.1146/annurev.en.37.010192.003225. URL http://dx.doi.org/10.1146/annurev.en.37.010192.003225. PMID: 1539941.

[149] M. J. B. Krieger and J. B. Billeter. The call of duty: self organised task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems*, 30:65–84, 2000.

[150] William Agassounon and Alcherio Martinoli. Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems. In *The First International Joint Conference on Autonomous Agents & Multiagent Systems, AAMAS 2002, July 15-19, 2002, Bologna, Italy, Proceedings*, pages 1090–1097, 2002. doi: 10.1145/545056.545077. URL http://doi.acm.org/10.1145/545056.545077.

[151] S. Yun, M. Schwager, and D. Rus. Coordinating construction of truss structures using distributed equal-mass partitioning. In *Proc. of the 14th International Symposium of Robotics Research (ISRR 09)*, pages 607–623, Aug. 31–Sept. 3 2009.

[152] Giovanni Pini, Arne Brutschy, Mauro Birattari, and Marco Dorigo. Interference reduction through task partitioning in a robotic swarm - or: "don't you step on my blue suede shoes!". In Joaquim Filipe, Juan Andrade-Cetto, and Jean-Louis Ferrier, editors, *ICINCO-RA*, pages 52–59. INSTICC Press, 2009. ISBN 978-989-674-000-9. URL http://dblp.uni-trier.de/db/conf/icinco/icinco2009-ra.html#PiniBBD09.

[153] dm M. Halsz, Yanting Liang, M. Ani Hsieh, and Hong-Jian Lai. Emergence of specialization in a swarm of robots. In Alcherio Martinoli, Francesco Mondada, Nikolaus Correll, Grgory Mermoud, Magnus Egerstedt, M. Ani Hsieh, Lynne E. Parker, and Kasper Sty, editors, *DARS*, volume 83 of *Springer Tracts in Advanced Robotics*, pages 403–416. Springer, 2012. ISBN 978-3-642-32722-3. URL http://dblp.uni-trier.de/db/conf/dars/dars2010.html#HalaszLHL10.

[154] Rafael Mathias de Mendona, Nadia Nedjah, and Luiza de Macedo Mourelle. Efficient distributed algorithm of dynamic task assignment for swarm robotics. *Neurocomputing*, 172: 345 – 355, 2016. ISSN 0925-2312. doi: http://dx.doi.org/10.1016/j.neucom.2015.05.117. URL http://www.sciencedirect.com/science/article/pii/S0925231215010516.

[155] M. S. Couceiro, R. P. Rocha, and N. M. F. Ferreira. A novel multi-robot exploration approach based on particle swarm optimization algorithms. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 327–332, Nov 2011. doi: 10.1109/SSRR.2011.6106751.

[156] Micael S. Couceiro, Patricia A. Vargas, Rui P. Rocha, and Nuno M.F. Ferreira. Benchmark of swarm robotics distributed techniques in a search task. *Robotics and Autonomous Systems*, 62(2):200 – 213, 2014. ISSN 0921-8890. doi: http://dx.doi.org/10.1016/j.robot.2013.10.004. URL http://www.sciencedirect.com/science/article/pii/S092188901300208X.

[157] Pololu Robotics & Electronics. Pololu 3pi robot, 2015. URL https://www.pololu.com/product/975.

[158] TableTop Robotics. The radio multi library for wixel wireless module, 2015. URL http://www.tabletoprobotics.xyz/radio_multi.html.

[159] Aaron Staranowicz and Gian Luca Mariottini. A survey and comparison of commercial and open-source robotic simulator software. In *Proceedings of the 4th International Conference*

on PErvasive Technologies Related to Assistive Environments, PETRA '11, pages 56:1–56:8, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0772-7. doi: 10.1145/2141622.2141689. URL http://doi.acm.org/10.1145/2141622.2141689.

[160] CYBERBOTICS Ltd. Webots simulator, 2015. URL https://www.cyberbotics.com/overview.

[161] Coppelia Robotics. Virtual robot experimentation platform (v-rep), 2015. URL http://www.coppeliarobotics.com/.

[162] ARM mbed. mbed lpc1768, 2015. URL https://developer.mbed.org/platforms/mbed-LPC1768/.

[163] Pololu Robotics & Electronics. Wixel programmable usb wireless module, 2015. URL https://www.pololu.com/product/1336.