UNIVERSITY OF SOUTHAMPTON

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING

Electronics and Computer Science

Exogenous Fault Detection And Recovery Solutions in Swarm Robotics

by

Adil Khadidos

Thesis for the degree of Doctor of Philosophy

Supervisors:
Dr Richard M. Crowder
Dr Paul H. Chappell

May 2017

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF PHYSICAL SCIENCES AND ENGINEERING
Electronics and Computer Science

Doctor of Philosophy

EXOGENOUS FAULT DETECTION AND RECOVERY SOLUTIONS IN SWARM ROBOTICS

by Adil Khadidos

A robotic swarm needs to ensure continuous operation even in the event of a failure of one or more individual robots. If one robot breaks down, another robot can take steps to repair the failed robot, or take over the failed robot's task. Even with a small number of faulty robots, the expected time to achieve the swarm task will be affected. Observing failure detection techniques requires an investigation of similar techniques in insects. The synchronisation approach of fireflies is an exogenous failure detection technique. This approach requires all the robots in the swarm to be initially synchronised together in order to announce a healthy status for each individual robot. Another exogenous failure detection approach is the Robot Internal Simulator. The concept of this approach is to have robots that are capable of detecting partial failures by possessing a copy of every other robot's controller, which they then instantiate within an internal simulator on-board to be run for a short period of time to predict the future state of the other robots. The work in this research draws inspiration from both approaches, which both still have several issues when they are implemented in swarm robotics. The enhanced technique developed in this research will depend on the input and output values in the robot's controller to diagnose other robots within the swarm during the entire swarm operation. In this research, communication plays an important part of the diagnosis procedure. While robots retain possession of their own controller values including their co-ordination, the receiver computes the distance between them based on the signal strength. A fault suspicion is generated if the computed distances do not match and an acknowledgement of the failure will be broadcast to the robotic swarm. This research explores the performance of the simulation experimental results. It has shown that failed robots are rapidly detected failures using the proposed technique. A mitigation procedure takes place after the faulty robot is shut down, either by pushing it away or allowing it to work as a communication bridge to operational robots.

Contents

List of Figures

List of Tables

Declaration of Authorship

I, Adil Khadidos , declare that the thesis entitled *Exogenous Fault Detection And Recovery Solutions in Swarm Robotics* and the work presented in the thesis are both my own, and have been generated by me as the result of my own original research. I confirm that:

- this work was done wholly or mainly while in candidature for a research degree at this University;
- where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated;
- where I have consulted the published work of others, this is always clearly attributed;
- where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work;
- I have acknowledged all main sources of help;
- where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself;
- parts of this work have been published as: listed in Section 1.5.

Signed:..

Date: 30 May 2017 ....

Acknowledgements

Firstly, I would like to express my sincere gratitude to my supervisor Dr. Richard M. Crowder for the continuous support of my Ph.D study and related research, for his patience, motivation, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better supervisor and mentor for my Ph.D study.

My sincere thanks also goes to Dr. Paul H. Chappell, who provided me an advice and insightful comments and encouragement throughout my work on the thesis.

Besides my supervisors, I would like to thank the viva committee: Dr. Victor Sanchez and Dr. Gary Wills, for their insightful comments and encouragement, but also for the hard question which incentivised me to widen my research from various perspectives.

I would especially like to thank King Abdulazez University for giving me the opportunity and the support for continuing my postgrad study overseas.

Last but not the least, I would like to thank my twin brother, Dr. Alaa Khadidos, and my best friends, including Dr. Khaled Alyoubi, for supporting me spiritually throughout writing this thesis and my life in general.

Chapter 1

Introduction

Robotic swarms with collective intelligent behaviour are likely to have advantages over single-robot systems. These advantages are provided by robustness, scalability, flexibility, adaptability and most importantly, reliability. As part of the efficiency in cooperation between swarm robotic members, a failure among individual robots may have an impact on the performance of the entire swarm's behaviour.

Identifying failures in swarm robots requires a number of detection mechanisms. A failure in the individual robot's hardware or software, such as communication, navigation, sensors and actuators could have an impact over the entire swarms behaviour. The main goal of detecting and identifying a specific failure is to determine faulty robots and to select the best technique to distinguish which robot has to be mitigated and recovered.

The work reported in this thesis will concentrate on simulation experimental analysis and evaluation of swarm robotics from the perspective of the effect of the impact of failures to individual robots and the effectiveness of the mitigation techniques.

During the simulation experiments in this thesis, the number of robots could have technical advantages with a small group, but the research purpose is to find solutions to practical swarm problems. However, the simulation experiments, during this research, will consider the swarm robots as a small group of robots in order to solve problems using a minimal amount of resources.

1.1 Swarm Robotics Concept and Definition

In order to understand swarm robotics, a number of concepts have to be defined.

1.1.1 Swarm Intelligence

Swarm intelligence is the term that is embraced by social insects and by optimization studies researchers used to describe the social insect metaphor, though losing much of its original robotics context (Bonabeau et al., 1999). The concept of swarm intelligence was employed in artificial intelligence, where the expression was introduced by Beni and Wang (1993) in the context of cellular robotic systems.

Swarm intelligence systems are typically made up of a population of simple individuals interacting locally with one another, and with their environment. The inspiration often comes from nature, especially biological systems.

1.1.2 Swarm Robotics

Swarm robotics are based on the principles of swarm intelligence, which emphasise self-organization and distribution in a large number of robots. With homogeneity and simplicity as design goals at the individual unit level, the main advantages of the swarm approach lie in the properties of scalability, adaptivity and robustness (Liu and Winfield, 2010). Table 1.1 shows the definition of each property of the swarm robotics approaches.

Figure 1.1 summarizes the characteristics of robotic systems including swarm robotics. An agent-based system has the capability of autonomous action in the environment in order to meet its design objectives (Jennings and Wooldridge, 1998). Single robot is limited in information processing capability and many other aspects, thus the cooperation of multi-robots are needed to complete the task in an efficient way. In most cases, to conduct a study or to do a task, it's preferable to use teamwork to complete the task efficiently. Multi-robot systems have made great progress and achieve great success in many areas (Yuan et al., 2007; Shi et al., 2012). Cooperative multi-robot systems depend on the individual robot very much, for further enhancing the robustness and scalability of the system. The action of a group of small robots in which each individual robot performs a simple role produces complex behaviour as a whole (Hinchey et al., 2007), thus swarm robotic system was born.

As with many technologies, there is no formal definition of swarm robotics that is universally agreed upon; however, some characteristics have been generally accepted (Higgins et al., 2009a). Swarm robotics was defined by Şahin (2005) as *"the study of how a large number of relatively simple physically embodied agents can be designed such that a desired collective behaviour emerges from the local interactions among the agents and between the agents and the environment"*.

The definition of swarm robotics can be expanded further using a number of criteria, as reported by Higgins et al. (2009b):

Table 1.1: Definition of key properties of swarm robotics.

Property	**Definition**	**References**
Robustness	Is the ability to cope with the loss of individuals. In social animals, robustness is promoted by redundancy and the absence of a leader.	Brambilla et al. (2012)
Scalability	Is the ability to perform well with different group sizes. The introduction or removal of individuals does not result in a drastic change of the performance of a swarm. In social animals, scalability is promoted by local sensing and communication.	Şahin et al. (2008)
Flexibility	Is the ability to cope with a broad spectrum of different environments and tasks. In social animals, flexibility is promoted by redundancy, simplicity of the behaviours and mechanisms such as task allocation.	Brambilla et al. (2012)
Adaptivity	Means that either an individual changes itself if necessary or the whole system changes itself. On an individual level, it means that an individual changes its own behaviour. Even if the individuals do not change, the system as a whole may still be adaptive.	Schut (2010)
Reliability	Is the probability that a robot or the entire swarm robotics system will perform its prescribed duty without failure for a specified period of time when operated correctly in a specified environment.	Winfield and Nembrini (2006)

- *Autonomous Robots*: Robots should have a physical interaction with the environment.
- *Swarm Size*: The availability of a large number of robots in the swarm.
- *Few Homogeneous Groups of Robots*: There should be relatively few groups containing large numbers of homogeneous robots.
- *Functionality*: The robots should be relatively simple and incapable such that the tasks tackled require the co-operation of individual robots.
- *Communication Capabilities*: Robots should only have localised and limited sensing and communication abilities. This constraint ensures that the coordination between the robots is distributed.

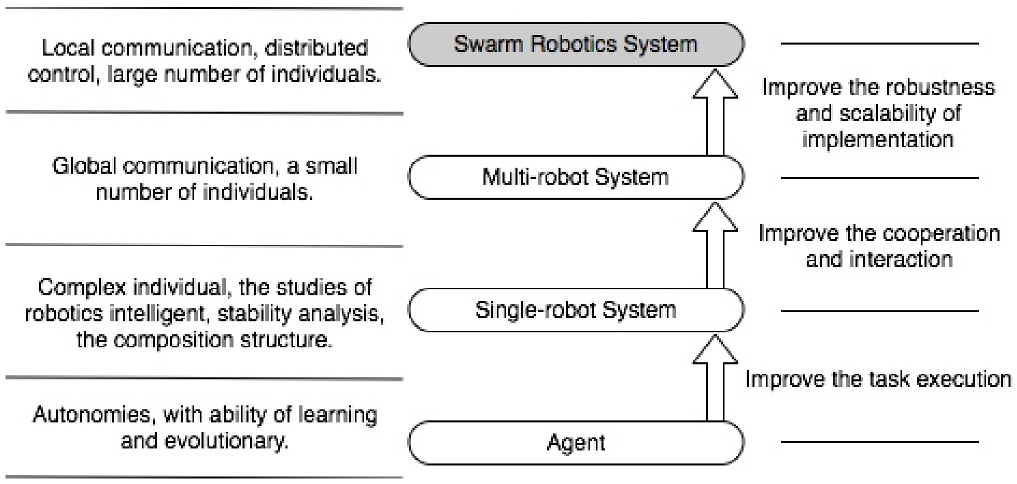

Local communication, distributed control, large number of individuals.
Swarm Robotics System
Improve the robustness and scalability of implementation
Global communication, a small number of individuals.
Multi-robot System
Improve the cooperation and interaction
Complex individual, the studies of robotics intelligent, stability analysis, the composition structure.
Single-robot System
Improve the task execution
Autonomies, with ability of learning and evolutionary.
Agent

Figure 1.1: The characteristics of various robot systems. The figure shows how moving from a single robot to a robotic swarm can improve performance. An agent is an autonomous individual with the ability of learning. A single-robot system is the improvement from an agent system in task execution. Enhanced cooperation and interaction in each robot leads to the ability of global communication between a small number of individuals. Then, the swarm robotics system emerges with improved robustness and scalability between a large number of individuals. Adopted from Shi et al. (2012).

1.2 Issues Arising in Swarm Robotics

It should be noted that a typical robotic swarm will contain a considerable number of individual robots. Hence a failure of a small number of robots will not lead to the swarm failure.

A robotic swarm needs to ensure continuous operation in the event of failures in an individual robot. Even with a small number of faulty robots, the expected time of achieving the task will be affected. Faulty robots will become lost and they will simply become obstacles that other functional robots in the swarm need to avoid (Winfield and Nembrini, 2006). However, the performance will be seriously affected if there are many faulty robots in the swarm, which will lead the robotic swarm into stagnation (Ismail and Timmis, 2010), whereas the entire robotic swarm will fail to achieve its given task.

Determining the perfect swarm size to achieve maximum reliability for fault tolerance requires additional explication, because this field has not been precisely considered for differences between multi-robots and swarm robotic systems. However, a reliability calculation is most important and a critical component when fault tolerant optimal swarm robotic design is required.

1.2.1 The Importance of Reliability in Swarm Robotics

As in many multi-robot systems, increasing task reliability and decreasing total cost to complete the system tasks are considered as one of the system advantages, (Zhu and Yang, 2010). Comparing with swarm robotic systems, the structure design of multi-robot systems (which is determined by global communication) has little attention to the system scalability, whereas the structure design of swarm robotic systems (which is determined by local communication between robots) has an advantage over the system scalability.

The determination of the multi-robot system scalability, in relation with the number of individual robots has been observed by Balch and Hybinette (2000). Their simulation experiment concentrated on evaluating the performance of the scalability approach in multi-robot systems, where the task was described as each robot is drawn to attachment sites arranged with respect to its neighbour to be driven loosely on molecular crystal formation in the arena. The first simulation experiment consisted of a multi-robot system with group size between one to eight simulated robots. With global communication applied, the performance evaluated was successful for achieving the multi-robot system task. Then, they observed another simulation experiment with a large team composed of 32 simulated robots which showed that the approach is scalable only when each robot relies on locally available information, namely, the locations of nearby robots. Global communication is not required, instead, local sensors (perhaps vision) can be utilised to generate effective formation behaviour in large robot teams.

Swarm robotic systems are considered scalable in terms of a reliable number of operational robots. However, in order to achieve the swarm task, a number of operational robots are required to achieve the swarm task.

The required number of individual swarm robots to achieve the swarm task can be determined by applying the constraint k-out-of-n redundancies approach (where k represents the number of the operational robots needed to accomplish the swarm task, and n represents the number of all the robots available in the swarm), (Sooktip et al., 2011). Despite that the k-out-of-n redundancies approach could consider the minimum number of robots needed to achieve the swarm task, the main concept is to draw attention to how the swarm robotic system would perform, even with a low number of faulty robots.

1.2.2 Detecting and Isolating a Robot with a Failure

Microbiological systems, including cells, organisms, and groups that possess a large number of subunits are inspiration for swarm robotics. Microbiological properties provide robustness and reliability to their own system (Ismail and Timmis, 2010). The approach of these systems undertake in order to detect faulty members and protect the entire system from the impact caused by these members. The objective of such protection is to isolate the faulty member while active members continue to perform the swarm task. This mechanism has been described by Timmis et al. (2010) as an immune system, which will be discussed more during this research.

In the context of swarm robotics, at the time a robot detects a failure, this faulty robot will broadcast a signal to other robots in order to move towards that faulty robot. Then, the next procedure is to isolate and mitigate the impact of the faulty robot to maintain the performance of achieving the robotic swarm's task (Winfield and Nembrini, 2006).

1.2.3 Mitigating and Recovering the Swarm Failures

The main objective of isolating a faulty robot is that operational robots are supposed to recover faulty robots so that the faulty robots will become active again and continue the tasks they were doing before the fault was identified (Ismail and Timmis, 2009, 2010).

Recovery procedures depend on the type of failure that has occurred. For instance, with the existence of communication, if a robot has a distance sensor failure (where values in the controller show that the robot is not moving and the distance sensors give no reading), it broadcasts a signal indicating that it has a sensor failure. Then other robots, that are working to isolate this faulty robot, have to take an action to either repair or unite with this faulty robot in order to complete the swarm task.

During this research, there will be more testing and evaluating of many failures related to navigation, sensors, actuators and communication. This evaluation will help to develop more mitigation and recovery features in swarm robotics.

1.3 Summary of Main Contribution

This research contributes to the area of failure detection and recovery solutions for swarm robotics. Specifically, it introduces novel failure detection and recovery model and evaluation analysis to verify the performance of the swarm robotics behaviour after applying the proposed model.

A number of failures in swarm robotics have been addressed in Section 2.1.2, but failures at hardware-level was determined to be the most common examples covered by a number of researchers. However, the use of the proposed failure detection and recovery model can provide useful insight into the understanding of the reliability property leading to the instantiation of this property to swarm robots foraging tasks.

The task of the swarm robotics, during the simulation experiments, has been affected with the presence of hardware failure. Therefore, applying the failure detection and recovery model to the swarm robotics shows an enhancement to the swarm performance and the swarm task has not been compromised anymore. The failure detection and recovery model was built to implemented in the robots' controller alongside with the main swarm task function in a way to not affect the robots' controller processing while it is doing the given task.

The main contribution of this thesis is the proposed failure detection and recovery model which was built in a way to not just detect the failure and recover it, but also is used to address the exact device that has failure. This will help in future to allocate more recovery procedures for swarm robotics.

Another main contribution of this thesis is the proposed communication relays, as a recovery procedure, among swarm robots. That allow the faulty robots to become at use to the operational robots. The failed swarm robots are able to reuse the remain energy power to operate the communication device to the robot to rebroadcast messages among the operational robots. To the best of the our knowledge, none of the previously designed swarm robotics has ever used the failed robots as a communication relays concept in order to increase the swarm robotics performance.

The collected results demonstrates that the proposed failure detection and recovery model allowed swarm robotics with individual failures to achieve the swarm task with not problems. In addition, and compared with simulation experiments before applying the proposed failure detection and recovery model, the results summarised that the proposed model helped to decrease the time from the swarm robots to finish the task in short time with less distance-travelled and less energy-expended. This could allow swarm robots to carry another task in raw in the future if possible.

1.4 Thesis Structure

The structure of this thesis is as follows, and is shown in Figure 1.2:

Chapter 1 provides an introduction to swarm robotics, including its properties and types of failures. It also introduces swarm intelligence.

Chapter 2 reviews literature relevant to this research. It first provides a background view and describes the decentralised mechanisms in failure detection approaches, including identification and analysis of failures. This is followed by a detailed description of different research experiments and challenges of the immune systems inspired by the Granuloma Formation. A current recovery approach is represented as a recovery solution for swarm robotic failure. Next, there is a discussion of the revised techniques that have been covered during this chapter. After that, the motivation and the research objectives, detailing an inspiration for the research topic and the areas that the research in swarm robotics can cover is presented. Finally, the research questions are presented.

Chapter 3 highlights the simulation experiments methodology and the simulation environment that is used during this experiment.

Chapter 4 presents the evaluation mechanism of the simulation experiment and a discussion of the experimental results.

Chapter 5 this chapter presents a discussion on the evaluation simulation experiment from the previous chapter for choosing the best swarm size for carryout the simulation experiments in this chapter. It investigates failure detection schemes to diagnose failures in order to recover/mitigate the robots with failures in the swarm.

Chapter 6 discuss an investigation into the performance of the swarm robotics with a complex task. At the beginning, a simulation experiment was deployed to evaluate the best swarm size with a large arena.

Chapter 7 discuss the performance of the simulated swarm robotics that was observed before and after implementing the exogenous failure detection model to a number of individual robots with the complex task. Also, it investigates the swarm behaviours after applying the mitigation procedure to faulty robots, by pushing them to the side of the arena.

Chapter 8 discuss another complex simulation experiment with a different method of applying mitigation solutions to the faulty robots. This requires faulty robots, with enough power energy to rebroadcast messages between operational robots over long distances within the simulation swarm robotics.

Chapter 9 highlights the discussions through the research in the thesis. The Chapter discusses the research questions presented in Section 2.10. It, also, discuss a specific details about the proposed exogenous failure detection model. Also, it will discuss, deeply, the mechanism for the failure diagnosis in specific devices in the E-Puck robot, and the how they could be mitigated or recovered.

Chapter 10 presents the conclusion of the research and the future work plans.

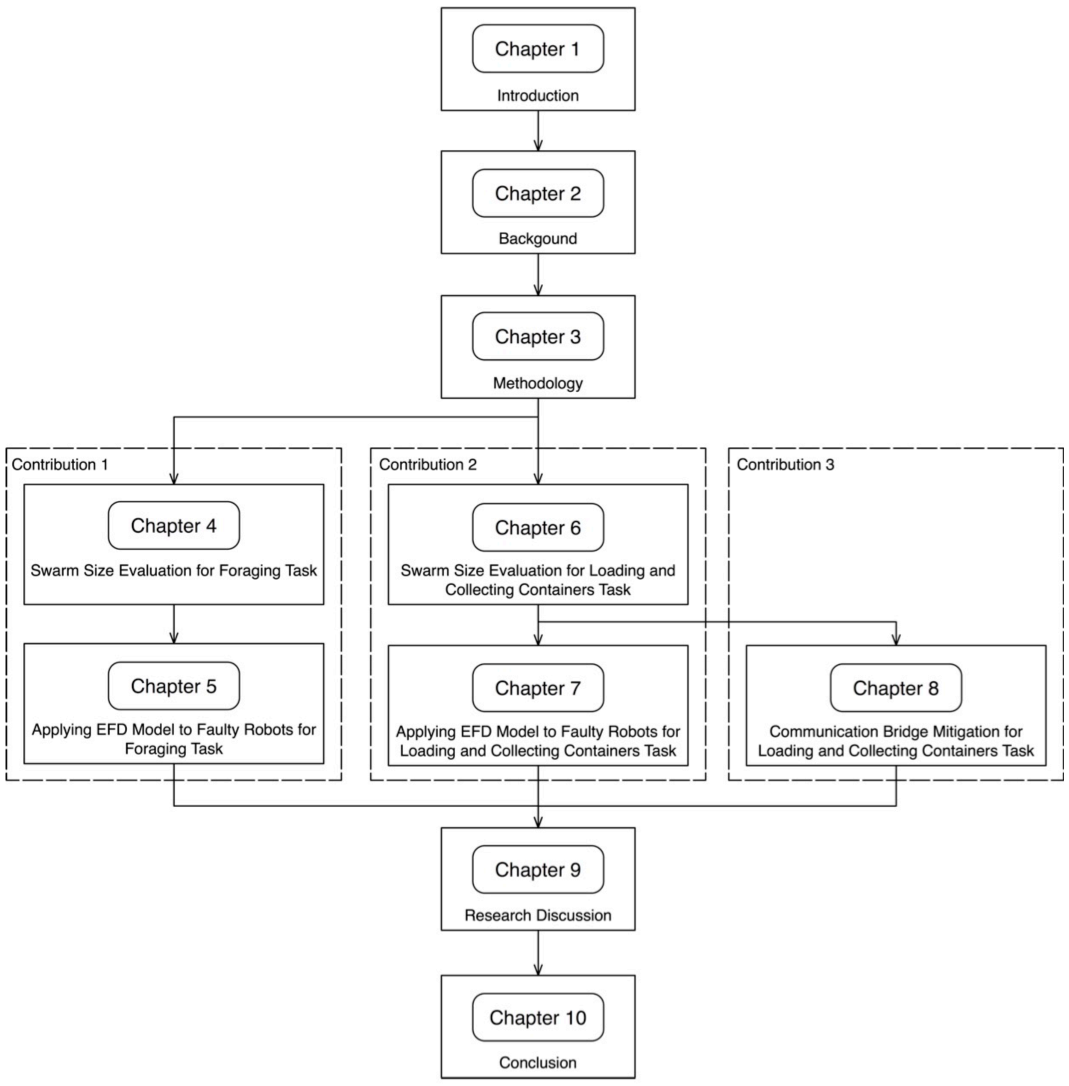

Chapter 1
Introduction
Chapter 2
Backgound
Chapter 3
Methodology
Contribution 1
Chapter 4
Swarm Size Evaluation for Foraging Task
Chapter 5
Applying EFD Model to Faulty Robots for Foraging Task
Contribution 2
Chapter 6
Swarm Size Evaluation for Loading and Collecting Containers Task
Chapter 7
Applying EFD Model to Faulty Robots for Loading and Collecting Containers Task
Contribution 3
Chapter 8
Communication Bridge Mitigation for Loading and Collecting Containers Task
Chapter 9
Research Discussion
Chapter 10
Conclusion

Figure 1.2: The structure of chapters in the thesis. The introduction to this work is presented in Chapter 1. Then followed by Chapter 2 that discuses the related work. Chapter 3 discusses the methodologies related to the simulation experiments in this thesis. The research of this thesis is mainly consists of the three contributions, which they are emphasised with the dotted boxes. The results collected from experiments in Chapter 4 used to evaluate the experiments discussed in Chapter 5 after the determination of the suitable swarm size. The results collected from Chapter 6 are used to evaluate the experiments in Chapter 7 after detecting the suitable swarm size. The evaluation of experiments in Chapter 8 rely on the results collected in Chapter 6 for the evaluation purposes. Chapter 9 discusses the approaches of the proposed Exiguous Failure Detection and Recovery model. Finally, the thesis conclusions and the future work are discussed in Chapter 10.

1.5 Publications

The work in this thesis has led to the following publication of a paper to the following conferences:

- Khadidos, A., Crowder, R. M., and Chappell, P. H. (2015a). Enhancing exogenous fault detection in swarm robotics by analysing transferable data. In Alford, N., FrÃ, J., et al., editors, *Proceedings of the Eighth Saudi Students Conference in the UK*, pages 385–395. World Scientific
- Khadidos, A., Crowder, R. M., and Chappell, P. H. (2015b). Exogenous fault detection and recovery for swarm robotics. *IFAC-PapersOnLine*, 48(3):2405–2410

Chapter 2

Background

It is widely recognised that insects coordinate their actions to accomplish tasks that are beyond the capabilities of a single individual. Swarm robotics was inspired from the observation of social insects, such as ants and bees, which stand as examples of how a large number of relatively simple individuals, with only local interactions, can interact to create collectively an intelligent system (Şahin, 2005).

In swarm robotics, a relatively large number of simple robots can perform difficult tasks that are beyond the capability of an individual. However, swarm robotics still have a limited reliability, especially at the hardware level. Even though the quality and robustness of the hardware is increasing, hardware failures are still quite common. So, achieving reliability in the swarm robotics is considered important to assure the robustness of the swarm system (Brambilla et al., 2012). Achieving reliability at the hardware-level in swarm robots requires to identify and address different failures and their effect on individual robots, as well as the entire swarm robotics performance, which includes the effect on simulation time for finish the swarm task. The objective of achieving reliability at the hardware-level is to develop detection mechanisms that identify the fault, as well as to determine faulty robots in the swarm and selecting the best technique to distinguish which robot has to be mitigated and recovered (Parker, 2012).

2.1 Fault Detection

Individual robots in the swarm may not need to diagnose each problem that arises during the task, such as failure of proximity sensors or communication failure, but the entire swarm system must have some means for compensating for a failure of a member of the swarm, so that the swarm team can continue to successfully meet their application objectives, (Parker, 2012).

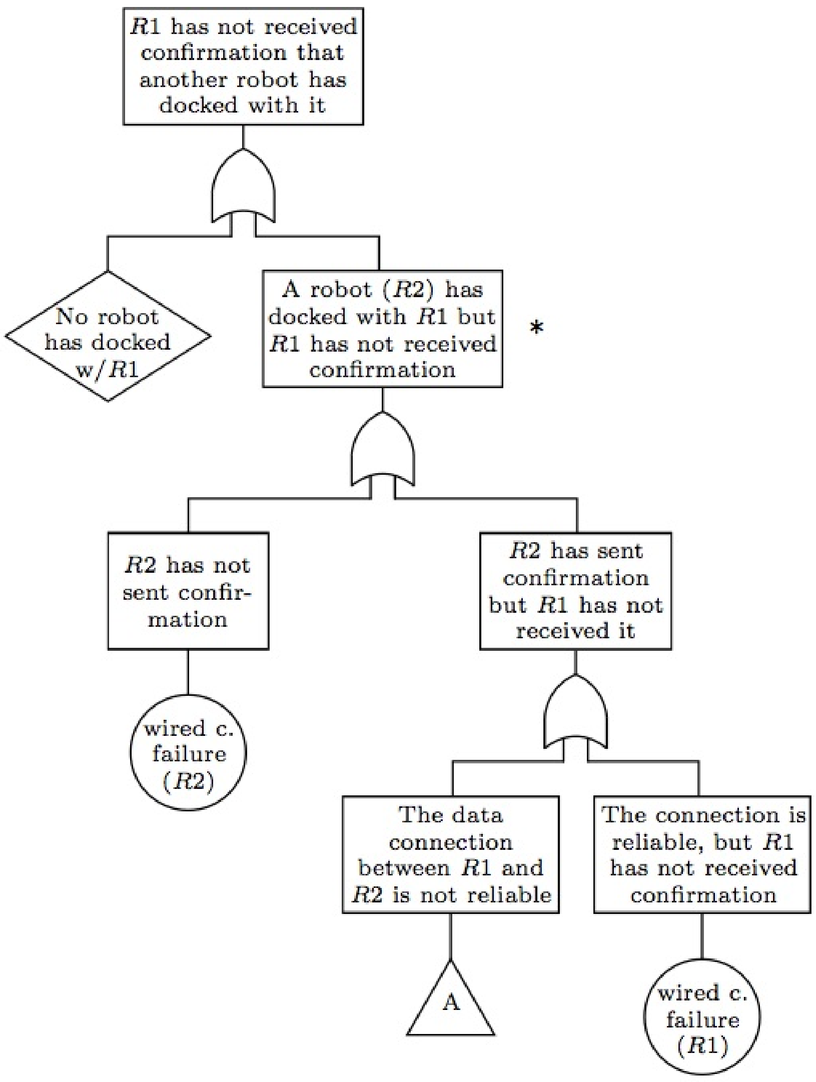
Parker (2012) made the assumption that swarm robotics failures may be categorized into three types, *known* faults, *unknown* faults and *undiagnosable* faults. During the design stage, it is important to ensure that the swarm robotic system is properly designed to overcome failures when they occur in order to complete the task. Hence, as part of the design process, a formal fault analysis needs to be undertaken, using techniques such as Failure Mode and Effects Analysis (FMEA), (Winfield and Nembrini, 2006), and Fault Tree Analysis (FTA), (Visinsky et al., 1994), where FMEA and FTA are accepted approaches by engineering.

The Failure Mode and Effects Analysis (FMEA) technique requires local wireless connectivity information to achieve swarm robotics objectives. The conceptual idea of this approach lies in broadcasting the robot's ID and the IDs of its immediate neighbours. If a robot loses a connection to a particular robot, and the number of remaining connected neighbours is less than a threshold, then the current robot assumes it is moving out of the swarm and turns around. Under the Failure Mode and Effects Analysis (FMEA) approach, the system designer attempts to identify all of the internal hazards (e.g., faults in robots or robot subsystems, including motor failure, communications failure, sensor failures, and control system failures) and external hazards (e.g., environmental disturbances, including obstacles and unstructured terrain, and communications noise) that threaten a robot's objectives, (Winfield and Nembrini, 2006).

In contrast to Failure Mode and Effects Analysis (FMEA), the Fault Tree Analysis (FTA) technique is a deductive method in which failure paths are identified using a fault tree drawing or graphical representation of the flow of fault events (Bloch and Geitner, 1990). Each event in the tree is a component failure, an external disturbance, or a system operation. The concept of the Fault Tree Analysis (FTA) approach requires that the analyst attempts to identify all of the potential causes of the event and the logical sequence of secondary events leading up to it, these together make up the fault tree, Figure 2.1, (Murray et al., 2012). The relationships between events at different levels in the tree are specified using boolean logic, where the outputs of functions at lower levels serve as the inputs to those at higher levels.

However, in the case of *unknown* and *undiagnosable* failures, the swarm robotics must have the means to detect encountered problems. The *unknown* faults cannot be anticipated by the designer, during the design stage, but can be diagnosed by the swarm system based on experience and available sparse information; and *undiagnosable* faults that cannot be classified autonomously and need human intervention, (Parker, 2012).

There are a number of techniques for detecting failures in robotic systems, especially when it comes to the perspective of swarm robotics (Parker and Kannan, 2006). In this research, fault detection approaches have been divided into collective behaviour data and health status over communication.



R1 has not received confirmation that another robot has docked with it
No robot has docked w/R1
A robot (R2) has docked with R1 but R1 has not received confirmation
*
R2 has not sent confirmation
R2 has sent confirmation but R1 has not received it
wired c. failure (R2)
The data connection between R1 and R2 is not reliable
The connection is reliable, but R1 has not received confirmation
A
wired c. failure (R1)

Figure 2.1: Fault tree examining the causes of a problem in the formation of a swarm behaviour. R1 and R2 represents two robots in the swarm that are awaiting confirmation of docking. Events are represented using rectangular boxes. At the top level of the fault tree, robot R1 is awaiting confirmation from other robots in the swarm. After analysis (asterisk mark where the fault analysis is occurring), the fault tree developer could consider that either no robot has docked with robot R1, or a robot (R2) has docked with robot R1 but for some other reason robot R1 has not received confirmation (These options are combined using an OR gate). There are two possibilities why robot R1 has not received confirmation from robot R2, either the connection is confirmed to be reliable, which means robot R1 has a wire failure, or the connection is unreliable, which will lead to further diagnosis (represented as a triangle with the letter A in the diagram). Adapted from Murray et al. (2012).

The collective behaviour data depends on predefined values of the input and output components, such as sensors and actuators. During robot operations, the controller system of each robot will keep track of its values. If the reading of the current value does not match the predefined value in the system, then the robot indicates a fault in one of its components (Parker, 2012). This mechanism will be discussed in the next subsection. However, one example of health status over communication detection is a Robot Health Signal, a commonly used technique in swarm robotics for detecting any failure (Parker, 2012). This technique depends on the communication between robots, where robots periodically broadcast health signals to their neighbours indicating their presence and the continuance of the operation. Similarly, the approach proposed by Christensen et al. (2009) for detecting failures is based on fireflies synchronisation. In this, all the robots are synchronised together to perceive the health status of other robots. If one robot is not synchronised, then it is assumed to be faulty and a response is initiated.

Researchers including Timmis et al. (2010); Lau et al. (2009); Halavati et al. (2007); Parker (2012) have observed and analysed these techniques and several others. They have been exploring the probabilistic aspects of how algorithms of both swarm intelligence and biological immune systems are similar from a scientific point of view. So, biological immune systems and swarm intelligence are complementary tools, and can be used effectively together to solve complex engineering problems.

2.1.1 Failure Analysis

Lau et al. (2011) explained a form of collective behaviour data for swarm robotics in the Error Detection and Recovery (EDR) paradigm. Error detection and recovery (EDR) consists of three stages: error detected in robot components, fault diagnosis, and recovery (Avižienis, 1967), Figure 2.2. Statistical analysis is one of the main concepts used for detecting failures in robots.

Lau et al. (2011) presented an approach using statistical results by using the adaptive data-driven error detection mechanism. Data-driven error detection approaches infer the presence of a fault through analysis of the operational data. This data consists of the number of collected objects, the energy used and the distance travelled, which are all collected during each control cycle. The control cycle describes the duration of the total number of objects that were collected over a fixed period of time. The results in their paper were based on a 250 second control cycle. Noting that the control cycle is different from the processor time step of the robot's controller which represents the controller duration for processing reading from sensors, then computes the outputs.

