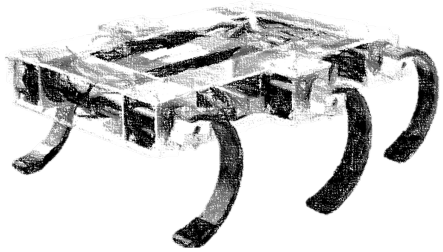
Autonomous Exploration by Cooperative Robots

Charu Agrawal



Delft University of Technology

Autonomous Exploration by Cooperative Robots

Master's Thesis in Embedded Systems

Distributed Systems Group
Faculty of Electrical Engineering, Mathematics, and Computer Science
Delft University of Technology

Charu Agrawal

August 20, 2019

Author
Charu Agrawal agrawalcharu1994@gmail.com

Title
Autonomous Exploration by Cooperative Robots

MSc presentation
August 26, 2019

Graduation Committee[1]

Dr. ir. Chris Verhoeven	Delft University of Technology
Prof. dr. ir. Dick Epema (Chair)	Delft University of Technology
Dr. ir. Massimo Mastrangeli	Delft University of Technology
Dr. ir. Stefanie Roos	Delft University of Technology

[1]In order of affiliating with the thesis

दादा के लिये,
जो गाँव से नगर आये।

To grandpa,
the man who moved to city.

Abstract

Imagine being lost in a desert with a bunch of friends, all of a sudden. Survival will be difficult. You will have mirages, distrust among friends and no means to leave landmarks on the sand. Unable to locate yourself, you will have no means to contact people with maps. The best you can do in such a situation is to stay together in the vicinity of each other and look for food and water. By staying together, you can see more and decrease faulty data; thereby increasing your survival probability.

Robots when left to explore the moon encounter the same issues. They do not have a Geo-Positioning System to locate them nor do they have a map. They have faulty sensor readings and might find it difficult to contact a human operator on earth all the time to solve issues on the moon. Since everything looks the same, there are no landmarks to memorise. As they walk around, their battery will also get exhausted. The more we equip the robot outside earth, chances of faults do not decrease, they increase. Therefore, there is a need to make primitive robots capable of autonomous exploration. We prefer sending more than one robot, inspired by the success of the collective strength of insects in harsh environments.

This thesis aims at engineering collective behaviour for a group of robots in such resource-less environments like the moon. We expect this collective behaviour to perform searching in time-critical events like earthquake-stricken areas. The thesis is designed to be implemented on legged robots called Zebros[2]. Using communication, they will collectively perform activities such that they appear as one body of tightly coupled autonomous units. We design three distinct algorithms[3] for such missions. Emergent behaviour is expected from the robots running these algorithms. The swarm should collectively choose the best among the possible options without disintegrating into subgroups.

[2] https://www.youtube.com/watch?v=Mrce6yapt4c

[3] https://www.youtube.com/watch?v=Yf3ToRk7YHY&feature=youtu.be

Preface

This document describes my MSc project on the subject of swarm robotics for Zebros. The thesis topic integrates research areas, that occupied the majority of my academic life: robotics, networking in embedded systems and distributed systems. This research aims to make it possible for robots to swarm everywhere. The research was performed at the Delft University of Technology with the collaboration of the Distributed Systems Group of the Faculty of Electrical Engineering, Mathematics, and Computer Science and the Zebro team from the TU Delft Robotics Institute.

I want to thank the many people who gave me great help during my research. In the first place my supervisors Dick and Chris, for their extensive contributions to my thesis report and research. Furthermore, I thank Massimo for information on swarm robotics and Netlogo. Without Stefanie, the last lap of the thesis would not have had been possible. The Zebro team including Otten, Hannah, Matthijs, Thijs and Maneesh for providing detailed information on Zebros. I would further like to thank Vishu, Sampi, Vijay, Natalia and Aritra for counselling me through the highs and lows of the research process.

Charu Agrawal

Delft, The Netherlands
August 20, 2019

Contents

Chapter 1

Introduction

"Collective intelligence can outperform an individual's ability."

Given a room with food crumbles spread high and low, under something, behind the curtains, ants can find it. How? They are very simple organisms. Given enough time, they can find every dead insect and every single crumble of food in the room. An ant has only 250,000 brain cells. However, 40,000 ants have the equivalent computational power to a human. Together, these 40,000 ants can do tasks which a single human being cannot. They can search large areas in parallel. If humans could walk as fast as an ant, he could compete in a horse race, which gives ants an advantage of speed. However, speed cannot be the only reason for ants to survive through the cause that extinguished dinosaurs from the earth. Along with their lightweight and small structure, their collective behaviour played a role in their survival through the Cretaceous period and since. The total weight of ants alive [45] on earth today is equal to all alive human's weight today[1]. This further implies that ants have 113[2] times more computation power than human computation on earth. Collectively, ants have better survival instincts than dinosaurs and better search capabilities compared to humans.

One of the robots being developed at the Technical University of Delft called Zebros. Zebros are aimed to achieve collective behaviour like that of ants in search tasks. Zebros are lightweight, small and can traverse most terrains. They walk while avoiding collision with moving and stationary objects. However, unlike ants, until now, Zebros lack a social sense of community which this thesis aims to provide. This thesis aims at creating co-

[1] This fact was stated in the year 2014 and is liable to change.

[2] This number is computed as follows:
Number of ants = 321,035,624,829,901,000 = x
Number of humans = 7,077,551,385 = y
Number of neurons in ants = 250,000 = a
Number of neurons in humans = 100,000,000,000 = b
Therefore,
Ratio of total neurons of all ants to neurons of all humans = (xa)/(yb) = 113

ordination between Zebros to perform search tasks collectively. These robots are given behaviours to achieve a common goal. With multiple robots collaborating to do a task, there is an increase in adaptability and flexibility, thereby increasing single robot's effectiveness. The field of engineering that deals with making robots coordinate with each other to achieve the collective intelligence of animals is called Swarm Robotics.

Multiple primitive robots are believed to outperform a single sophisticated robot [19][28][31]. The cost of building simple robots makes it feasible to increase the functionality of the swarm at the cost of losing some robots. The inspiration for 'many is better than one' comes from animals like ants and termites, who can do tasks impossible to humans. These tiny creatures do so many things in coordination, with little computational power that leaves biologist and engineers in awe.

Amongst the various institutes that work on implementing swarm behaviour in robotics, some distinguished ones are Kilobots from Harvard University [53]; e-puck mobile robots designed by Ecole Polytechnique Federale de Lausanne [23]; and Droplets of the University of Colorado Boulder [42]. However, these robot swarms are not intended to work everywhere. For instance, Kilobots can only function on smooth surfaces to allow reflection of light for communication; e-puck cannot walk in rough terrains and requires human assistance in case of failure; and droplets can only function in dark rooms. Most of the swarm robots require localisation techniques to swarm, which limits the robot to function only in a controlled lab setting. However, our robots are intended to collectively save victims in earthquake debris and explore uncharted areas of the moon. Although there exists no absolute localisation technique common to indoor, outdoor and extra-terrestrial environment setting, we aim at providing a generalised solution that works everywhere.

Across the world, engineers have implemented swarm behaviour in engineering applications like robotics, automobile and Internet of Things, inspired by biological findings of animal swarms. However, engineering capabilities do not go hand in hand with biological findings. Therefore, it is challenging to pick the right biological strategies and to define our problem statement in coherence with the present state of technology.

1.1 Problem statement

The problem statement for this thesis emerged from the need of swarming Zebros everywhere [27]. The ambition of the Zebro team is to create the swarming mechanism such that the Zebros utilise the benefits of collective intelligence using communication. Consequently, as a part of fulfilling the ambition, this thesis aims at providing collective decision-making support for ground robots that functions independent of the topology and environ-

ment. In this thesis, we engineer a generic swarm behaviour that is not dependent on the resources external to the swarm.

Space is the limit for swarming Zebros. An exciting problem statement is to be able to swarm Zebros on the moon[1]. If they can swarm on moon, they can swarm anywhere on earth, in fact in a more assisted way. Swarming on moon imposes constraints on the design and functionality of the swarm. Exploring an uncharted area on the moon eliminates the techniques that involve artificial setup. Unavailability of maps and GPS[3] on the moon, eliminates the use of these common localisation techniques. On the moon, communication via sound is not possible as sound does not penetrate in vacuum[4]. Communication using light requires the robots to be in line of sight, which is not feasible on the rough terrain of the moon. Online assistance from the earth is neither feasible and restrains the robots to practice complete autonomity. In the absence of a service centre on the moon, we need redundant hardware on the robots. Addition of each piece of hardware drastically increases the overall cost of the robot. Therefore, we need to keep the robots and the algorithms as primitive as possible. The mentioned techniques are commonly used by swarm roboticists, but we can not use them.

The surface of the moon is rough, so are the locomotion capabilities of Zebros. Zebros can traverse uneven terrain and climb heights up to 8 cm. Since they topple over themselves and slip repeatedly while traversing uneven ground, their motion is not predictable. Therefore, we can not calibrate their steps/ distance. This makes it cumbersome to localise using their walking distance, or make them move in a geometric formation.

In the absence of external help, we exploit local interactions between the different robots. We need a method of interaction such that it works everywhere and is independent of an external help. We need to engineer the behaviour of these individual robots such that they depend on each other and not on external controllers or devices.

With no central source of information, it is a challenging proposition to decide at a local level, about the global state of the whole swarm. A robot needs to predict the state of the whole swarm, with any one having complete information. Moreover, the swarm functioning should also be immune to failing robots within the swarm. Since the robots can get lost or dysfunctional during the mission, the algorithms should be run-time scalable.

Despite enormous literature on swarm robotics and swarm intelligence, there

[3]Global Positioning System

[4]moon's atmosphere is nearly vacuum [2]

are more "emerging behaviour" experiments than swarm robot experiments. However, this thesis provides another simulation displaying emergent behaviour[5] that is promising to be implemented in robots. After all, mimicking the millions of years of the natural evolution of ants will take quite some time. In this thesis, we propose three research questions listed as follows and we will seek answers to them:

RQ1 How can a group of autonomous legged robots collectively traverse unknown and unbounded areas?

RQ2 How can consensus be achieved by a group of autonomous robots without having a global overview?

RQ3 How to quantify the emergent behaviour of autonomous robots?

1.2 Approach

Our main intent with **RQ1** is to engineer the emergent behaviour that captures the requirements for Zebro missions. Unable to directly use any of the previously engineered swarm robotics technique, we derive inspirations from the swarm intelligence and multi-agent systems to formulate an implementable emergent behaviour.

The purpose of **RQ2** is to aid **RQ1** by proposing distributed algorithms. In the absence of a global observer, consensus can only be achieved with local interactions. The algorithms attempt to resolve disputes due to multiple opinions between the robots. This question has three different aspects that need to be answered: method of information dissemination, achieving consensus by decentralised local interactions and realisation by the robots about the termination of the decision making process.

For the research focus intended by **RQ3**, we need to quantify the local algorithms on the basis of expected global emergent behaviour. Emergent behaviour needs to be quantified such that the changes in local algorithms are captured through it.

1.3 Thesis outline and contributions

The remainder of the thesis is organised as follows. In Chapter 2, we introduce the state-of-the-art in swarm robotics and highlight their limitations.

[5]Emergent behaviour is defined as the changes in the global organisation of a system resulting from (un)defined code of conduct of its constituent subsystems.
Emergence refers to the patterns emerging at the global scale resulting due to changes occurring at the fundamental element.

Stemming from the shortcomings, we describe our approach to provide a solution for it. In Chapter 3, we propose three decision-making algorithms as a solution to the desired requirements and mission goals for Zebros. In Chapter 4, we compare the performance of the proposed algorithms in different experiments via simulations. Finally, in Chapter 5, we emphasise on the contributions and the future prospects of the algorithms.

The contributions and innovation in the thesis are the following:

1. We design a fundamental emergent behaviour for Zebro that has the potential to be developed to achieve all proposed missions of the Zebro team. We provide a swarming protocol for Zebro that equips them to autonomously conduct exploration on the moon and searching victims in disaster-stricken areas without absolute localisation. (Chapter 3)

2. We propose three decision-making distributed algorithms to equip the Zebros to provide the expected emergent behaviour. The first algorithm is a traditional swarming algorithm and is compared with two new communication-based algorithms. The three algorithms comply with the requirements of the Zebro missions. (Chapter 3)

3. We quantify the emergent behaviour of the three distributed algorithms through three performance metrics in six experimental setups. We analyse variation in the performance metrics on the agent's algorithms in different experimental settings. We also compare the algorithms with each other to deduce the best algorithm for each performance metric. We conclude with the most suitable algorithm to be sent on the moon. (Chapter 4)

Chapter 2

Swarm design in mobile robots

In this chapter, we elaborate on our problem statement. In Section 2.1, we describe the general approach followed to design swarm robotics behaviour. In Section 2.2, we introduce Zebro, its capabilities and the mission requirements. In Section 2.3, we present models for Zebros swarm inspired by the state of art. In the section, we reason for not being able to use any pre-existing design for the Zebro missions. In Section 2.4, we introduce varies disciplines of science that influenced our swarm design and how to incorporate findings from these domains into this thesis. In Section 2.5, we describe the algorithms/strategies that inspired our solution. In Section 2.6, we end the chapter with a final remark.

2.1 Swarm design approach

In this section, we describe the framework for designing a swarm. For the context of the thesis, we refer swarm robots as any set of robots with coordination capability. Depending on the robot's hardware and application requirements, the behaviour of the individuals is defined. These individual behaviours should lead to the achievement of global goals. The study of multiagent[1] interactions called swarm intelligence provides a wide variety of inspiration for swarm robotics from nature, psychology, mathematics etc.

In Figure 2.1, the system overview explains the process of establishing coordination in a given set of robots[2]. Since we already have functioning robots, we follow the bottom-up approach. The first requirement is to have mobile robots that can gather sensor data and process it. The robot is equipped with sensors required for the accomplishment of a mission. To develop coordination between them, communication is initiated among peers.

[1] Multiagent systems and swarm robotics is used interchangeably throughout the thesis.
[2] The term agent and robot are used interchangeably throughout the thesis.

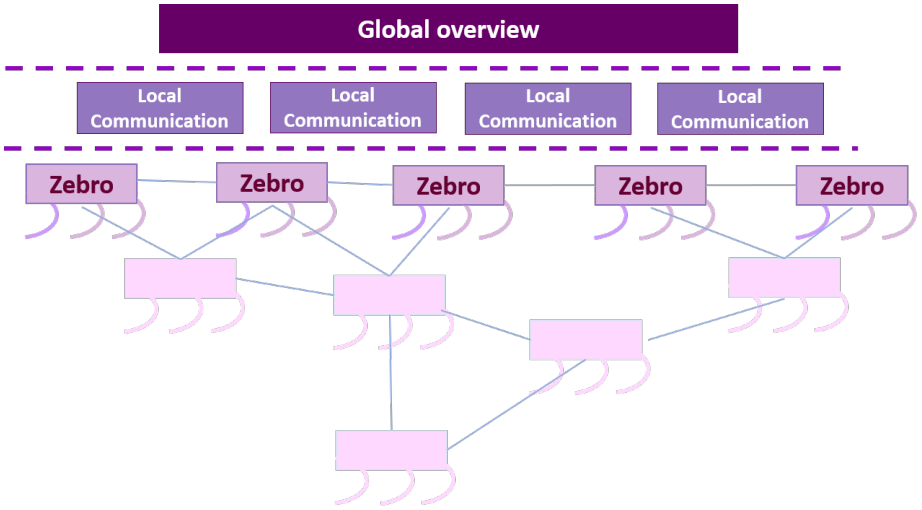


Figure 2.1: Design approach for designing swarming behaviour in a group of robots

The communication can occur as a multicast, broadcast or peer to peer communication. It can also be indirect communication like leaving physical traces, via a beacon or central source of information. Each robot becomes aware of at least its neighbours via local interactions. This attained local information is translated into global information. This translation can occur using extensive computation at each robot, or by a central source of information. The global information describes the emergent behaviour of the swarm. A swarm roboticist has to take great consideration while translating global goal to local behaviour. As local behaviour might not always be a linear function of the desired global behaviour. Moreover, extensive simulation/computation maybe required to understand the relation between local and global behaviour.

2.2 Zebro

The name, Zebro [3] is derived from the Dutch word **ze**s**b**enige **ro**bot translating to six-legged robot. These robots can walk on rough terrains attributing to its C-shaped legs that are inspired by Boston Dynamics' RHex robot [4]. The locomotive control is designed by Delft Center for Systems and Control (DCSC) at the Technical University of Delft. Zebro can walk on mud, climb stairs of certain heights and withstand up to 400 grams of weight on it. To visualise the abilities of Zebro, we recommend watching this YouTube link[3].

Zebro comes in many 'species'. Lunar Zebro [1] is the moon compatible

[3]QR code in Figure 2.2

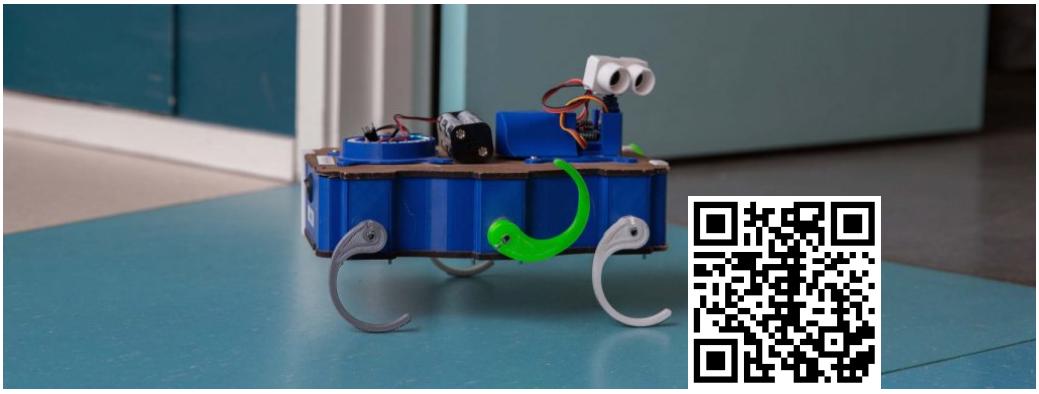


Figure 2.2: DeciZebro (Picture Credits: Lotte Hoes [5]) and QR code

version of Zebro, to be sent to the moon by 2022. Zebros are available in different sizes as well: PCB Zebro of the size of a matchstick box; KiloZebro has a length of one meter, and LightZebro made of lightweight material. We, however, are concerned with the fourth kind: DeciZebro shown in Figure 2.2. DeciZebro is of the size of an A4 size paper. The ideas and algorithms in this thesis can be directly transferred to all kinds of Zebros, other ground robots and mobile sensor systems.

2.2.1 Zebro capabilities

Zebro design provides the advantage of the robustness of movement on rough terrain. Zebros are incapable of calculating distance odometry [40] that assists majority of the state-of-the-art swarm robotics 2.3.6. Zebro is equipped with ultrasound sensors in the front for obstacle detection. It has myopic eyesight, i.e., it can sense obstacle at a distance comparable to its body length. The robot has a slow response to change in steering. Though not implemented yet, it can be programmed to take a sharp turn standing at the same place, accelerate or decelerate or made to walk backwards with the present hardware. Ultra-wide Band (UWB) radio communication is the latest addition to the Zebro design that assists communication. Appendix B provides more details on the design elements and properties of the Zebro.

2.2.2 Mission goals

Zebro team plans to do a lot more with the swarm than to perform moon exploration or search operation in a post-disaster scenario. The sensor network is proposed to find implementation in traffic and pollution observation, ship and logistics tracking, space exploration and self-deploying sensor networks [11].

In a nutshell, we cannot and do not want to hard code the robots for

a given task or location. We intend to create a simple generalised solution that works everywhere. Just like a group of ants are not bound to a location but can freely perform search tasks anywhere. Therefore, we target autonomous exploration in unbounded, uncontrolled and unknown areas without artificial environmental setting. Similar to ants, we aim to achieve a fully distributed, independent and decentralised system.

One important requirement of this study is the ability to visualise and simulate the robot coordination mission concept in realistic settings and the moon environment. That ability is key to **evaluate the algorithm performance, understand the impact of autonomy, explore different coordination approaches, design options, mission settings, and environment configurations** [58].

2.3 State of the art

In this section, we investigate the complex systems[4] that are analogous to our system. Therefore, while focusing on state of the art of swarm robotics, we provide references to emergent behaviour observed in psychology, networked embedded systems and biological swarms. We aim to adopt these models in Zebro swarm. The models are also improvised to comply with the requirements of the system. Until this thesis, the combination of our mission with such legged robots is not addressed by the swarm robotics community. Although the present state of the art in swarm robotics does not provide a direct implementation in Zebros, but does influence our final swarm design.

2.3.1 Indirect communication

Indirect communication refers to passing information without direct connection between transmitter and receiver. An intermediary is involved that holds the information for transmission. One of the most cited examples from nature used among swarm roboticist is replicating the indirect communication practice of ants using Stigmergy. Ants pass information by modifying the environment by leaving pheromone[5] trails. An ant retraces the path traversed by another ant by sensing the variation in pheromone left by other ants. The scent gets stronger as more ants traverse the same path. At any moment, an ant chooses to follow the stronger scent, as it represents a higher probability to find something of interest. Ants use cues extensively, so do dogs and baboons. Even wolves follow the scent of other wolves' urine to

[4]Complex systems are the distributed systems with no centralised control. The study of complex systems possess a more multidisciplinary approach than distributed systems.

[5]Pheromone: A chemical substance produced and released into the environment by some animals influencing the behaviour of other members of its species [6].

track them.

Unlike ants, robots cannot leave chemical pheromones for practical reasons [49]. However, a similar concept of ant stigmergy can be recreated, if robots leave physical traces as they walk. For instance, Zebros can leave magnetic pieces behind to be sensed by other robots using a magnetometer. Since this is not the same as scent, the metal pieces will create trash and will not disappear in the air. Moreover, magnetometer readings are highly influenced by the presence of magnetic materials in the vicinity. Addition of this module will also increase the cost and weight of each robot. At some point in the mission, all metal pieces will be spent.

Inspired by the chemical markers used by the insects (especially ants) for communication and coordination, the researchers exploited the notion of a "virtual pheromone". They implemented virtual pheromone by using simple beacons and directional sensors mounted on each robot [37]. Infact, the Pheromone Robotics project [49] implemented virtual pheromones via optically transmitted signals from each robot that may be propagated in a relay-type fashion. However, this approach is not suitable for rough terrains as robots might not always be in line of sight. Dr Khaliq [39] uses a floor of RFID tags that act as the Stigmergy medium. The information is stored by robots as it passes by and retrieved similarly. Pre-installation of RFID tags can not be used in our application because we explore uncharted area while avoiding previous artificial setup.

Even humans communicate by leaving milestones, as they explore new pieces of land. A roboticist can replicate the process by laying down beacons[6]. Beacons can be referred to as positioning systems used for mapping or information sources. They can be laid before starting the swarming [63] or during swarming as explained in Section 2.3.6. In Section 2.3.6, we present some models for beacon and discuss the possibility to use them in our mission.

Some swarm model like Kilobots [53], Shinerbots [44], Epuck [23], Robocup [41], TERMES [50] and LOCUST [12] use overhead cameras to inform each robot about its position with respect to the whole map. As we aim to explore without an artificial setup, we avoid the use of an overhead camera. Further, reliance on a central source of information like GPS or shared memory subsides the expected autonomity of the robot.

[6]Beacons are the systems that stand at a location to mark landmarks or relay information.

2.3.2 Direct communication

Unlike indirect communication, direct communication does not need a mediator to pass information among robots. The transmitter can convey messages to the receiver(s) when communication is established. Direct communication can further be categorised into two categories: targeted and untargeted communication. An example of targeted communication is peer-to-peer communication, i.e., message is targeted to a receiver. Untargeted communication like a broadcast or multicast, i.e., any agent who sensed the message, will receive it. A popular mechanism in swarm robotics is to make an untargeted communication. There are various means to broadcast/multicast information in swarms.

The method of communication plays an important role in strategising the swarm behaviour. It influences the choice of the swarming algorithm by effecting the means of information propagation. Some of the popular mechanisms are presented below:

1. **Firefly optimisation** is a distributed algorithm inspired by the synchronous blinking of fireflies. The fireflies appear to blink together without someone commanding them to. Each firefly synchronises its action with its neighbour(s). They synchronise their frequency of blink based on their neighbours cumulative amplitude.

   However, a firefly might have multiple neighbours pursuing different states. Therefore it takes time for the swarm to attain a stable state. Moreover, there is a delay in states, across the swarm.

   Similar to this is the clapping mechanism in human beings. As soon as someone hears a clap in an audience, she starts clapping and an avalanche of clapping starts. In the same way, the claps also come to a standstill.

2. **Elephant trumpets:** Asian Elephants communicate using a variety of sounds produced with their mouth or trunk. They make varied sounds when they greet, play, threaten, comfort, seduce or are excited. Moreover, continuous reinforcement of through sounds, keeps the herd together.

3. **Kilobots** [53] can organise around a thousand robots in a geometric formation with distributed information transfer. This robots are fragile and perform swarming on mirror-smooth surface. These ground robots communicate by reflecting light from the surface they walk on. They transmit visible light from their base, which reaches the other robot's base after being reflected from the surface.

4. **Coordination of Mars Rovers** [58]: A recently initiated mission of NASA[7] and JPL[8] is dedicated to exploring Mars caves using multiple rovers. The rovers start from the position of the Lander[9]. These Mars rovers spread out to procure scientific data while being with communication ranges of the other robots and the Lander. If a rover is lost, the fellow rovers are suggested to trace him based on its previous relative direction and distance. After the mission, these robot trace back their path to the Lander.

Approaches mentioned in points 1 and 2 use light and sound. These medium of communication are not inclusively available in our target environments. The absence of atmosphere disallows the use of sound on the moon. On the other hand, light needs line of sight which will be difficult to achieve on rough terrains. Kilobots mentioned in Approach 3, do not achieve anything more than organising themselves into geometric formations. Rather Kilobot's method of transmission is very specific and irreplaceable for exploration tasks on the moon. Not only do all surfaces reflect fine rays of light but also the base of Zebro will not always be parallel to the surface. Moreover, on rough surfaces, the achievement of the desired movement is always imprecise. Kilobots are at advantage of having to know the exact distance needed to traverse and being able to precisely travel that distance.

The project mentioned in approach 4 is very much similar to our mission. Robust swarming based on communication on the moon, with the same degree of autonomy. However, their work is also undeveloped in the simulation stage as ours. Though, they target to swarm only four robots, their algorithm is strictly distributed. By 2018, their work is also limited to the creation of the simulation infrastructure [58]. We look forward to their work, although their algorithms and details are not public.

2.3.3 Localisation techniques (Spatial orientation)

Swarms of aerial robots [63][51][30][7][13] and some ground robots [50][23] display collective behaviour in selective places only. These locations have overhead cameras, GPS systems, beacons, etc. installed for localisation. These devices provide global information. However, in for our mission statement, we will not have access to the location before sending the swarm to the sites. We will also not have a map of the moon or the earthquake-stricken area.

As seen from Table 2.1 no absolute localisation technique was encountered during this study that worked equally well in indoor, outdoor and extra-planetary locations. For instance, Global Positioning System (GPS) works

[7]National Aeronautics and Space Administration

[8]Jet Propulsion Laboratory

[9]Lander is the system that is fixed at the point of landing on the Mars.

Table 2.1: General localisation techniques

	Indoor	**Outdoor**	**Moon**
Global observer	GPS	GPS	No GPS available
Sound	Yes	Noise	Doesn't travel in vacuum
Previous setup[10]	Possible	Possible	Not possible
Line of sight (light)	Possible due flat surface	Not possible due to rough terrain	

very well outdoor but is not deployed for indoor use [35][11]. Also, to use a GPS on the moon, we need to first install it. Light-based localisation techniques require line of sight, which is unfeasible in rough terrains. Localisation on the moon can be done using maps. However, the available lunar maps do not possess the required precision of the maps [8]. Therefore, we sort to avoid absolute localisation and adhere our algorithm to relative localisation.

Without a proper map and information of the destination, sailors would navigate new locations using crude navigation tools like a compass. A compass will be helpful to desert wanderer to some extent as well. Therefore, we can use universality of the magnetic field as support to the swarm. The magnetic field's concept of attraction and repulsion is shared by the bird's swarming algorithm. The magnetic field can therefore directly use bird's flocking rules. However, magnetic material in vicinity, considerably affects the reading on the compass, especially in earthquake rubble. Nevertheless, moon has time-varying weak magnetic field. And unlike, earth's dipolar magnetic field, moon's magnetic field has fluctuating regional polarity [9]. Hence, we need another reliable source for knowing spatial orientation for navigation, thereby to localise robots.

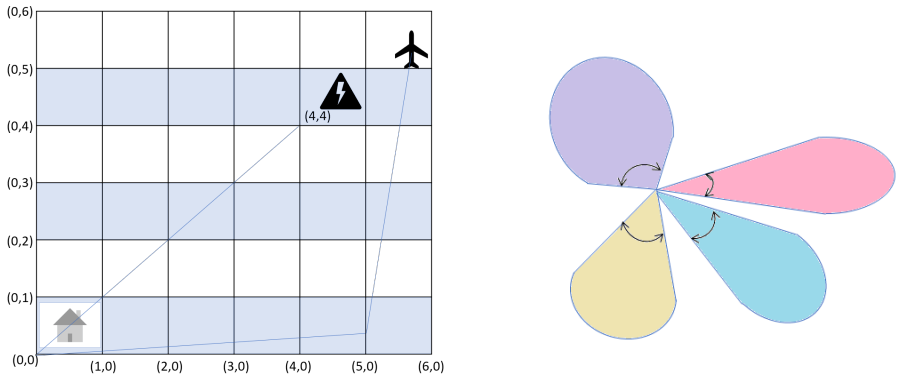
2.3.4 Mapping

Maps of earthquake affected areas can not be relied on as the infrastructure changed due to the calamity. To make the robots swarm in post-disaster scenarios they need to perform search and exploration without a map. Searching tasks are difficult by default in the absence of a map as we cannot plan the trajectory of navigation. But we study the methods that use maps and attempt to replace the need for prior mapping with mapping on the fly.

[11]Indoor localisation using GPS is not feasible due to multi-path effects from obstacles, inferences due to other wired and wireless network, and environmental factors like the mobility of people and atmospheric conditions.

SLAM (Simultaneous Localisation and Mapping) This is one of the most popular algorithms for search robots in unknown areas. This, however, is very expensive computationally, requires sophisticated sensors and accumulates errors. Though SLAM might have promising results, we do not explore this option because we want to make minimal hardware changes.

Grid forming A coordinate system can be formed by robots as they walk on unknown areas. This grid will assist these robots to find previous landmarks. Similar to the work of Coppola et al [25], in our proposed scenario shown in Figure 2.3(a), all robots start from the same place that is marked by origin(0,0). The charging station is at coordinate (4,4). To reach the charging station, the robot traverse 4 units in the west and 4 units to the north. However, due to the accumulation of odometry error with each step, the robot misses the charging station. It will not be able to locate itself with respect to the origin or the charging station. For following the grid, individual robots need to estimate distance with considerable precision or feedback to remove the errors[56].



(a) Grid forming

(b) Divide and Conquer

Figure 2.3: Proposed methods for mapping for Zebros: (a) The location is divided into grids. Robots are asked to reach a certain coordinate (say (4,4)). But the robot misses the location due to distance odometry miscalculation. (b) The robots spread in different directions, explore the locations and come back. At the origin they share the acquired information with fellow-mates.

Divide and conquer As shown in Figure 2.3(b), each of the robots starts from the centre of the swarm and moves away from the swarm. They explore certain areas and then come back to the centre. They calculate the radial parameters, devising the area coverage of the flock. Represented by the radius and angle of exploration. They share this information with all the

other robots. The other robots do not have to explore the explored areas. Now they start their mapping again in different directions. Ultimately, we will have a map of the entire place. Of course, they need to have feedback error correction techniques to compensate for odometric errors.

Ants can swarm everywhere, indoor and outdoor. They do not have a map or predefined directions for searching. They find paths by indirect communication technique as following trails of pheromones (scent) and by direct communication between each other. It will be an ideal scenario if robots swarm irrespective of location and terrain. Thus, we need to formulate ways in which robots can search without a map.

2.3.5 Landmarks

Leaving landmarks for reference while navigating is a common strategy in animals including humans. As mentioned in Section 2.3.1, Dr Zecca et al [64] use on-site Radio-frequency identification (RFID) landmarks to share information with swarming robots. They propose that primitive robots can learn information and instructions from on-site RFID landmarks to place themselves in geometric formation, synchronise with other robots in the area, or carry out cooperative tasks. We are restricted to use such landmarks because of the following reasons:

- The site under inspection is unknown and unexplored before.
- Leaving and using landmarks requires memory.
- No global map to locate the position with respect to the landmarks.

Inspired by landmark models in animal, we devised our land-marking model called NEWS model, for exploration on the moon. In Figure 2.4, the yellow robot moves towards the North, marking a yellow spot on the land. He conveys to the crowd the distance it traversed in the North. The second robot, orange robot, moves towards the north. Orange robot walks twice the distance to mark the landmark as 'N' (standing for North). The red robot then reaches the yellow landmark and walks as much distance possible towards the left, with enough battery left to go back to the point where he started (Spot 'S'). It marks the landmark as 'E' representing East. The green robot walks towards the right of the yellow spot, marking a new spot as 'W'(West). Now the other robots know the relative positioning of these marked landmarks. The robots walk around this area, searching for something of interest, say *X*. As soon as they find X, they inform the nearest landmark about the distance of X to the landmark. In this way, more robots can be sent to the location of X, if needed.

However, the model fails to account for poor odometry calculation in the legged robots as shown in Figure 2.5. Zebros have erroneous data of walked

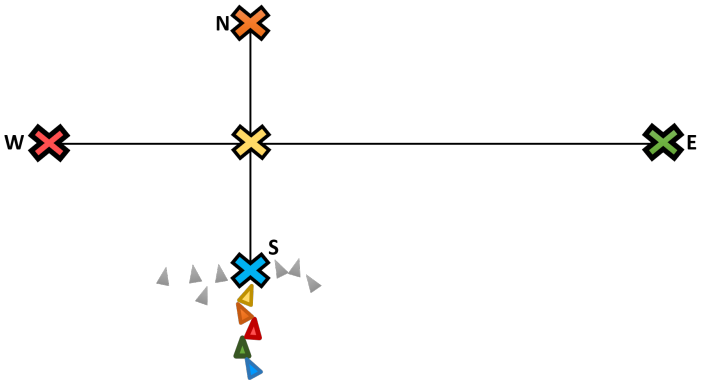


Figure 2.4: NEWS model: Landmarks are represented by crosses and robots by triangles. Three robots, coloured orange, green and red set out to set landmarks in North, East and West directions respectively, marking the landmark of their colour. These three robots, measure the distances from their origin (Landmark South) to assist the robots to reach those landmarks.

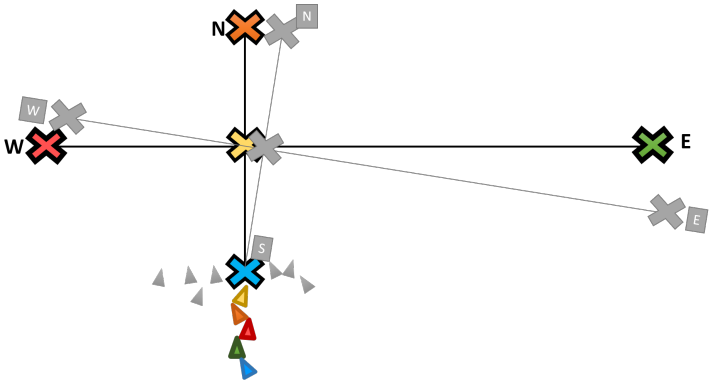


Figure 2.5: Error accumulation: When the grey robots try to find these landmarks, they are expected to get lost due to their erroneous distance calculation. The figure shows the error generated due to a wrong estimation of only one heading.

distance. The cumulative error will risk the loss of robots.

2.3.6 Distance odometry

The most common autonomous ground robots in swarm robotics are wheeled robot [23][46][38][29][47][34][43][20]. These robots use the distance odometry calculation for identifying their location in a map and with respect to each other. A legged robot like Zebro, can not measure distance like wheeled robots. No feasible method to use the odometry of legged robots was en-

countered during the study of the thesis. Another type of robots that are milestones in the history of swarm robotics are Kilobots [53] and I-Swarm robots [62]. They are stick-legged robots, that walk only on a very smooth flat surface. This smooth surface acts as a reflector for light as the medium of communication and odometry calculation. However, Zebros cannot also use such a method as its body is not always parallel to the surface. Therefore, it cannot deduce distances to other robots by the reflection of light from the surface. Moreover, we aim at using these robots on rough surfaces. Traditional swarm robotics methods fail to device a swarm design that functions equally well on different surfaces. Therefore, we need to device exploration method that does not use distance as an input parameter, or we need to find another method to calculate the distance between the robots.

2.3.7 Multi-robot formation

The multi-robot formation is a popular branch of swarm robotics. Robots organise themselves and walk in a pre-designed orientation forming structures like a chain or a semicircle. We speculate that robots walking in a formation will decrease the efficiency of the swarm in exploration tasks and avoid obstacles. Maintaining a formation is presumed to be difficult, perhaps challenging in the presence of obstacles.

Walking over stones, steps and hilly terrain is the biggest advantage of our robots, we cannot make an algorithm that overshadows its use. Consequently, we need a robust swarm design that allows the swarm to wander on all kinds of surfaces. However, it will be difficult to keep track of formation when robots as struggling with basic walking. In rough terrains, keeping the formation intact might be challenging. In case a robot is struck, it keeps everyone waiting or is lost. Therefore, walking in formation is not an ideal option for this thesis.

2.3.8 Conclusion

Similar to animals like birds, bees and ants, we want our swarm to be able to perform swarming everywhere. Therefore, we take the robot out of the laboratory setting and prohibit human supervision. We do not want to hard code the swarm behaviour for a single location or particular situation. Further, in the absence of a prior map, we cannot plan the paths or direction of movement. Dependence on previous maps is impractical in post-earthquake scenarios, as the infrastructure will be heavily damaged. No knowledge of the arena dimensions and the number of resource areas restricts the possibility of trajectory planning. Therefore, Zebro swarm should be capable of mapping, navigating, exploring and locating on the fly.

Senses of animals are stronger and well trained for the environment compared to mechanical robots. Our robots have different sensors and different

capabilities. They cannot leave a scent like ants. They cannot see and process data like animals. They cannot produce sound as a medium of passing indirect/direct information, but it won't work in the absence of atmosphere or during the daytime. They can, however, use radio communication as an electromagnetic field does not require an atmosphere and can work everywhere. They cannot measure the distance they walked from the previous location to final location, so cannot leave landmarks as birds or humans. They cannot have an elaborate discussion as we do before a democratic election, bandwidth and faulty communication being the limiting factors. Moreover, if we could send limited information via our communication medium, we could reuse some preexisting algorithm in computer science or swarm algorithms.

2.4 From swarm intelligence to swarm robotics

Leadership is often perceived as a practice by a single person called the leader. A leader is supposed to single-handedly takes decisions for the entire crowd. But it is evident from the study of Dr James Kennedy [17] that management is possible without a leader in charge. World democracies are an example of distributed leadership. Software applications like Splitwise, Wikipedia and Sharelatex are examples of distributed management. Such a trans-disciplinary study of decentralised multiagent systems to work collaboratively to achieve a task is called swarm intelligence (SI).

Swarm robotics is an application domain of swarm intelligence (SI). It involves implementing the theories and algorithm of SI into robots. Though, primarily inspired by biology, swarm intelligence umbrella's wide range of disciplines like biology, chemistry, neuroscience, psychology and cognitive science [17]. SI's study of emergent behaviour in multiagent systems can never be comprehensive. The wide range of inspirations and incongruency of these fields with robotics makes it near unachievable to find the 'best' solution.

The field of robotics has its own set of constraints, which are disjoint to biological findings. The unparallel success of the incongruent fields prohibits the direct translation of biological theories into algorithms for robots. In this thesis, we use the tools of complexity science, distributed systems and network science to bridge the gap between biological swarm and robotics. To narrow down the search space, we answer the basic questions in related disciplines to develop correlations between requirements and existing solutions.

Mathematical and theoretical biology Swarm robotics and swarm intelligence are inherently branches of biomimetics[12]. The term swarm is

[12]Biomimetics is the branch of biology that employs theoretical analysis, mathematical models and abstraction of living organisms to investigate the principles that govern the

inspired by animal swarms. The successful collectiveness of the swarm of animals contributed to a large number of algorithms in SI. Fishes, birds and ants are extensively being studied by swarm roboticist. Biologist and roboticists work together to formulate algorithms for swarm robotics. Theories of Biologists like Dr Iain Couzin [26] and Dr Thomas Seeley [54] have provided valuable counsel for this thesis. Geoecologist Hannah Härtwich contributed to the thesis with her study titled "Swarming in Nature and Robotics"[36].

Complexity science One important element of swarming is devising local rules that result in the achievement of the global goals. Complexity science deals with interactions of individual agents that lead to complex emergent behaviour.

Complexity science extends across engineering to nature via non-science disciplines like philosophy. Study of complex systems assists the understanding of the varied facets of the swarm dynamics as will be seen in Chapter 4. Tools and inspirations from complexity science assist the development of the right local rules. Some basic questions of complexity science, that are common to swarm intelligence that assists our understanding of the required swarm design are:

I What are the local and global rules robots should follow?
II What is the correlation between the local and global rules?
III What complexity parameters can be used to derive the efficiency of the swarm?
IV Which studied complex system resembles our system?

Distributed systems Swarm robotics is often termed as distributed robotics [14]. A distributed system is a collection of independent systems that appear to the observer as a single coherent system. A potential perspective to see a swarm of robots is as to visualise them as distributed systems. The common characteristics of distributed systems that are found common in a swarm of robots are scalability, asynchronous and dynamic behaviour to achieve decentralised cooperation using communication.

Dr Francesco Bullo, a prominent swarm roboticist, classified swarm robotic algorithms into three distributed algorithms [16]: flooding, leader election and averaging[13] algorithm. To reuse any existing and implementable distributed algorithms or devise one, we follow the classification by Bullo et al [16].

Agent is defined as each individual member of the swarm.
Utility is defined as the sensor values from individual members.

structure, development and behaviour of the systems [10].

[13]Also referred to as agreement/consensus algorithm

2.5 Inspirations for proposed algorithms

Various algorithms have inspired the design of the proposed algorithms. In this section, we describe the strategies that are consolidated together to result in the proposed algorithm as shown in Chapter 3.

2.5.1 Synchronising heading

Flocking algorithm [52] is used for synchronising heading of the flock with/without communication. Agents either sense or communicate the relative distance and relative heading with their neighbouring agents. Based on these two parameters the robots abide by the following three rules to avoid collisions, staying with the swarm and achieve synchronisation of heading.

- Cohesion: Agents steer towards other agents, if the distance between them is more than the minimum distance of cohesion.
- Separation: Agents steer away from other agents, whose distance is less than the maximum separation distance. This prevents the robots to collide with each other and avoid crowding.
- Alignment: To maintain the average heading of the swarm, the agents steer towards the average heading of neighbours.

Flocking simulations on Zebros were conducted and studied by previous members of the Zebro team [27][22] [48][21]. Therefore, this thesis did not make a radical shift from using the flocking algorithm in Zebro swarming.

2.5.2 Information dissemination

Gossip algorithm [15] is a distributed asynchronous algorithm for an arbitrarily connected network of nodes [32]. It is a communication protocol that applies to a system with an unknown number of nodes. Therefore, the algorithm is fault-tolerant to node failure and will adjust with the addition of agents to the swarm. Dynamic decision making is possible as the algorithm adapts to the changing weights on the nodes. Therefore, the gossip algorithm seeks to answer: "How information can be proliferated through a scalable and dynamic network?"

However, an issue with the gossip algorithm is announcing the end of the algorithm. In this dynamically changing unknown distributed network, how do agents realise the end of gossip via local interactions? The propagation is well addressed in the algorithm, but back-propagation is not defined. Moreover, there will be multiple layers of acknowledgements and back acknowledgements, how will the swarm manage multiple layers of acknowledgements?

2.5.3 Consensus

Bee's hive searching methodology is not based on absolute localisation and is decentralised with a democratic model of decision making. The homogeneous swarm is also run-time scalable, satisfying all the swarm characteristics quoted in Section 3.1.

Bees are very similar to Zebros, as they coordinate the same input parameters: relative heading and relative distance to their fellow neighbours. This striking similarity with bees inspired us to base our algorithm on bee democracy. In this section, we describe the consensus strategy of the bees and an issue with directly implementing the bee strategy.

Bee democracy

The search tasks in a bee's life involve finding nectar and a place to build a hive [33]. The hive location is of crucial importance, as it directly relates to the nectar-bearing flowers in the vicinity. Therefore, this long-term investment is given a considerable amount of time and is dispensed optimally planned resources. The plot of hive location searching algorithm closely resembles the distributed decision making strategy we want to achieve for Zebro missions. In this section, we explain the hive selection mechanism of the bees, followed by similarities of Zebro mission with bees. Furthermore, we inaugurate the different algorithms which are the prime focus of the thesis.