In this work, their initial experiment focused on the number of objects that were collected by one robot. The objective is to determine whether an error has occurred with statistical techniques using a sliding time window on the data to calculate some descriptive values, depending on the statistical technique of the sample, and to compare these values against those from the previous time window, Figure 2.3. The compared values of objects collected between cycle ten and the previous window, cycle nine, show a significant

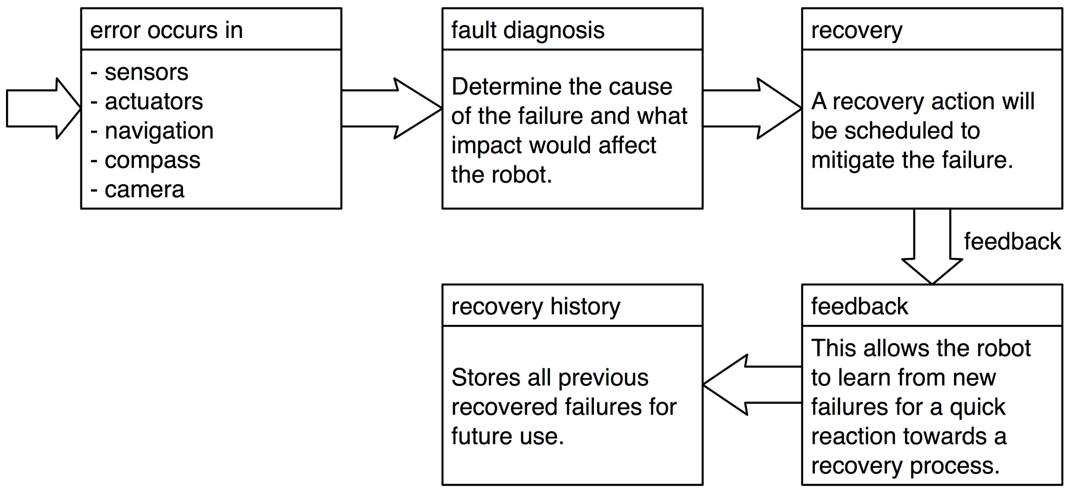
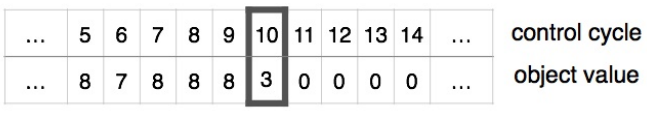

error occurs in
- sensors
- actuators
- navigation
- compass
- camera
fault diagnosis
Determine the cause of the failure and what impact would affect the robot.
recovery
A recovery action will be scheduled to mitigate the failure.
feedback
feedback
This allows the robot to learn from new failures for a quick reaction towards a recovery process.
recovery history
Stores all previous recovered failures for future use.

Figure 2.2: Stages in the Error Detection and Recovery (EDR) mechanism. When an error is detected and identified in a robot's component, based on the changes of predefined values of that component, the fault diagnosis mechanism starts to analyse the cause of the fault. This step also determines the nature and the exact location of the fault in the robot, based on predefined data. When a fault is identified, recovery measures can then be activated to carry out the process of dealing with the fault. If a fault still persists after recovery measures have been carried out, that information may be used as a form of feedback to the error detection and fault diagnosis mechanism for tuning and maintenance purposes, where the new fault data will be stored as recovery and maintenance history. The recovery history allows the robot to self-learn from previous failures, and the feedback can provide the swarm system developer with a new swarm robotics behaviour for future adoption. Adapted from Lau et al. (2011).

change which indicates the moment at which the robot failed to collect all the objects. Eight objects collected indicates a successful collection of all the objects, while three objects collected indicates failure to achieve the task. When these results are based on the activity of a single robot, it is uncertain whether this fault was influenced by the robot or the arena structure.



...	5	6	7	8	9	10	11	12	13	14	...	control cycle
...	8	7	8	8	8	3	0	0	0	0	...	object value

Figure 2.3: This example demonstrates the robot foraging task, which is to collect objects and return them to the home location. The number of objects are collected by the robot over a fixed period of time. A sample of a control cycle from five to fourteen cycles has been chosen in order to show where the fault occurred. The time window represents the control cycle and the collected objects. This figure shows that the collected objects in the time window at control cycle ten changes significantly, compared to the previous time window at control cycle nine. This indicates failure to collect the objects, where something has influenced the robot task. Adapted from Lau et al. (2011).

Detecting errors with data from a single robot is insufficient to differentiate between changes in data due to faults or to changes in the dynamic arena. However, Lau et al. (2011) advocated an alternative approach which includes data from other robots using local communication. Table 2.1 shows a data exchange between robot one and other robots within the communication range. Their important assumption was that the detection should determine whether a robot itself is faulty (self-detection) and not whether other robots are faulty.

Table 2.1: This table shows the data interaction through communication between objects collected by one robot and the other robots within the communication range. An example of control cycles five to fourteen has been chosen in order to show where the fault occurred. The objects collected in the table represent those collected by the entire swarm of individuals. Whenever an object is collected by a robot it informs the rest of the swarm of the total number of objects now collected by the swarm, then the other robots change their parameters according to the new change. Therefore, each robot will keep track of how many objects remain in the arena to be collected. Since the robots are mobile, it is probable that a robot might not interact with another robot during a control cycle, which is illustrated by blank cells in the table. Adapted from Lau et al. (2011).

	Objects collected (robot 1)	Objects collected (robot 2)	Objects collected (robot 3)	Objects collected (robot 4)	Objects collected (robot 5)
...					
5	8	7	8		
6	7	8	8	7	8
7	8	7			
8	8	7	5	5	
9	8	8	6		
10	3	4	4	3	2
11	0	1	0	1	
12	0	1	1		
13	0	2	1	0	1
14	0	0			
...					

Lau et al. (2011) acknowledged that implementing this into simulation tools and physical robots might be different. They found that the data of this experiment is also influenced by the environmental conditions, such as the availability of objects and the physical distribution of objects in the arena, which are likely to change as time progresses.

The goal of this research is to examine and evolve their approach, then implement it in a simulation tool. This is because their proposed communication strategies are still relevant and useful when dealing with swarm robotics failures, even in the case of uncertain environments.

2.1.2 Fault Diagnosis and Identification

Swarm robotic failures could occur in the robot's hardware or robot's software (internal hazards), or they could be classified as environmental disturbances (external hazards) Figure 2.4. For instance, the effect of the distance sensor failure on an individual robot is that the robot may collide with other robots or obstacles. Also, the effect of the motor failure in a single robot is to anchor the entire swarm in a fixed region, which could be a potentially serious problem. The environmental disturbances hazard might threaten a swarm robot collective, especially in the case of a dynamic arena, such as a moving arena surface, where this could generate noise within the swarm robotics communication or the localisation.

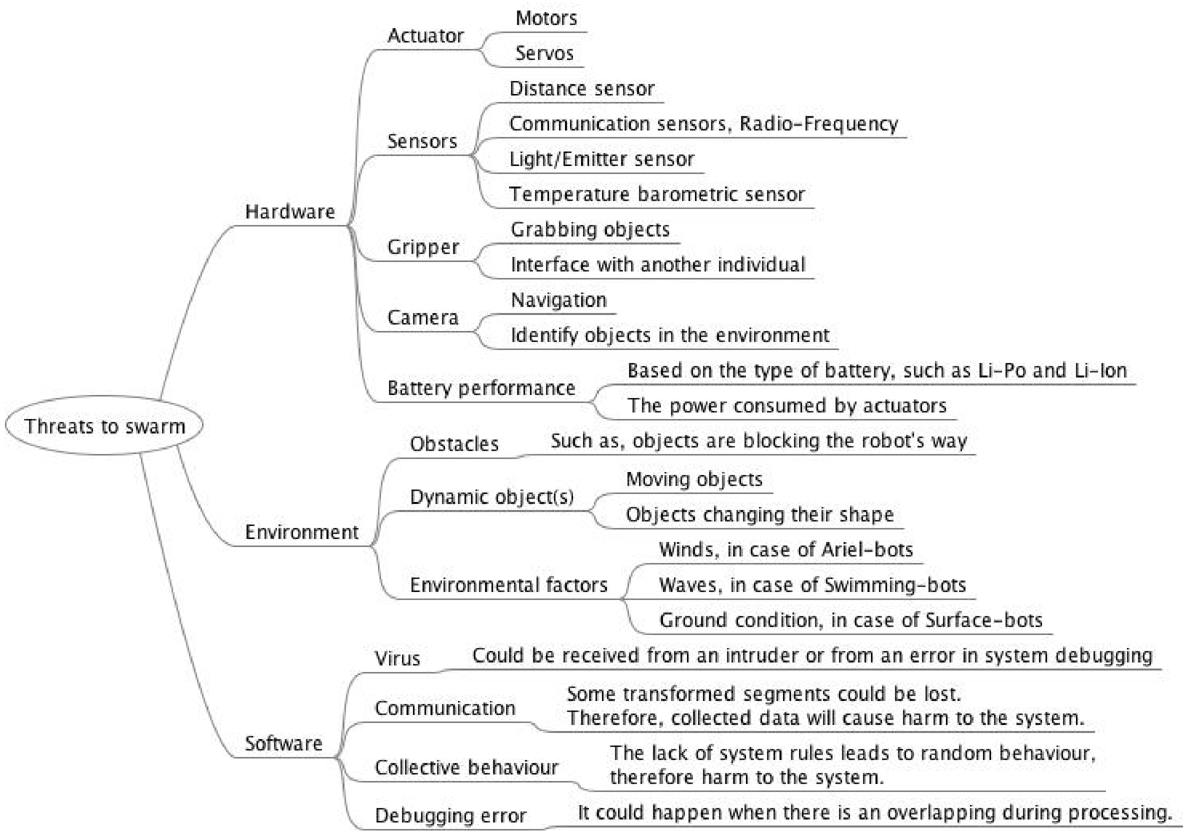

Threats to swarm
Hardware
Actuator
Motors
Servos
Sensors
Distance sensor
Communication sensors, Radio-Frequency
Light/Emitter sensor
Temperature barometric sensor
Gripper
Grabbing objects
Interface with another individual
Camera
Navigation
Identify objects in the environment
Battery performance
Based on the type of battery, such as Li-Po and Li-Ion
The power consumed by actuators
Environment
Obstacles
Such as, objects are blocking the robot's way
Dynamic object(s)
Moving objects
Objects changing their shape
Environmental factors
Winds, in case of Ariel-bots
Waves, in case of Swimming-bots
Ground condition, in case of Surface-bots
Software
Virus
Could be received from an intruder or from an error in system debugging
Communication
Some transformed segments could be lost.
Therefore, collected data will cause harm to the system.
Collective behaviour
The lack of system rules leads to random behaviour,
therefore harm to the system.
Debugging error
It could happen when there is an overlapping during processing.

Figure 2.4: A tree diagram identifying threats that could cause harm to the swarm robotics system.

Many researchers have been working on identifying failures in swarm robotics. Patterson and Patterson (2010); Higgins et al. (2009b); Winfield and Nembrini (2006) have analysed, understood and reported the failures that are most serious in terms of compromising the overall desired swarm robotics behaviours, which makes it possible to find the best technique for mitigating failures in swarm robotics.

The process of fault diagnosis and identification focuses on determining the specific cause of the fault. Parker (2012) has stated that many fault detection techniques in a single robot have the ability to diagnose and identify the exact fault that could occur in the swarm robotics, based on a failure state, such as slipping wheels state, faulty encoders state, stuck wheels state and so on. When the system determines with high probability that the robot is in a failure state, the identity of that state is known, which corresponds to the specific cause of the failure (Verma and Simmons, 2006).

Learning-based fault (LeaF) diagnosis was proposed by Parker and Kannan (2006) to diagnose and identify faults in swarm robotics. This system uses an adaptive causal model to represent possible faults in the system. When a robot encounters a fault, the system attempts to extract a relevant recovery action from previous experience. This system is able to effectively learn from its own faults, which makes it more robust and efficient for swarm robotics applications.

While swarm robot can suffer from a range of failures as shown in Figure 2.4, the work solely concentrates on failures at hardware-level in swarm robotics.

Hardware failures are the most common examples covered by a number of researchers, including Patterson and Patterson (2010); Higgins et al. (2009b); Winfield and Nembrini (2006). Hardware failures are possible to be detected because the hardware behaviours are possible to predicted during the development of the swarm robot controller based on the input and output variables of hardware devices in the robot. Unless the input and output variables from the robots' controller shows a consistent behaviour layout, the hardware device of the robot will be considered operational However, by validating and confirming that the robots' hardwares are operational it is possible to move on, in the future, and diagnosis failures at the software-level and so as the failures caused via the environment.

However, by addressing and bounding the swarm robotics failures specifically, the system developer will be able to implement these failures into the swarm robotics system. Even Parker and Kannan (2006) succeeded in enabling robots to diagnose and recover from pre-defined errors, but still their experiment with the causal model approach was based on a large-scale multi-robot system. As they stated, their work is still underway to further validate the LeaF approach in extensive experimentation both in simulation and on physical robots.

2.2 Approaches to Protecting a Swarm from Failures

After considering the concept of the learning-based fault (LeaF) approach by Parker and Kannan (2006), this section will discuss a relevant mechanism that has been inspired specifically from microbiological immune systems. The observation of the immune system in swarms has been documented by many researchers, an example proposed for swarm robotics was inspired by Granuloma Formation (Timmis et al., 2010; Adams, 1976).

2.2.1 Granuloma Formation Approach

Flugge et al. (2009) provided a description of Granuloma Formation. There are many parasitic diseases that are usually fatal if untreated, including Visceral Leishmaniasis, (Desjeux, 2004). Once a bacterium (which is the single form of bacteria) enters the cell, the cell becomes the carrier of a replicate bacterium. Before the infected cell disintegrates and becomes a threat to other cells, the immune mechanism, in the infected cell, activates its system and sends signals to other cells to form a wall around the infected cell as well as the bacterium, leading to the formation of a granuloma in order to isolate the infection from other cells Figure 2.5.

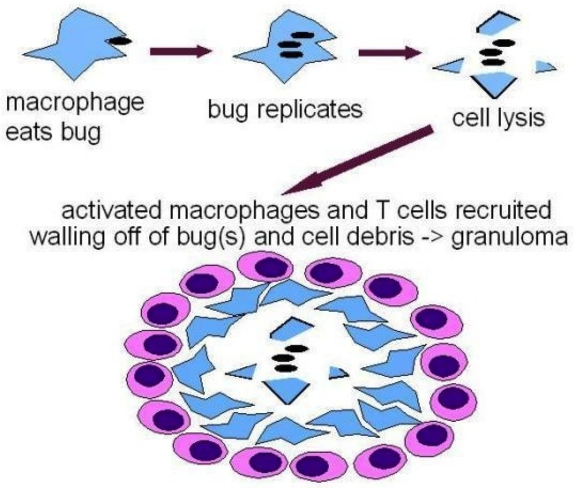

macrophage eats bug
bug replicates
cell lysis
activated macrophages and T cells recruited
walling off of bug(s) and cell debris -> granuloma

Figure 2.5: Simple concept of Granuloma Formation. The diagram shows the process of immune system activation. When the bacterium enters the macrophage (which is the cell that is attracted by the bacterium), the bacterium starts replicating itself until the cell disintegrates. Once that occurs, the cell sends signals to the nearest cells to completely surround the infected cell. From Ismail and Timmis (2010).

Neutrophils are another type of immune cell present in large numbers in the blood that respond quickly to the presence of a pathogen (such as a bacterium). Chtanova et al. (2008) describe the collective action of neutrophils in terms of swarm dynamics, because they develop from the initial arrest of a small number of neutrophils followed by a massive influx. This directed migration, where individual neutrophils move persistently towards the swarm over time, is probably caused by direct communication between cells via signalling molecules. Therefore, this signalling behaviour is attracted to neutrophils.

2.2.2 Immune System Algorithms

Immunological algorithms take inspiration from a diverse range of immune functions that occur across different immune systems. The observation of the immune system in swarms has been documented by many researchers, whose developments in the immune system have been based around different algorithms, for instance, the algorithm for artificial Granuloma Formation proposed by Ismail and Timmis (2009), Algorithm 1.

Algorithm 1 An artificial Granuloma Formation algorithm as applied to a robotic swarm. Adapted from Ismail and Timmis (2009).

```
begin
Deployment: Swarm robots are deployed in the environment
repeat
  Random movement of the swarm robot in the environment
  if Error detection then
    Signal propagation: Faulty robot will emit fault signal
    Containment: Containment of the faulty robots and isolation of their signals from
    the other robots
  end if
until forever
end
```

Timmis et al. (2010) work focused on the category of immune-inspired population-based algorithms from the perspective of swarm behaviour. They derived the immune system framework in swarm robotics from the algorithm proposed by Newborough and Stepney (2005), which enabled them to abstract the underlying concepts of the algorithms in a unifying structure, as follows:

1. **Create:** a population of novel individuals is created that represent candidate solutions to the problem being optimised by the algorithm.
2. **Evaluate:** each individual is evaluated based on predefined criteria that determine how well it solves the optimisation problem.
3. **Test:** a condition is tested to establish whether the algorithm terminates, returning an individual solution or set of solutions upon termination.
4. **Select:** a set of candidate solution individuals is selected to be used as the basis for the creation of the next generation of individuals.
5. **Spawn:** the new population of candidate solution individuals is generated for use in the next generation.
6. **Mutate:** variability is introduced to the algorithm via either altering the number of individuals of the new population or some other aspect of the algorithm.

The framework derived by Timmis et al. (2010) highlights the differences and similarities between immune and swarm robotics algorithms. During their work, they carried out an empirical analysis of a range of population-based algorithms inspired by different biological paradigms. This framework consists of six generalised stages, which constitute a single algorithm generation. It is structured as follows, where the key swarm steps are highlighted in italics:

1. **Create:** *a population of potential solution individuals is created at each generation. A potential solution is constructed by an ant agent iteratively following a series of path steps based on pheromone levels until a complete potential solution is generated.*
2. **Evaluate:** the best individual solution is the one with the shortest path.
3. **Test:** upon triggering the termination condition, the single best individual solution is returned as the output of the algorithm.
4. **Select:** no individuals from the current generation are selected for the next, as each generation creates its own population from scratch.
5. **Spawn:** no individuals are spawned for the next generation as none are selected.
6. **Mutate:** *an additional pheromone is laid at each path step of solution individuals proportionally to how good the solution is, whilst the pheromone is also reduced by a decay function.*

As seen in this algorithm, researchers are still observing algorithms from biological insects without considering the limitations to the extent of configurability between animals and robots.

2.3 Swarm Robotics Approach for Detecting Failures

While the robot is attempting to broadcast an assistance request signal, the broadcast signal will include the robot's controller input and output values to neighbours, to be computed and evaluated by the neighbours predefined values, and they then compare the results with their own values. If these values do not match, they assume that this robot has a suspected failure. Here are two approaches for failure detection that are relevant to the proposed exogenous failure detection and recovery model in swarm robotics of which will be discussed more in the thesis.

2.3.1 Fireflies Approach for Detecting Robot Failures

Christensen et al. (2009) took inspiration from the synchronised flashing behaviour observed in some species of fireflies. In their experiment, each robot was equipped with on-board light-emitting diodes (LED). So, operational robots could detect a non-flashing robot which was suffering a catastrophic failure.

Christensen et al. (2009) chose to deal with visual rather than radio and sound communication. This was because robots are unable to determine the location of the sender relative to their own frame of reference. Thus, the messages are separate from environmental localisation information.

Figure 2.6 demonstrates the Christensen et al. (2009) approach for exogenous fault diagnosis in the swarm robotics based on LED visual communication, which requires robots to be within close range, that is, 50 cm for detecting the LED on the other robot in order to estimate its location. Figure 2.6(a) shows that both robots are operational and synchronised and that their flashing LEDs are symmetrically together. Figure 2.6(b) shows that both robots are flashing correctly but they are not synchronised. This means that one or the other robot is non-operational and that they are suspicious of each other. The failure is considered detected whenever a robot flashes twice, while observing that another robot's LED is solidly on or it does not flash at all, as demonstrated in both Figure 2.6(c) and Figure 2.6(d).

2.3.2 Robot Internal Simulator Approach

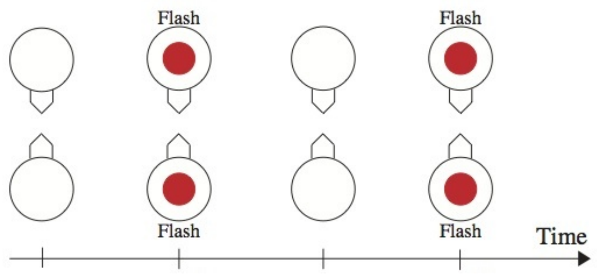
Another approach for exogenous failure detection was demonstrated by Millard et al. (2013). The concept of their work has robots that are capable of detecting partial failures by possessing a copy of every other robot's controller, which they can then instantiate within an internal simulator on-board. Thus, as long as a robot's controller can be instantiated in the simulation, its behaviour can be predicted. This means that the proposed method would be independent of the robot controller architecture, which could be used in both homogeneous and heterogeneous swarm robotics.

After a robot gets the data from other robots in the swarm, the internal simulation runs for a short period of time to predict the future state of the other robots based on their data. A failure will be indicated whenever there is a significant discrepancy between the predicted and observed behaviour of a particular robot.

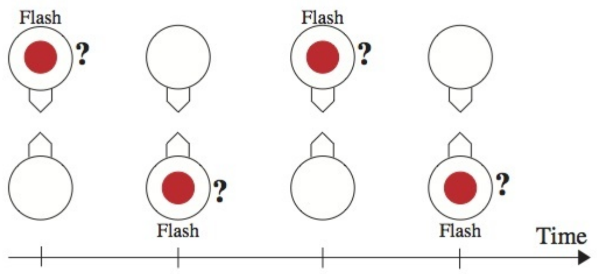
2.3.3 Weaknesses of the Fireflies and Robot Internal Simulation Approaches

The two approaches discussed in Sections 2.3.1 and 2.3.2 are good starting points for failure detection approaches in swarm robotics, but still these approaches have disadvantages compared to the exogenous failure detection and recovery model that are proposed in this thesis (Section 5.1).

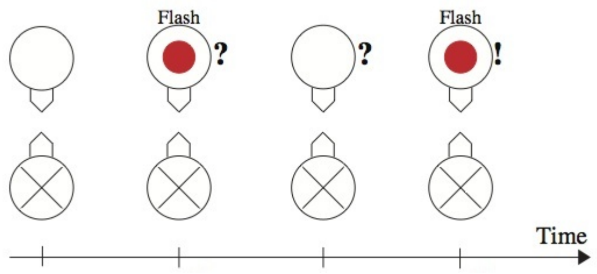
For instance, the fireflies approach requires robots, having camera onboard, to be at a close distance with their partner when they are diagnosis the health status of each



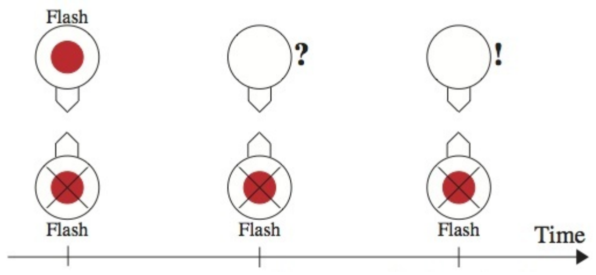
(a) Both robots are operational and synchronised.



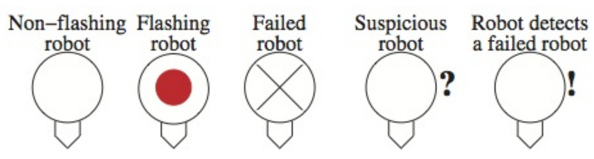
(b) Both robots are operational but not synchronised.



(c) One robot has a fault.



(d) One robot failed while its LED is on.



(e) The LED operational definitions.

Figure 2.6: The four possible scenarios for the Fireflies approach described by Christensen et al. (2009). Figure (e) shows the robots' status if they are not synchronised and a fault is acknowledged in one of them.

other. That means, if swarm robots are operating in a large arena size they will have less chances to diagnosis each other until they meet again after a while of wandering.

Also, the fireflies approach requires to diagnose health status of one robot at a time. Where they do not have the ability to diagnosis multiple robots at ones. If the diagnosis with one robot takes long time, that might mean the other robots that are close to these two robots will not have a chance to be diagnoses if they leave the camera sight of others.

The Fireflies approach requires a synchronisation flashing light of partner in order to detect health status between two robots in the swarm. With different environmental circumstances, the failure diagnoses would become difficult to be detected if the arena has a bright lights that could make it difficult for partners to detect the flashing lights of each other. This will compromise the main purpose of the fireflies approach.

On the other hand, the robot internal simulation approach consists of working offline, where robots have to predict the behaviours of their partners in the future time. Then, they get back and check if the predicted behaviour is matching or the robot would be considered failed. Unfortunately, the behaviour of swarm robots could not be predicted if they have to change their actions based on two factors, the arena size and the swarm task. The weakness of this approach is that failure diagnoses requires working offline. If in case the arena is small in size with a large number of individuals, robots will have to change their directions in order to avoid walls and other robots in their way constantly. At this time, predicting the location and the behaviour of other robots will be difficult. Also in case while a robot has already been predicted and is underway for recheck with partner but at the same time the robot start a task, this robot later will be predicted as failed while in fact is still operational.

In addition to that, robots have to be simple in construction (Section 1.1.2), this means robots does not need to have a complicated and expensive components such as a permanent memory for storing each other data for a long period of time, typically required by robot internal simulation approach.

However, the provided disadvantages discussed in this section have been considered during the development of the proposed exogenous failure detection and recovery model proposed in Section 5.1. This model works for diagnoses the health status of not just one partner but multiple partners within the communication range of each others. Also, it is likely to mention that this approach works online all the time between individuals even when a robot encounter an obstacle or while it has started a task.

The proposed exogenous failure detection and recovery model is not an extension to any provided approaches in this thesis. It is a new approach that was built to cover the concepts and gaps of approaches by other researchers.

2.4 Approaches in Recovery and Mitigation Solution

Ismail and Timmis (2009) suggested that when a robot faces a large power loss, then the other robots that are going to isolate this robot have to share power with each other to recharge its battery power. Once a power share operation is completed, the robots will then carry on with the tasks they were doing before.

Power loss should not be considered as a failure. When the battery power is running low, this means it only needs to be recharged. However, the concept of this mechanism was presented by Humza et al. (2009). In their experiment, they implemented a biological immune system framework on an E-Puck robot. The main objective was that a robot involved with a recharge docking station had to explore the environment and help recharge any distressed robot that it found, then return to the recharge docking station. This experiment introduced the idea of a heterogeneous swarm system, where different groups of robots exist in a swarm where some of them are working to detect and recover robots with low power, while the rest are working to achieve a given task.

However, there are a number of critical issues with their work. First of all, the recovery robots exist just to recharge other robots with power issues, but they are not involved in achieving the swarm's given task. The shortcoming of this arrangement is when these recovery robots have a failure themselves while recovering other robots. Another issue raised in their work is that these recovery robots do not have the ability to ensure the remaining power is enough to allow them to get back to the recharge docking station. This raises another issue that when they fail to return to the recharge docking station they become obstacles to other robots in the swarm.

From a critical point of view, there should be homogeneous robots in the swarm whose main objective is to achieve a given task together. The idea of a power sharing group has to be implanted in all robots, not just some of them. Thus, all robots must have the ability to recover each other.

Providing reliable swarm robotics requires the provision of system resources, (Higgins et al., 2009a). Due to resource constraints, including storage, communication and most importantly energy, this leads the robot to become inoperable, unless the resource is renewable. A proposed solution to prevent the wasting of the swarm robot's resources was suggested by Ismail et al. (2015). Their experiment relies on the use of residual energy in the faulty robots. So that, faulty robots can be used as a communication repeater to rebroadcast messages over a long range between robots. This of course requires a minimal amount of energy, while they are stationary, compared to the amount of energy needed to operate the actuators.

In Ismail et al. (2015) work, they have observed the failure behaviour based on the remaining energy, which is not useful to run the motors. In addition, the remaining amount of energy, in faulty robots, could only power up the communication unit for 27 minutes. However, the results from the simulation experiment in this research thesis may observe a different amount of remaining energy.

2.5 The Reliability Analysis of k-out-of-N

Stancliff et al. (2006) have supported the argument that larger members of less-reliable robots perform certain swarm tasks more reliably than smaller members of more-reliable robots. Their claim is relying on the use of the reliability analysis model for the swarm systems and components.

The main concept of using the reliability model R is assuming that a group of N robots are independent and have an equal probability of failure p. Thus, the system failure probability is the product of the robot failure probabilities, which could be applied in Equation 2.1.

$$R = 1 - p^N \tag{2.1}$$

Winfield and Nembrini (2006) argue that applying Equation 2.1 in swarm robotics could not determine that the overall system will likely not function properly if very few of the robots remain operational, i.e. this equation could generate incorrect assumptions, (Martinoli et al., 2012).

Winfield and Nembrini (2006) provided an enhanced approach which is defined with three possible states of the swarm robots, fully operational, partially operational or completely failed. The concept of their approach relies on the probability of robots failures in different states. The system reliability, R, for N robots is modelled as Equation 2.2, where p_0 is the probability of failed robots in the partially operational state, and p_f is the probability of failed robots in the completely failed state.

$$R = (1 - p_0)^N - p_f^N \tag{2.2}$$

The time of when the failures occurs in the swarm system could be vary from robot to another. Therefor, it is preferable to distinguish the total time of achieving the swarm task and the time before the impact of the first failure. The k out of N reliability model meet the requirements and provide correct assumptions with the swarm known parameters. Equation 2.3 represents the probability of swarm system failures P, where k is the number of operational robots and t is the simulation total time. The $MTBF$ represents the mean time before the impact of the failure, Equation 2.4.

$$P\left(k,N,t\right)=\sum_{i=k}^{N}\binom{i}{N}\left(e^{-t\lambda}\right)^{i}\left(1-e^{-t\lambda}\right)^{N-i} \tag{2.3}$$

$$\lambda=\frac{1}{MTBF} \tag{2.4}$$

An example of swarm system running with 20 robots, of which 10 robots are required to be operational, for 1800 seconds where $MTBF$ is 800 seconds, is represented in Figure 2.7.

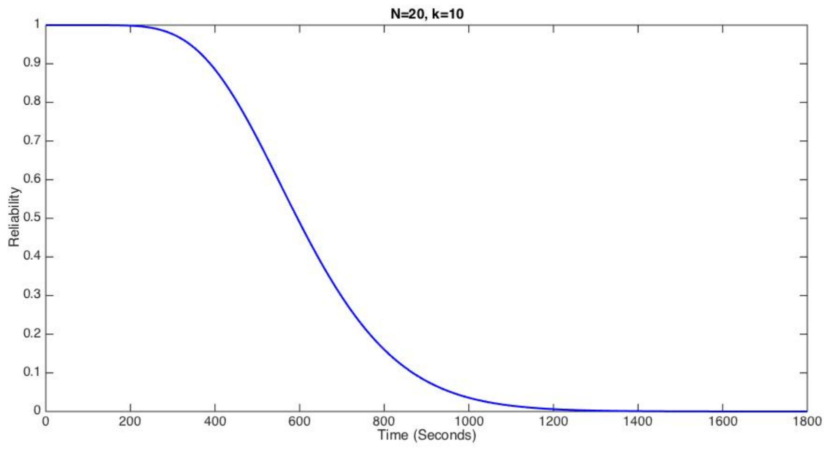


Figure 2.7: The reliability of a robot swarm modelled as a k-out-of-N system, with $k = 10$, swarm size $N = 20$ robots and $MTBF = 800$ sec (e.g.The reliability of 0.5 have a probability that the swarm robots will have significant risk to fail after 600 sec after the starting of the experiment which might lead to uncompleted swarm task when all robots are going to certainly failed around 1200 sec duration).

2.6 Discussion

Hinchey et al. (2007) have claimed that the advantages of swarm robotic systems over multi-robot systems is that individual robots are capable of performing a simple role produces with complex behaviour as a whole system. Achieving a complex behaviour will require the swarm robotic to have a mean of cooperation between individual robots. Therefore, wireless connectivity information is suggested to achieve the swarm robotics objectives (Winfield and Nembrini, 2006).

Although scalability is applied to swarm robotics (Şahin et al., 2008) (which means swarm robotics is capable of performing well with different swarm sizes), reliability could not be ignored (Bjerknes and Winfield, 2013), therefore, fault detection mechanisms are required.

The Failure Mode and Effect Analysis (FMEA) technique, used by Winfield and Nembrini (2006), has some important concepts that are helpful for predicting the reason for a faulty swarm robot, but still this technique only works for system developers in order to predefine possible faults for analysis purposes. Also, the concept of detecting the faulty robot requires individual robots to continuously broadcast their own ID's to neighbours. Whenever an individual robot is no longer connected with its neighbours, the neighbours will assume that this robot has encountered a failure, and the faulty robot needs to turn back 180 degrees to reconnect with the group. However, from the perspective of the swarm robotic system, this could not be considered a failure, even not a detection mechanism will determine the reason of this disconnection. Undertaking further analysis will be required, by the system developer, in order to determine the reasons behind the disconnection of this faulty individual. Therefore, the swarm robotic system is not able to undertake its own failure detection analysis to address it's encountered failures.

On the other hand, the Fault Tree Analysis (FTA) technique, by Visinsky et al. (1994), is used to initially identify all of the potential failure events, by the system developer, prior to building the swarm system. The reason this technique is not sufficient is that the swarm system cannot undertake detection procedures of encountered faults that are not identified by the system developer in the first place. Also, this technique has no smart mechanisms to build-up the sequence of events in order to generate new events dealing with the unidentified encountered failures. As a solution, there has to be a built-in mechanism, within the swarm robotics, in order to deal with encountered failures that could be unknown by the system developer in order to emphasis the robustness and the reliability of the fault detection mechanisms of swarm robotics.