Bees have two teams, say searchers and non-searchers. Around 2-6% are searchers from a swarm of 4000 bees [54]. It is unclear how they choose the teams. Searcher bees go in random directions, investigate and come back while the non-searcher team waits at the starting point (say origin). The searchers come back to the origin after finding a suitable potential hive location. They record the direction and distance of the site to the origin. The searchers then propagate their found location to the non-searchers. To do so, they wiggle (shake/vibrate) while standing on the top of the non-searcher bees. The characteristics of a wiggle represents the information needed to locate the newly found location. These bees wiggle in the direction of the potential location of the hive. The number of times it wiggles represent the distance of the location from the origin. For instance, a wiggle of 3 times, heading towards 45° to north represents 3 km in the direction of N°45. The frequency of the wiggle is determinant of the importance of the site. The non-searcher bees sit at the origin for a long time, and after some time (not known how much time), not all but some of them go themselves to see the site they consider to be the best. After investigating the locations, they will also start publicising the most suitable location according to them. It will take around 8-10 hours [54] before everyone decides to move to the 'best'

location. A single bee can convince only a couple of bees at a time. It goes to each bee and wiggles, those who sense the wiggle record the data. Therefore, once a bee is convinced of a particular site, the bee starts propagating it as its own personal best site. This is similar to the rumour routing protocol and gossip algorithms.

Issue

Like bees, Zebros are also aimed to explore uncharted and undiscovered locations without external help or a leader. However, unlike Zebros, bees have near-perfect senses to compute distance and direction. Their calculation of distance odometry is very reliable with suitable error-correcting mechanisms. We need to find compensation for this advantage of the bees.

2.6 Summary

Though a lot of amazing work has been done and well-received, none of them is fit for the Zebro design and the vision of the Zebro team. None of the existing algorithms seem to be exactly implementable on the Zebro swarm. More so, no one has tried to achieve a location independent swarming ground robots. This thesis exists to fill the knowledge gap that provides an implementable solution for Zebros. The proposed algorithms are an integration of the various incentives gathered from this chapter, especially Section 2.5.

Chapter 3

Collective behaviour of Zebros

In this chapter, we describe the three distributed decision-making algorithms suitable for the robots: Zebros. In Section 3.1, we catalogue the goal requirements, constraints and assumptions in our presented swarm design. In Section 3.2 we stage the experimental setup for our proposed design. In Section 3.3, we describe the expected emergent behaviour from the swarm that fulfils our mission goals. In Section 3.4, we present an overview of the three proposed algorithms for Zebros. In Section 3.5, we describe the implementation of the three proposed algorithms in the form of imperative pseudo codes.

3.1 Requirements

Based on the limitations of the current state of the art in swarm robotics (Section 2.3); mission goal (Section 2.2.2) and hardware of Zebros (Section 2.2.1), we present the requirements and assumptions for the swarm design proposed in this thesis.

Following constraints uphold for the existing design of the **Zebro**[1]:

1. No distance odometry is possible. As stated in Section 2.3.6, Zebros are not equipped to calculate steps taken or distance traversed.
2. For an economic solution, we avoid using expensive and sophisticated devices like cameras.
3. Zebros walk in the forward direction only.

The desired **swarm characteristics** are described below:

1. Decentralised: To avoid single point of failure, we avoid the use of a central source of information.

[1]Refer Appendix B for more details on Zebro design.

2. Run time scalable: The swarm should functions independent of the number of robots in the swarm. This is important to make the swarm functionality immune to the discovery or loss of fellow robots.
3. Autonomy: Autonomy is defined as self-governance and independence. Human supervision from earth to the moon is impractical and unfeasible. If humans could assist the robots on the moon, we wouldn't need to send robots on the moon in the first place.
4. Equal distribution of responsibilities: Given a situation where a robot is authorised to lead the swarm. If any damage occurs to this robot, the mission will be handicapped. If this leader robot is lost, then the mission will be aborted.
5. Homogeneous swarm: All robots have the same hardware and software. No robot enjoys the privilege of extra sensors or more computation power.

Environmental constraints engineered by our application requirements are described below:

1. No localisation is possible: Following the argument in Section 2.3.5, we prohibit the use of methods for absolute localisation.
2. Unknown Region: Unlike applications displayed in Section 2.3.1, 2.3.3- 2.3.6, our swarm should be equipped to perform exploration in unknown areas as well.
3. Unbounded Area: To increase the robustness of the swarm in unequipped location, we should design the swarm to function in unbounded areas.
4. External resource requirements: The swarm should not dependent on external equipment which we can not provide on the moon. Unlike swarm robots mentioned in 2.3.1, 2.3.3- 2.3.5, we should avoid the use of GPS[2] and overhead cameras in our design.

These requirements and constraints are furbished for Zebros such that they can swarm everywhere, even on the moon.

The following **assumptions** are made to simplify our problem statement.

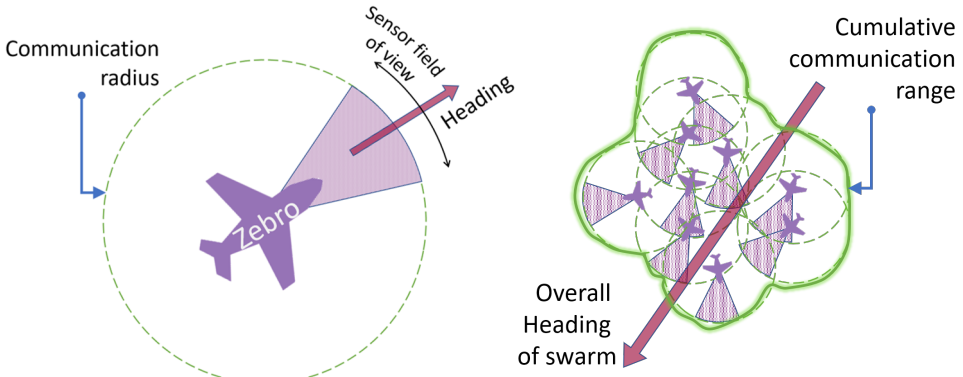
1. The communication module and sensors readings are error free.
2. Robots have inexhaustible battery.
3. Objects in the environment do not move.
4. In swarm robotics, communication is assumed to be easier than computation. Based on this assumption, we extensively communicate smaller but frequent messages. This reduces computation requirements and also provides redundancy for lost messages.
5. Communication delays are not considered in the model. However, there is scope for incorporating it in the model.
6. There are no defaulters or impostor robots in the swarm.

[2]Global Positioning System

3.2 Experiment setup

In the thesis, we represent the Zebro and the swarm of Zebros as shown in Figure 3.1. Zebro is represented with an aeroplane icon as in Figure 3.1(a). The heading of the aeroplane points towards Zebro's direction of movement. Though the sensing range is the same as the communication range, the field of view is constricted. The sensor can sense angle in the heading direction, in contrast to the omnidirectional communication field. The sensor as payload is to be determined on the mission of the Zebro Therefore, the sensor range, angle and the direction are arbitrarily chosen, and the solution presented in the thesis is independent of these sensor specifications. To identify the robots, they are assigned positive integers as ID.

Figure 3.1(b) shows a group of robots with increased field of view and communication range with respect to a robot. The movement of the robots emerge as a unit moving in the heading as shown by the arrow.

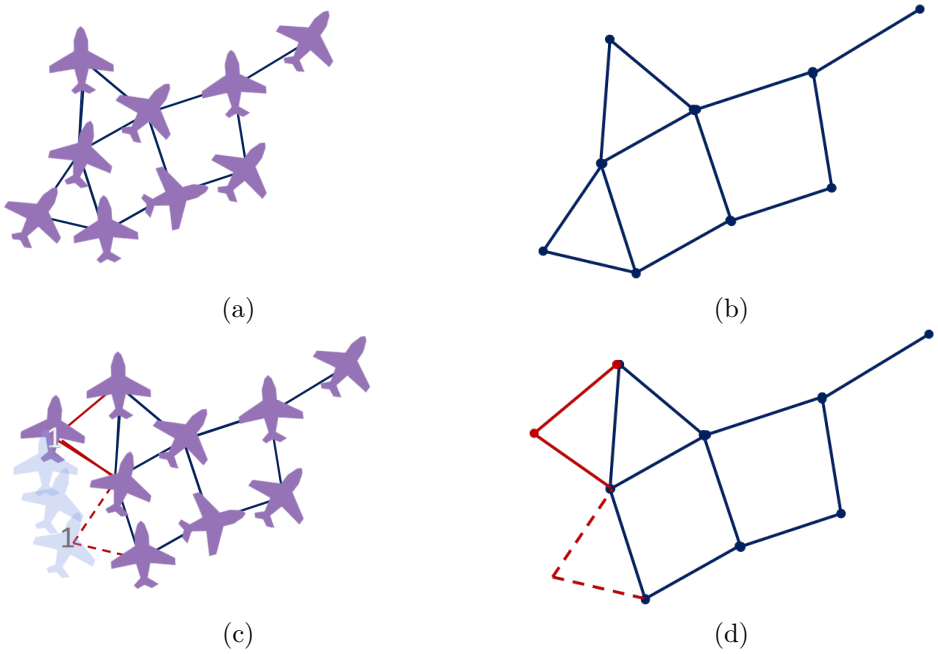


(a) A Zebro (b) A Swarm of Zebros

Figure 3.1: Symbolic description of the Zebro and a swarm Zebros: (a) A single robot with pointy vertex pointing towards the direction of it walks (b) The swarm of robots with increased field of view.

As shown in Figure 3.2, this distributed system can be modelled as an adaptive network. Each robot is modelled as a node and the communication connections are mapped as the links between these nodes. Weights on the nodes represent sensor readings of the robots. As the robots sense newer particle, the weights on the nodes change. The graph of the robot network is shown in 3.2 (b) and 3.2 (d). The communication links are constantly emerging and breaking. The Figure 3.2 (c) and (d) shows the breaking of links and emerging of new links as the robot with ID = 1 moving forward with respect to 3.2 (a) and (b) .

As per the requirements in Section 3.1, we construct an experimental setup



(a) (b) (c) (d)

Figure 3.2: Representation of the swarm as an adaptive network. (a) shows a swarm and (b) is the corresponding network representation of the swarm shown in (a). (c) depicts a transformed swarm where robot with ID 1 moves forward. (d) As the swarm transits from swarm state (a) to (c), certain links are broken and some new links are formed. Figure (d) represents the transition of the change observed in the network with respect to (b).

that replicates the environment that robots will witness on the moon: unbounded, unassisted, unknown to the swarm, no maps, no grids, no landmarks and no beacons. Figure 3.3 shows the setup.

The coloured pegs in Figure 3.3 represent objects that the swarm aims to search on their mission. The objects are sensed using some sensor[3] whose readings are fused, normalised over 100 and rounded to an integer. We call this integer as the utility of the object. We refer these objects as food particles throughout the following thesis. Hence, utility represents the value of the food particle. When a robot wants to acquire the object, it is said to have *consumed* the food particle[4].

It explicitly follows from the assumption that wherever there is nothing to be found, the utility is zero. Moreover, we reserve negative utility to represent an obstacles. The robots tell each other their utility values. The more positive the value is, the more desirable it is for the swarm. Also, no

[3]to be defined by the mission.

[4]The terminologies used in the thesis is inspired by biology.

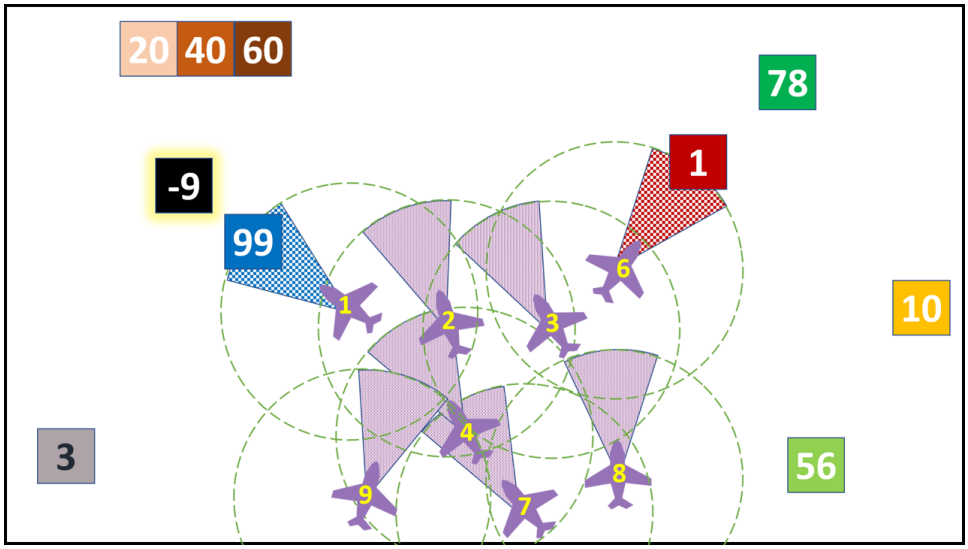


Figure 3.3: Scenario with food particles around the swarm of Zebros. The square pegs represent the food particles. All food particles are of different utility, represented by different colours. The network of robots is a connected graph. Agent ID 1 sense food particles with utility 99 and Agent ID 6 senses utility of 1. The emergent behaviour expects the swarm to consume 99.

two particles will have the same utility.

3.3 Emergence

In this section, we propose the cooperative and productive behaviour of the swarm that will facilitate the group of Zebros to accomplish missions on the moon. Thereby, we answer the first proposed research question in this section:

RQ1 *How can a group of autonomous legged robots collectively traverse unknown and unbounded areas?*

In Section 3.3.1, we define the behaviour of the swarm as seen by a global observer i.e., the emergent behaviour of the swarm. The shortcomings of the previously implemented swarm robotics led to the formulation of this new swarm design. In this thesis, we describe a generic swarm design that has the potential to accomplish the Zebro missions. Therefore, in Section 3.3.2, we describe the future prospects of the proposed swarm design that will actually accomplish the Zebro missions. In the remaining chapter we define the rules followed by individual agents, to result in the formulated emergent behaviour.

3.3.1 Emergent behaviour

With such a constrained problem statement as described in Chapter 1 and elaborated in Chapter 2, we have limited options. As we can not use methods of absolute localisation described in Section 2.3.3, robots rely on relative localisation. With no map[5] or previous knowledge of the environment, we design the swarm behaviour such that all the robots stay together.

By staying together these robots minimise the possibility of getting lost with respect to the swarm. On unbounded locations like the moon, if a robot leaves the swarm, it might never be able to get back to the swarm, ever again. Moreover, there is no provision for robots to get back to swarm as they are not assisted with proper odometry calculations, maps, landmarks etc, that could assist the robot to find its way back. Also, the network of the swarm with more nodes, will strengthen the collective intelligence as each node brings with it more information[6].

The robots localise relative to each other by sharing their relative heading and relative distance like birds as described in Section 2.5.1. They sense the distance using on board communication sensor: Ultra-Wide Band [55].

Many sensors do not work in vacuum and some require line of sight, we avoid those methods of communication. We use one of the most pervasive spectrum of electromagnetic wave for communication: radio waves. Ultra Wideband (UWB) communication module is feasible for the Zebro design [56]. UWB also provides relative distances and directions between agents [55]. More importantly, it will work outside the earth. It will yield more accurate distance measurements on rough terrains than wheel odometry.

The incapability of Zebros to walk in a systematic manner is considered in the swarm design by depending on heading consensus. If the Zebros were to collectively walk 1 kilometer in a rough terrain, they will fail to achieve the exact distance, owing to slipping and toppling over in the rough terrain. However, if they were to walk in a direction, even if they topple, they can calibrate the direction [55]. Therefore, we base our collective behaviour on direction consensus.

As the swarm moves as a unit, they sense food particles that come in their way. We want the swarm to collectively make a very simple decision: to choose the food particle with the highest utility available to the whole swarm. Accordingly in Figure 3.3, the swarm should consume the blue food particle.

Therefore, we define the emergent behaviour of the Zebro swarm as their abilities to stay together and make collective decisions. We expect them to

[5] Section 2.3.4

[6] in terms of sensor data

decide in favour of the highest valued particle that they find.

"A family who eats together, stays together"

This thesis provides a general-purpose solution for the Zebro missions. The application layer needs to designed to achievement of Zebro goals. Therefore, in Section 3.3.2 we describe how this devised emergent behaviour can pave way towards the application.

3.3.2 Scope of the behaviour

The emergent behaviour as described in Section 3.3.1, with the decision making and cohesive property of the swarm will be enable Zebros to achieve the following applications as well:

1. **Obstacle avoidance**
   The generic decision making algorithm can be used by the swarm to avoid obstacle collectively as depicted in Figure 3.4. Presently, Zebros are capable of avoiding obstacles, individually. With the emergent behaviour, they will be able to avoid obstacles collectively without separating.

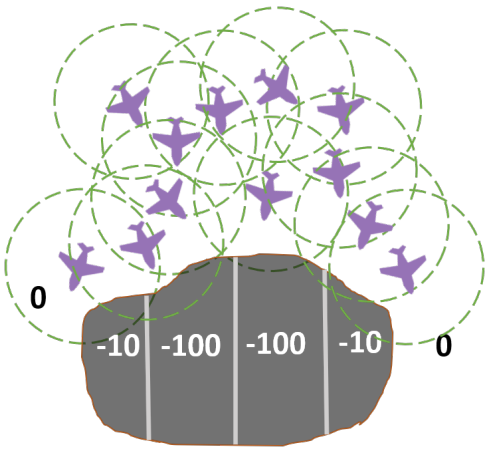


Figure 3.4: Obstacle avoidance scenario: As the robots detect an obstacle in front of them, they follow the path with the least negative value. In this case, they will tend to go in the direction where they sense a zero.

2. **Density: robot per unit area**
   The swarm can vary the cohesion between them to accommodate themselves with regards to the sparsity of the food/obstacle in a location. This behaviour is similar to the expansion of liquid molecule in response to exposure to high temperature.

For instance, robots spread out in areas of fewer obstacles to enhance their field of view. In crowded locations, they can coagulate into a tightly coupled formation. This will equip the swarm to have lesser tendency to separate in multiple groups. Such expansion and contraction will equip them to acquire multiple interests at the same time while staying together with the swarm.

The generic decision making algorithm will provide a mechanism to collectively decide the environment congestion and accordingly vary cohesion.

3. **Intelligent splitting/ chaining**
   As depicted in Figure 3.5, if the swarm could acquire multiple particles instead of choosing one between them, the overall utility of the swarm will increase. This is difficult to acquire without separation of the swarm in subgroups. If the swarm could collectively decide from where to split they could all stay together. This is possible only if the swarm has the intelligence to collectively decide the dynamics of splitting/-chaining, after deciding that it has to split.

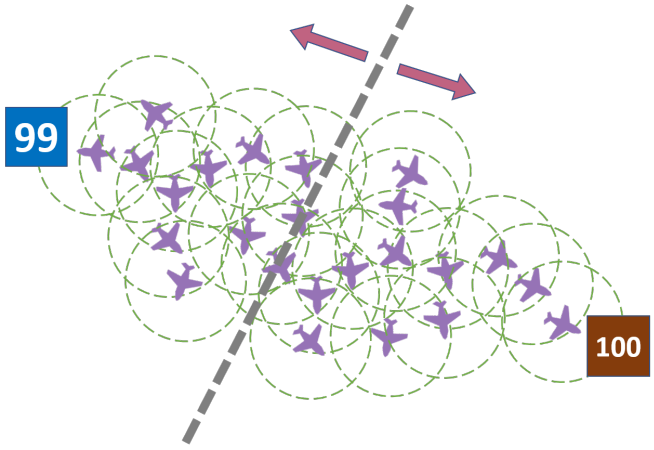


Figure 3.5: Group of robots split to acquire maximum utility. If the swarm senses comparable utilities, then they can benefit by acquiring both utilities, especially if one of them is the maximum utility in the landscape.

4. **Distribution of resources**
   The overall knowledge of the swarm will assist the distribution of resources among the robots based on their position in the network, previous charge status etc. As shown in Figure 3.6, the agent who found the particle can distribute the charge to the robots with critically less charge.

5. **Routing**: Routing will make it possible for the swarm to save re-

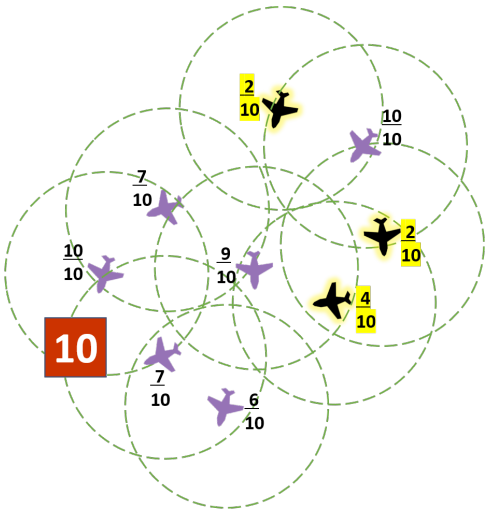


Figure 3.6: Distribution of charge based on charging status. The maximum charge is 10, the fractions represents the charge available to the agent. The highlighted agents have critically low charge.

sources on broadcasting information. The robots who found the food particles/obstacle can decide for the swarm, via routing, without involving the other agents in the process.

Therefore, the described emergent behaviour has the potential to perform tasks of an individual agent with more robustness.

3.4 Three Zebro algorithms

Three Zebro algorithms are proposed in the thesis, as described in Figure 3.7. The algorithms are written for individual robots to result in the emergent behaviour descried in Section 3.3. These robots constantly perform survival activities like obstacle detection, charging battery and maintaining the motion dynamics. These activities are already implemented in Zebros [48][27][40][21]. This thesis does not concern these activities. The activities in white represent the tasks performed by robots regularly. Further, we follow the flow chart top to bottom. The deviation from the tasks in white occurs when a utility is found. Following the discovery of one or more utility, the robots obey the shaded region as a flowchart from top to bottom in Figure 3.7. As soon as the utility is found information is proliferated through the swarm, directly or indirectly. When one or more utilities are sensed, then decision has to be made as to which utility to choose. On basis of the method of making the decision, the chart is divided into the

three algorithms: silent, immature and mature[7]. The three algorithms behave differently when they find utility. In all the algorithms, the robots continuously perform Reynolds' flocking and sense utility(if any).

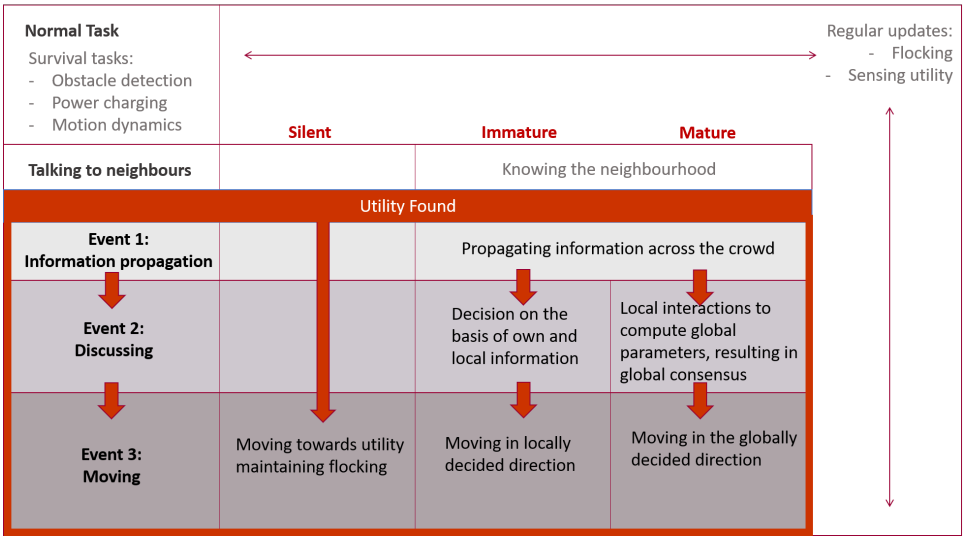


Figure 3.7: The basic flow of the three algorithms: the white regions represent the regular activities of the Zebro, whereas the shaded region is followed when one or more utility is found. The table is to be read from top to bottom, when utility is found.

The silent protocol involves no communication between robots[8]. The protocol is an adaption of the flocking algorithm as described in Section 2.5.1. The introduction of communication leads to the other two algorithms: immature and mature algorithms. The algorithms use communication to keep them updated about their neighbours and the information they have. The immature and mature robots communicate with neighbours using gossip algorithm as described in Section 2.5.2. In the immature algorithm, an individual robot takes a spontaneous decision based only on the immediate knowledge acquired from its neighbours. However, mature robots wait until a global consensus is achieved. The mature algorithm achieves global consensus, inspired by the strategy of the bee's democracy, as described in Section 2.5.3.

The varied behaviour of the algorithms in the presence of utility judges the

[7]Robots following silent protocol are referred to as silent robots. Similarly, immature robot and mature robots are the robots programmed with the immature and mature algorithm respectively.

[8]The silent algorithm is called 'protocol' because the algorithm is more like a pact between the agents, than an algorithm. As the robots do not interact with each other, they follow the pact to adhere to flocking rule and not to collide.

quality of emergent behaviour shown by the algorithms.

3.5 Algorithm implementation

This section describes the implementation of the algorithms in the form of imperative pseudo codes.

In Section 3.5.1, we introduce the silent protocol. The silent protocol synchronises the actions of the swarm without communication. This is the minimalist distributed algorithm characterising one of the most popular swarming algorithms: the Reynolds' flocking algorithm[9]. Though the silent protocol maintains the degree of autonomy as expected of the thesis[10], it fails to resolve conflict of opinions among the agents.

The silent protocol evolves into the immature algorithm by introducing communication among the agents as will be described in Section 3.5.2. Though immature robots attempt to keep the entire swarm informed about their findings, the swarm still breaks apart due to the difference in local best values and the global best [59].

Arising from the need of a global observer, the mature algorithm in Section 3.5.3 introduces a procedure to achieve global consensus via local interactions. Individuals are not allowed to move unless a global consensus is sensed. This ensures that opinions of all the individuals are taken into consideration to result in the global best.

We design a homogeneous swarm i.e., no agent possesses superior capabilities or authority over others. Therefore all the agents are programmed with the same code. Each subsequent subsection introduces an algorithm which is built on the preceding algorithm(s). Each section introduces its own new message types in addition to the previously used message types. As the section progresses, the algorithms' complexity builds up with respect to the preceding algorithm(s). Enumerated below are some facets common for all the algorithms.

1. Each agent communicates only with its own neighbours. An agent's neighbours are defined as all the agents in the given communication radius[11]. If multiple robots exist in the same direction but different distance to an agent, then they are also considered as the agent's neighbours.

[9] See Section 2.5
[10] See Section 3.1
[11] Communication radius = Sensing radius

2. Since the algorithm is presented for ground robots, no two robots can exist on the same location.
3. Though the algorithms are written for asynchronous agents, the simulator[12] uses global clock cycle to run the program.
4. As an agent's broadcasted message can be heard only by the neighbours, it is similar to multicasting within a swarm. Therefore when we mention a broadcast, it is assumed to be with respect to an agent and only its neighbours can receive the message.
5. If an agent senses more than one food particle at an instance, it reports the utility as the sum of the utilities of the particles.

As introduced in Chapter 1, this section answers the second research question:

RQ2 *How can consensus be achieved by a group of autonomous robots without having a global overview?*

3.5.1 Silent protocol

The silent protocol is the simplest of the three distributed decision-making algorithms proposed in the thesis. By introducing affinity or aversion of agents towards detected utilities, we extend the Reynolds' flocking algorithm into the silent protocol. As the name suggests the *silent* robots do not communicate.

Algorithm 1 Silent protocol

```
1: while true do
2:     procedure FLOCKING(heading, distance)
3:     procedure UTILITY FOUND
```

As stated in Section 2.5, Reynolds proposed three simple rules which assist a flock of birds to achieve consensus in their search and exploration tasks without communicating. Every bird calibrates two parameters for each of the neighbouring birds: the distance between them and their relative heading, equivalent to list f for Zebros as described in Table 3.1. The Zebros are intended to use the flocking strategy of birds by storing their relative direction and distance to all their neighbour in the queue F.

Based on their relative heading and relative distance, the birds change their speed or direction or both to keep up with the swarm. There are many variations of Reynolds' algorithm depending on the parameter influencing the movement of the agents: speed, direction or both. We propose change in direction of robots to flock, as the present Zebro design can turn but not

[12] Netlogo

accelerate. For the same reason, we have base our decision making strategy on heading consensus.

Table 3.1: Sensor value

Name	**Structure**	**Purpose**	**Queue**
f	$relative\_distance + absolute\_heading$	Flocking parameters	F
$sensed$	integer value	Utility of sensed food particle	

Algorithm 2 describes the implemented flocking algorithm. Whenever the distance of a neighbour is greater than the maximum permissible distance (say, $maximum - distance$), then the agent moves towards that neighbour to be nearer to that neighbour. This is termed as attraction/cohesion by Reynolds. However, if the distance between any of its neighbour is less than the minimum permissible distance (say, $minimum - distance$), then the agent directs itself away from the neighbour. Agents do so to avoid collision[13]. Reynolds calls this repulsion/ separation. In order to keep the swarm moving in the same direction, Reynolds introduced a third action called alignment i.e., each agent matches with its neighbour's heading. It does this by directing itself in the average direction of all its neighbours. The predetermined parameters dictate changes in the behaviour of the flock namely: minimum-distance[14], maximum-distance[15], minimum and, maximum turning angles[16].

The flocking algorithm is followed by the movement of agents towards/away from any detected utility. When an agent detects a utility in the vicinity, it starts following the Algorithm 3 while flocking. The algorithm gives the agent the tendency to direct itself towards a positive utility i.e., a food particle. If the utility is negative i.e., the object is an obstacle, the agent moves away from it, to avoid a collision.

The affinity towards the food particle often separates agents from the swarm [24] [65][27]. Therefore, the agents perform the procedure ALIGN once more after the detection of the food particle. This is intended to lessen the probability of the swarm to split into subgroups.

[13]Ultrasound sensors on the Zebros with SEPARATION procedure double checks the occurrence of collision avoidance.

[14]This radius is dependent on the size and swiftness of the Zebro

[15]This radius is dependent on the range of communication module

[16]The amount in degrees/radians with which an agent can turn in a given time period

Algorithm 2 Flocking

Input: list of heading and relative distance to neighbours.

1: **procedure** ALIGN(F)
2: $heading \leftarrow \frac{\sum heading\_of\_neighbours}{Number\_of\_neighbours}$ ▷ average direction of neighbours
3: **procedure** COHERE(F)
4: **if** $distance > maximum - distance$ **then**
5: $heading = heading\_of\_that - neighbour$ ▷ Move towards
6: **procedure** SEPARATE(F)
7: **if** $distance < minimum - distance$ **then**
8: $heading = heading\_of\_that - neighbour + 90$ ▷ Move away

Algorithm 3 Utility found

1: **if** $sensed > 0$ **then** ▷ Food-particle detected
2: head towards food-particle
3: **if** $sensed < 0$ **then** ▷ Obstacle detected
4: head away from obstacle
5: **procedure** ALIGN($heading, distance$)

3.5.2 Immature algorithm

The immature algorithm is a communication assisted version of the silent protocol. When a utility is found, the swarm following the silent protocol moves towards the utility. However, it is to be noted that more than one agent could have found utilities at the same time. In such cases, the silent robots tend to separate from the swarm. As described in Section 3.3.1, we do not expect such the swarm to split. Simulations [24][65][27] show that robots flocking in open spaces experience separation from the swarm.

Immature algorithm is intended to preserve the cohesion of the swarm by conducting a local election. the election is intended to choose the agent in the neighbourhood who sensed better utility than it or other neighbours. The result of the election is called $my-leader$. And the agent starts to follow the direction guided by the $my-leader$. While the movement happens and the swarm is flocking, the information of the local bests is spread across the swarm. Information from across the swarm becomes accessible to the agent. The new information allows the agents to choose an agent who sensed even a better utility.

Immature algorithm is synchronised with messages passing. As shown in Table 3.2, four messages are introduced in the algorithm. As an agent detects a utility it starts propagating message $m1$. Message $m1$ consists of its ID and the value of utility it sensed[17]. Agents store the incoming m1 data

[17]For the course of simplicity, we deal with positive utility i.e., food particles. Al-

Algorithm 4 Immature algorithm

1: **while** true **do**
2: **procedure** FLOCKING($heading, distance$)
3: **procedure** CHOOSING A LEADER ALGORITHM($m0, m1, m2, m3$)
4: **procedure** REALIGN($f, my - leader$)

Table 3.2: Message type

Name	**Structure**	**Purpose**	**Queue**
$m0$	STOP	Stop the swarm	
$m1$	ID	To create a list of neighbours	$M1$
$m2$	ID + $utility$	To compare utility values	$M2$
$m3$	ID + $max - utility$ + $my - leader$	To propagate the chosen leader	$M3$

Algorithm 5 Choosing a leader algorithm

1: **procedure** KNOWING THE NEIGHBOURHOOD
2: broadcast(m1)
3: **procedure** RECEIVE M1($m1$)
4: enlist all m1 into a matrix mat1 ▷ list of connected agents
5: **procedure** SENSED A UTILITY ▷ when an agent senses a food particle
6: broadcast ("STOP") ▷ stagnate the crowd
7: broadcast (m2)
8: **procedure** RECEIVE STOP
9: STOP
10: **procedure** RECEIVE M2($ID, utility$)
11: Enlist m2 into a matrix mat2 ▷ record various findings
12: broadcast (m3) ▷ broadcast personal best
13: **procedure** RECEIVE M3($ID, max - utility, my - leader$)
14: Enlist m3 into a matrix mat3
15: Choose maximum of own max-utility with max-utility of neighbours.
16: **if** $max - utilityofneighbour > max - utility$ **then**
17: my-leader ← my-leader of that neighbour

in a queue called $M1$. The agents compare its and its neighbour's utilities. The neighbour or itself, with maximum utility, is declared as $my - leader$.

gorithm for negative utility can be replicated similarly. The negative utility is reserved for obstacles.

Once an agent selects $my - leader$, it starts sending message $m3$. Message $m3$ consists of three bytes: own ID, the maximum utility encountered by it and the ID of my-leader. When an agent chooses, $my - leader$, it starts to realign[18] itself in the direction of the $my - leader$, unless it encounters a better maximum utility via $m2$ or $m3$. In the event when the robot has the best utility among his neighbours, i.e., it is it's own $my - leader$, then it just walks towards the sensed particle. Neighbouring agents keep comparing their maximum-utility with message $m2$ and $m3$. While this process of message passing is going on, the agents are flocking to keep close to each other without colliding.

Algorithm 6 Realign

Input: Matrix mat2

1: **procedure** REALIGN($my - leader$)
2: $heading \leftarrow heading\_of\_my - leader$ ▷ move in direction of chosen leader

As described in Section 2.5.2, the gossip algorithm is used to proliferate information throughout the swarm, which works without a predetermined routing mechanism. Gossip algorithm functions independent of the number of agents in the swarm. The asynchronous behaviour of the algorithm does not necessitate the need for a synchronised clock.

With increased communication cycles, the agent's local best comes closer to the swarm's global best. It is expected that immature robots are capable of achieving expected emergent behaviour.

3.5.3 Mature algorithm

One of the convenience available to the other implemented swarm robots, as mentioned in Chapter 2, is the presence of either a centralised control or a central source of information. In the mature algorithm, we intend to eliminate the requirement of a central source of information. We gather information about the entire swarm through recursive local interactions.

In the mature algorithm, we provide a method to achieve global consensus about the greatest utility available to the swarm. This consensus algorithm provides similar decisions to that recommended by a centralised controller, thereby eliminating the need for a centralised controller. Further, we generate an adjacency matrix that contains information about the connections in the graph generating the network structure of the swarm.

[18]Refer Algorithm 6

The immature algorithm added intelligence to the swarm compared to the silent protocol by providing local information through communication. However, the information propagation is limited to one hop (i.e., one neighbour) and the algorithm doesn't aim to achieve a global consensus. The individual agents are still not well informed of all the utilities encountered by the swarm at an instance.

The mature algorithm makes sure that all the agents are aware of all the sensed utility before choosing their leader. Therefore, there is only one leader ($the-leader$) instead of several personal leaders ($my-leader$). The mature swarm doesn't move until consensus on the best utility value is achieved. Once consensus is reached, all agents follow $the-leader$. This algorithm is called 'mature' algorithm as it accounts for all the opinions of its members before making the decision. This is advantageous compared to the immature and silent protocol, as global information is made available to all the agents.

In comparison to the immature algorithm, the mature algorithm has better provision to maintain the connectivity of the swarm. The splitting of the swarm into subgroups reduces. Though estimated to be slower than the silent protocol, the quality of decision in the mature algorithm should be better.

Algorithm overview

As described in Algorithm 7, as soon as a particle is sensed by the swarm, the STOP message is spread throughout the swarm and everyone stops. The STOP message also announces the start of an election. It is important for the swarm to stop, to keep the sensor reading static while the election is being held. The election is held via local interactions. The procedure of election is explained in Algorithm 8. As a result of the election, $the-leader$ is elected. $The-leader$ is the agent who detected the best particle sensed by the swarm during that election. When the election is over, the chosen leader ($the-leader$) guides the way to eat its detected particle. As $the-leader$ moves towards the particle, its neighbours sense the movement and flock towards $the-leader$. Soon, the entire swarm appears to be moving towards the particle. With the movement, the other contenders realise that they did not win the election. The leader walks with thrice the speed of flocking to prioritise consuming food over flocking. $The-leader$ does not flock while guiding the swarm. As the swarm approaches the particle, $the-leader$ consumes the particle and sends an ERASE message to the swarm. With the ERASE message, the information related to the decision making process is erased by the agents, $the-leader$ gives up its title and starts flocking again at normal speed.

Meanwhile, the individual agents start to contemplate the graph by recording their neighbour's links. All the agents are exchanging their link information with their neighbours. Repeated communications of agent's

Algorithm 7 Mature Algorithm

```
1: Variables reinitialised to 0
2: if utility found then                                   ▷ Swarm STOPs
3:     while EOE != 1 do
4:         procedure Election
5:     if EOE = 1 then                                     ▷ The election ended
6:         procedure Movement                  ▷ the − leader guides the swarm
7:         procedure Erase                                 ▷ Reinitialise variables
8:             EOE ← 0
9: procedure Developing a Graph
10: procedure Flocking
```

links, result in every agent knowing all the link pairs in the swarm. Agents store these links in its adjacency list to create an adjacency matrix of the swarm network. The adjacency matrix is verified to be complete when the adjacency matrix is equal to the transpose of itself as shown in Algorithm 11.

Election

The election is conducted when one or more particle is detected by the swarm. The agents who sensed particles are called contenders. The purpose of the election is to choose the best contender. The best contender is the one who sensed the highest utility among all the utilities available to the swarm at the time of election.

Algorithm 8 describes the election algorithm. The algorithm is divided into two algorithms: Propagation (Algorithm 9) and Back propagation (Algorithm 10). The various message used in the election algorithm are described in Table 3.3 and the assisting variable as described in Table 3.4.

Algorithm 8 Election procedure

```
1: procedure Propagation
2: procedure Back Propagation
3:     procedure End of Election
4:         EOE ← 1
```

After the start of the election, information of the sensed particle is spread across the crowd as message $n2$. Agents store the received and the sensed utility value in the queue $N2$. The maximum utility in the queue $N2$ is regarded as $max - utility$. $Max - utility$ is propagated by the agent, and the lower utility values are discarded.

Table 3.3: Some message type for the mature algorithm. Message type in gray-coloured text have been defined in previous algorithms as well.

Name	**Structure**	**Purpose**	**Queue**
STOP		Stop the swarm	
$n1$	**ID**	**To create a list of neighbours**	$N1$
$n2$	*utility*	Conduct election based on utility	$N2$
EOE		Marks the end of election	
ERASE		Resets variables	

Table 3.4: Variables for mature algorithm. Variables in gray-coloured text have been defined in previous algorithms as well.

Name	**Structure**	**Purpose**
sensed	**integer**	**value of utility**
heard	integer	The value of sensed food particles received from neighbours
$N2$	list of integer	contains all the sensed and heard values
$max - utility$	**integer**	**Maximum of food list**

As the information of the sensed particles propagates across the swarm, the agents classify their flockmates as a parent, child and siblings. Table 3.5 briefly describes the types of agent. The method of classification of these agents is explained in Algorithm 9.

During the election, apart from broadcasting their utility value the contenders wait for acknowledgement of its sensed particle. If a contender receives an acknowledgement of its food particle, then it knows that it is $the - leader$. $The - leader$ propagates the message "EOE = 1" which notifies everyone that the election has ended. This acknowledgement marks the back propagation of the result of the election. The back propagation algorithm is described in Algorithm 10.

Propagation

The propagation algorithm performs two main aims: propagates the sensed value and elects the best utility among them, and lays a foundation for back-propagation described in Algorithm 9. In this section, we first discuss the method of choosing the best candidate followed by generation of acknowledgements to initiate back propagation.

Algorithm 9 Propagation

1: **if** *sensed* utility **then** ▷ The agent is a contender
2: broadcast $n2$
3: $I\_know(X) \leftarrow 1$
4: $N2 \leftarrow sensed$
5:
6: **if** $n2$ received **then**
7: $heard \leftarrow n2$
8: $N2 \leftarrow heard$
9: $I\_know(X) \leftarrow 1$
10: $max - utility \leftarrow max(N2)$
11: **if** $sensed < max - utility$ **then**
12: **procedure** WITHDRAW FROM ELECTION
13: $sensed \leftarrow -1$
14: broadcast $max - utility$
15: ▷ The one who sends $n2$ is the parent.
16: ▷ The one who receives $n2$ is the child.
17: ▷ Parent and child occur in pairs.
18:
19: **if** $n2 = sensed$ **then** ▷ The flockmates are cousins
20: **if** ID < ID of sender $n2$ **then**
21: **procedure** WITHDRAW FROM ELECTION
22: $sensed \leftarrow -1$
23: ▷ The flockmates classify each other as cousins
24:
25: **if** $(I\_know(X) = 1)$ and (utility of $n2$ = X) **then** ▷ The flockmates are siblings
26: The flockmates are classifed as each others siblings

Electing When an agent receives $n2$ message, it adds the received utility (say X) in queue $N2$ and sets $I\_know(X)$ as 1. It compares the values in the food-list. The highest utility is classified as $max - utility$, which it propagates further as $n2$. The other utility in the food-list loses their significance. However, if a contender receives a $n2$ with a higher utility than his sensed utility, then it withdraws its contest and starts propagating the higher utility of received $n2$.

Framework for reception of acknowledgement: Back propagation is of vital importance for the mature robots, as it marks the end of the election and signifies that the swarm has chosen its $the-leader$. The propagation procedure lays the framework for the reception of acknowledgement thereby, finalising $the-leader$. During information propagation the agents are classified into various types as described in Table 3.5. We describe these agent types below. These agent types are specific to a food particle (say 'X').

Table 3.5: Mature robots categorise their flockmates into various agent types. There can be more than one of each type except, $the-leader$. The agent types are described below. Parent-child occurs in pairs, whereas siblings and cousins occur in sets of atleast two. Agent set in gray colour have been defined in previous algorithms.

Name	**Definition**	**Purpose**
Flockmate	**All the neighbouring agents**	**Agents within communication range**
Contender(s)	Agent(s) who found a food particle	They contest elections
Parent(s)	Agent informs child about the particle	Expects ack from child
Children	Agents who receive information of the particle via parent	Sends ack to parent
Siblings	Already informed pair of flockmates	Exception to flockmates being a pair of parent-child
Cousins	Flockmates who sensed the same particle	To reduce competition
$the-leader$	The chosen leader	Leads the way

As the contenders send the value to the neighbours in the form of message $n2$. The sender of the message is classified as parent with respect to receiver, who is its child. Parents will eventually expect an acknowledgement from child/children. The flockmates who know about a food particle without anyone telling it, they are siblings. Siblings know about the particle, they might or might not have the same parent[19].

It is also to be noted that two or more agents can sense the same particle. In such a case, they are called each other's cousins. The one with the highest ID contests on behalf of the particle. The rest 'younger' cousins, the ones with the lower ID number, withdraw themselves as contenders in the election.

[19]not important for the algorithm

These agent types are defined per food particle[20]. As different contenders exist for different food particles, the information generation of the food particle occurs at different points in the network. Therefore, the final acknowledgement is expected at the source of the information.
The messages $n3$ and $n4$ are carriers for the variables $I\_know(X)$ and $ACK(X)$ respectively.

Table 3.6: Message type for propagation and back propagation

Name	**Structure**	**Purpose**
$n3$	$I-know + utility$	Knowledge of existence of food particle 'X'
$n4$	$ack + utility$	Acknowledgement of food particle 'X'

Table 3.7: Variables for propagation and back propagation

Name	**Structure**	**Purpose**
$I\_know(X)$	Binary	Knowledge of existence of food particle with corresponding utility value 'X'
$ACK(X)$	Binary	Acknowledgement of corresponding utility

Back propagation

Children send acknowledgements to their parent for all the food particles they have received. An acknowledgement for a particle X, is sent if the following condition is fulfilled.

$$x_X = a_X + b_X + c_X$$

where,

x_X = Count of flockmates
a_X = Count of parent(s)
b_X = Ack received from child/children
c_X = Count of sibling(s)

[20] with utility 'X'

The acknowledgement is initiated by the leaf nodes. As the leaf node has no one further to propagate the message to, it starts the back propagation. For a leaf node, $a = 0$, therefore when $x = b+c$, it sends ACK to his parents.

$The - leader$ is the agent in the election who senses that end of the election. If a contender receives acknowledgement for its sensed food particle from all his flockmates, then it realises that it is $the - leader$. All his flockmates are his children, as it tells them about the particle it found[21]. Therefore, $the - leader$ is the parent of all his neighbours, it will not have any parent hence, $b = 0$, $c = 0 \implies x = a$.