Hinchey et al. (2007) claimed that individual robots are capable of performing a simple role along with complex behaviour as a whole system. Achieving a complex behaviour will require the swarm robotic system to have a mean of cooperation between individual robots. Therefore, a wireless connectivity information is suggested required to achieve the swarm robotics objectives (Winfield and Nembrini, 2006). Reliability, from the other side, could be considered as a part of the swarm's complex behaviour, where individual robots have to cooperate together to maintain the swarm performance.

As wireless connectivity information is applied, Parker (2012) has suggested two techniques for defecting potential failures through communication between individual robots, namely, collective behaviour and robot health signal. Unfortunately, these techniques cover only one failure aspect, they cannot be used to detect other failures than the predefined one, that is, they could only detect when a robot, suddenly stopped broadcasting its health status, which it will be considered having a fault without knowing what the fault is, or what caused this fault. However, Parker (2012) has observed and analysed these techniques and several others. He has been exploring the probabilistic aspects of how algorithms of both swarm intelligence and biological immune systems are similar, from a scientific point of view.

In the context of swarm failure detection, mechanisms of Granuloma Formation and neutrophils have been provided in this research in order to bring the principles of the immune system to swarm robotics. The main objective of the immune system technique in corresponding to swarm robotics is to enforce the failure detection and isolation mechanisms to provide reliability to the swarm.

The concept of the system immunity has been discussed by a number of researchers, including those that draw inspiration from fireflies by Christensen et al. (2009) and robot internal simulator by Millard et al. (2013).

In Christensen et al. (2009) work, they have set the control cycle step time to 0.15 seconds. In another words, during this period, the robot's controller reads data from sensors including the on-board camera and proximity sensors, then processes the data, and sends control signals to motors that drive the robot. Unfortunately, their failure detection technique will face a failure when robots commence a task or when they encounter each other after having been separated for a period of time, as their activations are likely to differ.

The work by Millard et al. (2013) was considered to expect the behaviour of other robots offline. whereas, during the internal simulator operation, a robot will not predict a fault if the other robot faces a sudden fault after broadcasting a health status. At the same time, it will become difficult for the operational robot to take a recovery decision.

Besides, it is still inefficient to rely on the on-board visual camera, as the robot's controller also has to deal with analysis images of the robot's LED.

However, in this research thesis, the simulation experiment of exogenous fault diagnosis will consider broadcasting communication messages (including the robot's information) between robots in the swarm. Also, localisation becomes more clearly stated with the existence of the navigation devices mounted in each robot. Therefore, the broadcasting robot's location becomes more explicit to neighbouring robots. Even more, swarm robots will be able to engage with multiple robots at a time, because there is no need for over-processing, in the controller, to analyse the image of the detected LED in each robot in the swarm.

The proposed exogenous fault detection model in this research thesis will proceed in addressing the failed component at the hardware-level to be diagnosed in order to be ready for the recovery/mitigation procedures. So far, the previous studies have discussed failure recovery for the energy management issue by Ismail and Timmis (2009) and Humza et al. (2009), where this issue is not a catastrophic failure. Swarm robotics failures, such as distinguishing between identified objects and other robots, limited communication range and overlapping code during processing are very important to investigate and test in real swarm robotics. During the progress in the research thesis, the approach proposed by Humza et al. (2009) will be developed in order to manage different failures recovery/mitigation.

Parker (2012) reported that reliability has been studied in individual and small robot systems over time. Large swarm systems illustrate the conservation of robustness to some extent, and reliability is a solid basis for component failure analysis (Kitano, 2007). Experimental studies are still required to characterise the trade-off relationship between robustness and performance of the swarm robots. This will lead to the basis for investigating reliability in swarm robotics. There still remains a wide variety of challenges in swarm robotics reliability, including:

- Metrics are needed that can evaluate the robustness and reliability of robot collectives (Kannan and Parker, 2007; Parker, 2001).
- General techniques for inferring the impact of partially failed robots are needed for a wide variety of collective robot applications, along with techniques designed for addressing these failures (Winfield and Nembrini, 2006).
- A quantitative model for representing the reliability of robot collectives is needed with which to determine the minimum number of robots needed for acceptable robot team performance (Winfield and Nembrini, 2006).

2.7 Motivation

As in robotic swarms, multi-robot systems are a collection of robots, working together to achieve a common goal (Higgins et al., 2009b). Swarm robotics differs from more traditional multi-robot systems in that their command and control structures are not hierarchical or centralised, but are fully distributed, self-organised and inspired by the collective behaviour of social insect colonies and other animal societies (Higgins et al., 2009a; Bonabeau et al., 1999). Self-organisation means that sometimes the collective behaviour, even if unpredictable, may well result in solutions to problems that are superior to ones that could have been devised in advance. The parallel drawn with social societies in the animal world extends to communication (Higgins et al., 2009b). Fault tolerance has already been extensively explored within the context of multi-robot systems with hierarchical command and control, notably in the work of Parker (1998).

While it may be true that swarm robotics can exhibit an unusual level of tolerance to failure of individual robots, it is not safe to assume that scalability and robustness are automatically properties of all swarm systems (Bjerknes and Winfield, 2013). Therefore, this is a problem that cannot simply be solved by adding more robots to the swarm. Instead, future large-scale swarm systems will need an active approach to deal with failed individuals if they are to achieve a high level of reliability (Millard et al., 2014).

The failure of a few individual robots may reduce the number of robots that are required for the team work. This may not be an issue with a large size of swarm robotics because of its applied scalability. However, in the real world, practical robotic swarms with a small number of complex robots may have a serious issue that prevents the task accomplishment. Even in the heterogeneous swarm robotics system, the need for physical and behavioural integration among the different hardware platforms results in a considerable amount of extra complexity for the design and implementation of each different type of constituent robotic agent. This integration complexity must be dealt with both in the hardware design, and at the level of behavioural control (Dorigo et al., 2012). The failure to one or a few of the complex individual robots may lead the robotic swarm into stagnation (Ismail and Timmis, 2010).

Also, considering the arena properties, from the perspective of the size and structure, swarm system reliability may have a significant impact on the entire swarm robotic task. For instance, in the outdoor open arena (where the arena size is unknown), swarm robotic size will be considered important when there is no specific preference for how many robots are needed for achieving a task, even after adding more robots. In this case, it is important to achieve reliability with the swarm system in order to finish the task.

The unknown properties of the arena, especially the size of the arena, may have a significant loss in the swarm's connectivity. Therefore, the problem of arena size maximisation maintaining the connectivity arises when the robots spread apart (Mathews et al., 2012).

2.8 Research Objectives

In the context of swarm robotics, the relatively simple robots can have any global pattern or behaviour (Bayindir and Şahin, 2007), that is, relatively simple robots can perform difficult tasks as well as a few complex ones (Magnenat et al., 2008).

A swarm of identical robots has limited reliability. Although the quality and robustness of the hardware is increasing, hardware failures are still quite common. So, achieving reliability in swarm robotics is considered necessary to assure the robustness of the swarm system (Brambilla et al., 2012).

Achieving reliability requires to identify and address different failures at the hardware-level, including the navigation device, and their effect on individual robots, as well as on the entire swarm robotics. The objective of this achievement is to find detection mechanisms that identify the fault, as well as to determine faulty robots in the swarm and choose the best technique to distinguish which robot has to be mitigated and recovered (Parker, 2012).

Figure 2.8 places this work at the intersection of biology, swarm robotics and simulation. The exact implementation will involve features drawn from both real swarm robotics experiments and biological systems and then include them in swarm robotics simulation experiments.

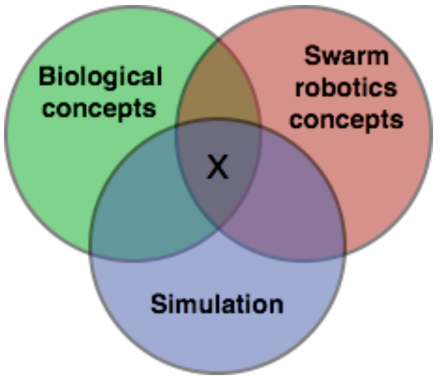


Figure 2.8: The conjunction of the three aspects discussed in this research, highlighting the area that the work will focus on.

Injecting the simulated robotic swarm with a failure in the navigation device of some individuals has proved that the impact of this failure can affect the overall time for achieving the given task. The proposed exogenous fault detection model has successfully achieved the research goal. Applying this model to a simulated robotic swarm having faulty individuals saved the time required to detect and address the existence of failures in each faulty robot. This allowed the simulated robotic swarm to mitigate these failures in order to achieve the swarm task in less time.

Not only was the time enhanced, but the travelling distance and the energy consumed were enhanced in the simulated robotic swarm.

The mitigation procedure consists of evacuating these faulty robots without affecting the entire simulated robotic swarm's task.

2.9 Research Aims

The swarm robotic system reliability may not be considered as a big issue that could threaten the system, due to the swarm robotic scalability, (Şahin et al., 2008), of which adding more robots can cover-up the failure. Even by considering scalability, the entire swarm robotic system may fail because of the persisting failure. Whereas, the work in this thesis demonstrates a number of motivation points that could proof swarm robotic reliability has to be reconsidered in order to emphasise the robustness of swarm robotics.

2.10 Research Questions

The research reported in this thesis will investigate these questions:

RQ1 Will the proposed exogenous fault detection principles satisfy the requirements for diagnosing different types of failures at the hardware-level in swarm robotics?

RQ2 Does the processing of the proposed exogenous fault detection affect the robot's controller functionality while the robot is achieving the swarm task?

RQ3 Why the failure detection and recovery procedures are important to swarm robotics?

RQ4 How could recovery procedures applied to the swarm robots in respect of hardware failures?

2.11 Summary

After in-depth analysis of the cited literature on failures detection approaches and those inspired from the immune system concepts in different approaches, the purpose of this research will be considering such failure detection and recovery concepts to come up with a reliable failure detection and recovery model that would be applied to swarm robotics despite the giving task. The failure detection systems is a different objective that will be investigated in more detail during this research. The study will address not only failures and their detection techniques, but also the mechanisms of failure mitigation approaches, from the perspective of microbiology by moving the faulty swarm robots away from operational ones.

Lau et al. (2011) demonstrated a data-driven error detection mechanism that provides a statistical analysis result. Their paper omits to provide important information, such as the system that was used to provide these results. Also, as mentioned in their paper, implementing this simulation mechanism with real swarm robots might give different results. The purpose of this research is to observe their work, then bring it to a development level for testing in simulation systems.

Many experiments have been inspired by Granuloma Formation system. The critical concept of using such systems is the limited validity of the system behaviours between microbiological units and physical robots, which are incomparable. Additionally, there is no experimental evaluation in swarm robotics that fully and quantitatively determines the real strengths or limitations of the similarities between animals and robots (Webb, 2000).

However, a relevant approach to providing reliability to the swarm robotics was inspired by some species of fireflies. Christensen et al. (2009) drew inspiration from the synchronised flashing behaviour observed in fireflies in order to produce their exogenous fault detection approach.

Another approach for exogenous failure detection was demonstrated by Millard et al. (2013) whose work on individual robots allowed them to possess a copy of every other robot's controller. These copies are then instantiated within an internal simulator in each robot. The purpose of this research is to carry on the concept of these approaches to come up with an enhanced exogenous fault detection model which will be applied in an experiment to check its validity.

Chapter 3

Research Methodology

The outcome of this research concentrates on the optimal exogenous fault detection in swarm robotics to complete the swarm task in short time with the best swarm robotics performance. To achieve this, the simulation experiments were carried out for different scenarios, thus making it possible to run simulation experiments much faster than it would take using real robots. The basic simulation time step can be adjusted to suit the experiments requirements. Besides, real world robots that are capable of achieving the research objectives are considered expensive, compared to the simulation tool.

The simulation software used in this research was Webots, developed by Michel (2004) and Cyberbotics (2013). The selected robot model for this simulation experiment was the E-Puck robot, due to its simulated components capability for achieving the experimental research objectives. Details and specifications will be explained in the next sections, including the E-Puck programming controller.

Some answers to the research questions in Section 2.10 were carried out briefly during this Chapter. A discussion relating to research questions is provided in Chapter 9.

3.1 Simulation Platform: Webots

Building up new robot models and setting up experiments can take only a few hours with a simulation tool. This is because simulations are easy to set up, less expensive, faster and more convenient to use, than real world robotic systems.

Webots is a professional mobile robotics simulation software that is widely used for educational purposes. Webots was initially developed by Michel (2004) and Cyberbotics (2013) at the Swiss Federal Institute of Technology in Lausanne.

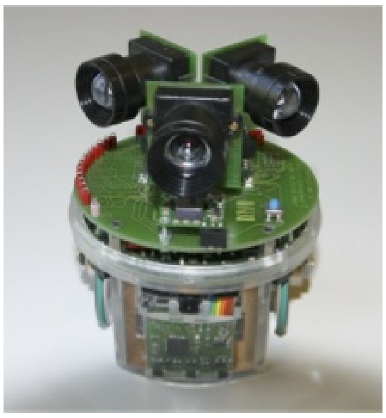
Webots have a number of essential features:

- Models include wheeled, legged and flying robots, and there is the ability to model a new robot.
- It includes variety libraries of sensors and actuators.
- Programming of the robot is possible in many languages including C, C++ and Java.
- It uses the ODE (Open Dynamics Engine) library for accurate physics simulation.
- It has the ability to simulate multi-agent systems, with global and local communication facilities.
- It transfers controllers to real mobile robots, including E-Puck and Khepera.
- It uses virtual time, making it possible to run simulations much faster than would be possible with a real robot.

3.2 The Robot Model: E-Puck

E-Puck is a small-scale differential wheeled mobile robotic platform designed for educational purposes. The E-Puck software was seasoned as open source, which enables it to be programmed to a suitable application based on the variety of uses of its sensors. In addition, E-Puck is designed as open hardware, which makes it able to accept several extension modules by stacking them on top of the E-Puck (as in real world E-Puck) or installing them in an empty slot inside the robot's body (Cianci et al., 2007), Figure 3.1. These configurations can also be applied to the E-Puck model in the Webots simulation software.

(a) E-Puck with the audio extension. This is the regular extension enabled in the E-Puck model in Webots software.



(b) A turret with three cameras on top of the E-Puck. It is able to add this extension in Webots manually from the controller program.

Figure 3.1: Physically different types of extensions can be stacked on top or installed inside the body of the E-Puck. These can either be applied to the real robot, or in the Webots model. Extension modules include a navigation module, compass, cameras, a Zigbee and a colour RGB LED ring. From Mondada et al. (2009).

3.2.1 E-Puck Robot Specification

Figure 3.2 shows the E-Puck simulated model is equipped with an asymmetric ring of a set of eight distance sensors (Table 3.2), ten LEDs, where eight LEDs are communication indicators between robots (and used for research observation), one LED is the main-board power indicator and the last one is located in the body of the E-Puck robot, to be used for different purposes including a low energy indicator (in the research simulation experiment, the body LED is used as an indicator after the E-Puck detects the object). In addition, the simulated E-Puck model has a navigation device that allows it to locate itself in the arena, and a compass for orientation. The simulated E-Puck is modelled with a front facing camera for visualisation, which in this simulation experiment was used as a colour detection sensor. Additional specifications are given in Table 3.1.

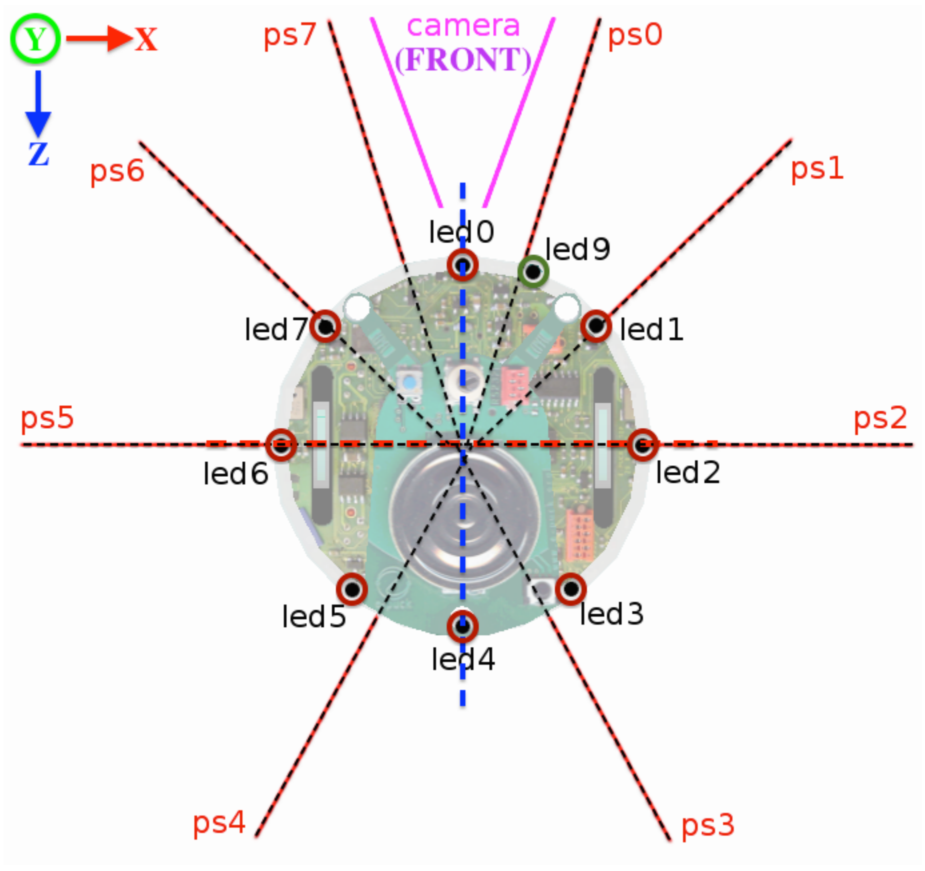


Figure 3.2: The E-Puck Sensors, LEDs and camera positions. Each E-Puck has a ring of 8 proximity sensors (PS0 - PS7) surrounding the robot, in addition to 8 red LEDs in a ring for status purposes. In the diagram, the LEDs are marked as black dots each surrounded by a coloured circle. LED8 is green and is located in the body but does not appear in the diagram. LED9, also green, is used as a power status that turns on when the battery is running low. The camera is located at the front of the E-Puck, giving the forward direction. The axis vectors are represented by the VRML standard, where the camera is pointing in the direction of the positive z-axis instead of the y-axis. Adapted from Cyberbotics (2013).

Table 3.1: E-Puck specifications. From Cyberbotics (2013).

Specifications	**Value**
Robot size	75 mm diameter
Wheel size	41 mm diameter
Axle length	52 mm
Motors	2 stepper motors
Max speed	13 cm/s
IR sensors	8 infra-red sensors
Camera	colour camera (640x480 resolution)
LEDs	8 red LEDs and 2 green LEDs

Table 3.2: E-Puck sensors specifications. The front direction of the E-Puck is given by the negative z-axis, which is the same direction as the camera. The plane zOx is oriented counter-clockwise, where the direction of the robot axle is given by the positive x-axis. The orientation of the sensors follow zOx plane. Adapted from Cyberbotics (2013).

Sensor Name	**X(mm)**	**Y(mm)**	**Z(mm)**	**Angle(degrees)**
ps0	10	33.0	-30	72.8
ps1	25	33	-22	44.1
ps2	31	33	00	0.0
ps3	15	33	30	298.5
ps4	-15	33	30	241.2
ps5	-31	33	00	179.9
ps6	-25	33	-22	135.8
ps7	-10	33	-30	107.2

As is the case for any differential wheels (E-Puck) robot set at its default position in Webots, in keeping with the Virtual Reality Modelling Language (VRML) standard, (Bell et al., 1995; Cyberbotics, 2013), the direction vector of the camera points in the direction of the positive z-axis. The forward direction of the E-Puck is given by the negative z-axis of the global coordinates.

3.2.2 E-Puck Controller Structure

The programming language used to build the controller to deal with different hardware components and software levels was C.

The E-Puck controller program is able to read the hardware input values, including proximity sensors, then set the motor speed accordingly. In addition, E-Puck robots are set to accomplish a simple foraging task. In particular, each robot is set to search for an object, then transport this object to a pre-set destination area in the arena.

The state diagram and the data flow structure of the controller are expressed in Figure 3.3 and Figure 3.4, respectively. When the E-Puck simulation starts, and before allowing any API function to be called, the controller initialises communication between the controller and the simulation environment. Then, the controller identifies each device installed in the E-Puck. This step must be set manually by the system developer, depending on the variety of devices installed in each robot if they differ.

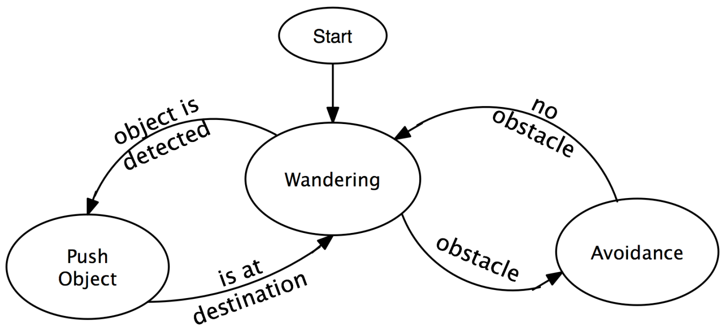


Figure 3.3: State digram of a swarm robotics foraging task. Robots start wandering until they detect an object. They, then, push the object to a pre-located destination area on the arena.

The E-Puck robots then start to identify its information including robot id, name, location in the arena based on the navigation module, and the north location to determine their direction, in addition to the sensor and actuator values.

The robot starts to wander in the arena and detect red objects. Detecting the colour of the object relies on the front camera located in each robot. This is representative of a pheromone-detecting technique in insects. If an obstacle is found, the robots start to find another way around to reach the object by using a set of surroundings sensors. When the object is detected, first, the robot stops close to the object and then starts moving around it until its side is aligned with the destination orientation. At the same time, the robot starts broadcasting a message to its neighbours to inform them that this object has been handled by him, so that other robots will not detect this particular object and can search for another object. Then the robot turns to face the object and starts pushing it until it reaches its destination, which is the green area in the arena. During the pushing procedure, if the object collides with another object or another robot, the robot fixes its position and pushes the object away from any interfering obstacle. The robot then searches for and detects another object until the entire task is finished.

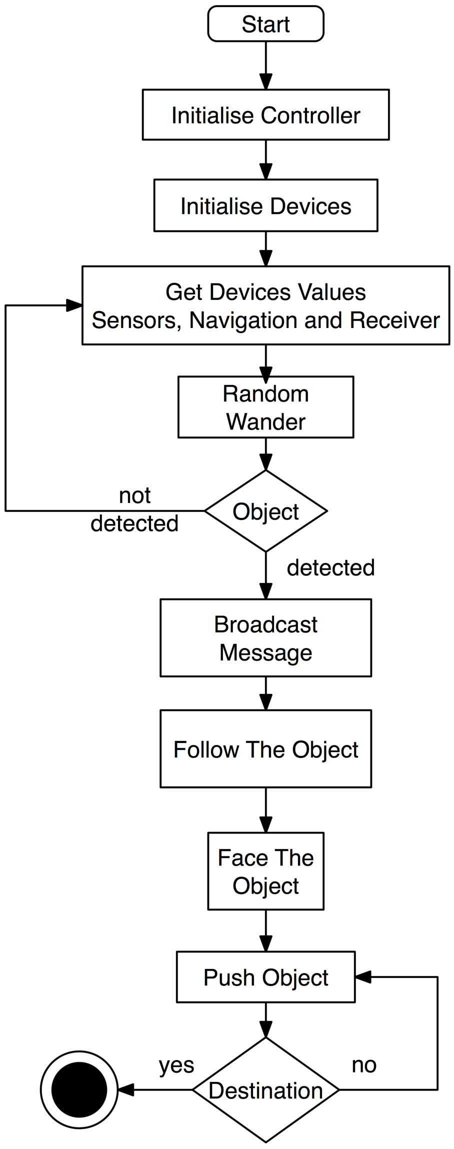


Figure 3.4: Data flow diagram of foraging task for searching for an object then moving it to the destination area.

3.3 The Properties of Simulation Experiment

Each of the simulation experiments was set to run ten times. All robots were equipped with the same controller program (explained in Section 3.2.2). The implementation of the proximity sensors in the model were designed to realistically represent the behaviour of the proximity sensor in a real E-Puck robot, where the output values of each proximity sensor will be slightly different from the others.

As the simulation experiment will be carried out in Webots, some details must first be clarified and explained for the accomplishment of this experiment, as well as to assure the robustness of a perfect simulation.

Arena The arena is modelled as a flat 1.5 × 1.5 meter square structured surface with grey colour, and surrounded by four walls (Figure 3.5). Later, depending on the experiment complexity level, the size of the arena will be increased to 5 × 5 meter square to add more difficulty to the simulated swarm task (Figure 3.6).

Destination Area The destination area is marked in brown and beige at the bottom of the arena. The distinction area represents the location of where objects will be collected. The destination area dimensions are 1.5 × 0.2 meters (Figure 3.5). The swarm robotics task is considered accomplished whenever all objects in the arena have moved and finally passed into the destination area. In the experiments with a bigger size arena, there are two destination areas (Figure 3.6) where containers first have to be transported to the right-top area, and then to the final destination at the right-bottom area, respectively.

Object Six cylinder-shaped objects are distributed on the arena (Figure 3.7), each with 100 millimetres in diameter. They are coloured red to facilitate their detection by the E-Puck camera, and also to stand out from the surrounding environment. Later on during the next experiments, the red objects will be placed in the middle of the arena. Also, instead of using the camera for detection, each object has an additional black collar that can be detected by the ground sensors in the robot.

Robot The robot used in this simulation experiment was the E-Puck with a navigation module and compass extensions for localisation and a front facing camera for detecting objects. Figure 3.8 shows the initial location of each E-Puck robot in the arena. Where Table 3.3 presents the position of each of the ten E-Puck robots in the arena. However, in the bigger arena, all robots will be placed at the left-bottom corner of the arena, Table 3.4.

Metrics for Evaluation The evaluation of experiments that were undertaken in this research were consist of choosing compatible metrics that are available in Webots. The discussion in Section 4.1 gives an overview of metrics used by other researchers. Webots simulation tool could provides a common metrics, the simulation time, distance-travelled and the energy-expended. Beside, theses metrics are combined with the relative devices used in E-Puck robot, which will be used to evaluate reliability of the robot's hardware (e.g. the navigation device). This will address the first research question **RQ1**.

However, throughout the thesis, the simulation experiments will have some changes depends on the swarm task complex and the ability of cooperation among individual robots before and after implementing failures. Moreover, the design and structure of the E-Puck robots' controller, that was demonstrated in Section 3.2.2, was developed later in Chapters 6 and 8 to cope with the requirements for detecting failures and recover faulty robots. The outcome from the developed E-Puck robots' controller will address **RQ2** and **RQ4** from Section 2.10.

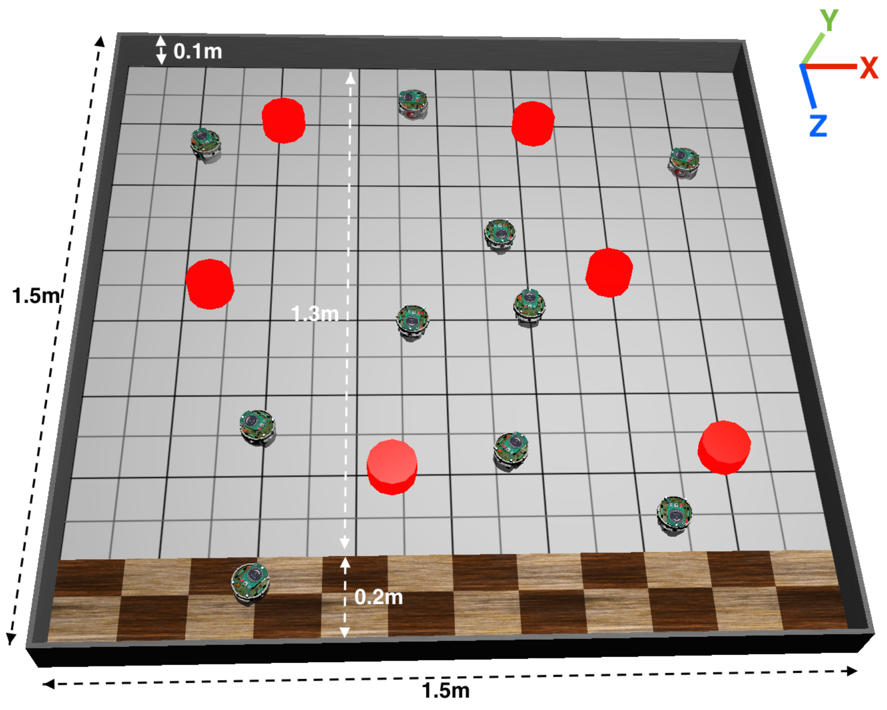


Figure 3.5: The simulation arena with ten E-Puck swarm robots. The red (large) cylinders are objects that are part of the swarm robotics task. The brown/beige pixel area at the bottom represents the destination to which the objects must be moved. The X, Y and Z axes at the top right corner represent the orientation of the arena in the VRML standard.

Table 3.3: E-Puck robots positions in the arena. The orientation relative to the -Z direction.

Robot Name	**X-Axis(m)**	**Z-Axis(m)**	**Orientation(degree)**
E-Puck 0	0.87	1.09	226
E-Puck 1	0.77	0.56	339
E-Puck 2	0.37	1.03	110
E-Puck 3	0.43	1.20	226
E-Puck 4	0.21	0.28	144
E-Puck 5	1.30	0.28	110
E-Puck 6	1.19	1.23	6
E-Puck 7	0.93	0.74	184
E-Puck 8	0.68	0.78	11
E-Puck 9	0.69	0.14	92

3.4 Simulation Challenges

A considerable number of issues have arisen with the simulation software, causing a delay during the simulation experiments and therefore a slow-down of the research progress, including:

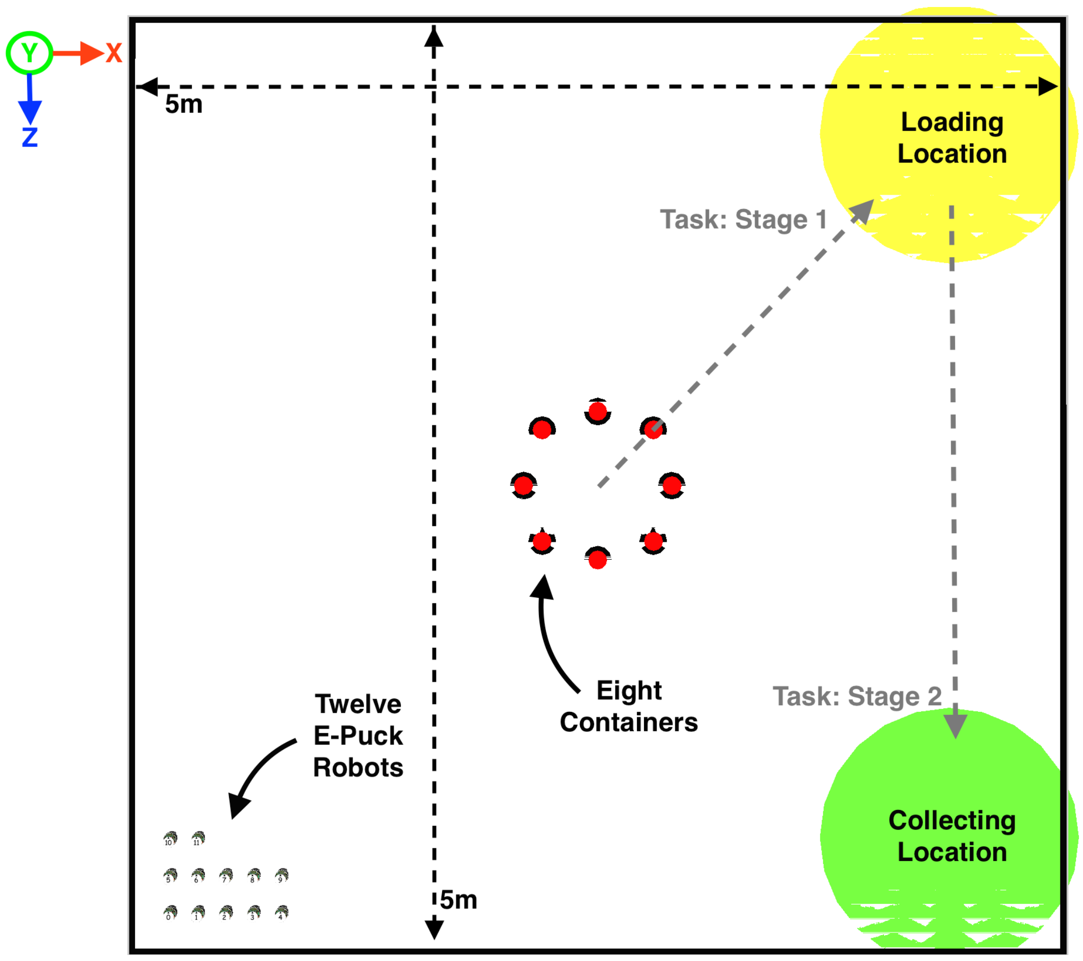


Figure 3.6: A larger size of the simulation arena with twelve E-Puck swarm robots located at the most left-bottom corner. The red cylinders, in the middle, are containers that are part of the swarm robotics task. There are two large circles at the two right sides of the arena. The top right circle represents the loading point, and the bottom right circle represents the collection point (final destination). The X, Y and Z axes at the top right corner represent the orientation of the arena in the VRML standard. From the Webots simulation platform.

Webots v7.4.0: During the experiment, warning messages were displayed after deploying the simulation. This caused bad camera readings and therefore wrong values. This issue was fixed after the release of the next version of Webots.

Webots v7.4.1: Difficulty running the software on MAC OS and Linux. This issue was solved temporarily until the release of a new version later on. In addition, the physics engine was causing slow simulation processing. This continued to provide wrong values that were noticed later on in the research.