The algorithm is designed such that there can be only one $the - leader$. The contenders who did not sense the particle with the highest utility, will always have $x_X \neq a_X + b_X + c_X$. For a non-winning contender $b = 0$, $c = 0$, and $x \neq a$. The condition, $x = a$ can not be true for anyone other than the contender who has the best utility.

Algorithm 10 Back propagation

```
1: if (x_X = a_X + b_X + c_X) and (I_know(X) = 1) then
2:     if b_X > 0 then
3:         Send ACK(X) to parents
4:     if (b_X = 0) and (sensed = X) then          ▷ ⟹ the − leader
5:         Broadcast EOE
6:         Set EOE ← 1
```

Developing the graph

To eliminate the requirement of the central source of information, local interactions are used to generate the graph of the network. The message M lists is introduced in Table 3.8. M comprises of all links connected to that node (agent). This message is spread throughout the network. As shown in Algorithm 11, this list is iterated with other link attachments of neighbours. All the links are subsequently stored with each agent to create an adjacency matrix.

Agents detect broken and new link by the change observed in the neighbour list $N1$. When the agents form new links, they add the new link in M, before broadcasting M. The agents update a broken link in M by conjugating a negation bit with the link. When the other agents receive a link with the negation bit, they delete the link from the adjacency matrix, else add it. Therefore, the adjacency matrix is updated with addition and deletion

[21] Any other agent who sensed the same particle is classified as a cousin, one of the two cousins withdraws his contestant, resulting in one contender per particle, as described in Algorithm 9.

of links. The update delay will depend on the size of the swarm and the communication speed.

Table 3.8: Message type for Algorithm 11

Message type	**Structure**	**Purpose**
M	list all links	To develop global graph

Algorithm 11 Developing the graph

```
1: procedure ENLIST OWN-LINKS                    ▷ Develop a list of own neighbours
2:     M ← list of own-links
3:     SEND LIST OF LINKS(M)
4: procedure SEND LIST OF LINKS(M)               ▷ Send own links to neighbour
5:     broadcast (M)
6: procedure RECEIVE LIST OF LINKS(M)            ▷ Store the links
7:     Adjacency_list ← M
8:     Delete duplicates
9:     M ← Adjacency_list
10:    SEND LIST OF LINKS(M)
11: procedure UPDATING M(M)                      ▷ Addition and removal of links
12:    own-links ← (-1)broken-links
13:    own-links ← new-links
14:    M ← list of own-links
15:    SEND LIST OF LINKS(M)
16: procedure CONVENE ADJACENCY MATRIX
17:    Adjacency_Matrix ← Adjacency_list
18:    if Transpose(Adjacency_Matrix) = Adjacency_Matrix then
19:        *Graph theories can be applied now.*
```

While the agents are choosing their leader, we develop the graph of the network. This will benefit the swarm in many ways. First, the completion of the adjacency matrix will confirm the proliferation of the information throughout the swarm at a local level. It will ensure that every agent had been reached in the message passing process. Secondly, the graph of the network so formed, can use various networking concepts to increase the efficiency of the system. Thirdly, we want individuals to be able to locate their position in the flock, with respect to their leader. If they could find the shortest path to $the - leader$, we can route the instruction to follow $the - leader$ in an efficient manner. It will even be advantageous to know the most traversed nodes, to make a quicker decision based on what the swarm is collectively 'thinking'.

Chapter 4

Evaluation of Zebro algorithms

Speed, quality, price. Pick any two.

— James M. Wallace

In this chapter, we infer the characteristics of the algorithms from simulation data. The problem statement of the thesis derived from the need for a collective decision making using communication in a group of autonomous robots in unknown, unassisted and unbounded site. Hence, the main goal of this chapter is to determine the decision making capability of the three algorithms in such scenarios. The communication module is not yet ready to be used[1], so we analyse the algorithm using simulation results.

In Section 4.1, we describe the simulator. In Section 4.2, we describe the experimental setup and performance metrics. In Section 4.3, we present the simulation results with inferences. In Section 4.4, we end the chapter by passing a judgement on the overall performance of the algorithms.

4.1 Simulation environment: Netlogo

To demonstrate proof of concept, we needed the simulator to support concurrent processes. We wanted a multi-agent simulator that models networks on a mobile distributed system. The simulator should be easy to use by different disciplinaries working in Zebro teams, including those who never programmed before. NetLogo[2] provides the required modelling environment [61].

[1] Appendix B

[2] The specifications of simulator setup are listed in Appendix A.

The simulator provides independent control over agents, food particles and communication links. At the same time, global values can be computed for analysis purpose. The simulator provides run time changes in parameters. The simulator has an entire library dedicated to calculating network parameters. The random generator creates a variety of experimental scenarios and robot positioning to validate the algorithms.

The NetLogo engine is single-threaded, so the agents must move one at a time in some order; they can't move simultaneously. The screen is updated after all the agents have had a turn; this visually preserves the illusion of simultaneity. The NetLogo application is one process and, the NetLogo engine is one thread within that process. The process scheduler is a cooperatively multi-tasked operating system [57]. Therefore the assumption that the system is concurrent is ambivalent. Also, the system is not absolutely asynchronous, the global clock synchronises the event cycles of the agents. Although the algorithms are intended to not need synchronisation.

The random and serial activation of the agents is in favour of the implementation in the real world. Any agent sends and receives the messages at any time, and does not synchronise its receiving with the sender. In the real world situation, there exists no synchronisation between transmitting and receiving will occur similarly. The robots will not be able to coordinate the sending and receiving events. This property influenced us to create algorithms that refrained from using event synchronisation.

Besides, the NetLogo code cannot be directly implemented on the robots. Moreover, Netlogo has programming limitations for developers looking for low-level programming. It is a strongly typed language compared to general-purpose programming languages like C and C++. There is no provision of message passing in the software. The data structures like array and matrix are not easy to use. The functionality of pointers is unavailable in NetLogo. Therefore, the initial pseudo-code had to be modified to fit in the Netlogo simulation environment. Moreover, such modification simplified the solution. With the lesser provision of storing the message data, the software implementation eliminated the non-essential aspects of the algorithms.

4.2 Methodology and metrics

In this section, we provide our solution to the third research question (**RQ3**), as described in Chapter 1.

RQ3 *How to quantify the emergent behaviour of autonomous robots?*

The agenda of the thesis is to provide algorithms for robots to perform autonomous exploration in the absence of external help, central source of in-

formation and/or central coordinator. Consecutively three Zebro algorithms namely: silent protocol, immature algorithm and mature algorithms are proposed in Section 3.5.

In this chapter, we study the swarm behaviour resulting from these three algorithms. We design six experiment setups by varying the swarm configuration and properties of the environment as described in Section 4.2.1. The environment is unbounded and unknown to the swarm. In Section 4.2.2 we provide measures to quantify the performance of the algorithms. The behaviour of the algorithms is classified based on **codependency on other robots for information, tendency to stay together and scalability**.

4.2.1 Experimental setup

In this section, we describe the experimental setup to understand the variation in swarm behaviour resulting from the three Zebro algorithms in different circumstances. We suggest six experiments as shown in Table 4.1. In these six experiments, we vary the number of robots, the number of food particles and the communication range of the robots.

These initial setups have varying parameters of the environment and the properties of the robots to observe the behaviour in the context of scalability, introduction conflicts and varying the factor concerning the cohesiveness. We run the three algorithms for all the six experiments.

Table 4.1: The six initial experimental setup

Experiment	**Number of robots (n)**	**Number of particles (p)**	**Communication range (c)**
E1	5	1	Short
E2	5	1	Long
E3	5	2	Short
E4	5	2	Long
E5	20	2	Short
E6	20	2	Long

These six experimental setups are represented as E-k, with $k \in [1, 6]$. Furthermore, the experiment running for each of the algorithms: silent protocol, immature algorithm, and mature algorithm, are termed as Ek-S, Ek-I and Ek-M respectively.

The experiments in Table 4.1 are developed by varying these three experiment parameters:

1. **Number of robots (n)**: Experiments 1-4 consist of 5 robots in comparison with 20 robots in Experiments 5-6. By varying the number of

robots, we want to evaluate the scope of scalability of the algorithms. Keeping other variables constant, to understand the change due to varying number of robots we compare **E3** with **E5** and **E4** with **E6**.

2. **Number of food particles (p)**: At the advent of the experiment, one food particle lies within the field of sensing of the swarm of robots in Experiments 1-2. We increase the number of food particles to two in Experiments 3-6. With two particles, we introduce a conflict of interest among the robots. Such a conflict is intended to test the quality of collective decision making of the swarm. With more particles, there will be more opinions creating more differences, further disintegrating the crowd. The analysis of the conflict provides a means to measure the integrity of the swarm: reliance of a robot with other robots for information and the loss of robots from the swarm due to conflict.

   We compare **E1** with **E3** and **E2** with **E4** to observe the response of the algorithms to conflict.

3. **Communication range (c)** We increase the communication range in Experiment 2, 4, and 6 in comparison to Experiments 1, 3 and 5. Increasing the communication range results in an increased number of neighbours. More neighbours imply faster information proliferation. Faster and more information propagation implies faster and better decisions.

   Higher value of communication radius also implies more connections per agent, resulting in a more connected network. A more connected graph implies lesser splitting of the swarm, implying more integrity of the swarm. The algorithms that perform well with smaller communication range will have higher network robustness. Higher network robustness allows the swarm to stay together in rough terrains and crowded areas.

   We classify Experiment 1, 3 and 5 as short range communication experiments (SRC) and Experiment 2, 4, and 6 as long range communication (LRC) [65]. In Category SRC, the neighbours of an agent are its physically neighbouring agents only. For Category LRC, every agent is a neighbour of every other agent. In Category LRC, an agent's message is equivalent to a broadcast to the swarm. To understand the change due to varying number of robots we compare **E1** with **E2** and **E3** with **E4** and **E5** with **E6**.

In the experiments, the initial setup of the robots is arranged on a connected random graph. Randomness in their positions and food particles is used to provide validity of the results in unanticipated circumstances.

The robots can sense food particles in the shape of a cone projecting in the forward direction of movement of the robot. The angle of this cone is 60 degree and the length of the cone is 3 units distance, whereas the length of the robot is 1 unit distance.

In the cases with two food particles, the relative position of the particles is oriented such that it tests the cohesiveness of the swarm. The particles are placed in a different direction of acquisition for the robots. Such that the swarm breaks if it attempts to eat both the particles. All the food particles are kept in such a way that they are sensed at the launch of the experiment.

Random numbers are chosen for the number of agents. It is kept in mind that one number is a one-digit number and the second is a two-digit number. The value of the food particles is also randomly picked. The communication range is kept such that it encapsulates two cases of network connectivity. First, where the network is barely connected. Any communication range shorter than this case will disconnect the graph. Second, where the network is a complete graph. Any communication range larger than this will be a waste of resources.

4.2.2 Performance metrics

After we have simulated the execution of all the algorithm in all the experimental setting, we analyse the results on the basis of the performance metrics mentioned in this section. We establish the following performance metrics on the basis of the properties we seek to judge in our algorithms:

1. **Quality of decisions:** $BestDecision?$ is a boolean metric representing the quality of decision taken by the swarm. The best decision for a swarm at an instance is to consume the food particle with the highest value available to the swarm. Any other decision is categorised as a wrong decision. $BestDecision?$ is true if the swarm choose the highest valued particle available to it at any given instance. $BestDecision?$ can not be calculated for single particle experiments: **E1** and **E2**.

   The rate of best decision (R_{BD}) is described as the ratio of the total number of runs in which the swarm chooses the best particle with the total number of runs, normalised over 100. R_{BD} describes the percentage of runs when the best decision was taken, i.e., $BestDecision?$ is true. The best quality of decision is expected to be with the swarm with the highest value of R_{BD}.

   R_{BD} can also be identified as a measure of predictability of the decision of the algorithms. It also provides a standard to measure the utilisation of collective intelligence.

2. **Unity of the swarm:** Split? is a boolean metric. It describes the integrity of the swarm. If the swarm splits into subgroups, then the *Split*? is true, else false.

   Split? rate (R_S) is described as the normalised value of the ratio of the number of times the swarm splits to the total number of runs. The resulting percentage is a measure of the number of times the swarm splits in a given experimental setup. The rate is an inverse measure of the unity of the swarm. It also provides an aggregate to measure the connectivity of the swarm network.

3. **Decision making time** is the time taken by a swarm between sensing a particle and executing the decision. The event of decision making is marked by different instances for different algorithms. We base our definition on the common trait within the algorithms by capturing the time between sensing and consuming the particle[3].

   We use three parameters to measure the decision making event:

   (a) *ticks*: *Ticks* is the simulated time. It is Netlogo's representation for the software's internal timer. Each agent in the model undergoes a singular random behaviour each tick [18]. Ticks are the number of event cycles taken by agents to make a decision. One *tick* represents an event cycle when all the robots run their codes once.

   (b) *time*: *Time* is the simulation time. It represents the system clock. Therefore *time* is the computer time the software took to make the decision.

   (c) *timepertick*: *Timepertick* represents the time taken per execution of a tick.

   Ticks, *time* and *timepertick* are calculated as the averages of ticks and time over multiple runs. Since the simulator is processing these concurrent events in a single thread, the actual time in the implementation will be different than *time*.

4.3 Observations

In this section, we develop an understanding of the order of the emergent behaviour in the distributed algorithm for the experiment variables and

[3] In the silent protocol, an agent moves after it decides. In the immature algorithm, agents decide between best particle while it moves. Agents following the mature algorithm move after everyone has decided on a common value.

performance metrics. Therefore, this subsection is divided into performance metrics. In each of these sections, we analyse the behavioural changes in the form of performance metric of every algorithm in general and also with varying experimental parameter. Further, the best algorithm for the performance parameter is also evaluated.

In the process, we develop an understanding of the experimental parameter's influence on the algorithm. We can declare the expertise of each of the algorithms. The process enables us to choose the best of the three algorithms based on optimum results for a single performance metrics and also the comprehensive best algorithm.

We gather performance metrics data from 20 runs of the three algorithms in all the six experiment setups as shown in Table 4.2, Table 4.5 and Table 4.7.We highlight the outliers in the tables, to foresee the best algorithm for the particular performance metric and the overall performance later in Section 4.4.

We identify outliers by measuring the central tendency of the data. the central tendency is measured by either mean or median. We focus our attention towards the half of the central tendency of the data where the performance is in favour of our problem statement. In the cells are shaded for one side of the median. The favourable property is the unshaded cells. We use mean when the distribution is near normal. The mean has one main disadvantage: it is particularly susceptible to the influence of outliers. The median is less affected by outliers and skewed data. We use the median where the variation between the highest and the lowest value is very large, for the mean to be affected by it.

4.3.1 Quality of decision

In this section, we present the observations and their interpretation to analyse the quality of decisions. As described in Section 4.2.2, we quantify the quality of decision making based on the rate of best decision (R_{BD}). Table 4.2 represents the values of R_{BD} for runs for all the six experiments for the three algorithms. Table 4.4 represents cumulative R_{BD} for the runs in all setups for each algorithm.

At instances, the silent protocol and the immature algorithm do not reach any sensed food particle. Even though the food particle is sensed the food particle is not eaten. Ignorance of food particles is a major benefactor to bring down the R_{BD}. Therefore, we compute the ignored cases as the rate (in %) of the occurrence of such event in Table 4.3. We call it the ignorance rate (R_I). R_I is used as a helper performance metrics to reason for one of the prime reasons for wrong decisions.

The ignorance cases are beneficial and undesirable at the same time. In the ignorant cases, the quality of decision is reduced and improvement in the

Table 4.2: Rate of *BestDecision*? (R_{BD}): The percentage of occurrence of the event when *BestDecision*? is true. The table displays data for all the three algorithms in the all the experimental setups. The shaded cells highlight the situation with R_{BD} lower than the mean (68.5%).

	Silent	**Immature**	**Mature**
E1	-	-	-
E2	-	-	-
E3	55	50	100
E4	70	60	90
E5	25	50	90
E6	40	95	100

Table 4.3: Ignorance rate (R_I): The percentage of occurrence of the event when no sensed particle was consumed. The table displays data for all the three algorithms in the all the experimental setups. The shaded cells highlight the situation with rate of ignorance greater than 0%.

	Silent	**Immature**	**Mature**
E1	20	20	0
E2	50	5	0
E3	0	0	0
E4	0	25	0
E5	10	0	0
E6	15	0	0

unity of the swarm is observed, as will be seen in this and the succeeding section. The cases with ignorance of food particle, agents tend to stay together with the swarm. But in those cases, the swarm is not consuming any particle.

Silent protocol

The decisions taken by the silent robots is based on little to no information limited information of the other agents. The only indication of the presence of detected particle received by a member of the swarm is by the change in the average heading of its neighbours. The change in heading does not exclusively reflect the presence of a particle. The fluctuation in the heading direction might also be influenced due to new robots joining the swarm or old robots leaving the swarm. The heading of an agent is the average of all neighbour's heading. Therefore, a fluctuation in heading is averaged over the number of neighbours, attenuating the cue. More so, the value of the

utility found is not reflected through the amount of change in direction.

Hence, the silent robots do not know if any other particle(s) is sensed, the utility of the particle(s) or the agent(s) who found it. Silent robots lack the basic information needed to make an informed decision. If the silent robots do not know the presence or the value of the particle sensed, then they can not choose between the best, rather they would always think they are the best. Deducing from their unsocial behaviour, self-centred nature and lack of knowledge, the silent robots are bound to make wrong decisions.

Increasing the number of robots, cutbacks the quality of decision. We observe a constant 30% decline in best decision cases for **E3** and **E4** to **E5** and **E6**, respectively. This implies that an increase in decision makers, jeopardizes the quality of decisions. Scalability aspect of the silent protocol is therefore compromised.

The decrease in R_{BD} can partially be attributed to an increase of R_I. More robots imply more influenced average heading of each robot. As mentioned in Section 3.1, the robots are given more affinity to stay together than to consume the particle for individual benefit.

A large number of robots are the strength of the swarm robotics as they cover more area and increase the field of sensors. The blessing of a large number of robots in swarm robotics is unprofitable for silent robots.

Increasing the communication range, prompts an improvement in the quality of decisions. There is a constant increase of 15% in R_{BD} for both experiment pairs under observation : **E4** $\Longrightarrow$ **E3** and **E6** $\Longrightarrow$ **E5**.

With increasing communication radius, all the robots are each other's neighbours. Therefore, a robot's heading is now instantaneously influenced by the robot on the farthest end of the swarm as well. Therefore, the heading of the robots is influenced by both particles.

It is noted that the ignorance rate also increases with an increase in communication range. The increase in the ignorance rate results in a decrease in the R_{BD} of experiment pair **E5**-**E6** with respect to **E3**-**E4**.

Therefore, for the silent protocol, the best decision is affected by the number of particles (30%) more than the increase in communication range (15%) Together decreased communication range and a large number of particles following the silent protocol will yield poor decisions. Therefore, a large number of silent robots are poor decision makers when loosely connected.

Immature algorithm

With the introduction of communication in the immature algorithm, robots are more informed than the silent protocol. They know the relative position of the agent who sensed a food particle and the amount of food particle

sensed. This equips immature robots to make an informed decision based on their own sensed utility as well as other agent's utility. A robot's neighbours make the information about their neighbours available to the robot. After a certain number of propagations, the information is spread across the crowd. The robots make decisions at every event cycle based on the information acquired in this and the previous event cycle. The agents hence, take actions before acquiring the complete knowledge of the swarm. Incomplete information may lead to instances of poor decision making especially when contenders are not neighbours.

Increasing the number of robots, the quality of decision improves or remains the same. This result is better than that of silent protocol, where the quality of decisions decreases with an increase in the number of robots. Increased information due to an increased number of neighbours is accountable for the positive outcome.

Increasing the communication range, increases the quality of decisions without an exception. Infact for **E2** and **E6**, the immature algorithm provides a similar quality of decision as to the mature algorithm. If it was not for the ignored case in **E4**, the LRC results for the immature algorithm will be the same mature algorithm. The high quality of decision in LRC, indicate the behaviour of the mature algorithm in the immature algorithm. Whereas the ignored cases reflect the traits of the silent protocol.

Well informed agents take the best decisions. LRC assists the robots to increase information. With LRC, every robot is each other's neighbours. Therefore, a sent message is broadcasted. The agents do not have to wait for multiple hops for the information to reach across the swarm. This increases the knowledge base of the swarm, quickens the process of decision making and improves the quality of decision.

Unlike the silent protocol, the **ignorance rate** does not have a pattern based on varying parameters.

Immature algorithm is a transition algorithm from silent to mature. It is expected to show variegated properties relating to both algorithms. The quality of decision improved as compared to the silent protocol, but it still makes poorer decisions in comparison to mature algorithm. With an increasing number of robots and increasing communication range, the quality of decision has improved but is not the best.

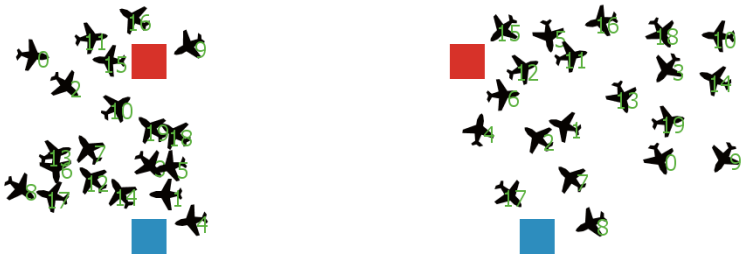
Immature robots profit from more number of robots in the swarm. It is therefore justified to use the immature algorithm for swarming applications. However, the quality of decision is low in SRC, which questions the use of the immature algorithm in rough terrains. Rough terrains tend to break the swarm, in such cases, an algorithm with robust network connectivity will go

a long way in maintaining the number of robots in the swarm and utilising collective intelligence.

Mature algorithm

Unanimously the mature algorithm provides the highest R_{BD} among the other algorithms, for all experimental settings. Mature algorithm's quality of decision is independent of variation in experimental parameters. The mature algorithm never ignores the presence of a particle as seen from Table 4.3.

However, the quality of decision shows peculiarity in **E4** and **E5**. We experience lower R_{BD} in **E4** and **E5** as compared to the cent percent R_{BD} in the other four experiments. These 4 out of 120 runs are regarded as wrong decisions. These anomalies are due to reason independent of the experimental factors. It is due to a simulator setup property and an algorithmic snag. There are two reasons for such behaviour are portrayed in Figure 4.1. Encountering such cases reveals that the mature robots are also prone to make wrong decisions.



(a) (b)

Figure 4.1: Two situations from **E4** and **E5** that led mature robots to make wrong decision. The utility of the blue particle is 95 and of red particle is 15. (a) The red and blue particles are in the same direction heading. The red particle is more likeable to eat faster because of its closer vicinity to the robot who sensed it. (b) More than one robot sensed the same particle, leading to conflict for the same particle.

The first reason for the anomaly is depicted in Figure 4.1(b), the smaller particle happened to be in the same direction as the bigger particle. Additionally, the smaller particle is closer to swarm than the bigger particle. The pixel size of the square-shaped food particle is in integer and the position of the robots is in float value. The not so equidistant nature of the food

particle and varied precision of position results makes it difficult to position the particles at equal distances to the robots. Such situations also occur in the silent and immature algorithm.

The second reason is related to the situation when two agents find the same particle as shown in Figure 4.1(a). The swarm follows the direction of the agent with the highest ID towards the particle. Silent and immature algorithm handle such cases differently. When multiple silent (or immature) robots sense the same particle, they move radially towards the particle, rather than following the direction proposed by one of them.
We obtain near-perfect result because the robots make an informed decision after knowing about all the findings in the swarm. The robots do not move without being informed of the status of the rest of the swarm. $R_I = 0$, also attributes to not be a delimiting factor to the R_{BD} and thereby keeping the quality of the decision high.

Inter-algorithm analysis

It is evident from Table 4.2 that the mature algorithm undisputedly makes better decisions than the immature algorithm and the silent protocol in all experimental scenarios. It is evident from Table 4.4, that mature algorithm makes 3 times better decisions than the silent protocol and immature makes 2 times better decisions than the silent protocol. Silent protocol's behaviour is unpredictable. With such non-deterministic and fickle nature, it is difficult to make them useful for a mission. The predictability of the mature algorithm makes itself the incontestable choice for deterministic decisions.

Table 4.4: Overall quality of decision among the three algorithms averaged over all the experimental runs.

Algorithm	R_{BD}	Performance
Mature	95	Best
Immature	64	
Silent	33	Worst

Additionally, the ignorance rate is a result of the average heading of the swarm to be different than heading of the food. It is a desired outcome that the swarm prioritizes flocking over their greed of consuming a food particle(s). The ignorance factor would not have occurred if the simulator was perfectly synchronous. Similar asynchronicity will appear in the implementation as well. Moreover, the agent will not miss a food particle if it rotates in the direction where it is sensed again. Such rotation avoids the food particle to go out of the sensing region of the robot. From Table 4.3 shows that silent protocol ignores more particles than immature, while

mature algorithm never misses a particle. Though the behaviour is comprehensible, but not consuming even one particle of the two is not justified. Mature algorithm tackles this issue as well.

4.3.2 Unity of swarm

Unity of swarm is a key characteristic to check the possibility of the swarming of Zebros on the moon. If the swarm splits into half each time a decision is taken, then the swarm might diminish eventually. The collectiveness of the swarm will suffer due to splitting, the intelligence will reduce correspondingly. It is to be noted that we are dealing with an unbounded area, robots separated from the swarm might never be able to find the swarm again.

Swarm splits due to lack of coordination, lack of information and hasty decisions. Zumaya et al in their paper [65] provided evidence of repetitive splitting of a flock of robots in open spaces. Zebro simulations [27] of flocking by Jurriaan de Groot also exemplifies the splitting behaviour.

Table 4.5: Split rate: R_S at which swarm splits into subgroups for the three algorithms in various experimental setup. The shaded background highlights the situation with split rate greater than 0%.

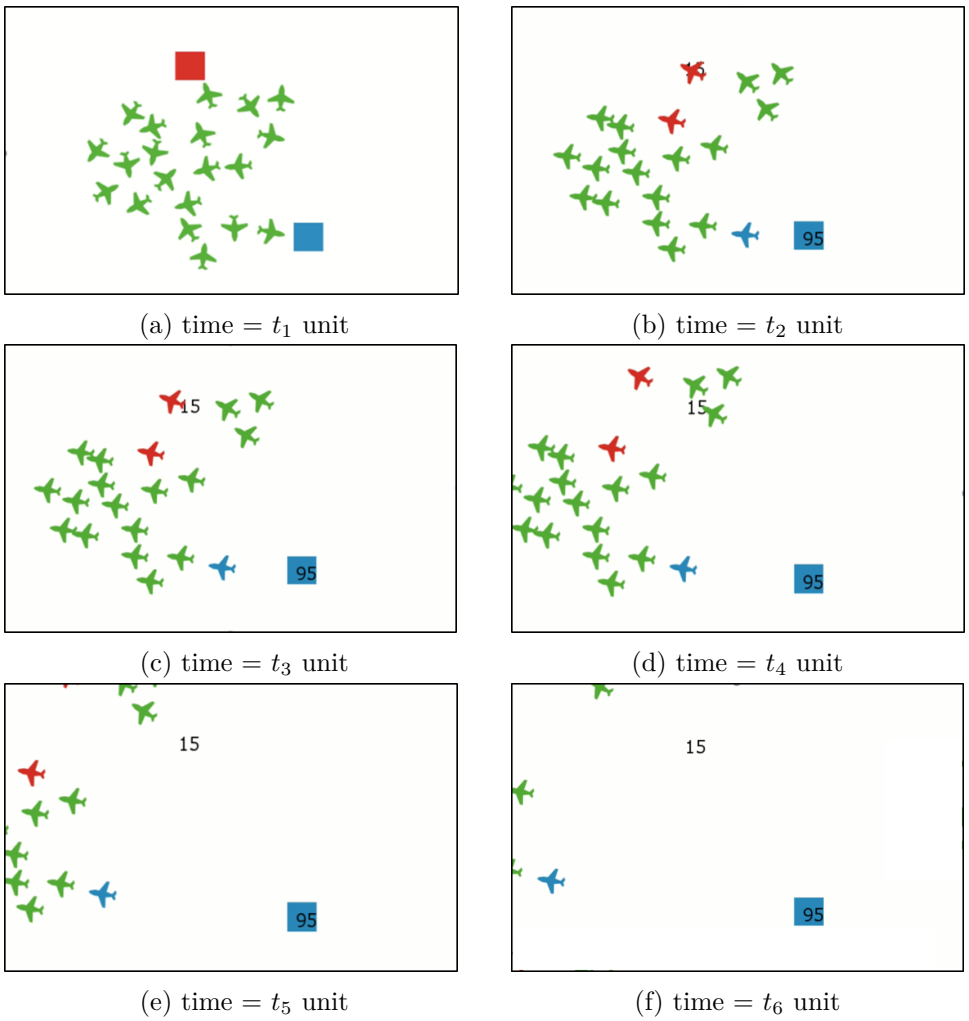
	Silent	**Immature**	**Mature**
E1	25	0	0
E2	0	0	0
E3	15	0	0
E4	5	0	0
E5	60	80	0
E6	0	0	0

We calibrate the unity of the swarm i.e., the ability of the swarm to stay together. As described in 4.2.2 we calculate the rate of *split* (R_S) as the number of time the swarm splits in all the runs. Table 4.5 represents the R_S. Unity of the swarm is inversely proportional to R_S.

Silent protocol

The silent protocol is expected to have a high R_S as it is essentially the flocking algorithm. Figure 3.5 displays the splitting of the swarm. In the figure, multiple particles are sensed in a decision cycle. However, the conflict due to these different particles is not the reason for the splitting of the swarm. Infact it is due to weak connectivity of the network that the swarm splits. The heading of an agent is influenced by only its neighbours, and uninfluenced by the other. The bridge formed by the red robots is the

weak connection of the swarm that results in the separation. The subgroups are unable to influence each other's heading, due to less connectivity. The swarm breaks because the two red robots are not enough to share heading directions with the subgroups on either side. Despite the split, the swarm did not even consume the best particle[4].



(a) time = t_1 unit (b) time = t_2 unit

(c) time = t_3 unit (d) time = t_4 unit

(e) time = t_5 unit (f) time = t_6 unit

Figure 4.2: Sequential screen-shots depicting splitting of the swarm in silent protocol in E5 setting, $t_1 < t_2 < t_3 < t_4 < t_5 < t_6$. The value of the red particle is 15, whereas the value of the blue particle is 95.

Increasing number of robot, the swarm splits into more groups of the swarm in SRC. It occurs because the only parameter that provides the swarm with the information about the presence of a particle is the change in the

[4]blue particle

average heading of the neighbours. This change in heading is diluted as the robots are farther away from the contender. With an increased number of robots, the information abates and the robots on the other side of the swarm are unaware of the discovery of another particle, leading to a separation. In LRC, separation is less frequent as every robot can communicate with each other in one event cycle. For the same reason, **increasing the communication range**, decreases the R_S. The increased R_I with increasing communication range benefits the swarm by upholding the unity of the swarm as seen from Table 4.3. **Introducing conflict** does not seem to influence the R_S.

In the simulation of Groot [27] the agents always break off from the swarm if they encounter multiple food particles in different directions. In the silent protocol, $R_I > 0$ indicates that the swarm has a greater affinity to stay with swarm than to consume a food particle. Therefore the swarm splits lesser in the silent protocol in comparison to the previous Zebro flocking simulations.

Immature algorithm

The immature algorithm never splits except for **E5**, where is splits drastically. **E5** represents the swarm with the largest number of robots who encounter a conflict. This sudden increase in R_S is similar to **E5** for the silent protocol, indicating that a high number of robots might result in immature algorithm behaving similar to the silent protocol. Further experiments with more robots will be more conclusive.

Howsoever, a swarm with the same number of robots with a dispute to resolve, do not split with a long communication range. Implying that immature algorithm behaves like the mature algorithm in LRC. Additionally, the quality of decision in **E6** for immature is similar to the mature algorithm. This reflects the hybrid behaviour of the immature algorithm.

Mature algorithm

Without an exception, the robots following the mature algorithm never split into subgroups. The mature algorithm maintains unity even in case of a weakly connected graph (as seen in the Figure 4.2 and 4.3(e)). Mature robots maintain network robustness. They do not make hasty decisions like silent and immature algorithms. Instead they wait until the best decision is chosen and then move altogether.

Inter-algorithm analysis

With varying parameters, the mature algorithm never splits. This fact advocates the use of the mature algorithm in unbounded areas, where the robots might get lost due to multiple decisions to be made simultaneously. On

the other hand, the immature algorithm performs very well, until it breaks the trend in **E5**, with more robots and shorter communication range. Resulting in its ranking below mature algorithm but still above silent protocol. Split protocol splits in almost all experiments without any preference to a single experiment parameter.

Long Range communication, saves silent and immature to split. Unity of the swarm can be trusted with LRC, in all the three algorithms [65]. Even for LRC scenarios, the weakly connected network should be sourced to mature robots. The LRC might turn into SRC with a lot of particles of interest or obstacles that might convince the swarm to split.

To conclude Table 4.6 represents the percentage of split in all the experimental setup. The mature algorithm performs the best while the overall performance of the silent is worst. The R_S of immature is high due to one experiment and silent robot split consistently through all the experiments.

Table 4.6: Overall performance for unity of the swarm among the three algorithms averaged over all the experimental runs.

Algorithm	Unity %	Performance
Mature	100	Best
Immature	87	
Silent	83	Worst

4.3.3 Decision making time

Decision making time is recorded from the instance an agent(s) senses a particle(s). The time is measured until the swarm consumes a particle. The consumed particle can be the particle that marked the beginning of the decision making time or any consecutively sensed particle. As described in Section 4.2.2 we record the simulation time (*ticks*) and system time (*time*) in Table 4.7.

Simulation time (*time*) is not a suitable parameter for measuring the duration of the decision making event. There are four reasons for holding this opinion. Firstly, *time* has high variances for the experiments as seen from Appendix C. The confidence interval is also very wide for the majority of the experiments. Secondly, the simulation time recorded is calibrated in microseconds which is comparable to the system's background activities. Therefore, simulation time is affected by factors outside the simulation environment as well. Thirdly, the high variance can be caused due to differ-

Table 4.7: Decision making ticks and time (time in milliseconds). The shaded cells have value higher than the mean for ticks and higher than median for time. mean(*ticks*)= 28.9, median(*time*)= 25.55.

	Silent		**Immature**		**Mature**	
	Ticks	Time	Ticks	Time	Ticks	Time
E1	89.5	55.4	29.5	12.4	8.9	20.3
E2	54.8	24.0	25.8	11.4	6.1	14.7
E3	67.4	38.7	31.6	16.9	8.5	10.1
E4	37.7	27.1	44.5	21.0	6.1	13.5
E5	17.0	38.6	25.5	32.6	9.5	375.8
E6	24.6	45.1	28.8	46.8	5.0	185.7

ences in distance of the food particles to the swarm[5]. Fourthly, it emphasizes on the fact that a wide variety of initial swarm formations were tested in the experiments.

Moreover, the actuation of the mature algorithm towards the chosen food particle is three times faster than the other two algorithms[6]. Therefore, time is not a reliable parameter to compare the algorithms. Hence, we emphasis our attention towards analyzing *ticks*.
Additionally, we expected the time and ticks to be related. But no uniform correlation was manifested across the table. Therefore, we investigate *timepertick* displayed in Table 4.8.

Table 4.8: Time taken per ticks (*timeperticks*). The shaded cells represent the cells with value of ticks greater than median (1.23).

	Silent	**Immature**	**Mature**
E1	0.62	0.42	2.30
E2	0.44	0.44	2.41
E3	0.57	0.54	1.19
E4	0.72	0.47	2.21
E5	2.30	1.27	39.56
E6	1.80	1.66	37.14

Timepertick is a measure of the execution time of the code as described in Section 4.2.2. *Timepertick* assists in reasoning for trends in *time*. Studying the *timepertick* revealed the difference in factors influencing the trend in

[5]the square food size is in integers and the position of the agents is in float with a precision of two decimal point

[6]It is to prioritize the movement of the leader over flocking. By moving three times faster, we equip the leader with a faster pace than walking speed due to the flock of robots.

time. We classify and explain the different factors below:

1. Algorithm functionality: Analyzing the time per event cycle with varying experimental factors provides a better understanding of the algorithm. *Timepertick* assists in estimating the time allocated to communication and computation or both. For instance, an increase in communication time is reflected in the duration of an event cycle[7] by varying communication range. With the introduction of conflict, communication as well as computation time increases.

2. Simulator property or experimental parameter?: A tick represents an event cycle for a run of the code of all robots. Increase in the number of robots implies that the more agents need to scheduled per tick. Therefore, at instances increase in the *time*, maybe caused due to simulator rationale. Keeping the number of robots constant, any trend in *time* caused thereafter might stem from variation in experimental parameters.

Silent protocol

Silent protocol is expected to be faster than the other protocol as it does not spend time communicating or waiting for the consensus. However, the data depicts that the protocol has a higher time per decision than the immature algorithm. Infact, the average of the time taken by silent is $\approx 40\%$ higher than the mean of the immature algorithm.

We are unable to find consistent patterns of increase/decrease that can be justified based on the protocol. We are unable to find trade-off between *ticks*, *time* and *timepertick*. We interpret two reasons to justify the data:

1. The spontaneity of the silent robots might be resulting in no patterns.

2. The large confidence interval as seen in Appendix C hints either a high variance or smaller sample size. We might need to perform more experiments to understand the behaviour of the silent protocol.

Immature algorithm

Increasing the number of robots, the *ticks* decreases and increase the *time* and *timepertick*. The number of *ticks* decrease due to quick access to more information resulting from more neighbours. More neighbours increase the connectivity between the robots as well, implying a lesser number of events (ticks) required to spread information. The increase in the *time* and *timepertick* is attributed to the single thread functioning of the simulator.

[7] *timepertick*

Introduction of conflict, results in increase in *ticks*, *time* and *timepertick*. Compared to SRC, a conflict in LRC witnesses a drastic change in the number of *ticks*.

With LRC, the number of events to reach decision increase because more information is available to the members of the swarm. The increased information is contributed by the presence of an extra particle and information acquired from the increased number of neighbours. The heading of a robot is influenced by new information acquired at the end of each tick. A final direction of the swarm is not instantaneously decided, rather it is not finalized until the swarm consumes the particle. The robots wiggles due to varied instruction coming from the flocking algorithm and particle information from a different contender. The robot's movement appears as if it is confused about the direction to head. In SRC, with less communication, it is easy for the swarm to avoid information. Therefore the number of ticks in SRC do not change as extremely with the introduction of conflict.

Without conflict, the immature algorithm behaves as a faster version of the silent protocol. Introduction of conflict initializes the communication functions of the algorithm, increasing the *time* and *timepertick*. Reception of $m1$ message[8] launches the sending of $m2$, computation from $m2$ message initializes the sending of $m3$ message and so on. This increases the time taken by individual robots to run the code per simulation cycle (tick). Therefore, *timepertick* increases. With the increase in *ticks* and *timepertick*, the *time* taken by swarm to come to a decision also increases.

Increasing the communication range, vary differently with and without conflict. In the case of conflict, the *time* and the *ticks* decrease with increasing communication range. Without conflict *ticks* and *time* decrease. *Timepertick* minutely increases with communication.

The observation of *ticks* and *time* is since without conflict the immature behaves like silent protocol. And with conflict, the algorithm 'tries' behaves like a mature algorithm.

Timepertick increases with increased communication range because more communication is done by robots to send the information to the neighbours. As the number of neighbours in increase with communication range. *Timepertick* also increases as sending and receiving information per robot increases.

Time and *timepertick* is effected mostly by the increase in the number of robots. *Ticks* is influenced by communication range. It is noted that immature robots behaviour is not unique. It is a fusion of silent and mature algorithm.

[8]refer to Section 4 for the purpose and content of the message type

Mature algorithm

Decision making in the mature algorithm is based on achieving a global consensus. Without a central coordinator, this global consensus is achieved by communication between the robot. Robots communicate with one another directly or via neighbours. This communication procedure is intensive with respect to the time needed to make the decision. The mature algorithm is, therefore, able to replace a central coordinator at the cost of increased communication time.

Increasing the number of robots, *ticks* increase by 1 tick. This increase is very small compared to the variation of *ticks* observed for the other algorithms. However, the *timepertick* increases phenomenally. The *time* shots up with the increase in the number of robots.

The increase in the tick represents the increase in the diameter of the swarm. Ticks for mature algorithm represents the number of events it took for the information of the sensed particle to be generated from the contenders, propagated across the crowd and returned as back propagation to the chosen leader. The small increase shows that the increase in robots from 5 to 20, did not phenomenally increase the diameter of the swarm.

Time increased because all the robots have to talk to all the other robots. In each event cycle (tick) every robot communicates atleast once. Therefore, the increase in the *timepertick* reflects the time taken to execute the communication by all the robots. The increase in *timepertick* is accountable for the increase in *time*.

Introduction of conflict, does not affect the number of ticks. However *time* and *timepertick* are reduced. This behaviour is counter-intuitive. We expected the *ticks* and *time* to increase the due to increased communication requirement of the swarm to conclude to a decision. Perhaps the introduction of the conflict does not unfavourably affect the time taken to resolve a conflict.

Increasing the communication range, *ticks* decrease as expected. Infact, all SRC experiments have the same number of ticks, so do all the LRC experiment setups. Moreover, the decrease in the number of ticks from SRC to LRC is roughly 33%, which is consistent for all the pairs[9]. However, the *time* and *timepertick* don't show a consistent pattern of change.

The consistent number of ticks in SRC experiments and LRC experiments separately depicts that other experimental factors[10] do not influence the number of ticks in the mature algorithm.

[9] E1-E2, E3-E4, E5-E6

[10] number of robots, the number of particles

The number of ticks decreases in LRC because it changes the communication type of the robot from multicasting in SRC into broadcasting to the swarm. In LRC, the information can be spread across the swarm in minimum one tick[11]. Therefore, lesser rounds of communication will be needed to come to a consensus, requiring lesser *ticks*.

Ticks are influenced only by variation in the communication range. As the amount of data transfer for the swarm is dependent on the number of robots. The number of robots affects the *time* and *timepertick* drastically. Introduction of conflict does not affect the time-dependent performance of the algorithm.

Inter-algorithm analysis

As we proceed from silent protocol through the immature algorithm to mature algorithm, we find increasing consistency in the number of ticks. Ticks are fairly consistent in the immature and mature algorithm. It is also deduced that the number of ticks decreases from silent to immature to mature algorithm.

Time for the silent protocol is decently constant to the changing experimental factors. Time taken by immature for decision making is less compared to silent protocol. The immature algorithm's time increases with an increase in the number of robots and particles. This behaviour of the immature algorithm corresponds to the time change pattern in the mature algorithm. We do not compare the time of mature algorithm with the other algorithms due to the varied speed of the robot in the mature algorithm.

Time per event cycle (*timepertick*) is fairly comparable between the silent and immature algorithm. However, the mature algorithm's *timepertick* is off the charts.

The *time*, *ticks*, and *timepertick*, seem to create a concoction of observations, still unable to completely justify the recorded behaviours. But some conclusions thoroughly validate the observations:

1. Silent protocol's decision making time is fairly constant. It reflects that the protocol functions independent of time.

2. In the observations, we witness that immature algorithm shows traits relating to both the silent and mature algorithm. This affirms the fact that it is a hybrid/ transit algorithm of the two.

[11]One tick can be taken to spread the information if the robot who sends the message is initiated by the simulator before the other robots receive it.

3. Ticks of the mature algorithms are only influenced by the communication range.

4. Mature algorithm's time is affected by the number of robots in the swarm as it waits for global consensus.

5. Since, the simulator is processing these concurrent events in a single thread, the time dependent performance metrics are misleading for the actual implementation. However, *ticks* is a very suitable parameter for analysis.

The significance tests of the Table 4.7 is provided in the Appendix C. For each experimental setting, we observe the overlap between the confidence intervals of different algorithms. Except for E4 and E6, we observe that no confidence intervals overlap. In E4 and E6, the intervals of the silent and immature algorithm overlap. These are the experiments with 20 robots possessing SRC. It provides evidence to our observation that silent and immature algorithm behave similarly in SRC for a large number of robots.

4.4 Summary

In the thesis, we defined intelligence for the swarm as the collective ability to resolve disputes. In this chapter, we analysed the trade-offs encountered in the algorithm based on maintaining the unity of the swarm, quality of decisions and decision making time. The importance of the information propagation was addressed by varying the experiment variable: communication radius. The scalability of the algorithms was analysed by varying the number of robots. The conflict resolution capability of the algorithms was put to test through varying the number of particles.

Obtaining the performance metrics in Tables (Table 4.2, Table 4.5 and Table 4.7), we shaded cells with the undesired characteristics. The number of shaded cells decreases as we move from silent to mature algorithm via the immature algorithm. Implying that the mature algorithm indeed does make the best decisions while maintaining the unity of the swarm, thereby endorsing the motivation of the thesis. The mature algorithm achieves the desired intelligence. However, the shaded cells in Table 4.8 invalidate the viewpoint that the mature algorithm is the fastest among the algorithms. Moreover, the comparison of *time* metric of the mature algorithm with other algorithms is unfair owing to its variable movement speed[12].

From the results of the performance metrics we also conclude that immature algorithm shows transiting behaviour between silent and mature algorithm.

[12]as mentioned in Section 4.3.3

Being a hybrid of the two algorithms, it sometimes behaves as either silent or mature or completely different than the two. For instance, without conflict, the immature algorithm behaves as a faster version of the silent protocol. With LRC, immature algorithm acts as the mature algorithm.