Webots v7.4.2: The Webots development team worked on fixing a lot of issues in this release and also made huge changes in the main prototype file of the E-Puck and the simulation world file. This forced a delay in the overall research that required

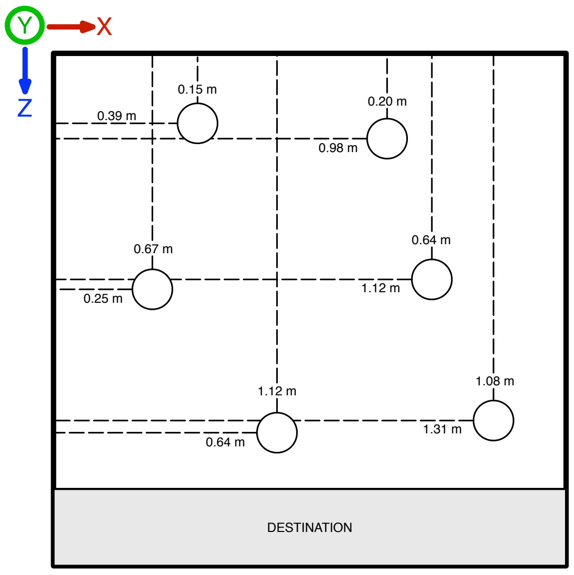


Figure 3.7: Location of the six objects in the arena. The circle represents an object. The measurement is in meters. The grey area at the bottom represents the destination to which the objects must be moved. The X, Y and Z axes at the top left corner represent the orientation of the arena in the VRML standard. This diagram is set to scale.

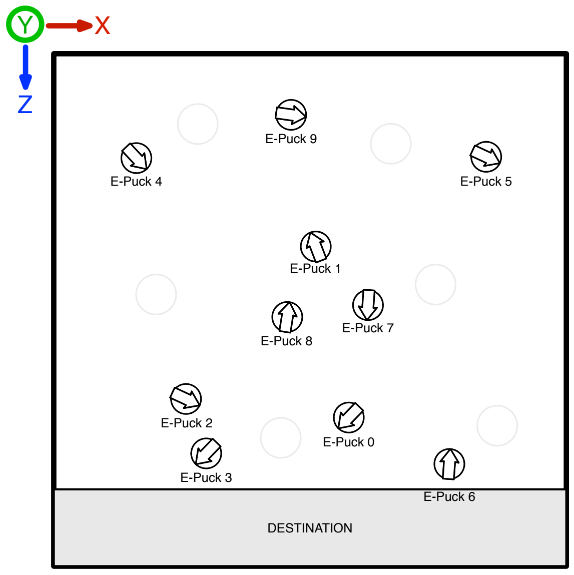


Figure 3.8: Ten E-Puck location in the arena. The circle represents the robot and the arrow represents the orientation. The measurement is in meters. The grey area at the bottom represents the destination to which the objects must be moved. The X, Y and Z axes at the top left corner represent the orientation of the arena in the VRML standard. This diagram is set to scale.

restructuring of the programming code again to accommodate the new changes. At this point, it was necessary to trace all the programming code manually, as the Webots software has no integrated debugging/tracing service.

Table 3.4: E-Puck robots positions in the large arena. The orientation relative to the north direction, which is -Z direction.

Robot Name	X-Axis(m)	Z-Axis(m)
E-Puck 0	0.2	4.8
E-Puck 1	0.35	4.8
E-Puck 2	0.5	4.8
E-Puck 3	0.65	4.8
E-Puck 4	0.8	4.8
E-Puck 5	0.2	4.6
E-Puck 6	0.35	4.6
E-Puck 7	0.5	4.6
E-Puck 8	0.65	4.6
E-Puck 9	0.8	4.6
E-Puck 10	0.2	4.4
E-Puck 11	0.35	4.4

Webots v7.4.3: Most issues had been solved in this release, but having more than four robots caused a delay in finishing all the experiments in short time.

Webots v8.0.1: There was a major new version released with an updated OpenAL (1.16.0). This release had a great fix to the delay issue from the previous release. However, after getting to the new experiments with the use of complex communication between individuals, another issue arose that caused a loss in transferred packets. It took a while due to the slow debugging and contact with the team support to figure out that the issue existed in Webots from the beginning. Another issue with this new release was some commands/functions which forced me to rebuild the affected parts in my code, and that has been dealt with after all.

Webots v8.2.0: The prototype files including the E-Puck robot and the arena had been affected since the release of version 8. So this version was released fixing most issues related to the prototype files (PROTO files).

Webots v8.3.0: After experiencing the problems with the previous release, this release was the most reliable with enhanced performance and enhanced processing speed. Although it has other determinable issues including the communication and packet buffering issues, still it can be surpassed.

The real time for finishing one experiment on the PC takes several hours, as the real total time to finish one run varies between half an hour to over an hour. A negotiation with the Webots software support team will be considered in order to find a way to speed up the processing of the experiment.

Beside the other issues faced with Webots, the camera is a major part of the reason for the slowness in the simulation processing. The camera is used to detect the object, so there will be more investigation in order to find another way to detect the object without using the camera at all. This will hopefully speed up the experiment processing time.

The E-Puck robot model has a set of five ground sensors that can be used for the line tracking. However, in order to overcome the issue with the camera to detect the objects, I added a black coaler under each object/container so that the robot can be able to detect the object whenever it's ground sensors gets a high reading.

Moreover, there is a possibility that the next experiments could be implemented with a real swarm of robots. This will enhance the evaluation of the proposed exogenous fault detection model on real swarm robots, compared with the simulated swarm robots.

3.5 An Experience View of Webots

After using Webots for an extended period of time, an overall of the its benefits and challenges can be discussed.

The Webots simulation software still not a complete platform for swarm robotics. It might be useful to use this tool with a different robot models, but still the implemented number of included agents could be affected when there is a need to use a mean of communication between individuals. In the case of the research experiments achieved in this thesis, there was need to build a matrix array for each individual robot with the size of the number of the robots in the simulated swarm. As there no need to include more than 20 robots, so it was easier solve this issue by implementing an array of the size 20 to hold all data for all partner robots.

However, there was a different variety of swarm-friendly simulation platform that could be use such as AeGoS or Player/Stage, but the decision lie on Webots as it was thought that it could be simple to use as simulation platform.

From a personal point of view, by experimenting with Webots for a period of time, it is concluded that Webots would not be completely suggest for use with complex experiments with swarm robotics, or even multi-robot systems. Instead, it could be useful if it is used to build robots from scratch, or use a single based robot model.

3.6 Summary

The simulation platform that has been used for all the experiments during work of this thesis was Webots. Webots is a professional simulation platform for mobile robots. Webots is widely used for educational purposes. A realistic model of insect behaviours was required. The robot model used for all experiments was E-Puck. This model includes all the resources that could suit the purposes of the research of this thesis. The basic structure of the E-Puck controller was built up for the wandering task (as a simple task). Then the other tasks were included like foraging and complex loading and a collection task. For the simple task experiments, the arena was structured as 1.5×1.5 meters square. Then, for adding more challenge to the simulated swarm, the arena was increased to a bigger structure, 5×5 meters square.

Chapter 4

The Swarm Size Evaluation for Foraging Task

Measuring the performance of the simulation experiment involves a number of metrics. Using these metrics will help to evaluate the swarm simulation experiments in the event that any changes occur, and also to what extent these changes affect the swarm system performance.

The swarm foraging task was deployed during the research simulation experiment during this case study. Choosing the right metrics for these simulation experiments are dependent on the swarm task and the observation of the experiment performance, including the proposed exogenous fault detection model in this research. However, the metrics used to evaluate the performance of this system have been chosen carefully to examine the changes in the swarm robotics before and after failure occurs to be applied to the expertness in Chapter 5.

4.1 The Simulation Experiment Performance and Metrics

Depending on the simulated experiment and the swarm task, it is important to carefully choose the right metrics for evaluation. For instance, Kernbach et al. (2013) work was measured with the swarm robot densities based on collective decision-making in the swarm without the existence of communication between individuals. Their work was based on the total number of individual robots that make a strong collective decision. Additionally, the work by Bayindir and Şahin (2007) uses the total energy consumption to measure the performance of the swarm system. Another basic metric is the total time to accomplish the task, as demonstrated by Peng et al. (2012). Ren and Tse (2012) work aims to cover an arbitrary unstructured space using a minimal number of robots. Their experiment measures the total coverage rate of the arena with 20 and 25 robots.

However, the following metrics are going to be used on all the experiments discussed in this thesis.

Simulation Time The simulation total amount of time the swarm robots spent to accomplish the task (in seconds). The task consists of searching for objects, then moving them to the destination area on the arena.

Distance-Travelled Is the total distance-travelled by all swarm robots in the swarm (in meters). Measurement of this distance starts when the simulation is initiated and ends when the task is accomplished.

Energy-Expended The total energy consumed by all swarm robots (in Joules). Every computed action will consume an amount of energy. The energy consumption is roughly analogous to the battery usage. The total energy consumption is the sum of all the energy used by the robots in the duration of the simulation experiment (until the entire simulated swarm task is accomplished).

Practically, the simulation was performed over the course of a single run and compared across multiple runs (ten runs). A key aim is to understand how reliable and efficient the swarm simulation is, without addressing the impact of failure.

At the time of the simulation run, all E-Puck robots start at the same location where they have been placed by the simulation software. The Webots simulation was programmed to store the first generated location for all the robots and objects so that, after the accomplishment of each swarm robotic task, the Webots simulation can recall all the E-Puck robots and place them back at the same initial location on the arena, and prepare them to be ready for the next run.

4.2 Size of the Robotic Swarm

During this simulation experiment, the swarm size was evaluated based on the number of individual robots in the arena. The results were collected after finishing the task ten times with each swarm size.

However, the chose of the swarm running time has no particular criteria except that the number of ten experiment runs has been applied by many other researchers like Liu et al. (2006); O'Dowd et al. (2014).

The swarm sizes that were evaluated are 1, 6, 7, 10, 14, 15, 16 and 20 robots. The arena is the 1.5×1.5 meter square size, as shown in Figure 3.5.

The size were expanded originally from experiments with 1, 10 and 20 robots. Where after collecting the results with theses sizes, it was easy just to go with 20 robots but this will take long time in reality and will be difficult to chose the right size for carrying the experiments in Chapter 5. Additional swarms of six and seven robots were added to determine the changes between a swarm with 1 and 10 robots. Also, the additional sizes of 14, 15 and 16 were added that shows a swarm with 16 robots have finished the swarm task in slightly less time than with 20 robots. However, related to the issues mentioned in Section 3.4 that causes a delay in collecting the results, and because a swarm with 10 robots in Webots gives result that was close to all larger swarms, as shown in the figures, ten was selected. This value maximised swarm size, and ensured that the stimulation times were not excessive.

The evaluation will be based on the comparison of the three metrics: time, distance-travelled and energy-expended per robots and per swarm.

4.2.1 Task Description: Swarm Foraging

The simulated swarm task chosen for detecting the best performance of the simulated swarm size is swarm foraging. The reason for choosing the foraging task is due to the capability of the robot's cooperation, where this could bring the greatest attention for the simulated experiment observations (Krieger et al., 2000).

Typically, simulated swarm foraging involves a number of robots deployed in an arena to search for specific objects and transport collected objects to a specific location on the arena (Winfield, 2009; Lau et al., 2011). The simulated foraging behaviour consists of wandering in the arena, while avoiding obstacles, until the robots encounter and detect the object, and then prepare themselves for the pushing procedure. Figure 3.3 shows the foraging swarm task in simple form, before applying the exogenous failure detection and recovery concepts.

During the experiments in this thesis, the collected results shows that for the task of foraging, swarm robots can take advantage of the collective efficiency while relying on hardware usage when all robots are operational. However, the design of E-Puck model in Webots is capable of dealing with foraging task. While the foraging task requires cooperation between robots, which could add complexity to the swarm behaviour, it is important to continuously carry out the same task for all the experiments in this thesis. That is due to the reasons for justifying the capability of the proposed failure detection and recovery model processing along while doing the task and cooperating with other robots at the same time (**RQ2**). While the processing of the proposed failure detection and recovery model does not compromise the swarm task, it would be possible to test the proposed model with more swarm tasks in future.

The observed data was collected based on retrieving all the deployed objects by pushing them to the destination area on the arena. The simulated swarm task is considered accomplished whenever the arena is freed from all deployed objects.

4.2.2 Results

The simulation experiment was set to run ten times with swarm sizes of 1, 6, 7, 10, 14, 15, 16 and 20 E-Puck robots respectively.

The box plots in Figure 4.1 show a decrease in the simulated time. Also the decrease in average in distance-travelled (Figure 4.2(a)) and average energy-expended (Figure 4.2(b)) compared, gradually, with the increases of simulated swarm size.

Figure 4.3 also show the total of distance-travelled (Figure 4.3(a)) and energy-expended (Figure 4.3(b)) per swarm run.

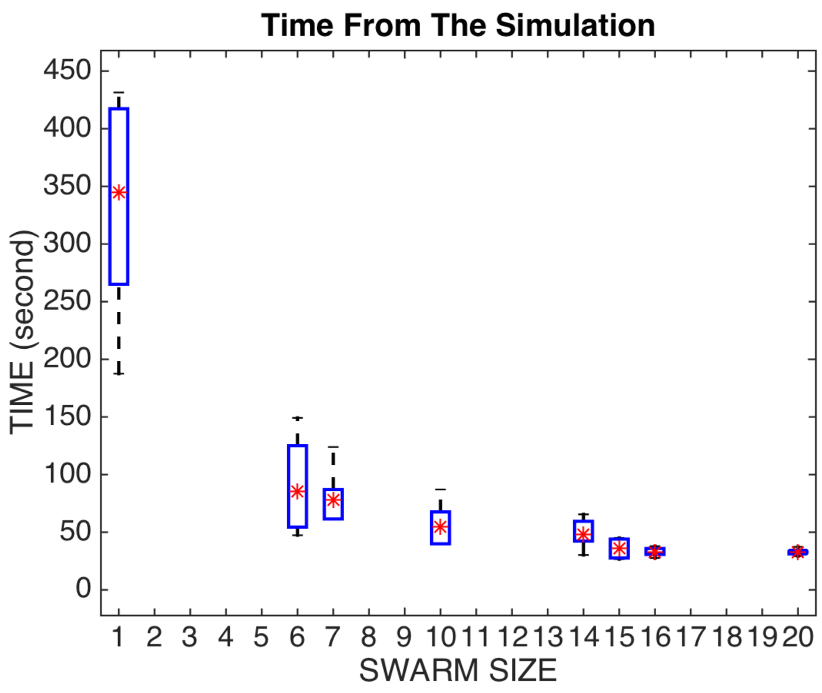
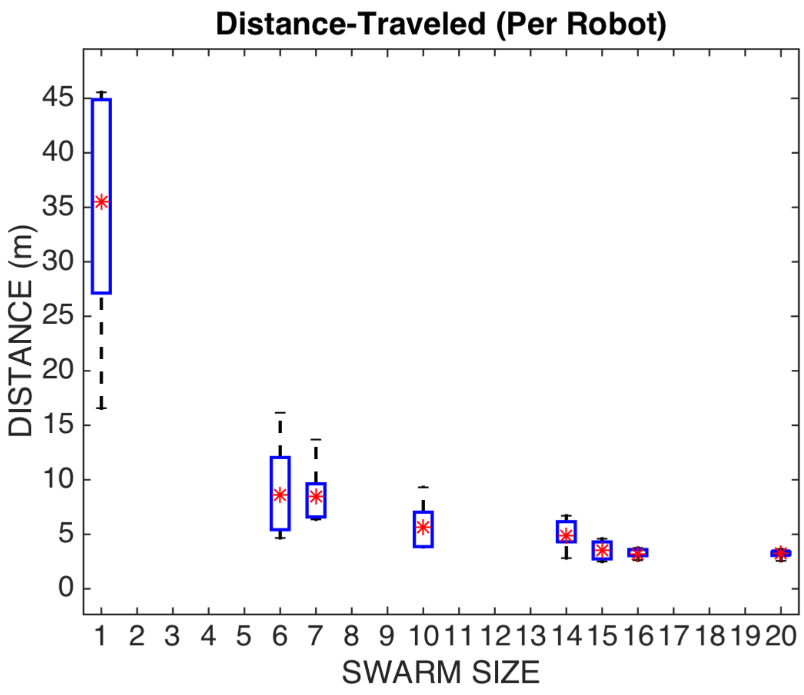


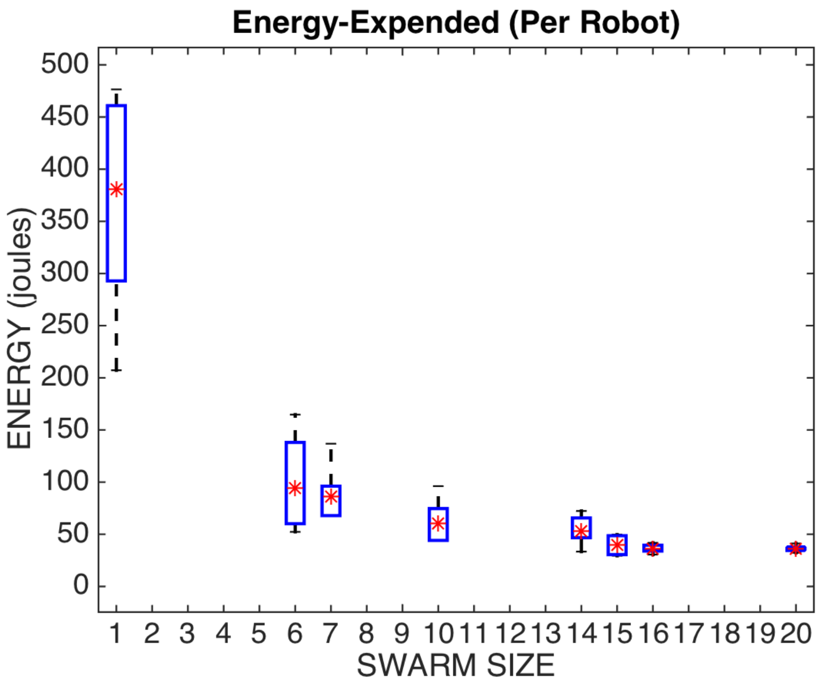
Figure 4.1: Box plots showing the time from the simulated experiment for clearing the arena compared with the swarm size. Each run was repeated ten times. The box represents the skewness in the data. Stars represent the mean, the top line represents the longest running time and the bottom line represents the shortest running time during this experiment.

The simulated time in all experiments represents the average of the total time taken after the simulated swarm task was accomplished. While the distance-travelled and the energy-expended are different according to the statistical analysis per robot and per swarm run. Table 4.1 presents the simulation experiment's data that was collected per robot.

Table 4.2 showing the data was collected from the simulated experiment per swarm run.

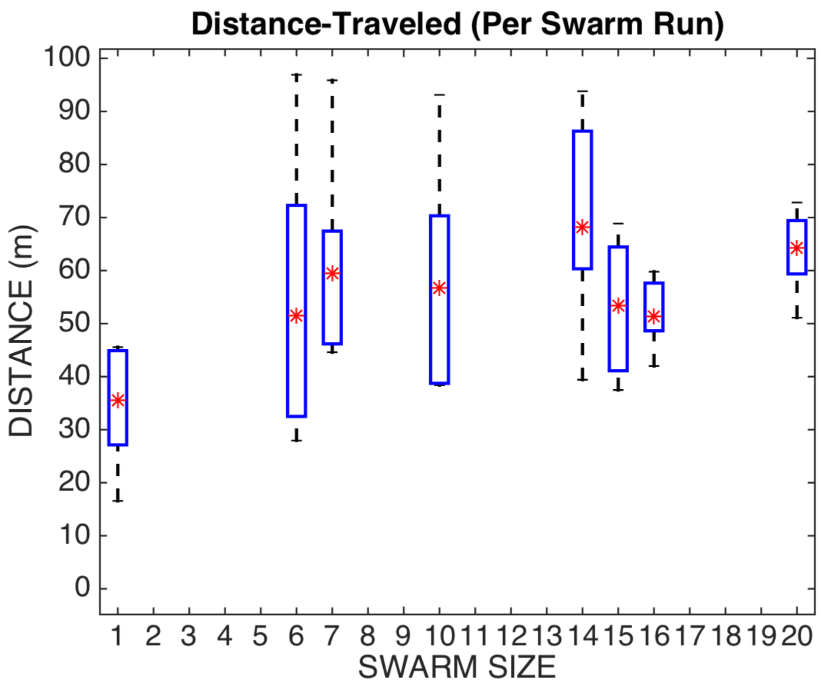


(a) The average distance-travelled per robot for clearing the arena compared with the swarm size.

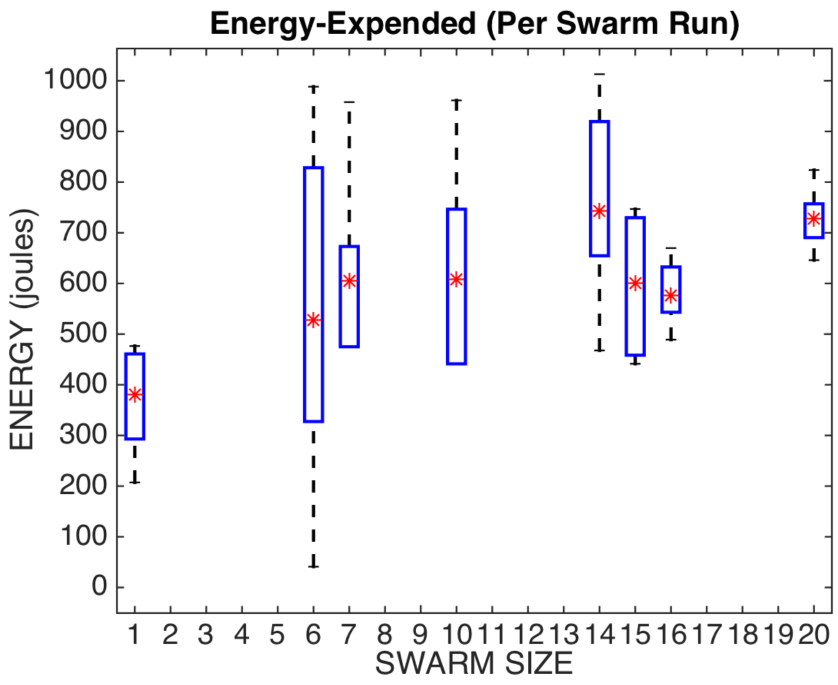


(b) The average energy-expended per robot for clearing the arena compared with the swarm size.

Figure 4.2: Box plots showing the comparison between the performance obtained from the swarm sizes per robot. Each run was repeated ten times. The box represents the skewness in the data. Stars represent the mean, the top line represents the longest running time and the bottom line represents the shortest running time during this experiment.



(a) The total distance travelled per swarm run for clearing the arena compared with the swarm size.



(b) The total energy-expended per swarm run for clearing the arena compared with the swarm size.

Figure 4.3: Box plots showing the comparison between the performance obtained from the swarm sizes per swarm run. Each run was repeated ten times. The box represents the skewness in the data. Stars represent the mean, the top line represents the longest running time and the bottom line represents the shortest running time during this experiment.

Table 4.1: The mean of the simulated time, distance-travelled and energy-expended for the simulated swarm sizes: 1, 6, 7, 10, 14, 15, 16 and 20 robots to clear the arena.

Swarm Size (Number of Robots)	Time From Simulation (Seconds)	Distance-Travelled Per Robot (Meter)	Energy-Expended Per Robot (Joules)
1	344.95	35.53	381.00
6	85.59	8.58	94.52
7	78.35	8.50	86.53
10	55.01	5.67	60.75
14	48.09	4.87	53.10
15	36.27	4.22	40.12
16	32.58	3.21	35.98
20	32.97	3.23	36.41

Table 4.2: The total of distance-travelled and energy-expended, in addition to the simulation time, for the simulated swarm sizes: 1, 6, 7, 10, 14, 15, 16 and 20 robots to clear the arena.

Swarm Size (Number of Robots)	Time From Simulation (Seconds)	Distance-Travelled Per Swarm (Meter)	Energy-Expended Per SWARM (Joules)
1	344.95	35.53	381.00
6	85.59	51.50	527.15
7	78.35	59.48	605.68
10	55.01	56.71	607.45
14	48.09	68.12	743.45
15	36.27	53.31	600.84
16	32.58	51.32	575.62
20	32.97	32.97	728.20

4.2.3 Discussion

After obtaining the simulation experiment results with different swarm sizes, it is time to investigate the performance of the swarm simulation experiment based on the results observed in Section 4.2.2.

From Figure 4.1 and Figure 4.2, it is probable that the simulated swarm performance will improve as the swarm size increases. The simulated swarm robots get significantly better with a swarm size from 10 to 20 robots. The simulated time (Figure 4.1) for swarm sizes between 10 and 20 robots is relatively flat. This means that swarm sizes between 10 and 20 robots are likely to show only a slight change in the overall simulated swarm performance. Figure 4.2(a) and Figure 4.2(b) show similar observations for distance-travelled and energy-expended.

However, increasing the number of robots in the simulation swarm causes the simulation experiment to take a long time to simulate one complete task. Therefore, as the difference in the performance between swarm sizes of 10 and 20 robots is rather small, 10 robots is the suggested size for providing a good evaluation for the next experiments.

After determining the best swarm size to use during the next experiments, it is necessary to analyse the combinations of metrics, simulation time, distance-travelled and energy-expended together, Figure 4.4.

The statistical represented in Figure 4.4 was used to measure how close the data are to the fitted regression line of the related experiments for swarm task completion with ten robots. These statistical measurements are a new metric method for verifying the importance of applying the proposed exogenous failure detection and recovery model to the swarm robotics after implementing failures. The collected data in Figure 4.4 later will be compared with the collected data from a swarm with failed individual robots. These data will show that in the presence of failed individuals the regression line will change as long as the swarm behaviour is distracted with the faulty robots while the operational robots are working to finish the task.

The optimum combination is where the energy is proportional to the simulation time, where the measurement of R-squared indicates that energy-expended and simulation time are a perfect fit. This is clear from in Figure 4.4(c), where the trend-lines in both diagrams are similar.

In Figure 4.4(a), the relationship between the simulated time and the distance-travelled are linear in each run, which means that the speed observed in each run is linearly when compared to the other runs. A couple of runs are perfectly compatible with each other. This relationship would be affected if there existed any failures in any swarm run.

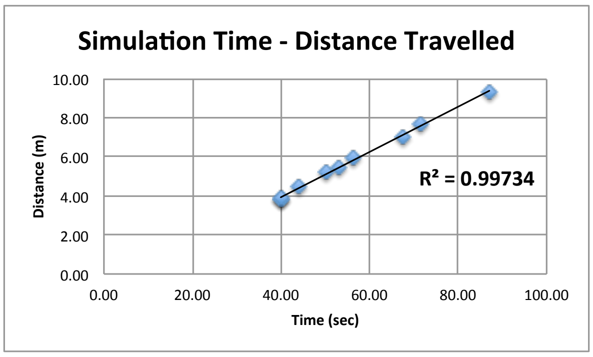
However, these metrics will be taken into consideration and compared with the results obtained from the next experiments to determine whether any changes could be observed in the future experiments.

4.3 Summary

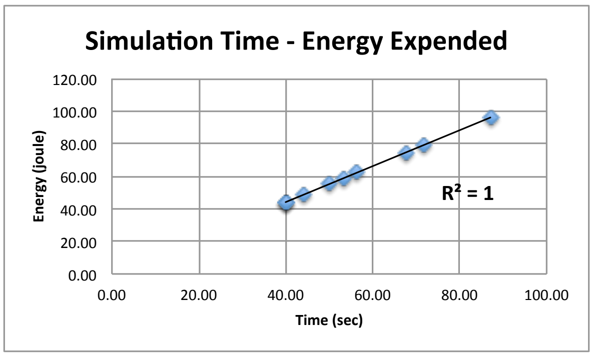
The evaluation performance of swarm robotics experiments requires a number of metrics. The time from the simulation, the distance-travelled and the energy-expended were the three metrics chosen to be carried out for the experiments in the entire theses.

Swarm foraging task was deployed during the research simulation experiment, were robots have to collect a number of objects to the collection location in the arena. The swarm task would be considered accomplished whenever all objects are collected in short time.

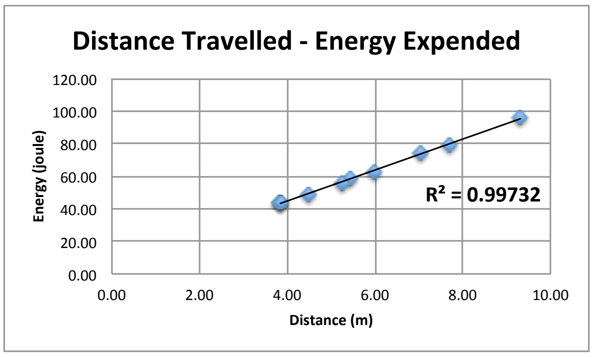
The observation from the swarm size evaluation experiments shows that a number of ten robots is an acceptable number for the experiments in the next chapter.



(a) The compatibility of time and distance-travelled with 10 robots to clear the arena from objects.



(b) The compatibility of time and energy-expended with 10 robots to clear the arena from objects.



(c) The compatibility of distance-travelled and energy-expended with 10 robots to clear the arena from objects.

Figure 4.4: The correlation between the time, distance-travelled and energy-expended for a swarm size of 10 robots.

Chapter 5

Failures Impact on the Swarm Foraging Task

Swarm robots can exhibit a number of failures that could affect individual robots at the software or the hardware level. Failed individuals are capable of threatening the entire swarm robotics performance. It is therefore considered important that the swarm robots are able to deal with different types of failures that could constitute a threat to the swarm robotics task. The swarm task encountered during simulation experiments is the swarm foraging. The foraging task requires swarm robots to search for objects and transport them to a specific location (Winfield, 2009; Lau et al., 2011).

The simulation experiment of this research will thoroughly investigate associated failure symptoms, at the hardware-level, then introduce a unique form of failure detection technique.

To draw attention to the reliability of the swarm robotics, it is important to evaluate the robustness of the swarm robots through a simulation environment. This will eventually lead to the evaluation of the new technique for detecting faults and mitigating their impact on the swarm.

5.1 The Proposed Approach for Exogenous Failure Detection

After examining the two approaches, fireflies (Christensen et al., 2009) and the Robot Internal Simulator (Millard et al., 2013), it is possible to observe several gaps in both approaches, including the lack of direct communication between robots and the space limitation between each attempt at simulator diagnosis where a fault could occur during the analysis period.

In this research, the exogenous fault detection approach combines several techniques drawn from other researches, including those of Christensen et al. (2009) and Millard et al. (2013).

In this approach, each robot in the swarm works to diagnose the health status of its neighbouring robots. The achievement of this approach requires robots to cooperate together to ensure the reliability of the swarm robotics. Then they are compared against the values possessed by other robots' controllers. The values possessed by the other robots' controllers are renewable, which means that each robot acquires fresh values but does not hold them unless a robot is suspected of having a fault. After the robot confirms that there is no suspicion, these values are overwritten with other values, or released if there is nothing suspicious at all.

Figure 5.1 represents the exogenous fault detection model used during these research experiments. It involves all the input and output values through the robot controller.

Communication is necessary between the robots during the exogenous fault diagnosis step. As a robot has the ability to receive broadcast messages from multiple robots, this feature will allow the robot to diagnose even multiple robots while they are also being diagnosed by others.

5.2 Exogenous Fault Diagnosis Concept

The computational procedure shown in Figure 5.1 plays an important part in the robot's controller. As a robot has the ability to diagnose itself, it also uses the same procedure to diagnose other robots within the communication range. This makes the swarm evaluation simple, as there is no need to involve a third-party agent from outside the swarm of robots to resolve issues.

With the communication available between robots, each robot broadcasts a set of data as a packet including its own co-ordinates on the arena: orientation, motors values and sensors values. It also includes its computed diagnosis results in the same packet after it receives a response. This allows the receiving robot to diagnose and compute individual data that is received from the broadcaster robot among the swarm, then to compare the outcome results against the received computed results from the same robot.

Distance is the common concept that could be computed from the two devices onboard a swarm robot, the communication device and the navigation device. Based on the possessed robots' co-ordinates that were broadcasted among individual robots within their communication range, robots will be able to compute the distance between them. On the other hand, individual robots compute the distance based on the signal strength of the received message from the partner robot.

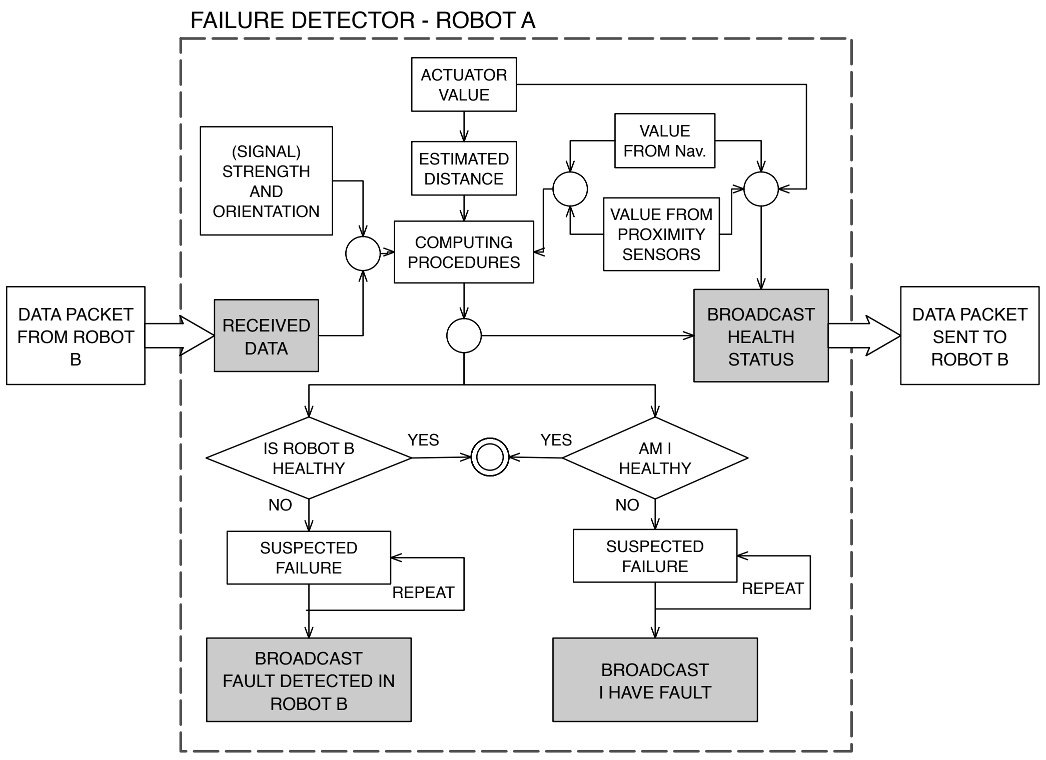


Figure 5.1: The exogenous diagnosis approach for failure detection. In this approach, robots diagnose each other through a direct communication between them. Each robot broadcasts its health status signal to its neighbours. The controller of robot A receives the sensors, coordination, and motors values. Then, the controller computes these values, together with values received from robot B. At the same time, the controller of robot A broadcasts individual values along with the diagnosed information to the nearest robot in its communication range. When robot B receives these values and information, it starts to diagnose the values and computes the movement of robot A based on the values of the communication module, including the signal direction and the distance between robots' A and B. If none of these values match, then both robot A and robot B broadcast a suspicious report on each other. At this stage, robot A and robot B need to confirm which one has the failure. They follow the same procedure to check if the suspicion exists with another robot in the swarm. The faulty robot will be acknowledged whenever a healthy robot confirms which suspicious robot has the fault. The faulty robot will be reported to the entire swarm of robots. This will allow the swarm to take an action toward recovering/mitigating the faulty robot.