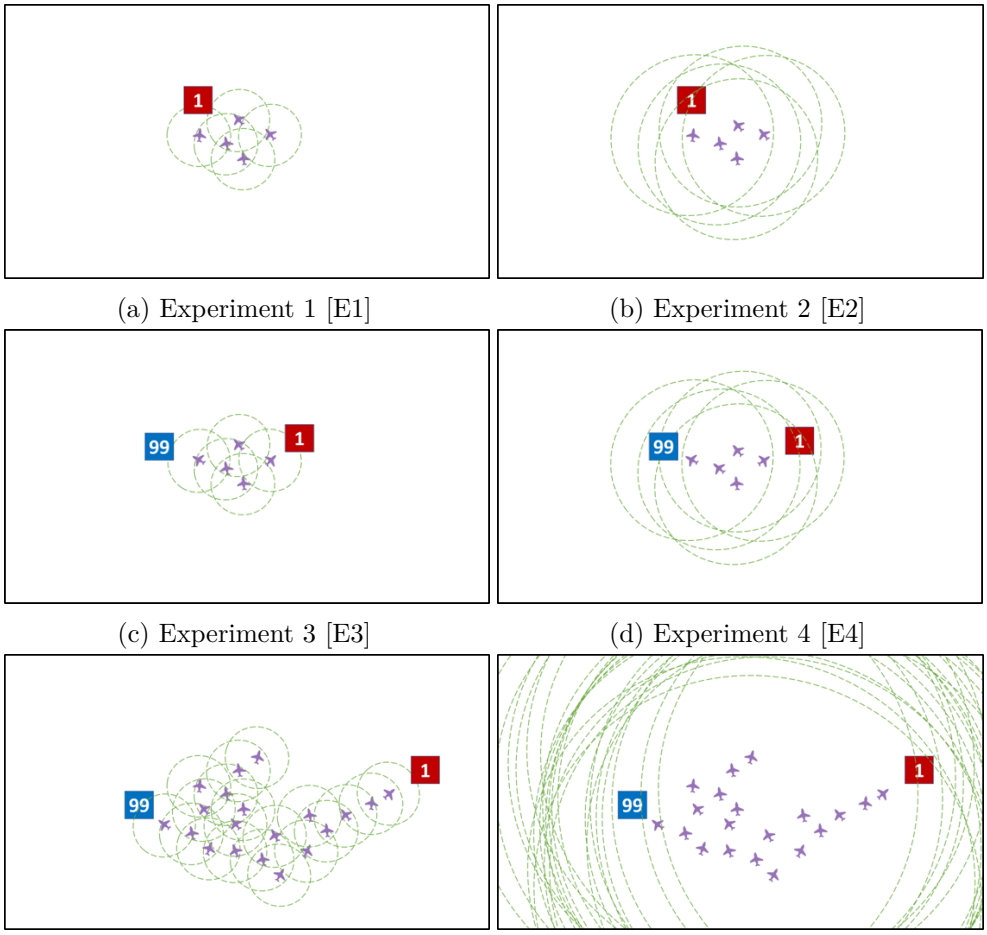
We present Table 4.9 as a conclusive verdict of the performance of the algorithms. Mature algorithm optimises the quality of decision and integrity of the swarm but the speed is not its forte[13]. For an algorithm with average performance metrics, the immature algorithm is the choice. The silent protocol does not excel in any performance metric.

Table 4.9: Overall performance of the three algorithms for all the performance metrics

Algorithm	**Quality of decision**	**Unity**	**Time**
Mature	Best	Best	Worst
Immature			Best
Silent	Worst	Worst	

The silent protocol can make compatible decisions without communication and therefore should be preferred when communication is unavailable or is an expensive resource. The silent protocol is not a good choice when it comes to unbounded areas, as the split rate is high, and the loss of robots might not be recoverable. The silent robots do not even realise the presence of conflict encountered by the swarm. The decisions made by the silent protocol are not predictable and therefore the choice of the best decision is not reliable.

[13] see Table 4.8



(a) Experiment 1 [E1]

(b) Experiment 2 [E2]

(c) Experiment 3 [E3]

(d) Experiment 4 [E4]

(e) Experiment 5 [E5]

(f) Experiment 6 [E6]

Figure 4.3: Experiment setup: Airplanes depicting Zebros, green circles representing communication radius, red and blue squares representing the food particles with utility as 15 and 95 respectively.

Chapter 5

Conclusions

In Section 5.1 we summarise the problem that we solved. In Section 5.2 we mention the uniqueness of our algorithms that contributes to the swarm robotics community. In Section 5.3 we propose future research that can be conducted on Zebro algorithms. We conclude the thesis with Section 5.4.

5.1 Summary

We avoid dependencies on external devices or human assistance during runtime, we circumvent the need for absolute localisation by creating global information by local interactions and with relative localisation. The robots rely on each other for decision making. In order to make the most of the collective behaviour, the robots need to stay together and trust each other's input. Therefore, we defined emergent behaviour as the decision making strategy of the swarm: the ability of the swarm to choose the best utility while being with the swarm.

We modified the traditional flocking algorithm into the silent protocol to provide a benchmark algorithm for the two new algorithms: immature and mature algorithm. We quantify the performance of the algorithms based on their capability to stay together and make an informed decision. We figured out that if any robots were to be sent on the moon, the mature algorithm should be the first choice as they show the expected emergent behaviour. The mature robots[1] make the best decisions while maintaining the unity of the swarm. This behaviour is useful to all the proposed Zebro missions on the earth and on the moon. Additionally, in order to implement the Zebro algorithms, no hardware changes are needed.

[1]Robots following mature algorithm are called mature robots.

5.2 Contributions

The proposed algorithms are **lightweight** in terms of implementation, communication, computation and memory storage. In contrast to our algorithms, swarming methods like Simultaneous Localisation and Mapping (SLAM) and adaptive predictive control (APC) do not conserve resources and are tedious to implement. SLAM requires a camera to be installed which communicates an array of images. These images are a bulk to store, communicate and compute. Due to the installation of cameras on each robot, the overall cost of the project increases by the size of the swarm times the cost of the camera. On the other hand, adaptive predictive control is computationally very heavy and only works for an already known number of robots.

The algorithms proposed in this thesis base their decisions on Boolean messages and normalised sensor readings ranging within $+/- 256$. The cost function is a simple comparative operation consisting of $<, >$ and $=$. The storage is minimal as the robots refresh their memory after every event of decision making. The mature algorithm stores an array (N*N) of Boolean values of the size of the swarm (N). Moreover, this array does not contribute to the decision making process and still, the memory required for the mature algorithm is less compared to SLAM and APC.

The election methodology of the Zebro algorithms is unbiased and with equal rights to all the robots. Even the leader does not practice any special privileges. A leader is merely an agent who detected better particle(s) from his view. The leader loses its title after consuming the particle(s). This property of the algorithm makes the swarm **fault-tolerant** to the loss of the robots. Moreover, addition of robots during run time is also possible. Therefore, the algorithms are as **scalable** and **fair as a democracy**.

The mastermind behind the success of swarm intelligence is collective intelligence. Collective intelligence profits from the collaboration and communication of the distributed agents, such that there is **no need for a central controller**. Moreover, avoiding the use of a central controller makes the swarm more robust and economises the mission. As no robot is a central controller or possesses any special privileges, the mission will not be stranded by loss of any robot. Without the need for a global information source, we save the cost spent on installing a Global Positioning System, overhead cameras or shared memory. Unlike other swarming algorithms in robotics, our algorithms are completely distributed and not dependent on any central controller.

Our robot's coordination is not dependent on the environment. The algorithm is **location independent** and does not require any previous knowledge of the region. Therefore, the swarm can traverse any uncharted and

undiscovered areas. Practically they can swarm anywhere in the universe.

Coming up with swarm mechanism for legged robot had been challenging and rewarding. The robots topple over oneself, they slip in repeated attempt to climb rough heights. They can neither count the number of steps they walk, nor calculate the distance travelled from a location. The proposed algorithm works for robots with such unsystematic and unpredictable method of walking, that makes our algorithm **athletic**. Therefore, the algorithm can work with any kind of robots especially for the like of Zebro who are poor in calibrating the distance odometry: humanoids and army tanks.

Glitches and mismatch of sending and receiving event may cause **asynchronicity**. Such asynchronicity occurred in the previous communication module of Zebros. Therefore we use event-based information proliferation mechanism: gossip algorithm.

Using the event-based synchronisation technique, robots do not need a global clock or an onboard clock to synchronise actions. Therefore, we save the bandwidth in the communication header by not using bytes for synchronising time. Moreover, if a robot reboots, it does not need to synchronise its time.

There exists no **generic** swarming algorithm that can be implemented on all kinds of robots. The swarming algorithm of other roboticist is closely related to their sensors, movement properties of the robots or environmental benefits: like a dark room, smooth flooring or a previous environmental setup. Unlike other robot swarming algorithms, our algorithm is **modular** in design. It can be implemented on all kinds of ground robots, walking on any surface, with any amount of sensors and any time of the day. The only requirement is a wireless communication module to provide collaboration among the robots.

The robots do not need to communicate more data than mentioned in the thesis. Any additional sensor payload on the robot will not influence the communication load. Additional sensors will increase the computational power to fuse the readings, normalise the value over 256^2 and send it over a byte. Thereby not increasing the communication data. This property will keep the communication resources insulated from changes in environment, robots or applications domain.

Often, the bottleneck to the collective behaviour of the distributed systems stems from **communication restrictions** [60]. It is also evident from the previous communication module of Zebro that the swarm size affects the amount of data transmitted by a robot. As we increase the number of ro-

[2]If colours can be represented in 256 divisions, so can any sensor reading.

bots, the time slot available for transmission per robots becomes smaller, questioning the system's scalability. However, the short range communication analysis in Chapter 4, shows that this issue is taken care of. Mature algorithm proves equal efficiency in the short range communication[3] as in long range communication[4]. The mature algorithm does not need the robots to communicate with all the other members of the swarm. Infact any number of robots connected in a line topology[5] will also compute the best decision without splitting in the mature algorithm.

Additionally, the proposed algorithms can be used as a communication protocol for controlling the dynamics of distributed multiagent embedded systems in varied engineering applications. Therefore, these algorithms will find direct application in **ad hoc** networks like Mobile ad hoc network (MANETs), Vehicle ad hoc network (VANETs) and Internet of things (IoT). Dynamically changing networks of complex systems like **adaptive networks** can also find a use of these algorithms.

5.3 Future work

The proposed algorithm is a general-purpose decision making layer to all the proposed Zebro missions. Albeit the success of the simulations, the implementation of the algorithms on the Zebro is equally important. Unexpected issues with communication module might counter the success of the Zebro mission.

Computing the global picture of the swarm via local interactions unfolds multifarious possibilities from the implementation-oriented rich field of network science. The global knowledge will make the scope of the emergent design described in Section 3.3.2 possible.

Using networking parameters like betweenness, the swarm can predict the possibility of occurrence of splits before it happens. The swarm can spread out intelligently to optimise the utility function. The position of an agent can help the robots to distribute resources like charging based on their previous charging status. The immature algorithm provides an advantage of time over mature algorithm. Networking parameters can help the immature algorithm to enhance its other performance metrics: decision making capability and maintaining the integrity of the swarm.

The compilation of the adjacency matrix via local interactions unfolds several possibilities of this adaptive network to implement strategies of static

[3]The degree of the robots is minimalist to maintain the connected graph.

[4]Every robot is in connection with every other robot.

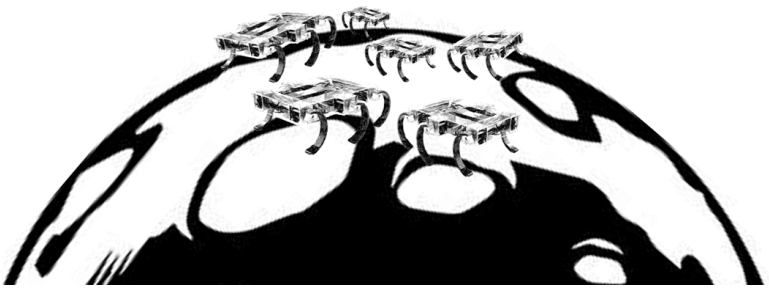
[5]Agents are connected to two adjacent neighbours, except the ones at the end who are connected to only one agent.

network. Time can be optimised by routing information via shortest paths between contenders to contest auction. Instead of the entire swarm conducting elections, the contenders can conduct an auction to decide between the best values. The swarm can also coagulate or expand its formation if an agent can see what the entire swarm can see.

The thesis is a convergence of biology, distributed systems, network science, and robotics/embedded systems. The direction that the proposed swarming algorithms lead will add value to all the four research domains.

5.4 Conclusion

Mature Zebros can be sent on the moon.



Epilogue

From decades, biologist and roboticists have been putting efforts in analysing swarm behaviours and patterns, to devise local rules governing their global behaviours, showcasing cooperation. However, traditional approach of roboticists in replicating the swarm behaviour is flawed as they include centralised assistance. Whereas, the strength and the robustness of swarm robotics lie in the decentralised coordination of agents. The author believes that swarm roboticists try to superficially imitate the swarm behaviour rather than reconstructing it. At the moment, the needed quality and quantity of local rules on swarms are simply unattainable by biologists. Coincidentally or naturally, the study of ad hoc networks, together with the study of social networks of humans can help find solutions.

With the similarities between ad-hoc networks and swarm network, researchers will be able to understand information propagation, dynamics of change in network and ways to deal with it. Social networks, on the other hand, are studied extensively. Social science researchers are now able to collect behaviour data and to evaluate their theories at a scale that never be reached by traditional offline methods. It is much easier to replicate social network theory than biological rules, rules which are not even devised now. Something similar has been done in this thesis. For example, in this thesis, we have, from a computer scientist's point of view, explored network science and distributed theories to stay close to an implementable solution, rather than mootly copying animal behaviour. We have demonstrated the robustness of the system without a leader using reliable and implementable sciences like network science and distributed systems.

Maintaining primitive behaviour and expecting the results of animals is also overrated. Animals have a wealthy set of well trained sensors. Robot prototype lack such precision and training. Animals also record faulty data but the quick processing and better sensor fusion techniques, overcome the flaws in the sensory data. The team should invest in introducing tilt sensor for balance, Inertial Measurement Unit (IMU) and magnetometer for Spatial orientation. If the aim is to reach the ninja anatomy that ants already possess to climb, build and repair structures, ninja sensors are needed.
Also, it necessary to believe that there will be times when the collective cannot outperform an expert.

Bibliography

[1] Lunar Zebro Mission. http://zebro.space/mission-2/. Retrieved online on August 2019.

[2] Mission page of National Aeronautics and Space Administration (NASA) https://www.nasa.gov/mission_pages/LADEE/news/lunar-atmosphere.html. Retrieved online on August 2019.

[3] Zebro. www.zebro.org. Retrieved online on February 2019.

[4] RHEX: Boston Dynamics. https://www.bostondynamics.com/rhex. Retrieved online on February 2019.

[5] DeciZebro. http://www.lottehoes.nl/lisanne-bouwde-mee-aan-de-decizebro-de-zesbenige-oliedomme-insectrobot/. Retrieved online on February 2019.

[6] Wikipedia page on Pheromone. https://en.wikipedia.org/wiki/Pheromone. Retrieved online on February 2019.

[7] Rapid Unarmed Autonomous Vehicle. https://www.rapiduav.com/. Retrieved online on February 2019.

[8] Lunar maps. https://www.space.com/30904-awesome-moon-maps-nasa-usgs.html. Retrieved online on August 2019.

[9] Wikipedia Page on Magnetic Field of the Moon https://en.wikipedia.org/wiki/Magnetic_field_of_the_Moon. Retrieved online on May 2019.

[10] What is mathematical biology. Centre for Mathematical Biology, University of Bath. www.bath.ac.uk. Retrieved online on June 2018.

[11] https://tudelftroboticsinstitute.nl/research. retrieved online on february 2019. Technical University of Delft Robotics Institute.

[12] Online source from US Navy mission https://www.onr.navy.mil/en/Media-Center/Press-Releases/2015/LOCUST-low-cost-UAV-swarm-ONR. Retrieved online on February 2019., April 2015.

[13] Javier Alonso-Mora, Andreas Breitenmoser, Martin Rufli, Paul Beardsley, and Roland Siegwart. Optimal reciprocal collision avoidance for multiple non-holonomic robots. In Martinoli A. et al, editor, Distributed Autonomous Robotic Systems 2013, volume 83 of Springer Tracts in Advanced Robotics, Berlin, Heidelberg, 2013. Springer.

[14] Gerardo Beni. From swarm intelligence to swarm robotics. In E. Sahin and W.M. Spears, editors, Swarm Robotics WS 2004, volume LNCS 3342, pages 1–9. Springer-Verlag Berlin Heidelberg 2005, 2005.

[15] Stephen Boyd, Arpita Ghosh, Balaji Prabhakar, and Devavrat Shah. Gossip algorithms: Design, analysis and applications. In IEEE 24th Annual Joint

Conference of the IEEE Computer and Communications Societies., volume 3, pages 1653–1664, Miami, March 2005. IEEE Infocomm.

[16] Francesco Bullo, Jorge Cortes, and Sonia Martinez. Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms. 3 June 2008.

[17] Russell C Eberhart, Yuhui Shi, and James Kennedy. Swarm Intelligence, chapter 6 (SWARM INTELLIGENCE), pages 187–219. San Francisco, CA, 2001. referring to Section 1.2.

[18] Justin Caccavale, David Fiumara, Michael Stapf, Liedeke Sweitzer, Hannah J Anderson, Jonathan Gorky, Prasad Dhurjati, and Deni S Galileo. A simple and accurate rule-based modeling framework for simulation of autocrine/paracrine stimulation of glioblastoma cell motility and proliferation by l1cam in 2-d culture. BMC systems biology, 11(1):124, 2017. publisher: BioMed Central.

[19] Y Uny Cao, Alex S Fukunaga, and Andrew Kahng. Cooperative mobile robotics : Antecedents and directions. In Autonomous Robots, volume 4, pages 1–23, 1997.

[20] Gilles Caprari and Roland Siegwart. Mobile micro-robots ready to use: alice. In in Proceedings of the IEEE IRS/RSJ International Conference on Intelligent Robots and Systems (IROS '05), page 3845–3850, Edmonton, Canada, August 2005.

[21] Marcel Ceelen, Cees van der Geer, Floris Rouwen, and Stijn Seuren. Fundamentals for an autonomous zebro swarm. Msc thesis, Delft University of Technology, January 2015.

[22] Pengqi Chen. Distributed algorithms design for zebro swarming. Msc thesis, Delft University of Technology, August 2017.

[23] Christopher M Cianci, Xavier Raemy, Jim Pugh, and Alcherio Martinoli. Communication in a swarm of miniature robots: The e-puck as an educational tool for swarm robotics. In Winfield A.F.T. Şahin E., Spears W.M., editor, Swarm Robotics. Lecture Notes in Computer Science, volume 4433. Springer, Berlin, Heidelberg, 2006.

[24] Cinder. Guide for cinder 0.9.1. https://libcinder.org/docs/guides/flocking/chapter1.html, 2017.

[25] Mario Coppola, Jian Guo, Eberhard Gill, and Guido CHE de Croon. Provable self-organizing pattern formation by a swarm of robots with limited knowledge. Swarm Intelligence, 13(1):59–94, 2019.

[26] Iain D. Couzin. Collective cognition in animal groups. In Trends in cognitive sciences, volume 13 of 1, pages 36–43. Elsevier Current Trends, January 2009.

[27] Jurriaan de Groot. Swarm behaviour for the zebro robot. Msc thesis, Delft University of Technology, November 2017.

[28] Gregory Dudek, Michael RM Jenkin, Evangelos Milios, and David Wilkes. A taxonomy for multi-agent robotics. In Autonomous Robots, volume 3 of issue 4, pages 375–397. Kluwer Academic Publishers, December 1996.

[29] Ali Emre Turgut, F. Gökçe, H. C. Çelikkanat, Levent Bayindir, and Erol Sahin. Kobot: a mobile robot designed specically for swarm robotics research. Techincal report, KOVAN Research Lab, Department of Computer Engineering, Middle East Technical University, 2007.

[30] Eric W Frew and Timothy X Brown. Airborne communication networks for small unmanned aircraft systems. In Proceedings of the IEEE, volume 96 of 12, pages 2008–2027. IEEE, December 2008.

[31] Brian P Gerkey and Maja J Matarić. A formal analysis and taxonomy of task allocation in multi-robot systems. volume 9, pages 939–954, 2004.

[32] Mohsen Ghaffari and Calvin Newport. How to discreetly spread a rumor in a crowd. In International Symposium on Distributed Computing, pages 357–370. Springer, 2016.

[33] Russell Golman, David Hagmann, and John H. Miller. Polya's bees: A model of decentralized decision-making. Technical report, Science Advances, 2015. Published online 2015 Sep 18. exclusive licensee American Association for the Advancement of Science. Distributed under a Creative Commons Attribution NonCommercial License 4.0 (CC BY-NC).

[34] Roderich Gross and Marco Dorigo. Towards group transport by swarms of robots. In International Journal of Bio-Inspired Computation, volume 1 of 1-2, pages 1–13, 2009.

[35] Yanying Gu, Anthony Lo, and Ignas Niemegeers. A survey of indoor positioning systems for wireless personal networks. IEEE Communications surveys & tutorials, 11(1):13–32, 2009.

[36] Hannah Hartwich. Swarming in nature and robotics: How to apply nature's strategies in robot swarming. Technical Report, 13 April 2018.

[37] Aleksandar Jevtić and Diego Andina de la Fuente. Swarm intelligence and its applications in swarm robotics. In 6th WSEAS Int. Conference on Computational Intelligence, Man-Machine Systems and Cybernetics, Tenerife, Spain, December 2007. International Conference on Computational Intelligence, Man-Machine Systems and Cybernetics (CIMMACS).

[38] Ebrima Jobe, James McLurkin, and Chutima Boonthum-Denecke. R-one swarm robot: Developing the accelerometer and gyroscope. In Proceedings of the Twenty-Fifth International Florida Artificial Intelligence Research Society Conference. Association for the Advancement of Artificial Intelligence (www.aaai.org), 2012.

[39] Ali Abdul Khaliq. From Ants to Service Robots: an exploration in Stigmergy-Based NAviagtion Algorithms. PhD thesis, Orebro University, May 2018. ISBN 978-91-7529-253-3.

[40] Laurens Kinkelaar. Adaptive gait switching control structure using max plus in legged locomotion. Msc thesis, Delft University of Technology, June 2018.

[41] Hiroaki Kitano, Milind Tambe, Peter Stone, Manuela Veloso, Silvia Coradeschi, Eiichi Osawa, Hitoshi Matsubara, Itsuki Noda, and Minoru Asada. The robocup synthetic agent challenge 97. In AI Challenges, pages 62–73. Robot Soccer World Cup, Springer, Berlin, Heidelberg, August 1997.

[42] John Klingner, Anshul Kanakia, Nicholas Farrow, Dustin Reishus, and Nikolaus Correll. A stick-slip omnidirectional powertrain for low-cost swarm robotics: Mechanism, calibration, and control. In 2014 IEEE/RSJ Intelligent Robots and Systems (IROS), pages 846–851, Chicago, IL, USA, 2014.

[43] Sergey Kornienko, Olga Kornienko, and Paul Levi. Minimalistic approach towards communication and perception in microrobotic swarms. In Proceedings of the IEEE IRS/RSJ International Conference on Intelligent Robots and Systems (IROS '05), page 2228–2234, August 2005.

[44] Enyu Luo, Ji Liu, Alexandra Bacula, Yuting Ng, Tamer Basar, and Grace Xingxin Gao. Shinerbots: Navigating large-scale robot swarms inspired by golden shiner fish. University of Illinois at Urbana-Champaign, 319 Coordinated Science Laboratory, 1308 West Main St. Urbana, IL 61801, 2017.

[45] Ross Martin. OMG True!: Incredible and Amazing Facts. BookBaby, November 2014.

[46] Francesco Mondada, Edoardo Franzi, and Andre Guignard. The development of khepera. In 1st International Khepera Workshop, volume 64, pages 7–14. HNI-Verlagsschrienreihe, Heinz Nixdorf Institut, 1999.

[47] Francesco Mondada, Luca Maria Gambardella, Dario Floreano, Stefano Nolfi, J-L Deneuborg, and Marco Dorigo. The cooperation of swarm-bots: physical interactions in collective robotics. In IEEE Robotics and Automation Magazine, volume 12 of 2, pages 21–28, 2005.

[48] Mattijs Otten. Decizebro: the design of a modular bio-inspired robotic swarming platform. Msc thesis, Delft University of Technology, April 2017.

[49] David Payton, Mike Daily, Regina Estowski, Mike Howard, and Craig Lee. Pheromone robotics. In Autonomous Robots, volume 11 of 3, pages 319–324, The Netherlands, November 2001. Kluwer Academic Publishers.

[50] Kirstin Hagelskjaer Petersen, Radhika Nagpal, and Justin K Werfel. Termes: An autonomous robotic system for three-dimensional collective construction. In Robotics: Science and Systems VII, Cambridge, MA, 2011. MIT Press.

[51] Bart Remes, Dino Hensen, Freek Van Tienen, Christophe De Wagter, Erik Van der Horst, and GCHE De Croon. Paparazzi: how to make a swarm of parrot are drones fly autonomously based on gps. In International Micro Air Vehicle Conference and Flight Competition (IMAV2013), pages 17–20, Toulouse, France, September 2013.

[52] Craig W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In Computer Graphics, volume 21 of 4, pages 25–34, Anaheim, California, July 1987. ACM SIGGRAPH '87 Conference Proceedings.

[53] Michael Rubenstein, Christian Ahler, and Radhika Nagpal. Kilobot: A low cost scalable robot system for collective behaviors. In In Proceedings of 2012 IEEE International Conference on Robotics and Automation (IRCA), pages 3293–3298, Saint Paul, Minnesota, May 2012. Washington, D.C.: Computer Society Press of the IEEE.

[54] Thomas D. Seeley and Susannah C. Buhrman. In Behav Ecol Sociobiol (1999), volume 45 of 19, pages 19–31, Berlin Heidelberg, May 1998. Springer-Verlag. https://doi.org/10.1007/s002650050536.

[55] Yuichiro Shimizu and Yukitoshi Sanada. Accuracy of relative distance measurement with ultra wideband system. In Electronics and Communications in Japan (Part III: Fundamental Electronic Science), volume 87, pages 26–36. Wiley Online Library, 2004.

[56] Thijs ter Horst. Uitra-wideband communication and relative localisation for swarming robots. Msc thesis, Delft University of Technology, Delft, March 2019.

[57] Seth Tisue and Uri Wilensky. Netlogo: Design and implementation of a multi-agent modeling environment, May 2004. Presented at SwarmFest.

[58] Martina Troesch, Tiago Vaquero, Amos Byon, and Steve Chien. A journey through an autonomous multi-rover coordination scenario in mars cave exploration. Technical report, Jet Propulsion Laboratory, California Institute of Technology, U.S. Government sponsored, 4800 Oak Grove Drive Pasadena, CA 91109, 2018.

[59] John N. Tsitskilis. Problems in decentralized decision making and computation. PhD thesis, Laboratory for Information and Decision Systems,

Massachusetts Institute of Technology, Cambridge, Massachusetts, December 1984. LIDS-TH-1424.

[60] Kagan Tumer and Adrian K Agogino. Overcoming communication restrictions in collectives. 2004 IEEE International Joint Conference on Neural Networks (IEEE Cat. No. 04CH37541), 2:1127–1132, 2004. IEEE.

[61] Uri Wilensky. Netlogo. Technical report, Center for Connected Learning and Computer-Based Modeling, Northwestern University. Evanston, IL, 1999. http://ccl.northwestern.edu/netlogo/.

[62] Heinz Woern, Marc Szymanski, and Joerg Seyfried. The i-swarm project. In ROMAN 2006 - The 15th IEEE International Symposium on Robot and Human Interactive Communication, pages 92–496, September 2006.

[63] Peter R Wurman, Raffaello D'Andrea, and Mick Mountz. Coordinating hundreds of cooperative, autonomous vehicles in warehouses. In AI magazine, volume 29 of 1, pages 9–20, March 2008.

[64] Giulio Zecca, Paul Couderc, Michel Banâtre, and Roberto Beraldi. Cooperation in a swarm of robots using rfid landmarks. In 2008 International Workshop on Robotic and Sensors Environments, Ottawa, Canada, 17-18 October 2008.

[65] Martín Zumaya, Hernán Larralde, and Maximino Aldana. Delay in the dispersal of flocks moving in unbounded space using long-range interactions. Scientific Reports, 8(2045-2322), 10 2018. Article number: 15872.

Appendix A

Simulation Environment

In this appendix, we give the specifications of the hardware and software setup used to obtain data in Chapter 4.

A.1 Software specifications

As described in Chapter 4, we use Netlogo [61] to simulate our algorithms. Given below are the specifications of the Netlogo version used:

- Version: 6.0.4 (June 4, 2018)
- Extension API version: 6.0
- Java HotSpot (TM) 64-Bit Server VM 1.8.0_172-b11 (Oracle Corporation)
- Scala version 2.12.4
- Java heap: used = 23 MB, free = 188 MB, max = 910 MB
- JOGL: (3D View not initialised)
- OpenGL Graphics: (3D View not initialised)

A.2 Hardware specifications

The hardware configuration of the computer on which the simulations were run are quoted below:

- 7 core processor
- Installed RAM of 8GB
- Operating system: Windows 10.0 (AMD64 processor)
- Frequency: 2.5GHz to 2.6GHz

Appendix B

Configuration of Zebro

This appendix is to elaborate on the properties of Zebros as mentioned in Section 2.2.1. The specifications in the section are documented when the problem statement of the thesis was conceived, as of January 2018. The newer additions to the Zebro design are quoted in grey coloured text. These additions are not considered in the thesis.

B.1 Technology Readiness Level

The technology readiness level (TRL) is used to measure the stage of development of the product/technology. TRL is a helpful knowledge-based standard and a shorthand for evaluating the maturity of technology or invention. TRL is an integer between 0-10 [48].

- TRL of communication module: 0
  At the beginning of the thesis, the communication module was being implemented. That module failed to give the desired results. The Zebro team is looking for an alternative communication module.

- TRL of locomotion module: 4.
  Zebro can walk, so the critical function of moving through rough terrains using C-shaped legs has reached the proof of concept. Still, many improvements need to be made. Design considerations must be applied at a fundamental level like revisiting decisions on motor type and leg detection methods [48].

- TRL of localisation and swarming: 1
  No localisation technique is available. Simulations of Zebros swarming using some flocking algorithms have been performed [27][22]. However, these algorithms have not yet been implemented on the Zebros.

B.2 Communication module

The DW1000 ultra-wideband transceiver has been used for communication and ranging measurements, and a communication layer called Anarchic TDMA (AN-TDMA) has been implemented to support Tangolation. The DWM1001-DEV development board from Decawave is used, which combines a nRF52832 ARM Cortex-M processor, a DW1000 transceiver for ultra-wideband communication, and a 6.5GHz UWB channel 5 antennae. Below are some specifications of the communication module:

1. **Local timer**: A 16MHz hardware timer local to the nRF52832 microprocessor.

2. **DW1000 system timer**: The 125 MHz system timer of the DW1000 transceiver. The transceiver can be programmed to transmit a message at a specific time-stamp.

3. **Local timer synchronisation**: The hardware timers of the nRF52832 microprocessor are used to predict the system time of the DW1000.

4. **Amount of transmittable information**: The amount of transferable data per second depends on the size and density[1] of the swarm. For instance, for a swarm of 16 robots, 32 transmissions per robot per second are possible. However, with 32 robots only 16 transmissions per robot per second are possible.

5. **Data rate**: 20 kb/s to 250 kb/s

6. **Update rate**: 1 Hertz

7. **Maximum swarm distance**: 60 meter

8. **Minimum swarm distance**: 1 meter

The concept of TDMA (Time Division Multiple Access) is used where agents waits for their turn. When the swarm is small or the density is low, Zebros can send much more often than when there are many Zebros close by [56].

Presently, the module is not working, due to magnetic inferences caused by the motor of the Zebro, and other magnetic objects in vicinity. Alternative solutions like removing such errors or replacing the communication module altogether are being considered.

[1] robots per unit area

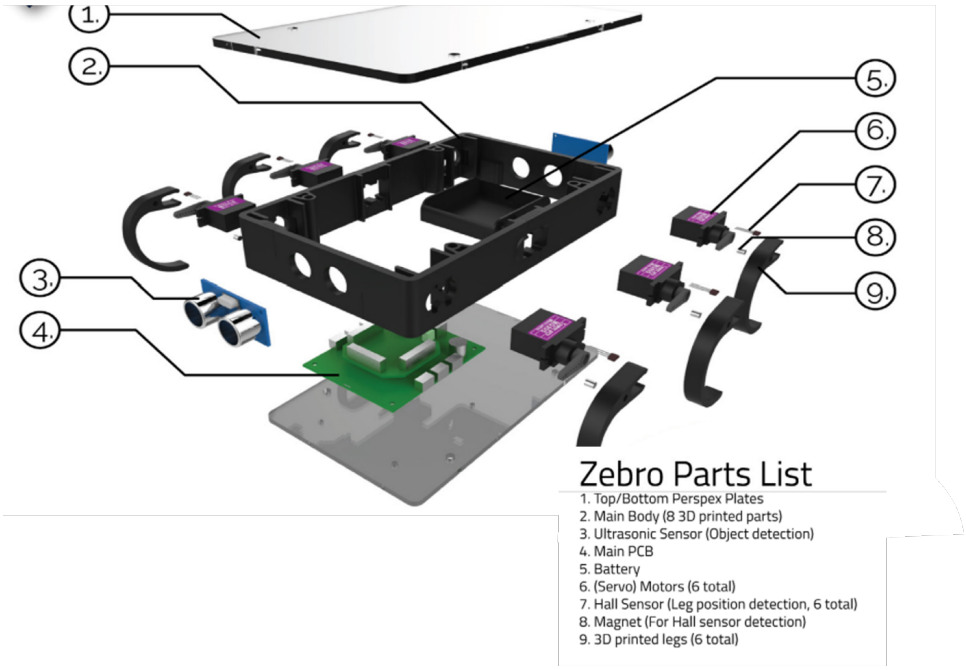


Figure B.1: Zebro Design [48]

B.3 Movement parameters

The following are some specification of the movement properties of the swarm [40] [21].

1. Speed: 1-2km/hr
2. Acceleration: zero
3. Walking time: 3 hr
4. Movement direction: Forward

There are six DC motors each with running at 170 RPM, 22.04kgf/cm, rated for 12V with 3.8A stall current.

B.4 Physical structure

Zebro has a modular design. The internal structure of the Zebro is shown in Figure B.1. The physical measurements and the cost of the DeciZebro is given below [48]:

1. Dimensions: 268 * 210 * 160 (in mm)
2. Radius of legs: 50 (in mm)
3. Maximum climb height: 120 (in mm)
4. Weight bearing capacity of Zebro: 400 grams
5. Total weight: 1800 grams
6. Price per Zebro, series 1: € 402,-

7. Price per Zebro, series 100: €230,-
8. Estimated per Zebro, 1000+: €150-200,-

B.5 Processor

The specifications of the computation unit in the Zebro are given below:

1. Board: Raspberry Pi Zero Wireless (v 1.3) with a broadcom processor
2. Processor: ARMv6 CPU (BCM2835)
3. External memory: MicroSD card with a capacity of at least 8 GB.
4. Power requirement: 5-5.25V
5. Operating frequency: 1GHz
6. Processing memory: 512MB RAM
7. CPU: Single-core

B.6 On board sensor

The two sensors on the Zebro are [21]:

1. Ultrasound sensor
2. Accelerometer
3. Camera

B.7 Power requirement

The power requirements for the system are described below.

1. Locomotive system: >12V, 10 A
2. Zebro controllers: 5V, 3A
3. On board battery: 14.4V, 4000 mAh

Appendix C

Significance tests

The significance tests of Table 4.7 as shown here. The data is presented in the same organisation as of Table 4.7.
The category numbers on the X axis represent the following:

- 1 $\Longrightarrow$ Ticks for silent
- 2 $\Longrightarrow$ Time for silent (in ms)
- 3 $\Longrightarrow$ Ticks for immature
- 4 $\Longrightarrow$ Time for immature (in ms)
- 5 $\Longrightarrow$ Ticks for mature
- 6 $\Longrightarrow$ Time for mature (in ms)

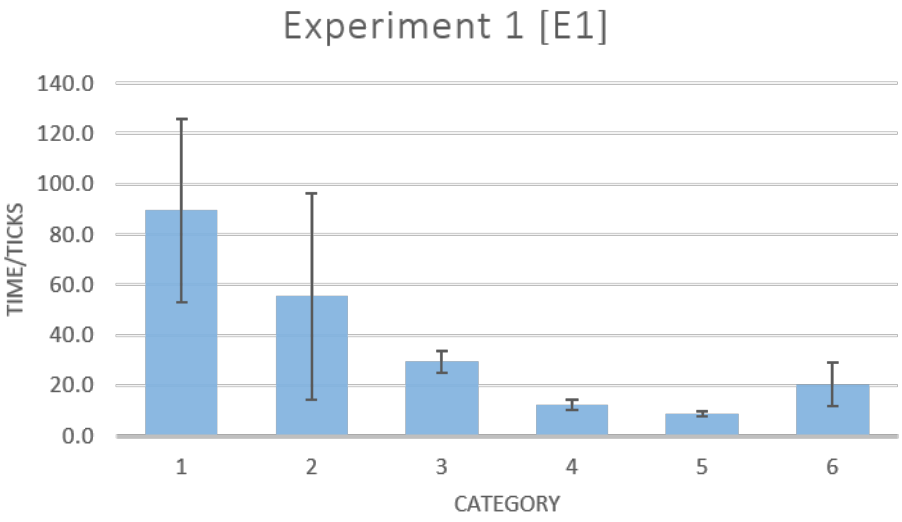


Figure C.1: Significance test for Experiment 1

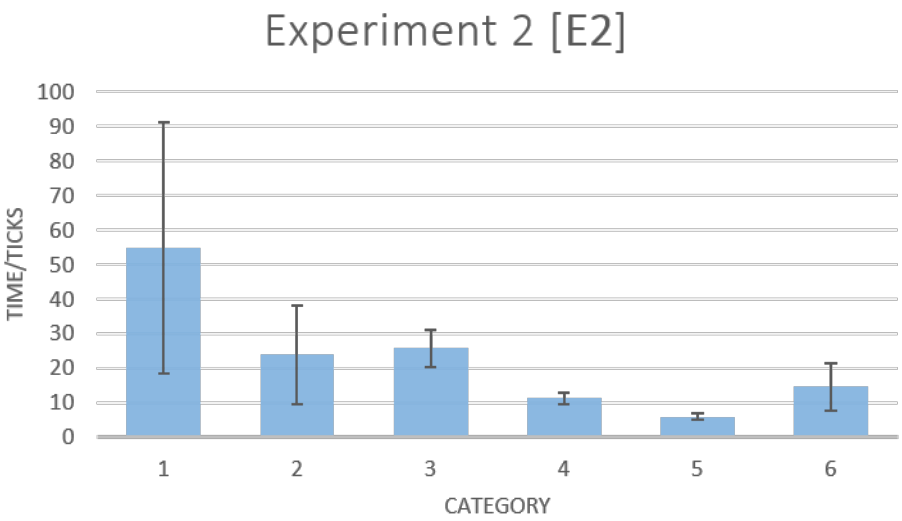


Figure C.2: Significance test for Experiment 2

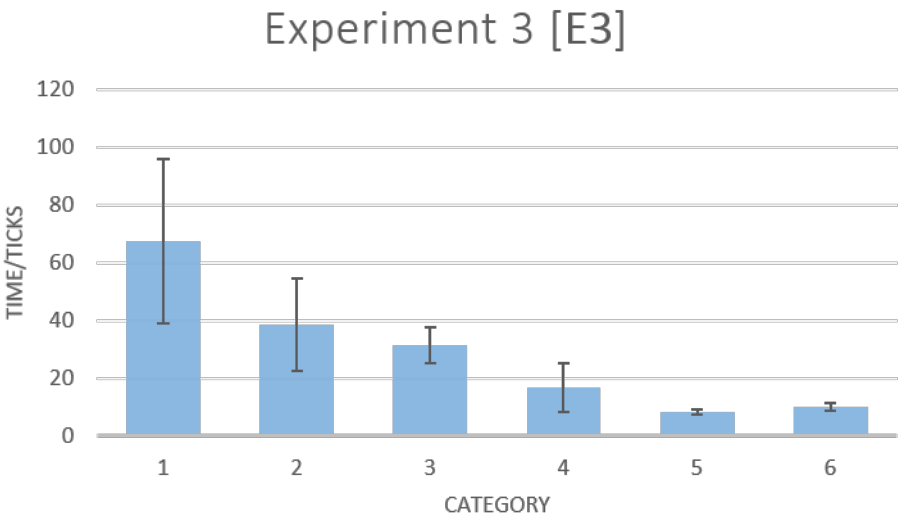


Figure C.3: Significance test for Experiment 3

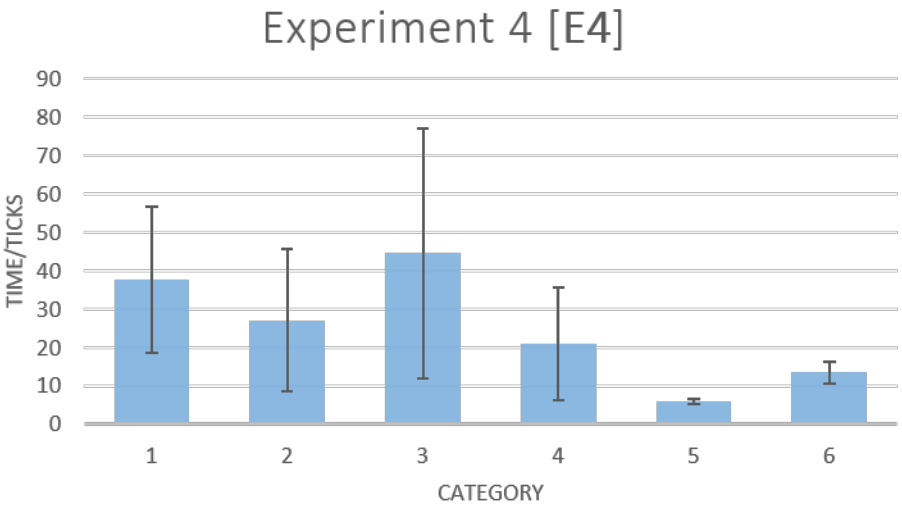


Figure C.4: Significance test for Experiment 4

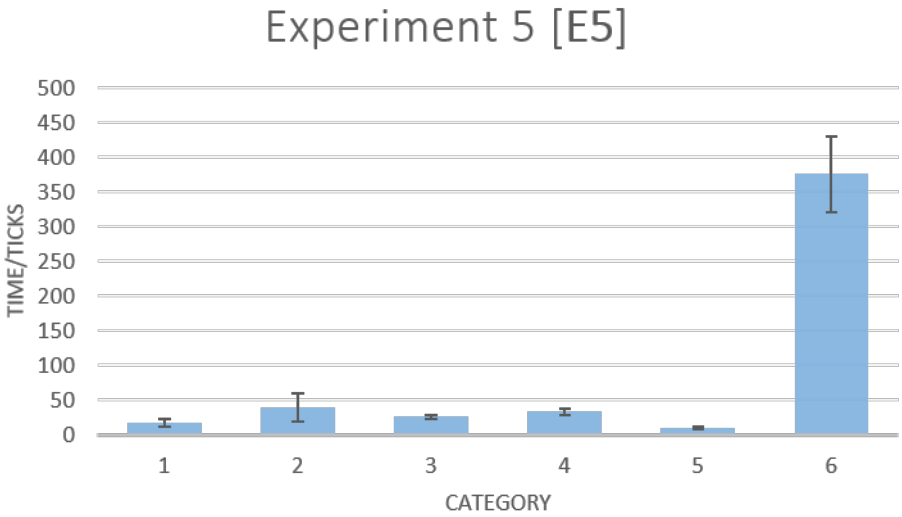


Figure C.5: Significance test for Experiment 5

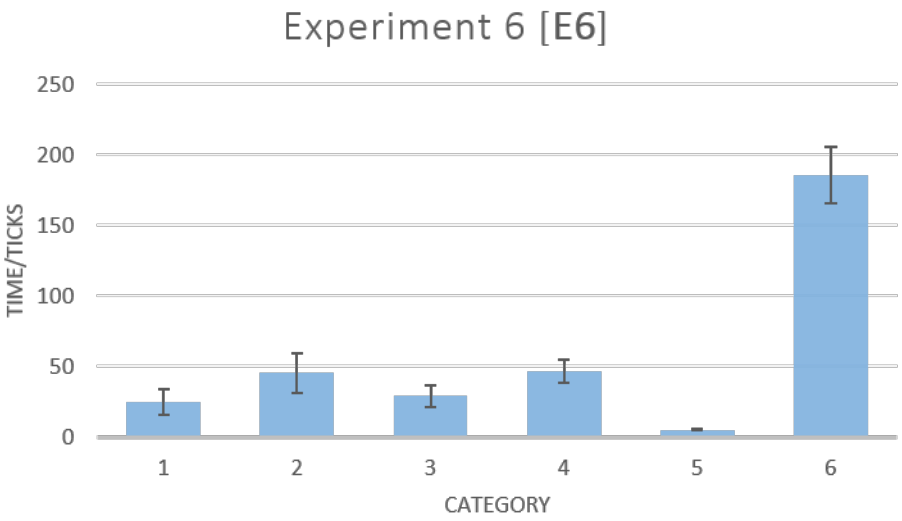


Figure C.6: Significance test for Experiment 6