While robots are within the same communication range, they also use signal strength to diagnose any failures in neighbours. Signal strength helps to compute the distance between the emitter and the receiver robots. Distance is a measurement parameter that can be used to assure the healthiness of individual robots based on the data given. If the computed distance in one robot does not match with the received computed distance from a partner, and so if the distance computed from the co-ordinates does not match with the distance computed from signal strength, then these robots will become suspicious.

Realistically, the distances computed from different devices could not be totally accurate. However, a ratio value were considered for measuring the two distance values that where computed in robots' controller. Depending on the system developer and the type of the swarm robot model used, the ratio value could be vary. In the experiments with the E-Puck robot, the contrast rate value was set to 80 per cent accuracy matching. This contrast rate value was tested only on the E-Puck robot, but it could work with other swarm robot models as well.

Figure 5.2 shows how robots A, B and C diagnose each other, based on the data provided. When robots A and C are within each other's communication range, robot A is assumed to receive robot C's health packet including identification and co-ordination. X_C and Z_C are the co-ordinates of robot C, and so on with robots A and B. The time represents the time of the simulation where all the robots are initiated together. Robot A computes the distance $d_{navigation}$ (a,c), as in Equation 5.1. At the same time, robot A computes the distance d_{signal} with the use of the communication signal strength, as in Equation 5.2. Robot A includes robot C's identification along with its coordination and both computed $d_{navigation}$ and d_{signal} in one packet, then broadcasts it to the swarm. Both robot B and robot C are going to receive robot A's packet, but it will be accepted by robot C and rejected by robot B because it has robot C's identification. Robot C then computes its $d_{navigation}$ and d_{signal} and includes robot A's identification, then broadcasts it to the swarm. Robots are considered healthy whenever $d_{navigation} = d_{signal}$.

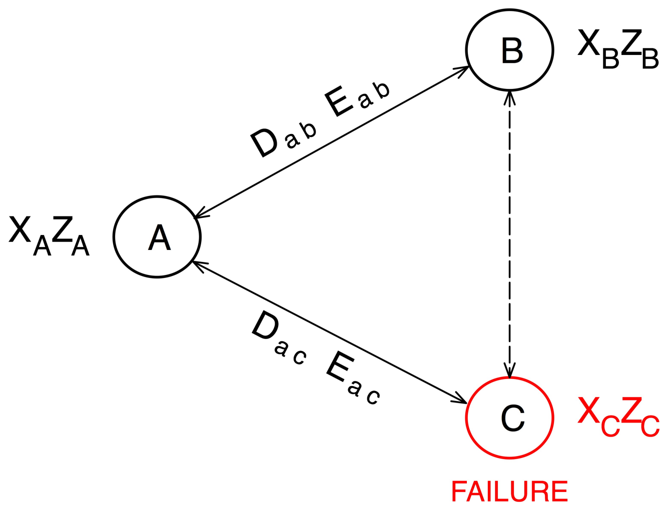


Figure 5.2: A case scenario where 3 robots are analysing the data received from their partners while they are travelling in the arena. This figure shows the orientation of robots A, B and C with arrows. Robot C have a fault in its navigation device.

$$d_{navigation} = \sqrt{(X_C - X_A)^2 + (Z_C - Z_A)^2} \tag{5.1}$$

$$d_{signal} = f(SignalStrength^2) \tag{5.2}$$

However, during this simulation experiment, the navigation device in robot C has been injected with a fault, where at Δt time if robots' A and C compute that $d_{navigation} \neq d_{signal}$, then they store their current computed values and check with a third robot by following the previous procedures. Whenever robot A computes that $d_{navigation} = d_{signal}$, or robot C computes $d_{navigation} \neq d_{signal}$, they both now check each other's computed values to identify the cause of the failure in robot C. Based on the data computed in robots' A and C, the cause of the failure is in the navigation device in robot C.

5.3 Injecting Multiple Failures into a Simulated Swarm

The previous experiment concluded that the best swarm size to observe the next experiments was 10 robots. A number of 3 robots were injected with faults (noise) in their navigation device. The effect of the fault initiates with the start of the simulation experiment.

The remaining robots were fully operational. The exogenous failure detection technique was applied to all the robots' controllers.

The simulated swarm robots' task is foraging (as described in Section 4.2.1), where the robots have to bring all six objects to the destination area. Whenever a robot was detected as faulty, the controller stops the robot's movement, then shuts it down to avoid causing any further impact on the simulated swarm robots. The reason for shutting down faulty robots is to prevent continual broadcasting of false values to healthy robots. Over time, if faulty robots are not shutdown, the suspicious robot is going to continually determine if it has a fault or not, which will cause the suspicious robot to be stuck in the exogenous fault detection technique loop, therefore, impacting the entire swarm robotic task. A further experiment demonstrates the reason for the shutdown of faulty robots is represented in Section C.1.

When the arena is cleared of all objects, the results are then logged before the simulation recalls for the next run, until the results of all 10 runs have been collected.

The evaluation of this experiment will be based on the comparison between the reaction of the simulated swarm's behaviour before and after it was injected with a fault.

Note that during this experiment there was no reaction toward recovering or mitigating the simulated swarm robotics after the fault was detected.

5.3.1 Results

In this experiment, the presence of the fault was successfully detected in all the simulated swarm robotics experiments. Table 5.1 shows the reaction simulation time of the 3 faulty robots after detection and shutdown. The collected experiment results include the simulation time required by the robots to detect the suspicious robot, checking with another healthy robot, identifying which device has the fault, and then shut itself down.

Table 5.1: Reaction time from the simulation experiment of the 3 faulty robots from first detection of the 3 failures until the shutdown of all 3 faulty robots.

Shortest Time	0.16 Sec
Mean Time	0.2 Sec
Longest Time	0.22 Sec

Meanwhile, the operational robots continue towards finishing the simulated swarm task until the task is accomplished.

The box plots in Figure 5.3 show the differences between the simulated swarm robots before and after they are injected with a failure.

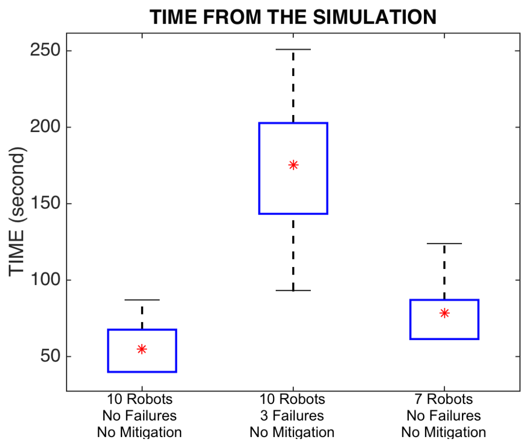
The diagram indicates the significant increases in the simulation time, distance-travelled and energy-expended when the simulated swarm is injected with a failure. These increases are compared to the simulated swarm without the existence of the fault. Table 5.2 shows the mean of the three metrics: simulation time, distance-travelled and energy-expended, of the simulated swarm robots.

Table 5.2: Average time, distance-travelled and energy-expended for 10 runs before and after the simulated swarm was injected with 3 failures. In addition to a comparison to a simulated swarm with 7 operational robots.

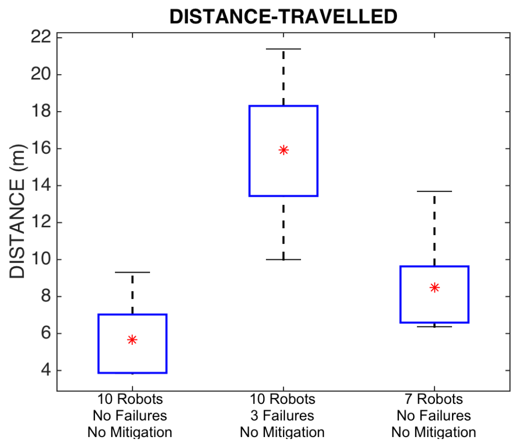
Metric	10 Robots No Failures	10 Robots 3 Failures	7 Robots No Failures
Time from Simulation (Seconds)	55.01	175.34	78.35
Distance-Travelled (Meter)	5.67	15.93	8.50
Energy-Expended (Joules)	60.75	135.64	86.52

5.3.2 Discussion

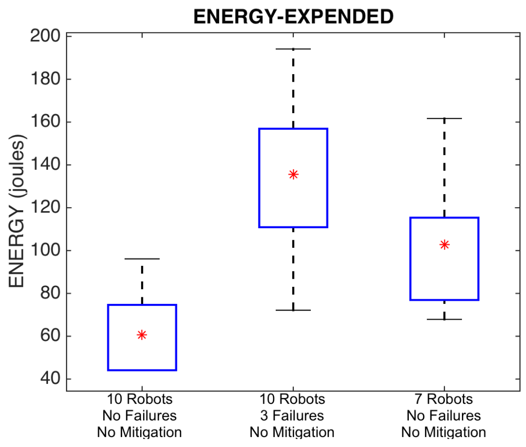
After the robots become suspicious of the existence of a fault in another, they need to confirm and identify this fault. Identifying the fault requires at least a third healthy robot. If the third robot comes under suspicion too, then they all carry on finding a healthy one for confirmation. However, when the robot is confirmed to be faulty, it



(a) The impact of failures on the simulation time to clear the arena of objects.



(b) The impact of failures on the distance-travelled to clear the arena of objects.



(c) The impact of failures on the energy-expended to clear the arena of objects.

Figure 5.3: Box plots showing a comparison of the performance of the simulated 10 swarm robots before and after injection of a failure. The data in these diagrams correspond to the simulation time, distance-travelled and energy-expended.

immediately shuts itself down to avoid causing any harm/interference to the operational robots, and thus to the entire simulated swarm task.

The existence of the failure causes the simulated swarm robots to take longer to finish the task than a swarm with no fault. The fault also negatively affects the simulated swarm's distance-travelled, which forces robots to travel longer distances. This would reduce the actuators' performance over time on real robots. Furthermore, the existence of the fault causes wasteful expenditure of the robot's energy, whether it is simulated or a real robot.

In Figure 5.4, the combination between the simulation time and the energy-expended still shows a proportional relationship in both Figure 5.4(b) and Figure 5.4(e), which means that using either one of these two metrics during the evaluation provides the same results. The proportional relationship between the simulated time and energy-expended is totally clear in Figure 5.4(f) and Figure 5.4(d), where the trend-lines are comparable in both diagrams.

However, as claimed in the previous experiment, the relationship between simulation time and distance-travelled changes after the injection of a failure (Figure 5.4(d)). This means that the speeds in each swarm run are no longer linear as they were in the simulated swarm experiment with no failure (Figure 5.4(a)).

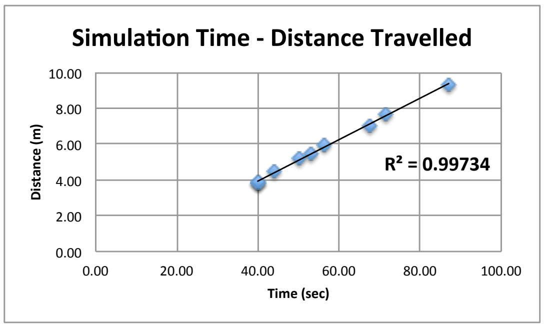
The next simulation experiment will observe the simulated swarm robots after mitigating the fault. This requires the nearest robots to locate the nearest faulty robot, then approach it in order to push it away from the arena.

5.4 Mitigating the Impact of Failures on the Simulated Swarm Robotics

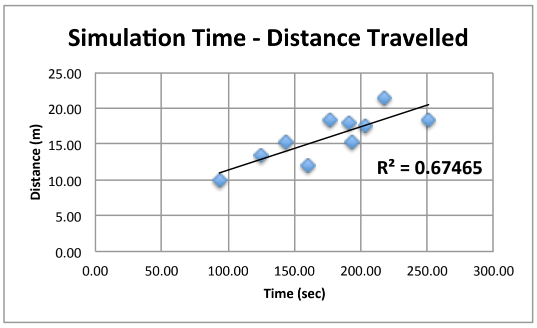
This particular simulation experiment will focus on evaluating the simulated swarm robotics after mitigating the cause of hardware failure. A comparison between results obtained from applying the mitigation to the simulated swarm and results from the previous experiment will be discussed in this section.

In the previous simulated experiment, faulty robots shut themselves down after they were confirmed to be faulty, based on the exogenous fault detection model.

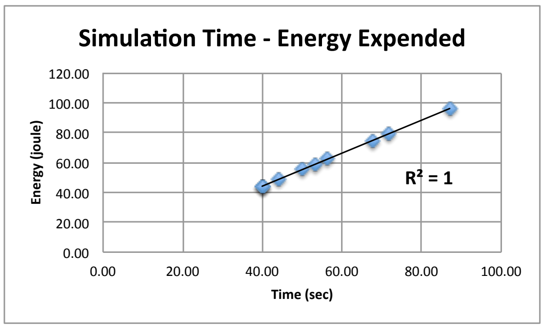
The procedures for mitigating the impact of the faulty robot depend on the last values received by the healthy robot. During this stage, the healthy robot starts to analyse the cause of the fault. If the fault occurs in the navigation device, then the healthy robot locates the faulty robot based on the orientation and distance as computed from the signal strength (Section 5.2).



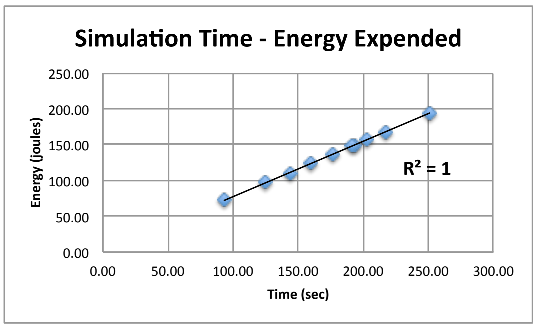
(a) The compatibility of time and distance-travelled with 10 robots, with no failures, to clear the arena from objects.



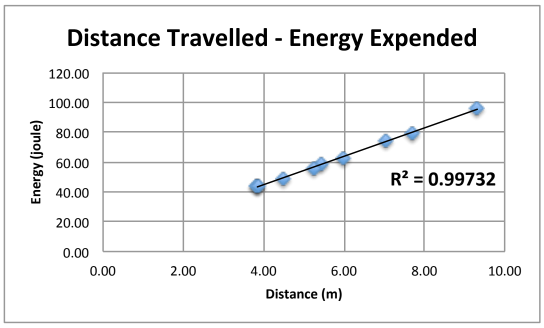
(d) The compatibility of time and distance-travelled with 10 robots, of which 3 are failed, to clear the arena from objects.



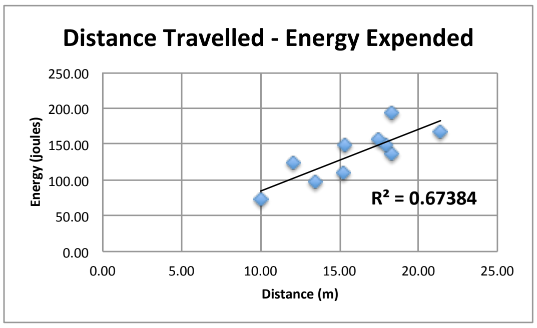
(b) The compatibility of time and energy-expended with 10 robots, with no failures, to clear the arena from objects.



(e) The compatibility of time and energy-expended with 10 robots, of which 3 are failed, to clear the arena from objects.



(c) The compatibility of distance-travelled and energy-expended with 10 robots, with no failures, to clear the arena from objects.



(f) The compatibility of distance-travelled and energy-expended with 10 robots, of which 3 are failed, to clear the arena from objects.

Figure 5.4: The correlation between the time, distance-travelled and energy-expended for a swarm size of 10 robots. Diagrams (a), (b) and (c) represents a swarm with 10 operational robots having no failures. Diagrams (d), (e) and (f) represents a swarm with 10 robots of which 3 have failures.

After getting the distance and orientation, the healthy robot can now move in a straight line until it reaches the faulty robot. When the healthy robot approaches the faulty robot, the healthy robot pushes the faulty robot to the nearest side of the arena. This will keep the faulty robot away from the operational robots while they finish the simulated task.

5.4.1 Results

In this experiment, all the faulty robots have been successfully moved to the side of the arena. Figure 5.5 shows the decreases in the simulation time, distance-travelled and the energy-expended compared with the previous experiment, where the robots have a fault, and after mitigating the impact of the faulty robots.

All three metrics were affected after applying the mitigation procedure to the faulty simulated swarm robots. Table 5.3 shows a comparison of the mean of simulated time, distance-travelled and energy-expended between the simulated swarm robots after the faulty robots have been shutdown, and after the mitigation procedure has been applied.

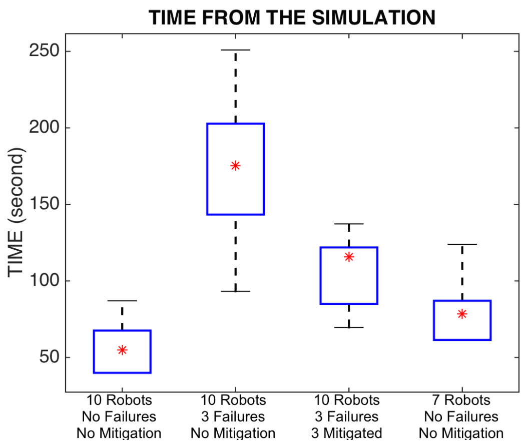
Table 5.3: The average time, from the simulation experiment, the distance-travelled and the energy-expended before and after detecting the 3 failures then mitigate them. In addition to the results from a simulation swarm with 7 operational robots (with not failures) shows how the simulated swarm behaviour has been improved before and after mitigating the faulty robots.

Metric	10 Robots No Failures No Mitigation	10 Robots 3 Failures No Mitigation	10 Robots 3 Failures 3 Mitigated	7 Robots No Failures No Mitigation
Time from simulation (Seconds)	55.01	175.34	115.85	78.35
Distance-Travelled (Meter)	5.67	15.93	12.65	8.50
Energy-Expended (Joules)	60.75	135.64	89.61	86.52

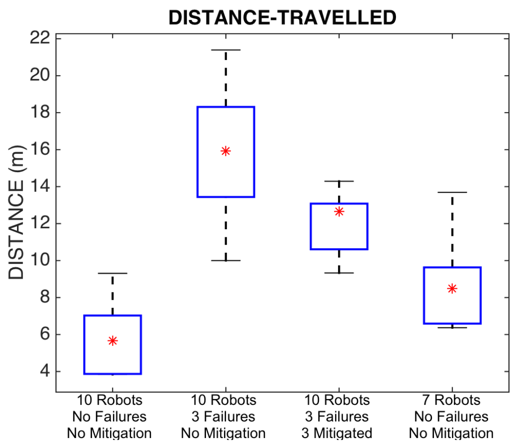
5.4.2 Discussion

In Figure 5.5(a), after applying the mitigation procedure, the simulation time has significantly decreased. This means that the simulated swarm robots managed to finish their task in less time, even with the existence of the faulty robots. The mitigation allows operational robots to travel freely without being blocked by faulty robots. The mitigation procedure also allows the simulated robots to accomplish their task with less travelling (Figure 5.5(b)).

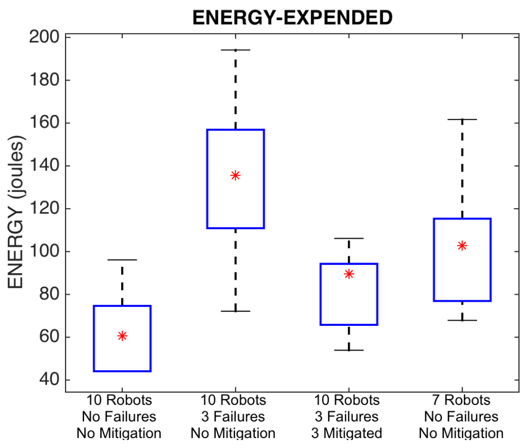
Apart from the energy-expended after the mitigation procedure takes place, Figure 5.6(b) and Figure 5.6(e) show that the trend-line of the energy-expended is proportional to the simulation time, where the measurement of the trend-line, R-squared, indicates that both energy-expended and simulation time are still perfectly fitted during all three experiments.



(a) The effect on the simulation time to clear the arena of objects.



(b) The effect on the distance-ravelled to clear the arena of objects.

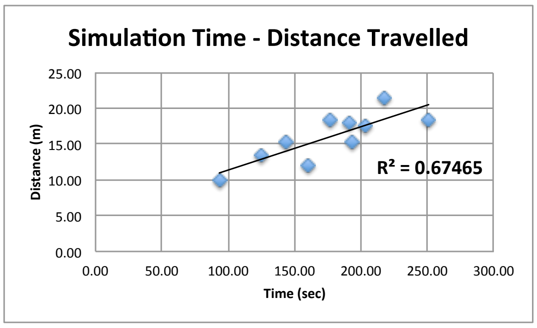


(c) The effect on energy-expended to clear the arena of objects.

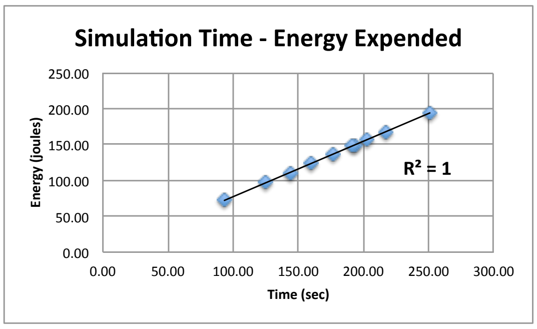
Figure 5.5: The observed performance of i) 10 operational robots; ii) 10 operational robots of which 3 robots failed (become stationary); iii) 10 robots of which the 3 failed are pushed to the side of the arena; iv) 7 operational robots.

From Figure 5.6(d) it is observed that the relationship between the simulation time and the distance-travelled has improved after the mitigation, compared to the relationship between the simulation time and distance-travelled with the existence of a failure (Figure 5.6(a)). This is because the speed has been enhanced after mitigating the failure in the simulated swarm.

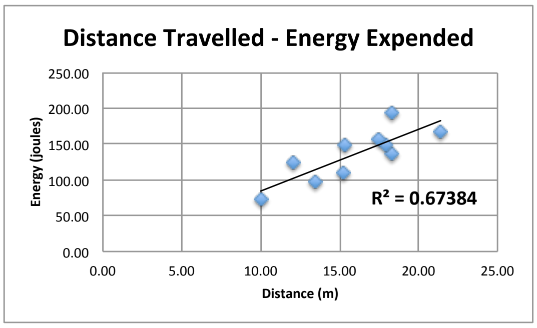
As there is no difference in the simulation time and energy-expended (discussed previously), the relationships between them are shown in Figure 5.6(e) and Figure 5.6(b).



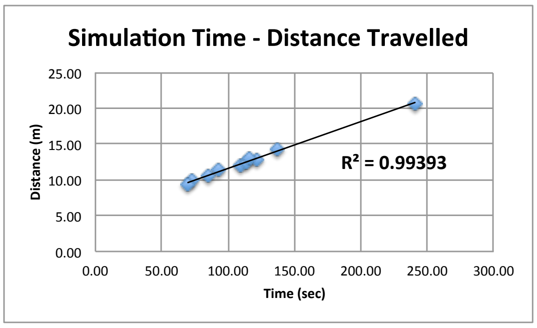
(a) The compatibility of time and distance-travelled with 10 robots, of which 3 are failed, to clear the arena from objects.



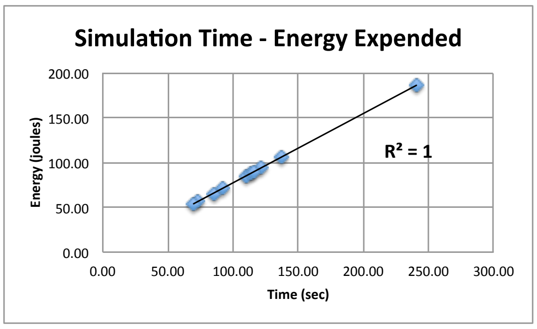
(b) The compatibility of time and energy-expended with 10 robots, of which 3 are failed, to clear the arena from objects.



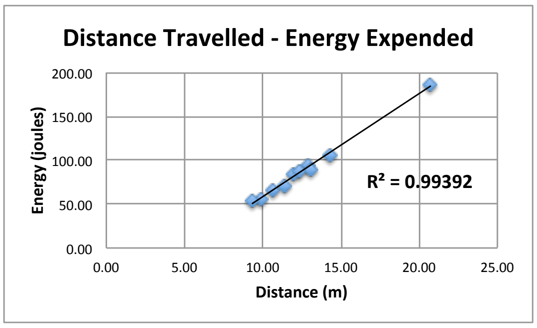
(c) The compatibility of distance-travelled and energy-expended with 10 robots, of which 3 are failed, to clear the arena from objects.



(d) The compatibility of time and distance-travelled with 10 robots, of which 3 failures are mitigated, to clear the arena from objects.



(e) The compatibility of time and energy-expended with 10 robots, of which 3 failures are mitigated, to clear the arena from objects.



(f) The compatibility of distance-travelled and energy-expended with 10 robots, of which 3 failures are mitigated, to clear the arena from objects.

Figure 5.6: The correlation between the time, distance-travelled and energy-expended for a swarm size of 10 robots. Diagrams (a), (b) and (c) represents a swarm with 10 robots of which 3 have failures. Diagrams (d), (e) and (f) represents a swarm with 10 robots of which 3 failures are mitigated.

5.5 Summary

It is considered necessary for swarm robotics to be able to deal with different types of failures that could constitute a threat to the swarm robotics task. Achieving reliability requires identification and addressing the different failures and their effects on individual robots, as well as the entire robot swarm. The objective of this achievement lies in detection mechanisms that can identify and address the fault in the swarm and then mitigate it.

The proposed model in Figure 5.1 was used to provide reliability to the simulated swarm robotics during this research.

Figure 5.5 shows the variation in the simulation time, distance-travelled and expended-energy of the simulated swarm experiments without failure, after failure had been injected, and after mitigating the failure. It is clear that the use of the exogenous fault detection model enhanced the simulation time, distance-travelled and expended-energy. Therefore, it is good to use the exogenous fault detection model in order to save time, distance-travelled and expended-energy.

More investigation of this model will be conducted in order to accommodate more features for the mitigation and recovery level.

Chapter 6

The Swarm Size Evaluation for Loading and Collecting Task

Brutschy et al. (2014) claim that researchers often consider task abstraction in their experiments, rather than focusing on the details of task execution. They focus on how robots interact and cooperate to perform tasks (such as foraging) without direct interaction between robots and objects. The retrieval of an object from a source to a destination is usually abstracted into a trip between the two locations in the arena. However, the task execution can take a different dimension, such as in complex swarm tasks, especially in the physical interaction form. Considering partition/division of the swarm task between multiple robots may require more observation and details to understand the reliable task execution.

A swarm robotics task could be subdivided into several subtasks to be handled by a group of robots. So that, each robot can perform a subtask together, or individually with each member in the group.

Task partitioning is the general term referring to subdividing the swarm robotic task into subtasks. Task partitioning was defined by Jeanne (1986) and Pini et al. (2014) as a technique to organise work that consists of decomposing a task into a number of subtasks. The concept of this is to allow different groups of swarm robots to perform each of the subtasks, either in parallel, or in a defined order and different moments in time.

The observation of the case study during this chapter will help to improve the industrial robots at warehouses to keep on cooperating while transporting containers.

The simulation experiment, during this chapter, will be deployed without injecting the swarm with failures. The result from the this simulation experiment will be used as a base measurement to be compared and evaluated with the executed results after injecting the simulated swarm with failures. The behaviour of swarm robotics with injected failures will be demonstrated later in Chapters 7 and 8.

The reason for implementing subtasks in swarm robotics is to observe the performance of swarm robotics while handling a complex task.

6.1 Simulation Experiment

The discussion regarding the previous simulated experiments (given in Section 4.1), focuses on the simulated swarm robotic performance from the perspective of how the simulated swarm will be affected whenever changes occur to the swarm robots. This was performed when each swarm member handles one simple task by itself (with no cooperation). The results collected from the simulated swarm experiment also have provided details about the possible number of simulated swarm members that are needed to accomplish the task in an efficient time. However, for the case study that will be discussed during this chapter, the simulation swarm robotics will perform a cooperation behaviour between multiple members. In this case, cooperation will require robot members to partition and share their resource capabilities with each other. For instance, cooperative robots will have to share their proximity sensors reading over communication for the purpose of accomplishing the swarm task. This will improve the ability of the robots to transport large and heavy objects if the effect of one robot is not enough to achieve the swarm task.

In order to achieve the efficiency of swarm robotics cooperation, a mean of communication between robots is required to achieve the swarm task. However, still there will be no leader or stationary base communication to control the swarm members. So that, within the communication range, whenever a robot encounters the container, the robot should broadcast its information to the nearest neighbouring robot in request for cooperation, in order to accomplish the task.

The communication range is limited to a short range (0.4m). That is due to the specification of the communication module unit installed in the E-Puck robot. Although the E-Puck has the ability for a handle long distance communication (a range over 0.4m), short range can guarantees the delivery of packets completely with no lost. Thus, the cooperation between robots might be affected depending on the arena size. Therefore, the right number of E-Puck robots are required for the provided task during experiments in this case study.

6.1.1 The Arena Structure

The simulated swarm robotics in this chapter will be performing in a 5×5 meter square arena size. A number of eight containers are placed in the middle of the arena. Each container is surrounded with a black collar that can be detected by the ground sensors that are installed under the robot's body.

Regarding the swarm task during this experiment, the arena will have a loading location and a collecting location. The simulated swarm's task is to transport all containers, first, to the loading location, then to the collecting location. The collecting location will hold all the containers after been loaded and transported until the accomplish of the swarm task, Figure 3.6.

6.1.2 The E-Puck Robot Controller Structure with the Embedded Cooperation Attribute

Particularly, the controller of the E-Puck robot, in this case study, was designed especially to accommodate the cooperation feature between two robots, by sharing each other's data, Figure 6.1. When the first robot detects the container, it starts to position itself at the rear side from the container and facing the loading location. At this point, the robot waits for assistance from the nearest partner, within the communication range, in order to transport the container. When the second robot detects the container, or when it receives an assistance message, the second robot positions itself at the front side of the container and facing the loading location. At this moment, both robots inform each other that they are ready to transport the container, then start moving together.

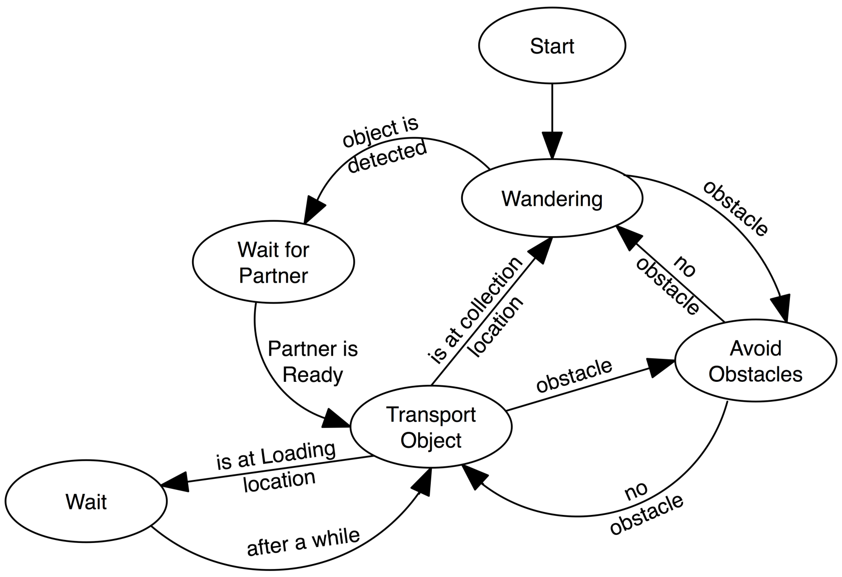


Figure 6.1: The structure of the E-Puck controller that shows how robots cooperate together in order to transport the container.

The mechanism of the cooperation will allow two robots to perform as one independent robot by partially sharing the ability of their resources. For instance, the rear robot, from the container, would be fully aware of the front robot sensing, especially when the front robot encounters an obstacle, so that they can take action to avoid this obstacle.

Figure 6.2 explains the mechanism of how robots sharing each other sensors in order to avoid obstacles, while they are transporting containers.

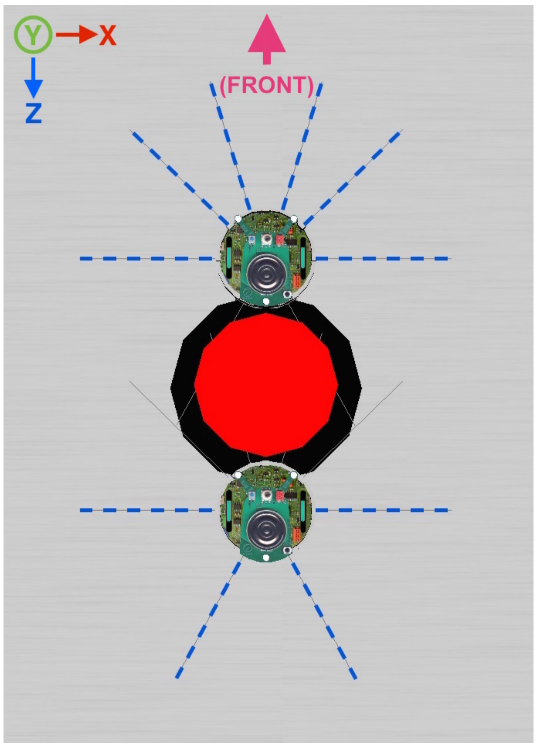


Figure 6.2: The cooperation between two robots allows them to collect data and share it to undertake action decisions, such as avoiding obstacles. The red cylinder, surrounded with a black circle, is the container that has to be transported. The blue dashed lines show the direction of the active proximity sensors on each robot and shared with the partner. So, both robots in this figure work as one single unit.

6.2 Task Description: Loading and Collecting Containers

The swarm behaviour for the simulated experiment could be represented as an application for the industrial zone or warehouses. Although industrial robotic systems do not represent swarm robotics, the swarm robotics principles still apply during these next simulated experiments.

During the simulation experiments in this chapter, the arena will be obstacle-free, where robots will be wandering freely with no collisions with obstacles (except the collisions with each other). All containers, in this case, will be gathered at the centre of the arena. Before the simulation run, all robots are placed at the corner of the arena on the other side from the loading and collecting locations.

The behaviour of the cooperative swarm robots consists of a number of steps as follows:

1. At the beginning of the simulation, robots begin wandering within the arena.
2. Robots use ground sensors for searching and detecting the black-coloured collar at the bottom of the container, therefore the container will be considered detected.
3. When the robot detects the container, it starts computing the coordination and orientation to the loading location, then moves and positions itself at the rear side of the container.
4. At this moment, the robot broadcasts its information, including its position and waits for assistance from another robot.
5. Whenever another robot receives the broadcasted message within the communication range, it orientates itself and starts to move to position itself at the front side of the container and near the first robot.
6. Now, both robots send a ready signal to inform each other that they are ready to transport the container to the loading location on the arena.
7. While they are in transporting mode, both robots share their sensing capability in order to avoid obstacles.
8. At the loading location, robots reposition and orientate themselves with the collection location.
9. Then, they continue transporting the container to the collection location.
10. Whenever all containers are collected, the swarm task is considered accomplished.

Step 7 may not be considered as a requirement step because the arena is obstacle-free, but this step is still required as robots may have a collision with each other during this particular simulated experiment.

The data flow diagram in Figure 6.3 presents the behaviour of two robots cooperating together to transport the container to the loading zone, then to the collecting zone.

6.3 The Performance of the Complex Swarm Task

The simulation experiment in this section will observe and evaluate the efficiency of the simulated swarm robotic size in the large arena size (5×5) meter square. Because there is a need for cooperation between at least two robots, the observation of the simulation experiment will start with 4 robots. Then the size will increase with every even number. The simulated swarm robotic sizes are: 4, 6, 8, 10, 12, 14, 16 and 20 E-Puck robots respectively.

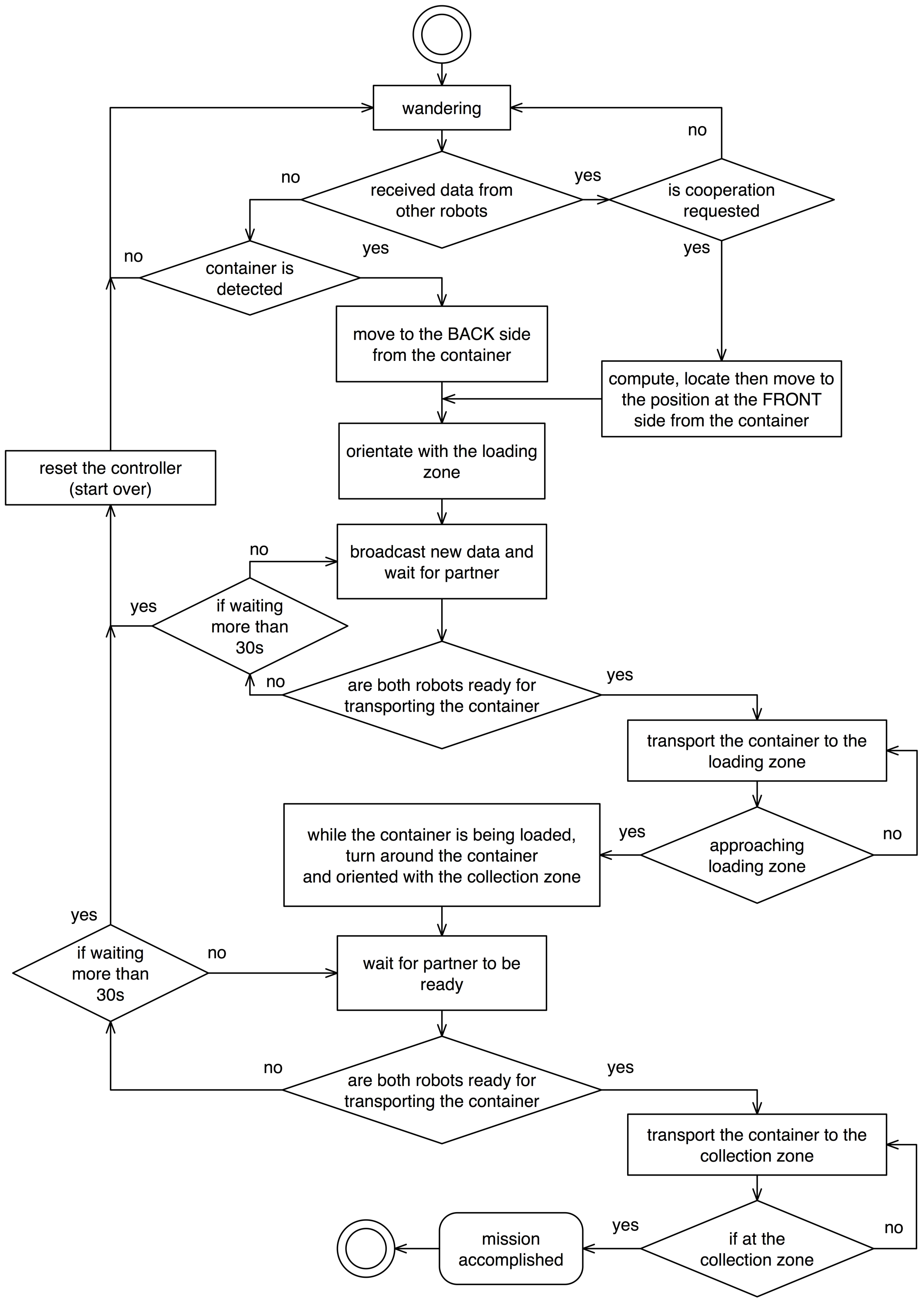

wandering
received data from other robots
no
yes
is cooperation requested
no
yes
container is detected
no
yes
move to the BACK side from the container
compute, locate then move to the position at the FRONT side from the container
orientate with the loading zone
reset the controller (start over)
broadcast new data and wait for partner
no
if waiting more than 30s
yes
no
are both robots ready for transporting the container
yes
transport the container to the loading zone
while the container is being loaded, turn around the container and oriented with the collection zone
yes
approaching loading zone
no
yes
if waiting more than 30s
no
wait for partner to be ready
no
are both robots ready for transporting the container
yes
transport the container to the collection zone
mission accomplished
yes
if at the collection zone
no

Figure 6.3: The E-Puck controller behaviour for the cooperation between two robots while cooperating to each other for transporting the container.

The evaluation will determine the possible number of robots needed to accomplish the swarm task within the least time t and distance-travelled. Although the energy-consumption is proportional to the simulated time, it will still be considered and represented in this chapter.

6.3.1 Results

The simulation experiment was repeated ten times for each swarm size. The results were collected and demonstrated as box plots in Figure 6.4, Figure 6.5 and Figure 6.6.

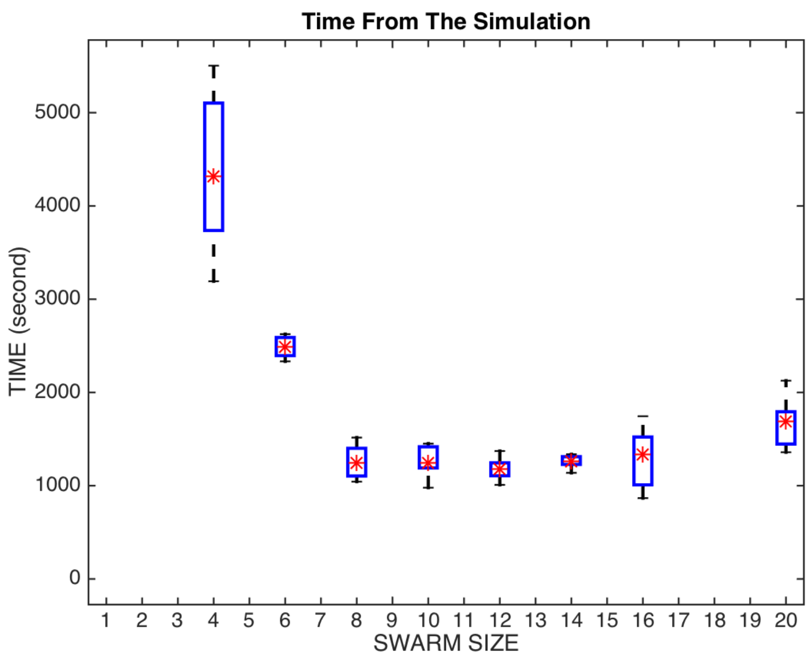
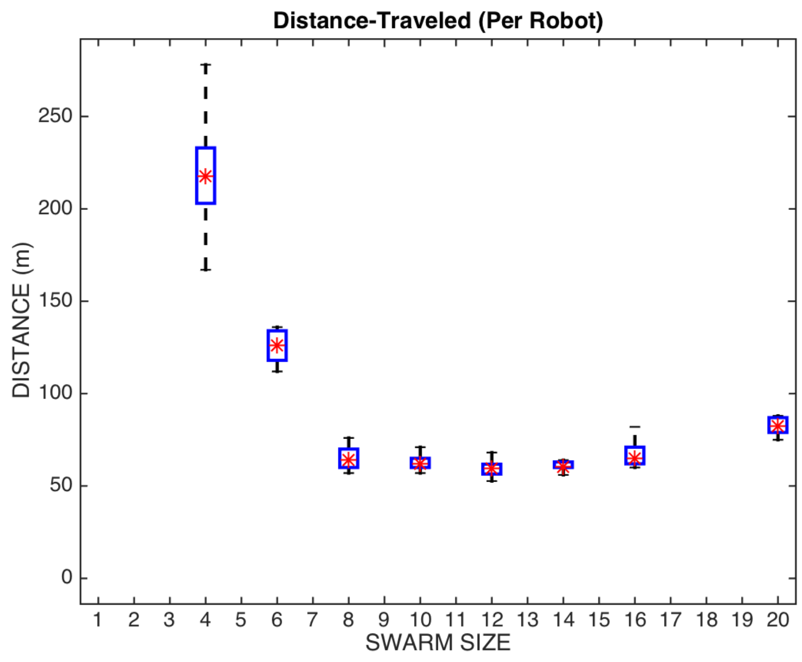


Figure 6.4: Box plots showing the time from the simulated experiment for clearing the arena compared with the swarm size. Each was repeated ten times. The box represents the skewness in the data. Stars represent the mean, the top line represents the longest running time and the bottom line represents the shortest running time during this experiment.

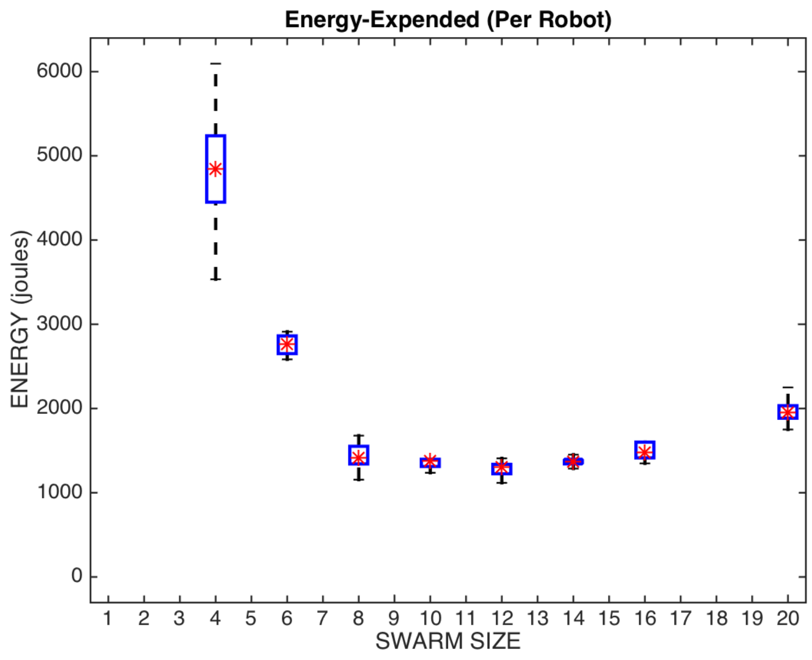
Table 6.1 summarises the results collected after running the experiment ten times for each swarm size. These results represent the average time from the simulation, the distance-travelled per robot, and energy-expanded per robot, for each swarm size, for loading and collecting containers at their specific locations in the arena.

Table 6.1: The average time, distance-travelled per robot and the energy-expended per robot for the simulated swarm sizes: 4, 6, 8, 10, 12, 14, 16 and 20 robots to load and collect containers at the specific locations on the arena.

Swarm Size	Simulation Time	Distance-Travelled	Energy-Expended
4	4314.42	217.51	4841.08
6	2486.56	126.05	2768.27
8	1241.92	64.09	1412.58
10	1247.11	61.74	1373.38
12	1174.24	59.59	1300.09
14	1259.35	60.44	1372.04
16	1336.61	64.93	1482.19
20	1686.08	82.42	1955.55

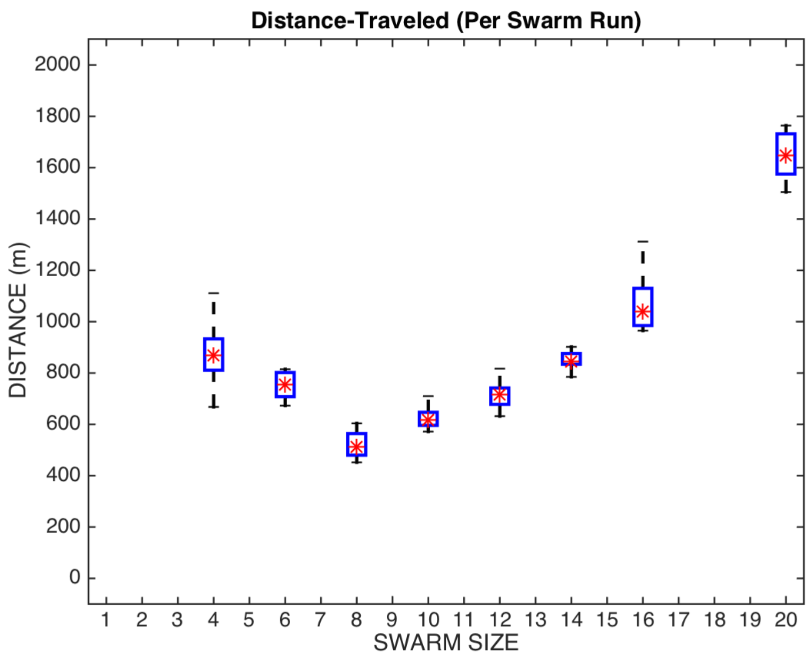


(a) The average distance-travelled per robot for loading and collected containers compared with the swarm size.

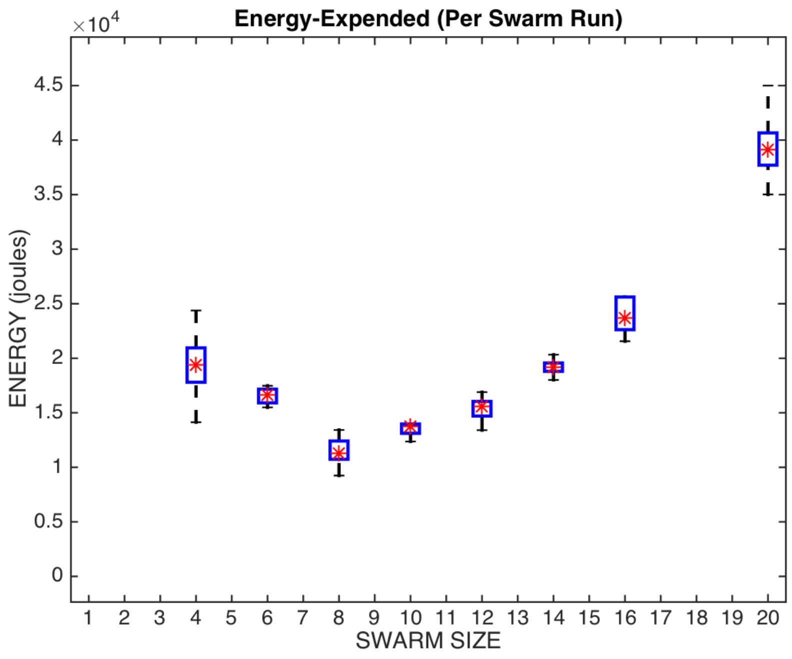


(b) The average energy-expended per robot for loading and collected the containers compared with the swarm size.

Figure 6.5: Box plots showing the comparison between the performance obtained from the simulated swarm sizes over ten runs. The box represents the skewness in the data. Stars represent the mean, the top line represents the longest running time and the bottom line represents the shortest running time during the experiment.



(a) The average distance-travelled per swarm run for loading and collected the containers compared with the swarm size.



(b) The average energy-expended per swarm run for loading and collected the containers compared with the swarm size.

Figure 6.6: Box plots showing the comparison between the performance obtained from the simulated swarm sizes over ten runs. The box represents the skewness in the data. Stars represent the mean, the top line represents the longest running time and the bottom line represents the shortest running time during the experiment.

Table 6.2 summarises the results collected after running the experiment ten times for each swarm size. These results represent the average time for the simulation, the distance-travelled per swarm run, and the energy-expanded per swarm run, for each swarm size, to load and collect containers at their specific zones in the arena.

Table 6.2: The average time, distance-travelled per swarm run and the energy-expended per swarm run for the simulated swarm sizes: 4, 6, 8, 10, 12, 14, 16 and 20 robots to load and collect containers at the specific zones on the arena.

Swarm Size	Simulation Time	Distance-Travelled	Energy-Expended
4	4314.42	870.02	19364.33
6	2486.56	756.33	16609.62
8	1241.92	512.76	11300.66
10	1247.11	617.42	13733.83
12	1174.24	715.03	15601.06
14	1259.35	846.12	19208.61
16	1336.61	1038.86	23714.99
20	1686.08	1648.47	39110.98

6.3.2 Discussion

Regarding the simulated swarm task, which were described in Section 6.2, individual robots require partners to achieve the simulated swarm task. The overall observation from the collected results shows that the simulated swarm robotic system performance increases whenever the swarm size increases until the performance reaches a peak with 12 robots in the simulated swarm. The performance starts to decrease again when the simulated swarm size gets larger.

The results are demonstrated in Figure 6.6(a) and Figure 6.6(b) shows that 8 robots, with the given arena and swarm task specification, are enough to achieve the simulated swarm task for transporting containers to the loading and collecting locations. By considering the measurement of efficiency and performance of the simulated swarm, 8 robots show a great efficiency in distance-travelled and energy-expended. That means 8 robots could do the simulated swarm task in less simulation time, less distance-travelled and energy-expended. But, for the sake of evaluating the exogenous failure detection model in the forthcoming experiments, it is necessarily to have a large sample of simulated swarm robotics. That is due to the need for operational robots while there are multiple failed ones. Therefore, Figure 6.5, Figure 6.5(a) and Figure 6.5(b) show the simulated swarm performance for each swarm size. The time from the simulation, the distance-travelled and the energy-expended shows great overall performance when the swarm size is increasing. However, when the swarm size reaches 8 robots, the performance almost remains constant until it reaches 20 robots. The performance of the distance-travelled and the energy-expended per robot (Figure 6.5(a) and Figure 6.5(b)) are proportional to the time for the simulation (Figure 6.4). Although 12 robots can finish the simulated swarm task in less time, the performance with 12 robots remains acceptable with a time efficiency simulation.

The reason for choosing 12 robots instead of 8 robots is lie on two factors. Firstly, as was defined by Şahin (2005) in Section 1.1.2, a swarm robotics is consist of a large number of individuals and with local interaction among them. This was one reason for choosing the largest of the best size. In addition, the experiment in this Chapter consists of detecting and recovering failures of individual robots, therefor the shortest time was noted for individuals to finish the swarm task was with 12 robots. By considering the distance-travelled and the energy-expended between 12 and 8 robots in Table 6.1, a swarm with 12 robots managed to accomplish the task with less distance-travelled while consuming less energy than the experiment with 8 robots.

After determining the best swarm size to use during the next experiments, it is necessary to analyse the combinations of metrics, simulation time, distance-travelled and energy-expended together, Figure 6.7.

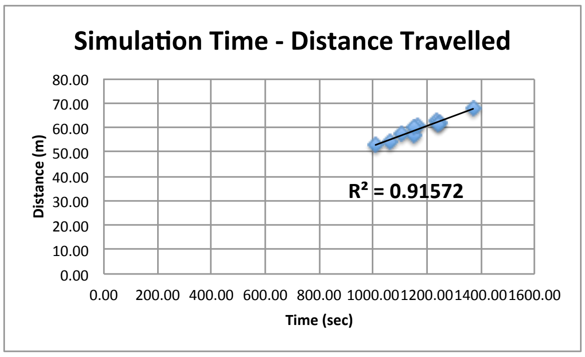
For the forthcoming experiments, including evaluating the simulated swarm with a number of faulty robots compared to a simulated swarm after detection and mitigation the failures, 12 robots will be chosen for analysing the swarm performance.

6.4 Summary

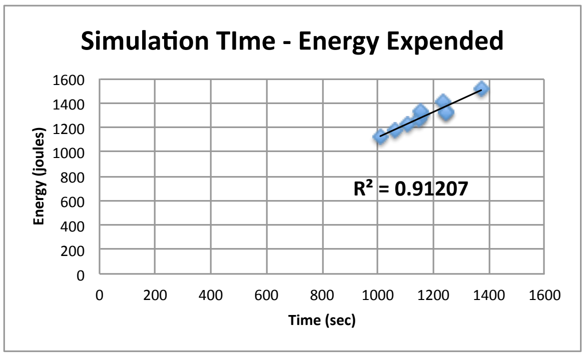
The simulation experiments in this chapter have taken part in a larger sized arena, compared to the arena size in the previous chapter. In addition, the task became more complex as individual robots were required to cooperate in pairs in order to achieve the swarm task.

The task used during the experiments required the pair of robots to transport the container first to a loading zone then to the collection zone.

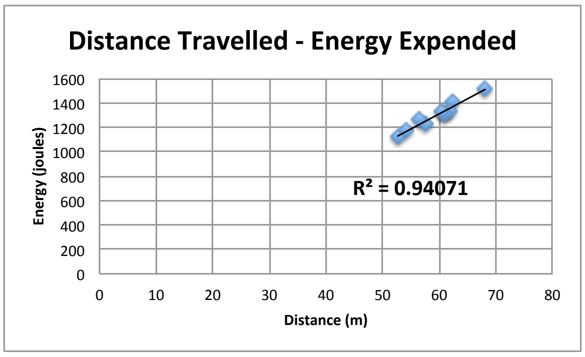
Based on the statistical analysis and the performance, 12 robots was the right swarm size are going to be undertaken for the forthcoming experiments when there will be failures injected to multiple robots while they are wandering and/or achieving the swarm task.



(a) The compatibility of time and distance-travelled with 12 robots to load then collect all containers in the arena.



(b) The compatibility of time and energy-expended with 12 robots to load then collect all containers in the arena.



(c) The compatibility of distance-travelled and energy-expended with 12 robots to load then collect all containers in the arena.

Figure 6.7: The correlation between the time, distance-travelled and energy-expended for a swarm size of 12 robots.

Chapter 7

Failures Impact on the Swarm Loading and Collecting Task

Result from the simulation experiment in Chapter 6 will be used as a base measurement to be compared and evaluated with the executed results after injecting the simulated swarm with failures. This will be used to investigate the effectiveness of the proposed exogenous fault detection model after applying it to the following experiments in this chapter.

Failures will be injected to multiple robots at the hardware level, precisely the failure will affect the robot's navigation. The failure in the navigation device will causes error reading of the robot's positions and so computational process such as distance-travelled. Also, failed robots will be shutdown after they are confirmed having the failure. Therefore, the overall energy-expended of the simulation swarm will be affected.

As part of evaluating the efficiency of the exogenous fault detection model and the mitigation procedure, it is important to observe the behaviour of swarm robotics after encountering failures in individuals while individual robots are handling subtasks. The objective of this case study is to determine whether the proposed exogenous fault detection model can still detect failures even with the existence of complex swarm tasks.

7.1 Simulation Experiment

The same configuration of the simulation experiment was set in Chapter 6 will be used for experiments in this chapter.

Additionally, the exogenous failure detection and the mitigation procedure will be applied, for handling failures in the simulation swarm robotics experiments.

The arena has a loading location and a collecting location. The simulated swarm task is to transport 8 containers, per pair of robots, first to the loading location then to the collecting location. The collecting location will hold all the containers after been loaded and transported, Figure 3.6.

The controller of the E-Puck robot, in this case study, was designed to consider the cooperation between two robots, Figure 6.1. When the first robot detects the container, it starts to position itself at the rear side from the container and facing the loading location. At this point, the robot waits for assistance from the nearest partner, within the communication range, in order to transport the container. When the second robot detects the container, or when it receives an assistance message indicating the location of the container, the second robot positions itself at the front side of the container and facing the loading location. At this moment, both robots inform each other that they are ready to transport the container, then start moving together.

The mechanism of the cooperation will allow two robots to perform as one independent robot by partially sharing the ability of some of their resources. For instance, the rear robot, from the container, would be fully aware of the front robot sensing, especially when the front robot encounters an obstacle, so that they can take action to avoid this obstacle.

7.2 Task Description: Loading and Collecting Containers While Handling Failures

During the simulation experiments in this chapter, the arena will be obstacle-free, where robots will be wandering freely with no collisions with obstacles (except the collisions with each other and failed robots). All containers, in this case, will be gathered at the centre of the arena. Before the simulation run, robots are placed at the corner of the arena on the other side from the loading and collecting locations. The location of robots and containers are set to relocated back to their initial location after the simulation swarm task was accomplished for each swarm run.

The data flow diagram in Figure 6.3 presents the behaviour of two robots cooperating together to transport the container to the loading location, then to the collecting location.

7.3 The Impact Failures During the Swarm Complex Task

In the next simulation experiment, failures were injected at random times after the running of the simulation. The effect of the randomness in time could impact the robots at any time, either while the robot is wandering and searching for the container, or during the robot task (when the robot is cooperating with another robot and both are transporting the container to the loading and collecting zones).

Failures in the simulated swarm also affects either a total of four or six robots, respectively, of the 12 robots. These faulty robots have been randomly chosen by the simulation supervisor-controller. The supervisor-controller in Webots is used to monitor and observe the entire simulated swarm behaviour, including the robots and containers positions. However, each time the simulation run, the failure affects different robots at different times. For each simulation run, the supervisor-controller randomly selects four and six robots out of the total number of robots in the arena, then the supervisor assigns a random time, between the starting time of the simulation to the average time been collected from Section 6.3 with 12 operational robots.

7.3.1 Results

The collected results from the simulated swarm robotic experiment with four failures out of 12 robots are represented in Figure 7.1. These collected results are compared against the collected results from the previous simulation experiment, in Chapter 6, with 12 and 8 operational robots.

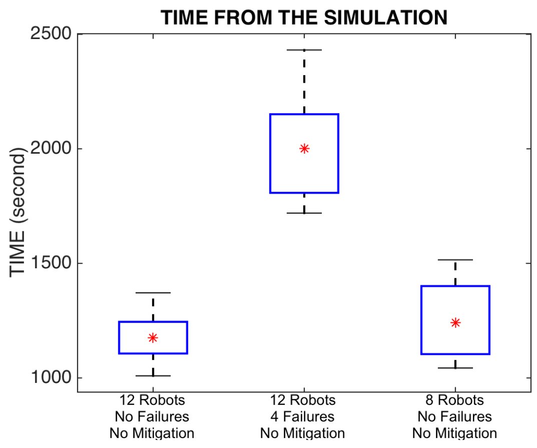
Table 7.1 summarises the results of the time from the simulation, the distance-travelled and the energy-expended for loading and collecting containers on the arena with four failures out of 12 robots.

Table 7.1: Average time, distance-travelled and energy-expanded for ten runs before and after the simulated swarm was injected with four failures. In addition to a comparison to a simulated swarm with 8 operational robots.

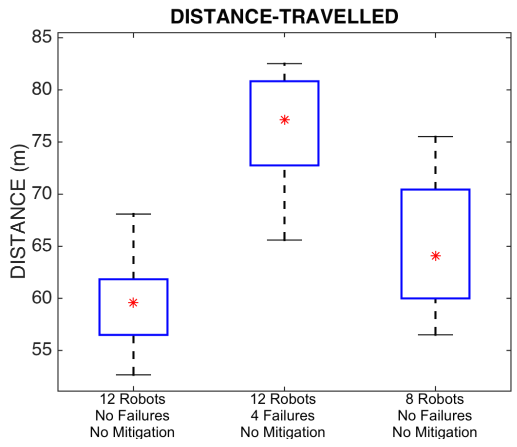
Metric	12 Robots No Failures	12 Robots 4 Failures	8 Robots No Failures
Time from simulation (Seconds)	1174.24	2002.30	1241.92
Distance-Travelled (Meter)	59.59	77.15	64.09
Energy-Expended (Joules)	1300.09	2227.34	1412.58

Figure 7.2 shows the collected results after increasing the number of faulty robots to become six failures, which represents the effect on the behaviour of the simulated swarm with half number of operational robots.

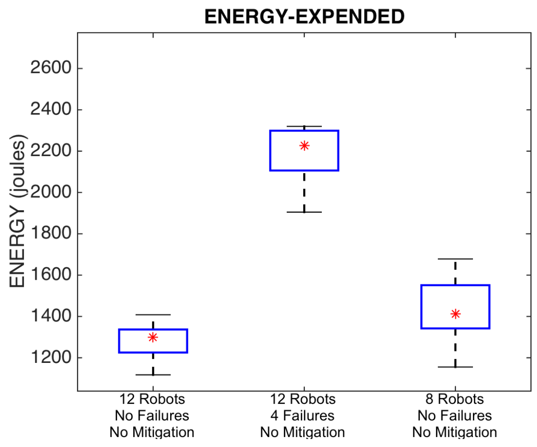
A failure of half the number of simulated swarm robots even shows how the overall performance is even getting worse than the simulation swarm with four failures. Results



(a) The impact of four faulty robots on the simulation time to load and collect the containers.

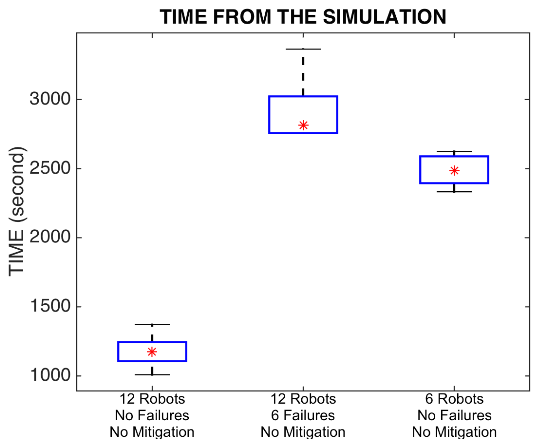


(b) The impact of four faulty robots on the simulation swarm distance-traveled for loading and collecting the containers.

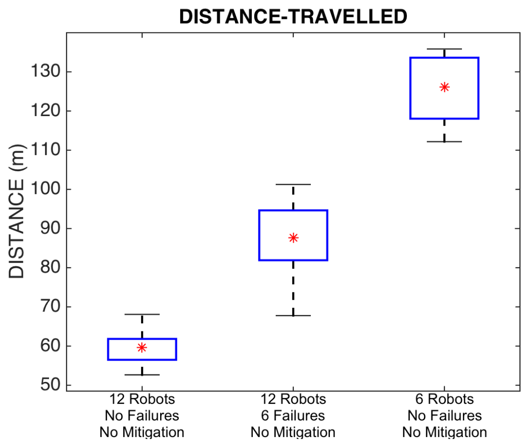


(c) The impact of four faulty robots on the simulation swarm energy-expanded for loading and collecting the containers.

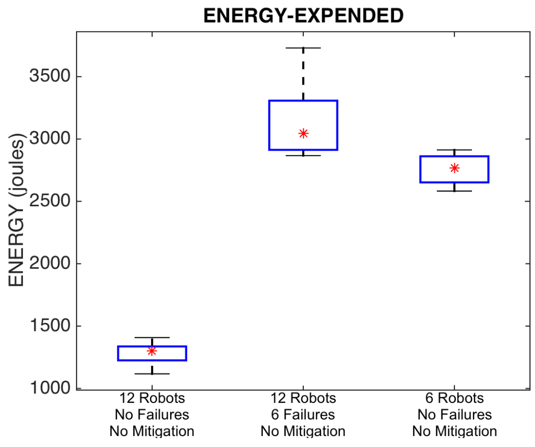
Figure 7.1: Box plots showing a comparison of the performance of the simulated 12 swarm robots before and after injection of failures in four robots. The data in these diagrams correspond to the simulation time, distance-travelled and energy-expended.



(a) The impact of six faulty robots on the simulation time to load and collect the containers.



(b) The impact of six faulty robots on the simulation swarm distance-traveled for loading and collecting the containers.



(c) The impact of six faulty robots on the simulation swarm energy-expanded for loading and collecting the containers.

Figure 7.2: Box plots showing a comparison of the performance of the simulated 12 swarm robots before and after injection of failures in six robots. The data in these diagrams correspond to the simulation time, distance-travelled and energy-expended.

collected from the time from the simulation, the distance-travelled and the energy-expended for loading and collecting containers on the arena, with six failures out of 12 robots, are summarised in Table 7.2.

Table 7.2: Average time, distance-travelled and energy-expanded for ten runs before and after the simulated swarm was injected with six failures. In addition to a comparison to a simulated swarm with six operational robots.

Metric	12 Robots No Failures	12 Robots 6 Failures	6 Robots No Failures
Time from simulation (Seconds)	1174.24	2814.19	2486.56
Distance-Travelled (Meter)	59.59	87.61	126.05
Energy-Expended (Joules)	1300.09	3043.13	2768.27

7.3.2 Discussion

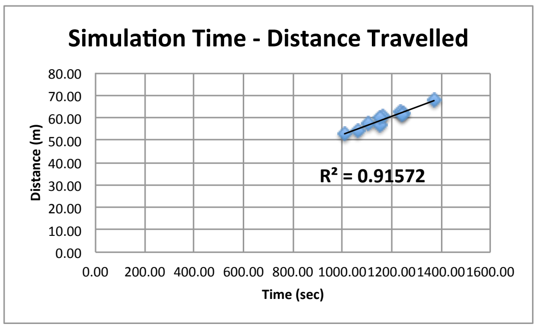
After collecting results, with four and six failures out of 12 robots, failures have a great impact on the time from the simulation, the distance-travelled and the energy expended for the robots to load and collect all containers.

The failure in the simulation has been randomly injected into robots at random time after the start of the simulation. The failure effect caused the robots to terminate immediately. At this stage, the operational robots struggled to continue finishing the simulation swarm task while avoiding the faulty robots that are stationary in their way.

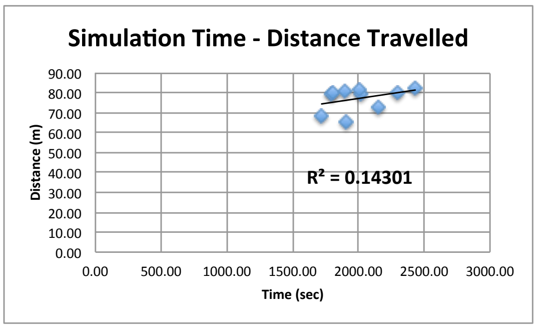
With the presence of four failures out of 12 robots, the behaviour of the simulation time, the distance-travelled and the energy-expended, for loading and collecting the containers, has jumped more than the behaviour was noted in the experiment with 12 fully operational robots, and so as with 8 operational robots (Figure 7.1).

Failures in the simulated swarm show a major impact to the simulation time and the energy-expended (Figure 7.2(a) and Figure 7.2(c)) with six faulty robots out of 12 robots. On the other hand, the distance-travelled linearly increased, compared with 12 and six robots. That is due to the period of time before the impact of failures in the robots, where the robots showed participation toward collecting a few containers before they failed.

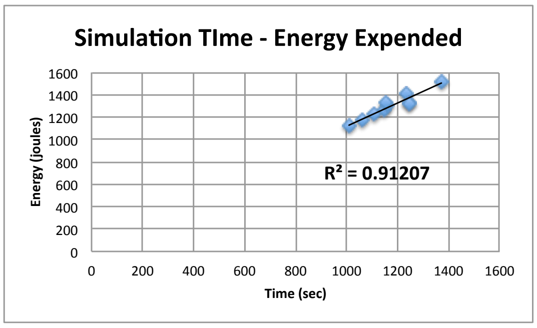
Still the behaviour of the simulated swarm with the existence of failures devastated the overall performance, Figure 7.3 and Figure 7.4. Therefore, the need for mitigating the failures could enhance the performance. The next experiment evaluates the efficiency of the proposed exogenous failure detection model.



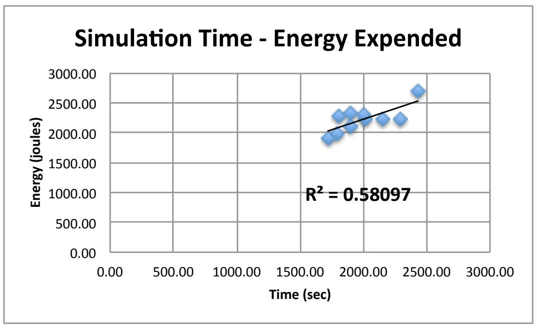
(a) The compatibility of time and distance-travelled with 12 robots to load then collect all containers in the arena.



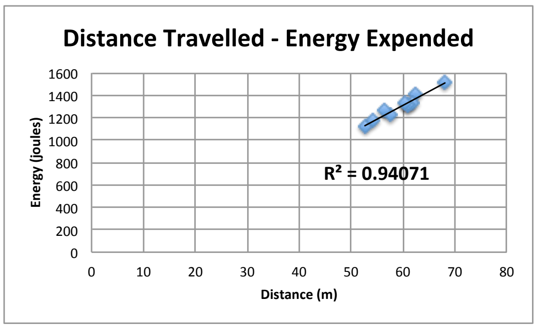
(d) The compatibility of time and distance-travelled with 12 robots, of which four are failed, to load then collect all containers.



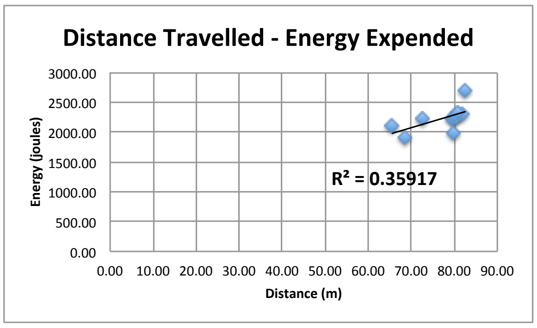
(b) The compatibility of time and energy-expended with 12 robots to load then collect all containers in the arena.



(e) The compatibility of time and energy-expended with 12 robots, of which four are failed, to load then collect all containers.



(c) The compatibility of distance-travelled and energy-expended with 12 robots to load then collect all containers in the arena.

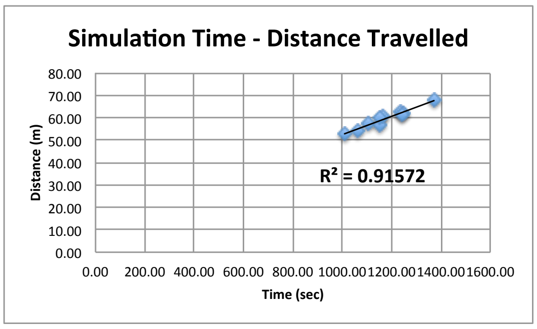


(f) The compatibility of distance-travelled and energy-expended with 12 robots, of which four are failed, to load then collect all containers.

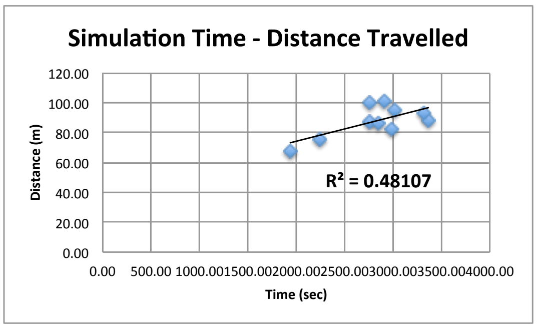
Figure 7.3: The correlation between the time, distance-travelled and energy-expended for a swarm size of 12 robots. Diagrams (a), (b) and (c) represents a swarm with 12 operational robots having no failures. Diagrams (d), (e) and (f) represents a swarm with 12 robots of which four have failures.

7.4 The Efficiency of Detecting and Mitigating Failures During the Swarm Complex Tasks

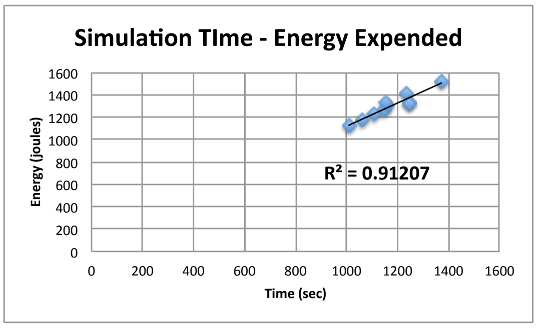
The proposed exogenous fault detection model, which was described in Section 5.1, shows a significant effective for detecting the faulty robots in shot time. In the previous simulated experiments, Section 5.3, the failure was injected into the E-Puck robots



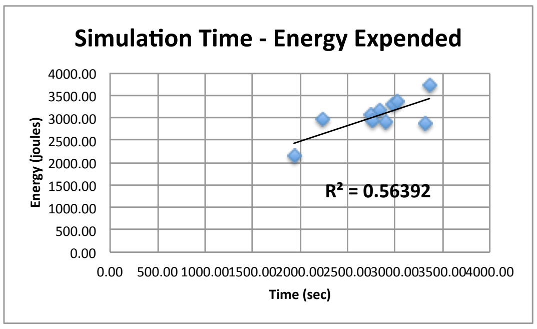
(a) The compatibility of time and distance-travelled with 12 robots to load then collect all containers in the arena.



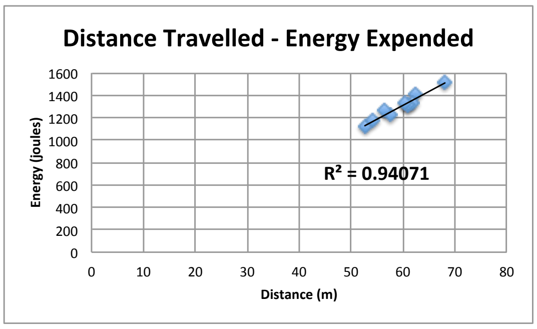
(d) The compatibility of time and distance-travelled with 12 robots, of which six are failed, to load then collect all containers.



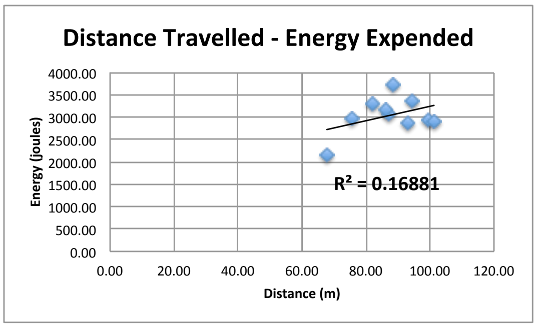
(b) The compatibility of time and energy-expended with 12 robots to load then collect all containers in the arena.



(e) The compatibility of time and energy-expended with 12 robots, of which six are failed, to load then collect all containers.



(c) The compatibility of distance-travelled and energy-expended with 12 robots to load then collect all containers in the arena.



(f) The compatibility of distance-travelled and energy-expended with 12 robots, of which six are failed, to load then collect all containers.

Figure 7.4: The correlation between the time, distance-travelled and energy-expended for a swarm size of 12 robots. Diagrams (a), (b) and (c) represents a swarm with 12 operational robots having no failures. Diagrams (d), (e) and (f) represents a swarm with 12 robots of which six have failures.

during the initialisation of the simulation run. This means that those faulty robots did not have enough time for wandering in the arena, therefore to participate with the rest of the operational simulated swarm robots.

The complexity in the next simulated experiment will lie in the steps required to overtake the swarm robotics stagnation.

Observing the efficiency for detecting failures during complex tasks requires understanding the behaviour of the controller of the E-Puck robot. Additional to the data flow diagram that was reprinted in Figure **??**, the following data flow diagram (Figure 7.5) presents how operational robots react towards those with failures. It shows a typical controller behaviour, for the loading and collecting task, after adding the exogenous failure detection sections. During every controller cycle, the robot's controller diagnose all I/O data that are shared between cooperating robots and with other wandering robots within the communication range. So that, during each control cycle, robots are required to analyse other robots' operations while they are achieving their own operation without affecting the overall task.

The diagnosis process in the exogenous failure detection sections (Figure 7.5) is detailed is Figure 7.6. The diagnosis procedure works for detecting up-normal behaviour in E-Puck robot. If the robot is suspicious, then, it has to be diagnosis by another robot within the communication range. The diagnosis work as computing the distance between the two robot through the exchanged data, including the coordinates X and Z and the signal strength. Based on these collected data, both robots diagnosis each other then broadcast the statues of each other to the swarm. If both robots are suspicious in each other, then a third robot can confirm the one with failure.

During the simulation, as part of the mitigation procedure, operational robots push faulty ones to the side of the arena, and away from the location of containers, after they detect the failure.

7.4.1 Results

The collected results shows effectiveness after implementing the exogenous failure detection feature in the simulated swarm robotics.

The failures in all the ten repeated experiment runs were detected successfully by all the operational robots. Figure 7.7 shows the performance of the simulated swarm with four faulty robots before and after been mitigated in a swarm with 12 robots. These data are compared against the collected results of the simulated swarm with 8 and 12 operational robots. Table 7.3 summarises the results of the time from the simulation the distance-travelled and the energy-expended for loading and collecting the containers on the arena with four recovered failures out of 12 robots.

7.4.2 Discussion

It is clearly obvious that the exogenous failure detection and the mitigation procedure have a noticeable impact on the results between a swarm with faulty robots, that impedes other operational robots, and after mitigating these failures.

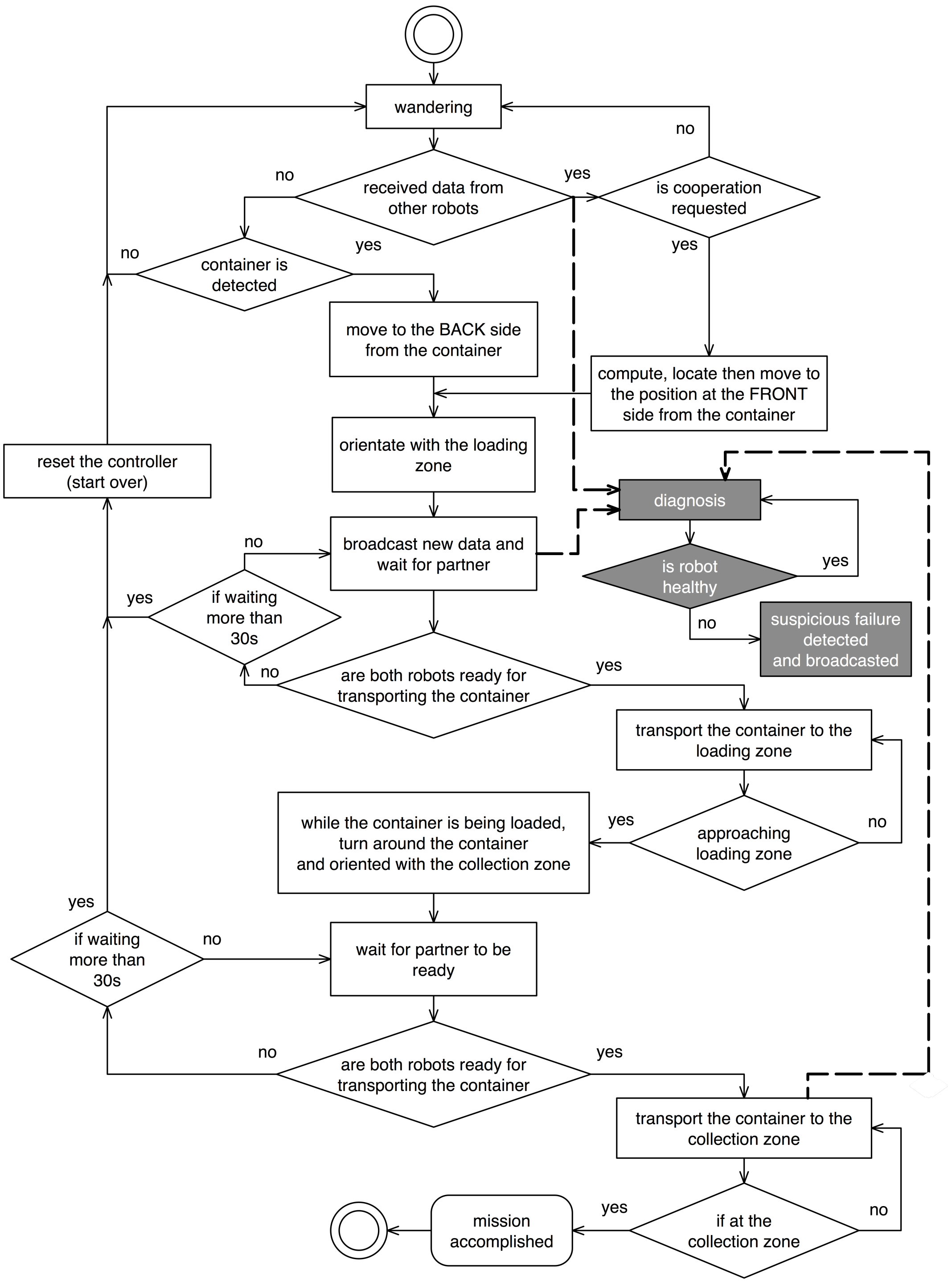

wandering
received data from other robots
no
yes
is cooperation requested
no
yes
container is detected
no
yes
move to the BACK side from the container
compute, locate then move to the position at the FRONT side from the container
orientate with the loading zone
reset the controller (start over)
diagnosis
broadcast new data and wait for partner
no
is robot healthy
yes
no
suspicious failure detected and broadcasted
if waiting more than 30s
yes
are both robots ready for transporting the container
no
yes
transport the container to the loading zone
while the container is being loaded, turn around the container and oriented with the collection zone
approaching loading zone
yes
no
wait for partner to be ready
if waiting more than 30s
yes
no
are both robots ready for transporting the container
no
yes
transport the container to the collection zone
mission accomplished
if at the collection zone
yes
no

Figure 7.5: The E-Puck controller behaviour for the cooperation between robots while Exogenous Failure Detection is applied. The Exogenous Failure Detection sections are highlighted in grey colour.

The mitigation procedure even shows the enhancement in the distance-travelled (Figure 7.7(b)) with four failures out of 12 robots. Also Figure 7.8, shows how a simulation

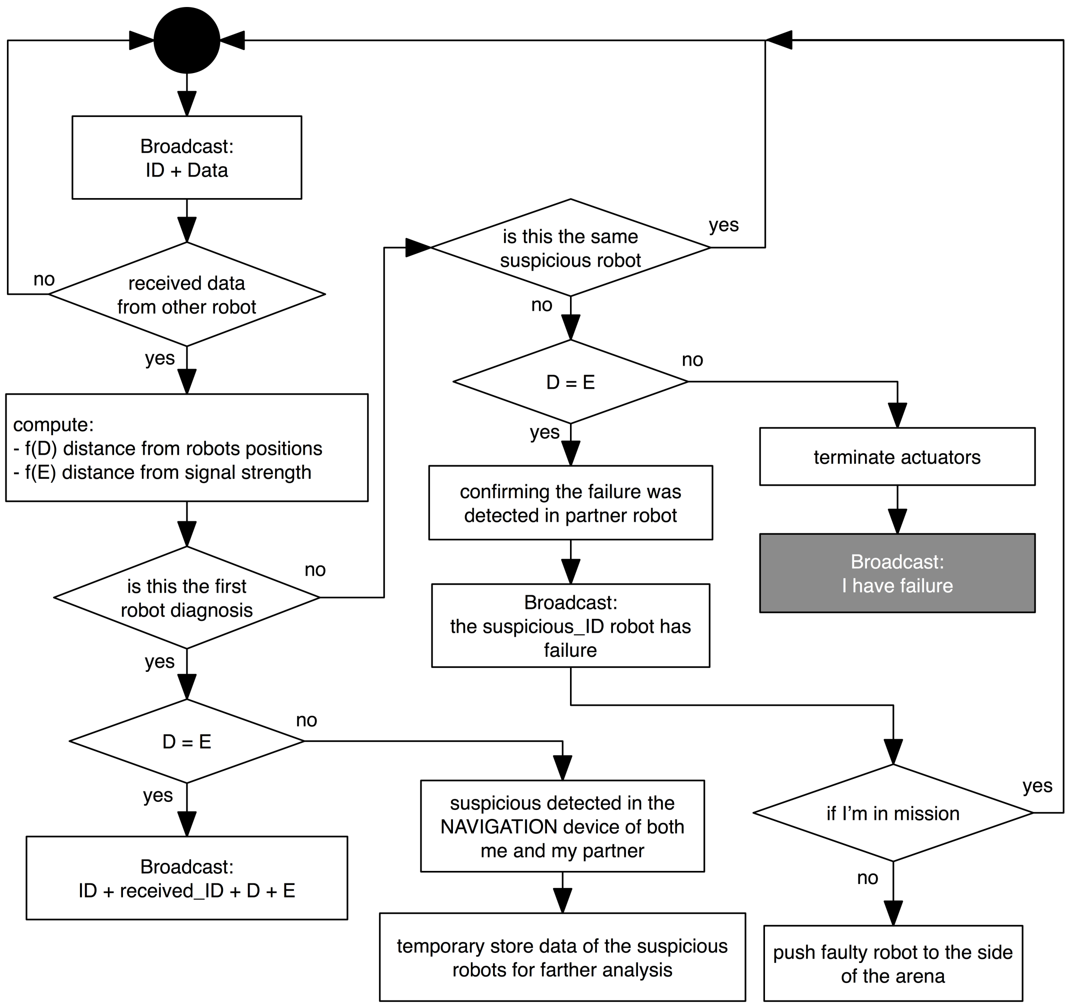

Broadcast: ID + Data
received data from other robot
no
yes
compute:
- f(D) distance from robots positions
- f(E) distance from signal strength
is this the first robot diagnosis
no
yes
D = E
no
yes
Broadcast: ID + received_ID + D + E
is this the same suspicious robot
yes
no
D = E
no
yes
confirming the failure was detected in partner robot
Broadcast: the suspicious_ID robot has failure
suspicious detected in the NAVIGATION device of both me and my partner
temporary store data of the suspicious robots for farther analysis
terminate actuators
Broadcast: I have failure
if I'm in mission
yes
no
push faulty robot to the side of the arena

Figure 7.6: The E-Puck controller behaviour for the cooperation between robots while the specific failure diagnosis part from the exogenous failure detection model. The symbol D represents the computed distance from the position of two E-Puck robots. The symbol E represents the computed distance from the signal strength of the receiver E-Puck robot. The black circle refer back to the rest of the E-Puck controller including the robot task.

swarm robotics with six mitigated failures has improved the simulation time, distance-travelled and the energy-expended performance to become even better than a simulated swarm with six operational robots. That is due to the faulty robots had achieved part of the swarm task before they become faulty. In addition to that, the mitigation played a role to save time for the remaining operational robots to finish the task with free obstacles (after the faulty robots are terminated and were pushed away to the side of the arena).

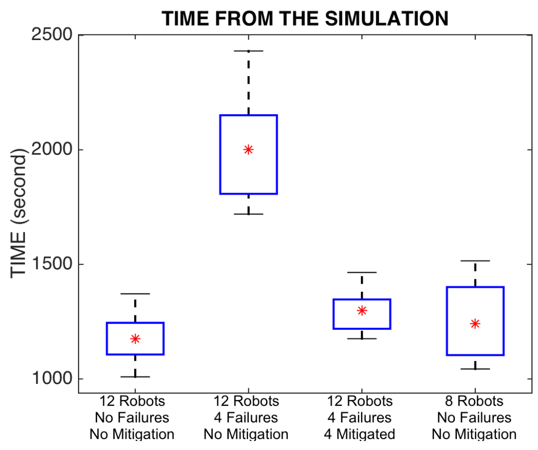
The exogenous failure detection model and the mitigation procedure has shown how the impact of failures can make a difference to the simulated swarm robotics performance. Therefore, the existence of such a model could enhance the performance.

Table 7.3: Average time, distance-travelled and energy-expanded for ten runs before and after the simulated swarm was injected with six failures and then mitigated. In addition to a comparison to a simulated swarm with six operational robots.

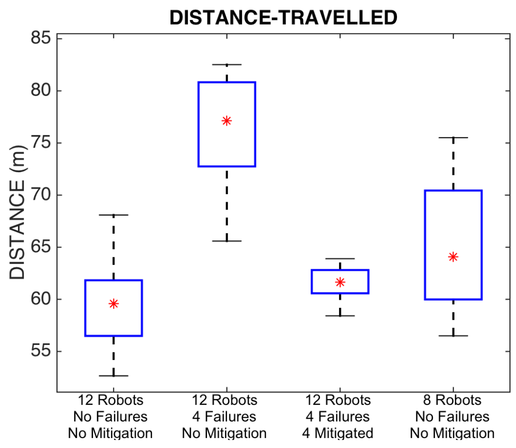
Metric	12 Robots No Failures No Mitigation	12 Robots 4 Failures No Mitigation	12 Robots 4 Failures 4 Mitigated	8 Robots No Failures No Mitigation
Time from simulation (Seconds)	1174.24	2002.30	1299.15	1241.92
Distance-Travelled (Meter)	59.59	77.15	61.65	64.09
Energy-Expended (Joules)	1300.09	2227.34	1676.38	1412.58

Table 7.4: Average time, distance-travelled and energy-expanded for ten runs before and after the simulated swarm was injected with six failures and then mitigated. In addition to a comparison to a simulated swarm with six operational robots.

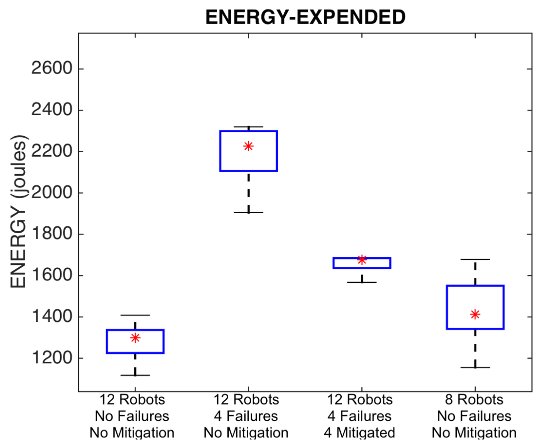
Metric	12 Robots No Failures No Mitigation	12 Robots 6 Failures No Mitigation	12 Robots 6 Failures 6 Mitigated	6 Robots No Failures No Mitigation
Time from simulation (Seconds)	1174.24	2814.19	1722.37	2486.56
Distance-Travelled (Meter)	59.59	87.61	61.39	126.05
Energy-Expended (Joules)	1300.09	3043.13	1897.23	2768.27



(a) The recovery effects of four faulty robots on the simulation time to load and collect the containers.

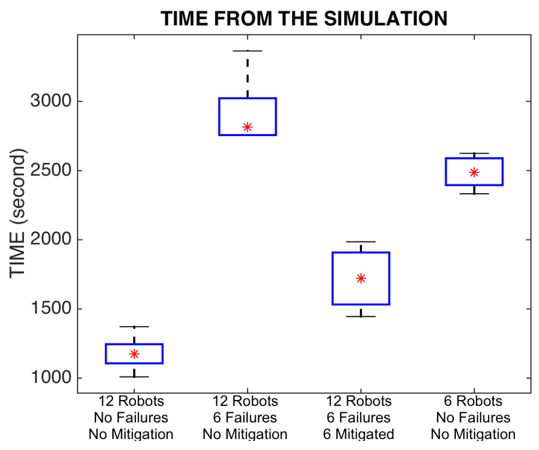


(b) The recovery effects of four faulty robots on the simulation swarm distance-traveled for loading and collecting the containers.

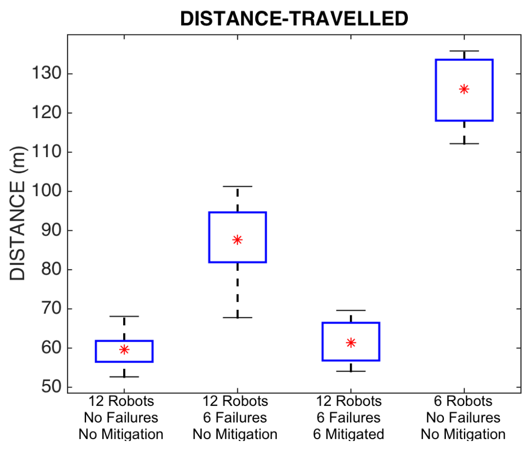


(c) The recovery effects of four faulty robots on the simulation swarm energy-expanded for loading and collecting the containers.

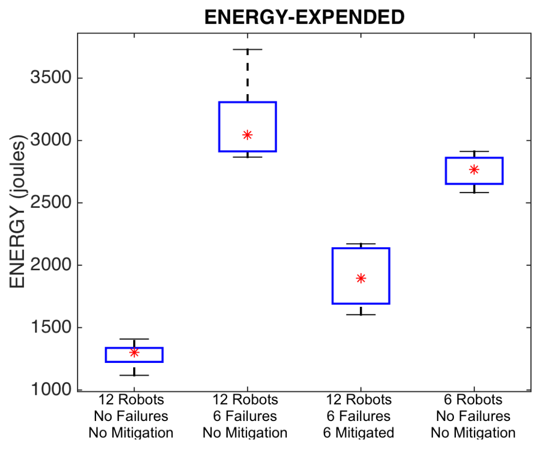
Figure 7.7: Box plots showing a comparison of the performance of the simulated 12 swarm robots before and after mitigating four faulty robots. The data in these diagrams correspond to the simulation time, distance-travelled and energy-expended.



(a) The recovery effects of six faulty robots on the simulation time to load and collect the containers.

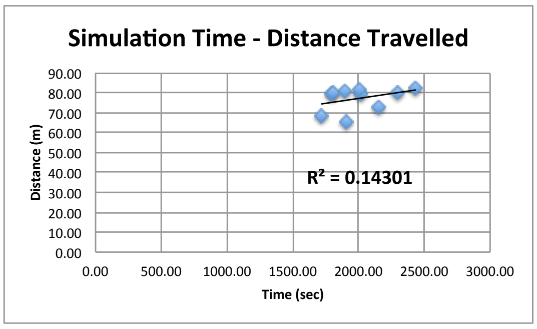


(b) The recovery effects of six faulty robots on the simulation swarm distance-traveled for loading and collecting the containers.

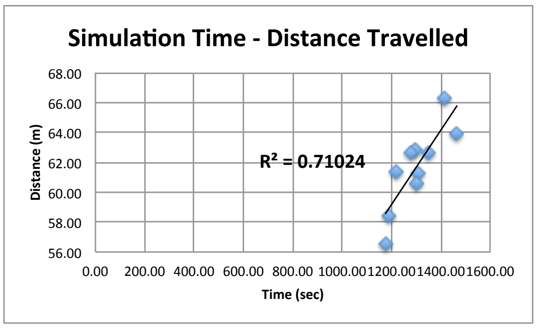


(c) The recovery effects of six faulty robots on the simulation swarm energy-expanded for loading and collecting the containers.

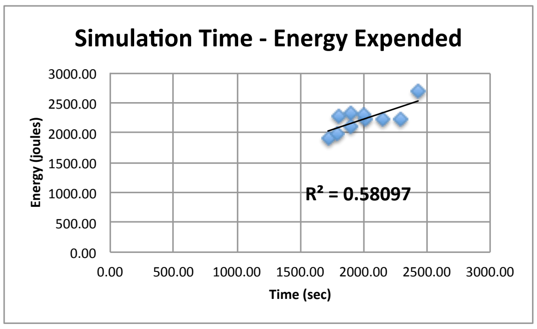
Figure 7.8: Box plots showing a comparison of the performance of the simulated twelve swarm robots before and after recovering six faulty robots. The data in these diagrams correspond to the simulation time, distance-travelled and energy-expended.



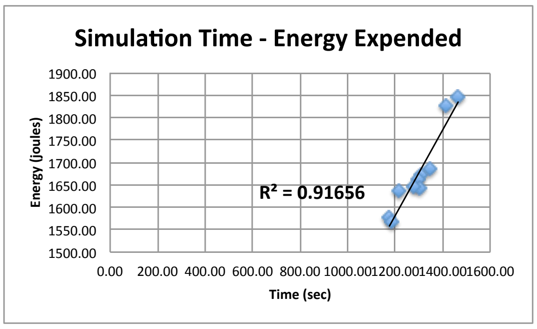
(a) The compatibility of time and distance-travelled with 12 robots, of which four are failed, to load then collect all containers.



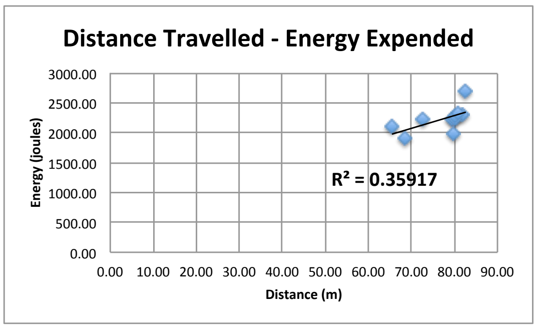
(d) The compatibility of time and distance-travelled with 12 robots, of which four failures are mitigated, to load then collect all containers.



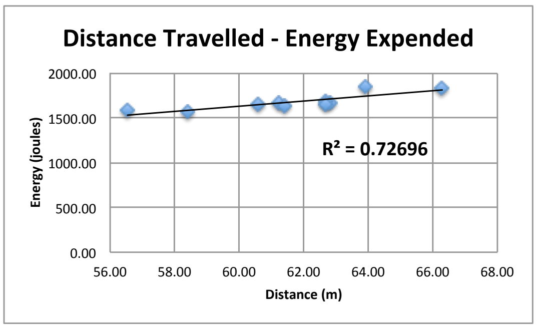
(b) The compatibility of time and energy-expended with 12 robots, of which four are failed, to load then collect all containers.



(e) The compatibility of time and energy-expended with 12 robots, of which four failures are mitigated, to load then collect all containers.

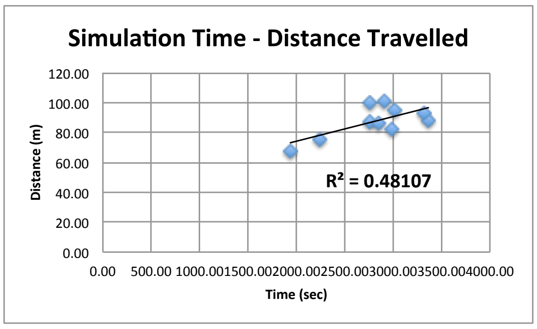


(c) The compatibility of distance-travelled and energy-expended with 12 robots, of which four are failed, to load then collect all containers.

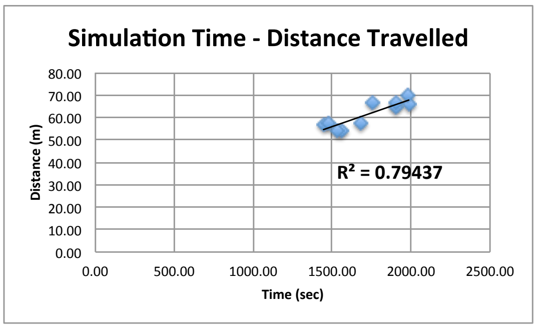


(f) The compatibility of distance-travelled and energy-expended with 12 robots, of which four failures are mitigated, to load then collect all containers.

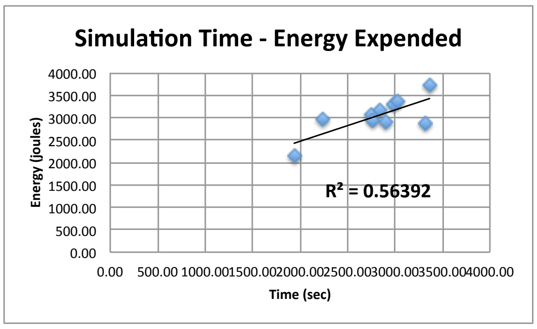
Figure 7.9: The correlation between the time, distance-travelled and energy-expended for a swarm size of 12 robots. Diagrams (a), (b) and (c) represents a swarm with 12 robots of which four have failures. Diagrams (d), (e) and (f) represents a swarm with 12 robots of which four failures are mitigated.



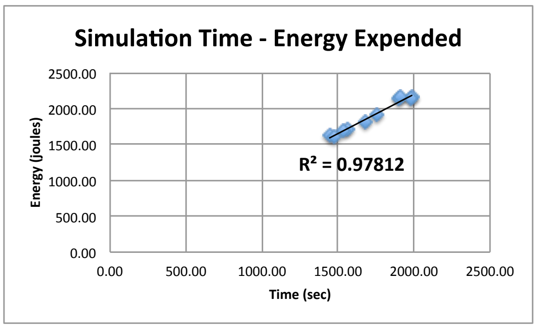
(a) The compatibility of time and distance-travelled with 12 robots, of which six are failed, to load then collect all containers.



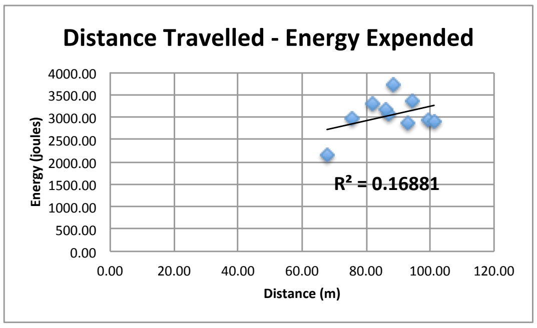
(d) The compatibility of time and distance-travelled with 12 robots, of which six failures are mitigated, to load then collect all containers.



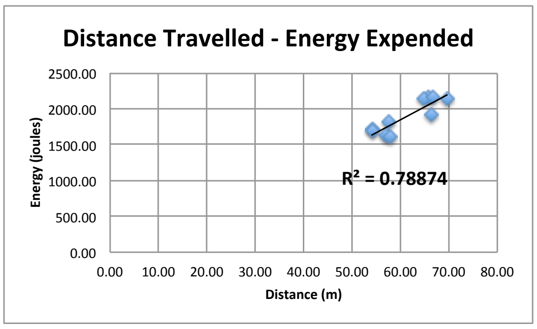
(b) The compatibility of time and energy-expended with 12 robots, of which six are failed, to load then collect all containers.



(e) The compatibility of time and energy-expended with 12 robots, of which six failures are mitigated, to load then collect all containers.



(c) The compatibility of distance-travelled and energy-expended with 12 robots, of which six are failed, to load then collect all containers.



(f) The compatibility of distance-travelled and energy-expended with 12 robots, of which six failures are mitigated, to load then collect all containers.

Figure 7.10: The correlation between the time, distance-travelled and energy-expended for a swarm size of 12 robots. Diagrams (a), (b) and (c) represents a swarm with 12 robots of which six have failures. Diagrams (d), (e) and (f) represents a swarm with 12 robots of which six failures are mitigated.

7.5 Summary

The simulated swarm task for loading and collecting containers was identified complex, as individual robots were required to cooperate in pairs in order to achieve the swarm task.

The task used during the experiments required the pair of robots to transport the container first to a loading location then to the collection location. While robots are on their task, some individual robots may encounter a failure. The failure prevents the robots from moving. This could cause operational robots to travel further and for longer time in order to avoid faulty robots (i.e., obstacles).

The exogenous failure detection was implemented on the simulated swarm robotics with a number of failures: four and six failures out of 12 robots. The results show how the simulated swarm performance has been enhanced after mitigating these faulty robots by terminating failures and pushing them to the side of the arena.

Chapter 8

Reusing the Faulty Robots as Communication Bridges

Failures in the swarm robotics could occur at any time from the beginning of the task. The collected results from experiments in Chapters 5 and 7 does not include the remaining energy was left in the failed robots. The mitigation procedure used for handling failures was by shutdown faulty robots then push them away to the side of the arena. The mitigation procedure helped to clear the arena from obstacles caused from faulty robots.

A fair amount of unused resources in the faulty robots, including the communication device, could be reused for the benefits of the swarm task. Reusing the leftover resources will depend on how much energy is left in the faulty robot that can be reused to operate a certain device.

However, the suggested solution could provide a swarm robotics to reuse and count on faulty robots to become a communication bridge between operational robots. This solution could mitigate the swarm robotics and it could save the time and the distance-travelled to achieve the swarm task.

8.1 Failure Recovery and Mitigation

Understanding failure modes requires identifying all the effects that failures may encounter in swarm robotics to build up a general overview of the swarm's reliability.

The proposed exogenous failure detection model in this thesis (Section 5.1) shows an enhancement in the reliability simulated experiments. Also, the suggested solution to mitigate the faulty robots have shown an enhancement to the time and distance-travelled after shutting down and transporting faulty robots to the side of the arena, so that they do not become obstacles to the operational robots.

The mitigation for reusing the faulty robot's resources requires an observation of the required amount of energy to run the communication device while the robot is stationary.

8.1.1 The Proposed Approach for Mitigating Energy in Swarm Robotics

The main concept of failure mitigation, regarding swarm robotics throughout the previous simulation experiments, was applied to the E-Puck robot model in the Webots simulation platform.

Figure 8.1 shows a scenario of four robots communicating with each other. Assuming that two robots are faulty, robots 1 and 2 are operating as communication repeaters. From the scenario, robot 3 has detected the container and is in the waiting stage for co-operation from an operational robot to transport the container. The closest operational robot that can undertake the task is robot 0. Since both robots, 0 and 3, are not within communication range of each other, support from the communication units in the faulty robots, 1 and 2, will associate to rebroadcast a message packet to robot 3, through robot 1 then robot 2, to be received by robot 0.

The message packet from robot 3 contains useful information (including the identification of the location of robot 3). Therefore, robot 0 can travel to the location of robot 3 for assistance.

8.1.2 Observing the Required Energy for Communication Operation

The computation of the energy-expended of the E-Puck model in the Webots simulation is dependent on the consumption of the total amount of joules (with fully charged battery) and the consumption value of each device in the E-Puck model.

As part of the mitigation procedure, which was provided in previous experiments (Section 5.4 and Section 7.4), E-Puck robots have to respond to an immediate shutdown whenever the failure is detected. This means that the E-Puck robot will no longer need to fully operate, with the exception of the LED indicators (LEDs indicators in E-Puck uses for observation purposes for informing the developer of the swarm system which robot is failed) Figure 8.2. That means the robot's controller should be operating at a low level to control LEDs only.

Of course if the robot failed at the beginning of the simulation, then the robot should have almost a fully charged battery that could be of use to the operational robots in the swarm. Therefore, the robot's controller, the LEDs and the communication unit,

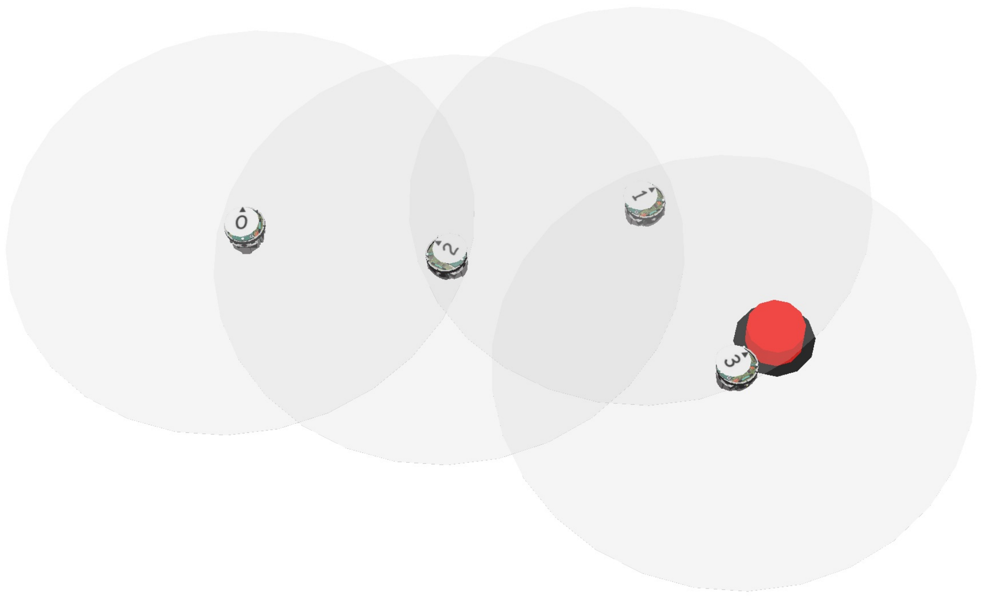


Figure 8.1: A simulated swarm robotic with four E-Puck robots, of which two robots are operational and two are failed. Robots 0 and 3 are operational and robots 1 and 2 have failed. The small dot on top of the robot number represents the front of the robot. The communication range of each robot is represented with the large grey circle around the robot. The cylinder-shaped object with the black collar represents the container.

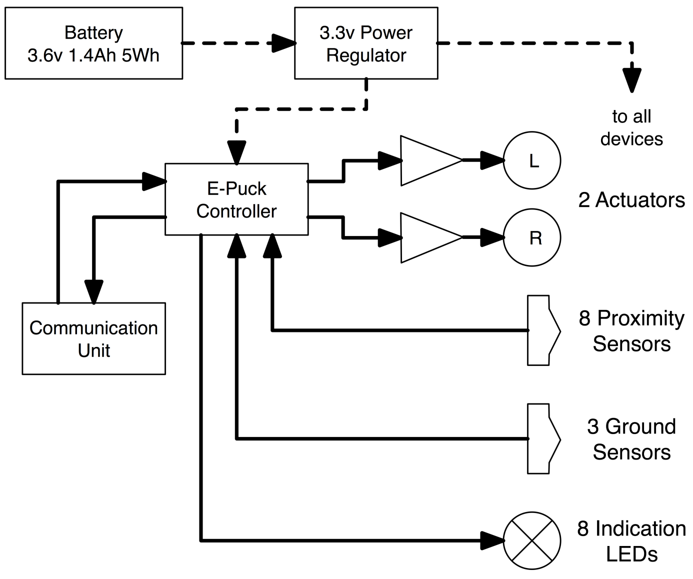


Figure 8.2: The E-Puck's electronics structure at the hardware level. Adopted from Mondada and Michael (2007); Cianci et al. (2007).

except for the actuators, can be used to guide and assist the operational robots that are faraway from each other in the arena, as was represented in Figure 8.1.

The provided information by the E-Puck developers was not enough for observing the consumption amount of the actuator motors in the E-Puck. This required a deeper investigation to understand the way an E-Puck robot consumes energy in regards to each device. The aim of this investigation is to apply the right variables to the energy-expended function of the E-Puck in the Webots simulation.

Mondada and Michael (2007); Cianci et al. (2007) provided a few data of how much each E-Puck device could consume. The power consumption data provided by the manufacturer is the following:

- The E-Puck is equipped with a 3.6v (1.4Ah) battery, 5Wh capacity, which is sufficient for about 2-3 hours of intensive use.
- When the E-Puck is not in use it draws less than 1.5W, with the radio on (ready to receive).
- When the E-Puck is in use, and the processor is under heavy load, it consumes about 3.5W.

From the provided data, it is possible to calculate the time needed for the E-Puck's controller to run while the actuator motors are inactive, which is 2.9W. This produces the capacity that an E-Puck can operate for approximately 7 minutes in addition to operating the communication unit without the actuator motors.

In addition to the previous calculation, a simple simulation experiment was performed in Webots using two E-Puck robots, where one has a failure and another one is fully operational.

The failure of the first robot started from the beginning of the simulation experiment, while the operational robot was allowed to wonder within the arena until both ran out of energy.

The results have been collected after the total swarm energy was ran out, Table 8.1. On average, it was noticed that faulty E-Puck robots hold enough energy that the robot can run for approximately 12 minutes.

Table 8.1: Reaction time from the simulation experiment of two robots, where one has a failure and the another one is fully operational. The results were collected after each individual robot ran out of energy.

Metric	Operational E-Puck	Faulty E-Puck
Time from Simulation (hours)	3.16	3.67

The remaining energy in faulty robots, can be reused by the entire swarm robots in many ways. As it was proposed by Ismail et al. (2015), this extra remaining energy can be transferred from faulty robots to operational robots during the swarm task. Sharing the remaining energy, from faulty robots to operational robots, may not be a required recovery procedure. Alternatively, the energy-expended in swarm robotics will be further observed, empirically after applying the exogenous failure detection model and the mitigation procedure.

It should be noted that the remaining energy (after shutting down the actuator motors only) is in E-Puck robots are different from one to another. That is due to the time the failure occur in the specific E-Puck. Also, this does affect the total time in the simulation.

The experiments in the next sections will show a way of recovering the remaining energy in the faulty robots in order to assist operational robots while they are stationary. Also, this will be another proposed recovery and mitigation procedure for swarm robotics.

8.2 Failure Mitigation of the Reused Energy in Swarm Robotics

The particular interest of work in this research was the proposed exogenous failure detection model and its recovery and mitigation procedures. The recovery/mitigation procedure that was applied to the previous simulation experiments is terminating and pushing the faulty robots to the side of the arena, so that they do not become obstacles to the operational robots while it continues towards accomplishing the swarm task. The results, after applying the recovery/mitigation procedure, did show an improvement in the total time and the distance-travelled of the simulated swarm robotics.

The configuration of the simulation experiment in this chapter will be slightly typical of the configuration of the previous experiments. The use of the extended energy, that is left in the faulty robots, will be taken into consideration as another form of recovery/mitigation in swarm robotics. The approach of the new recovery/mitigation procedure will be explained more in the next sections.

8.2.1 Task Description: Stationary Communication Bridge Through Faulty Robots

The arena was structured as obstacle-free, where there are no precisely placed obstacles, in the way of wandering robots, except for the interferences with the faulty robots and with each other.

A number of eight containers are placed in the centre of the arena. At the beginning of the simulation, all robots were located at the corner of the arena, on the opposite corner from the loading location, Figure 3.6.

The simulated swarm robotic task was to transport the container, with support from another robot, to the loading location, and then to the collection location in the arena.

The swarm robotic task will be considered accomplished whenever all containers were collected.

It is worthy to mention that the main purpose from this simulation experiment is the reuse of resources from faulty robot. Where the faulty robots will be extending the remaining energy to provide assistance to the operational robots to accomplish the simulated swarm robotic task. The specific approach for recovering the swarm robots, by applying a communication bridge from the faulty robots, was explained in Section 8.1.1.

8.2.2 The E-Puck Robot Controller

The robot model used for this experiment was the E-Puck. A typical E-Puck robot controller for the provided swarm robotic task was designed to achieve cooperation between individuals.

The only change in the E-Puck robot controller was produced by the faulty E-Puck robots, Figure 8.3. That is related to the special way in which E-Puck robots with failures could be useful for the operational robots.

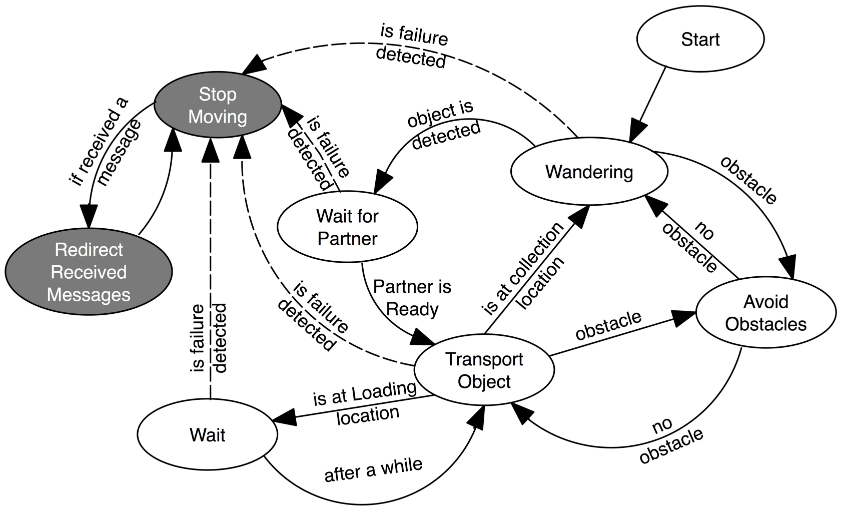


Figure 8.3: The structure of the E-Puck controller that shows how robots cooperate together in order to transport the container, in addition to the communication bridge feature for transferring broadcasted messages between operational E-Puck robots through faulty ones.

The exogenous failure detection approach was implemented in all E-Puck robot controllers. That means, when a failure is provoked in an individual, all robots within the communication range will be notified of the failure.

8.3 The Performance of the Swarm Robotics with the Communication Bridge Feature

The approach of the recovery and mitigation procedure in the previous simulation experiments was described as terminating the faulty robots so that they do not become obstacles to the operational robots. Even the distance-travelled and the energy-expended amounts have been considered until the failure occurred, along with the operational robots. That means, these two metrics, especially the energy-expended, were no longer computed as the faulty robot will no longer need to wander anymore.

The simulation experiments in this chapter were built upon the remaining energy in the faulty robots. So that, the faulty robots could be useful and not a waste to the entire simulation swarm robotic.

The results collected from the next simulation experiment were dependent on the faulty robots, but instead of shutting them down, the faulty robots would continue operating while they were stationary. The reason behind that is to allow these faulty robots to work as a communication bridge to transfer broadcasted signals from one faraway robot to another one.

8.3.1 Results

Two simulation experiments were undertaken with a different number of faulty robots, out of a total of 12 robots. Each simulation experiment was repeated for ten times.

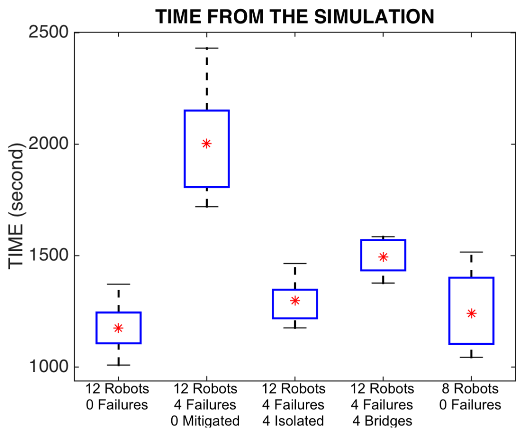
The results collected from the first simulation experiment with 12 robots, of which four robots were faulty, is demonstrated in Figure 8.4. Table 8.2 shows a comparison between a swarm with 12 operational robots, four of which are faulty, after mitigating the four faulty robots and a swarm with eight operational robots.

The second simulation experiment used a swarm with 12 robots, of which six are faulty, Figure 8.5. Also the comparison of the results is demonstrated in Table 8.3.

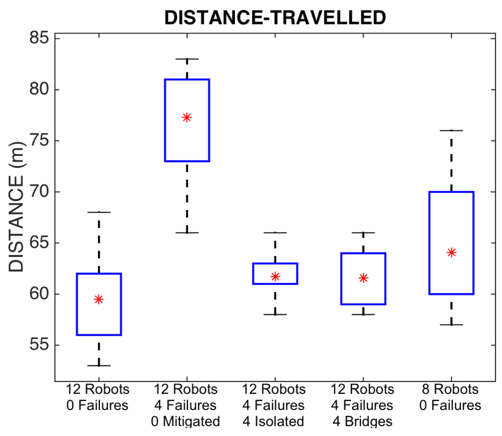
In each case the results of the mitigated swarm is compared with (i) no failures, (ii) no mitigation and (iii) after applying the mitigation were discussed in Section 7.4 and (iv) a swarm is equal in size to the number of operational robots (i.e. eight robots for the swam with four failures).

8.3.2 Discussion

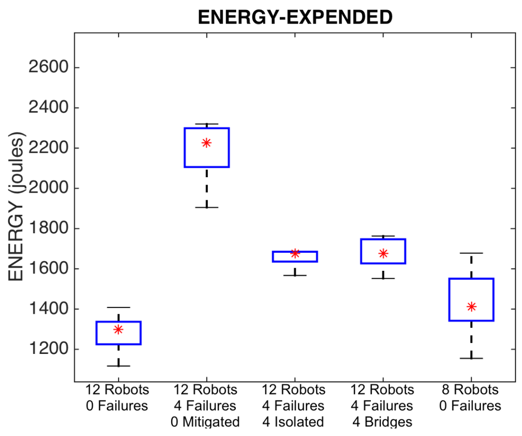
The collected results show an overall enhancement to the simulated swarm robotics after applying together both the proposed exogenous failure detection approach and the proposed mitigation.



(a) The impact of four failed robots on the simulation time to load and collect the containers.



(b) The impact of four failed robots on the simulation swarm distance-travelled for loading and collecting the containers.



(c) The impact of four failed robots on the simulation swarm energy-expended for loading and collecting the containers.

Figure 8.4: Box plots showing a comparison of the performance of the simulated twelve swarm robots before and after recovering from the four robots failures. The data in these diagrams correspond to the simulation time, distance-travelled and energy-expended.

Table 8.2: Average time, distance-travelled and energy-expended for ten runs before and after the simulated swarm was injected with four failures and then mitigated. In addition to a comparison to a simulated swarm with four operational robots.

Metric	12 Robots 0 Failures	12 Robots 4 Failures 0 Mitigated	12 Robots 4 Failures 4 Isolated	12 Robots 4 Failures 4 Bridges	8 Robots 0 Failures
Time from simulation (Seconds)	1174	2002	1299	1493	1242
Distance-Traveled (Meter)	60	77	62	62	64
Energy-Expended (Joules)	1300	2227	1676	1677	1413

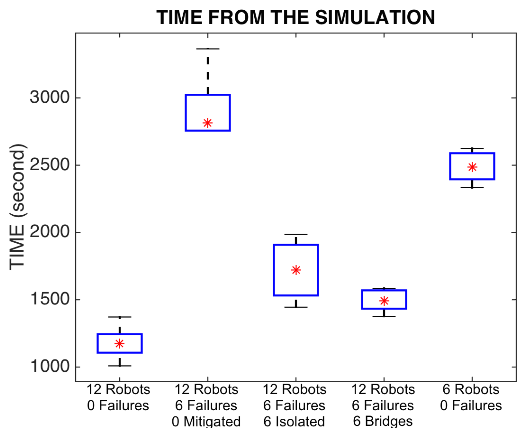
Table 8.3: Average time, distance-travelled and energy-expended for ten runs before and after the simulated swarm was injected with six failures and then mitigated. In addition to a comparison to a simulated swarm with six operational robots.

Metric	12 Robots 0 Failures	12 Robots 6 Failures 0 Mitigated	12 Robots 6 Failures 6 Isolated	12 Robots 6 Failures 6 Bridges	6 Robots 0 Failures
Time from simulation (Seconds)	1174	2814	1722	1727	2487
Distance-Travelled (Meter)	60	88	61	61	126
Energy-Expended (Joules)	1300	3043	1897	1885	2768

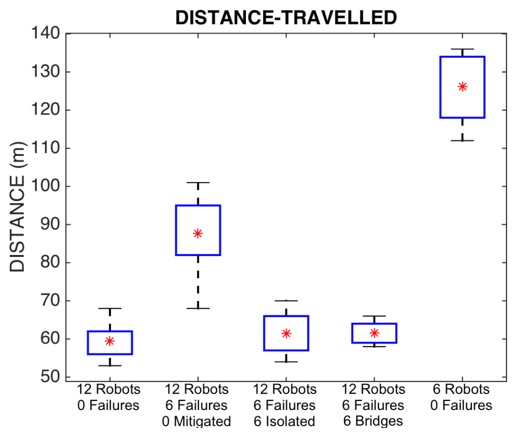
Instead of being obstacles to operational robots, faulty robots, now, can cooperate together with operational robots towards finishing the swarm task. That is, through passing messages between operational robots across various distances.

It is obvious that the time to finish the swarm task in the simulation tends to be increasing whenever there are failures, four and six failures out of 12 operational robots (Figure 8.4(a) and 8.5(a)), respectively. In addition, while failures persisted, the simulated swarm robots tended to travel for a long distance (Figure 8.4(b) and Figure 8.5(b)) and consume more energy (Figure 8.4(c) and Figure 8.5(c)). That is due to the fact that operational robots get busy avoiding the faulty robots while still achieving the swarm task.

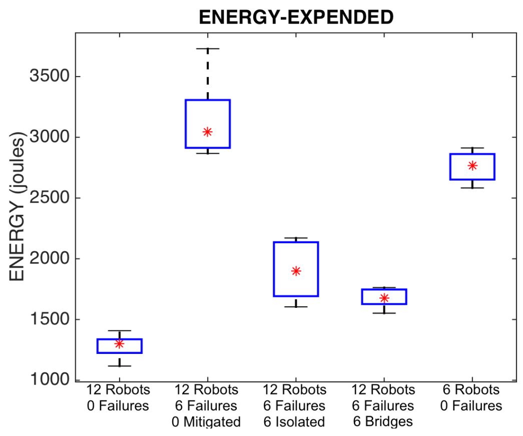
Also, swarm robots, in the simulation experiment, normally perform to cooperate with other robots before failing. While transporting the container, with existing cooperation between two robots, the task operation could be distracted in the middle between the loading and collecting locations on the arena should one of these cooperated robots fail.



(a) The impact of six failed robots on the simulation time to load and collect the containers.



(b) The impact of six failed robots on the simulation swarm distance-travelled for loading and collecting the containers.



(c) The impact of six failed robots on the simulation swarm energy-expended for loading and collecting the containers.

Figure 8.5: Box plots showing a comparison of the performance of the simulated twelve swarm robots before and after recovering from the six robots failures. The data in these diagrams correspond to the simulation time, distance-travelled and energy-expended.

In this type of situation, the operational robot will wait for another operational robot within the communication range to take over the task of the faulty robot, thus, carrying on the swarm task. This procedure might take a long time which causes a delay to finish the swarm task. Especially, if the operational robot has to wait for a long time for another operational robot to take over the position of the faulty one.

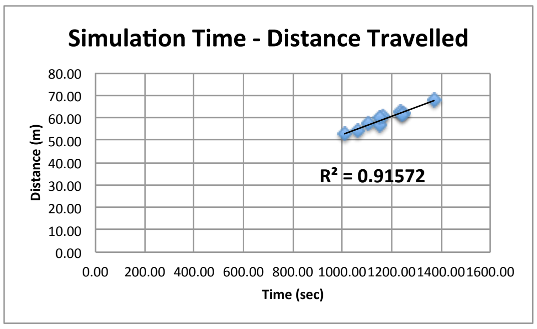
However, the proposed mitigation, in this chapter, shows an effective improvement in the total time, distance-travelled and the energy-expended from the simulation when faulty robots fail in the middle of the simulation swarm task.

It has to be mentioned that the proposed mitigation procedure in this chapter may not be quite as good as the proposed mitigation in Section 7.4. Where previously, robots with failures are pushed to the side of the arena so they don't become obstacles, whereas in the current simulation experiment, robots become obstacles in the middle of the arena. But the benefit is that the faulty robots assist operational robots by transferring their broadcasted messages over the large arena. So, at least the outcome shows an improvement after applying the mitigation procedure. Based on this concept, it could be assumed that the more the robots failed and becoming communication bridges the more the operational robots cooperates over long distances and so finishing the task faster.

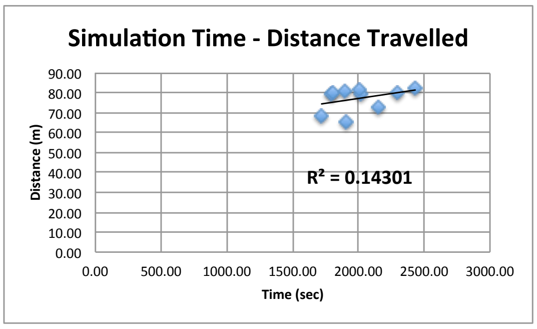
Figure 8.6 and Figure 8.7 shows the comparisons in time from the simulation, the distance-travelled and the energy-expended before and after implementing failures (four and six failures) in the simulated swarm robotics.

Results in Figure 8.8 and Figure 8.9 do not show a perfect fit, this is due to the different times at which failures occurs to each experiment run. So, this may not be a perfect way to measure the efficiency of the applied mitigation in this chapter, of which reusing failed robots as communication bridges. That is due to the fact these faulty robots are still operational and consuming energy compared to the previous experiment where faulty robots are shut down immediately after the failure was detected. In another words, the energy-expended in these faulty robots are still observed during the entire swarm task, even in case they were becoming communication bridge and they does not help with the swarm task. In that case, it would be difficult, as a system developer, to determine if all or most failed robots became useful to the operational swarm robots, otherwise they should be shut down and pushed to the side of the arena as they are not needed any more.

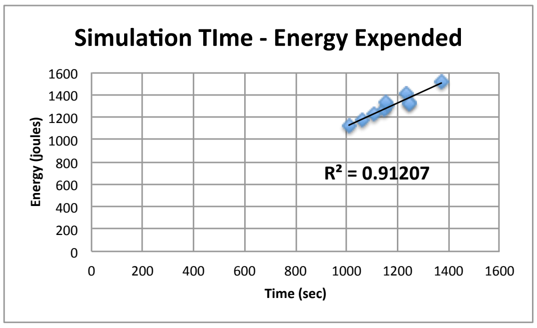
On the other hand, mitigation procedures are important to swarm robotics. This proves that the efficiency of the total time and the energy-expended with a swarm of 12 robots and a swarm of four out of 12 robots is approximately 78 per cent. Even comparing this with a swarm of only eight robots gives an efficiency over 80 per cent of time and energy-expended.



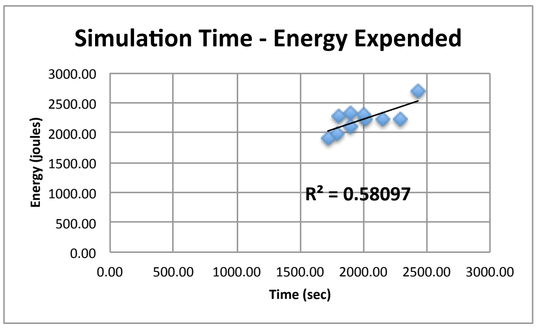
(a) The compatibility of time and distance-travelled with 12 robots to load then collect all containers in the arena.



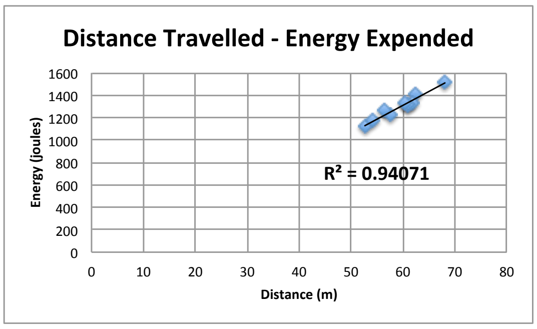
(d) The compatibility of time and distance-travelled with 12 robots, of which four are failed, to load then collect all containers.



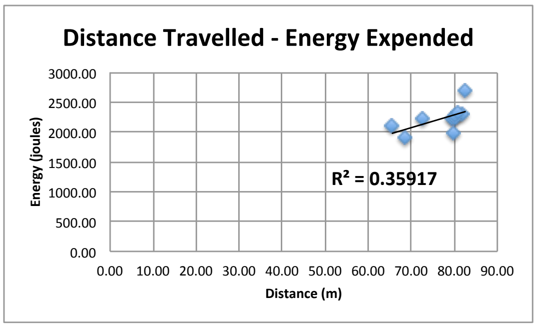
(b) The compatibility of time and energy-expended with 12 robots to load then collect all containers in the arena.



(e) The compatibility of time and energy-expended with 12 robots, of which four are failed, to load then collect all containers.

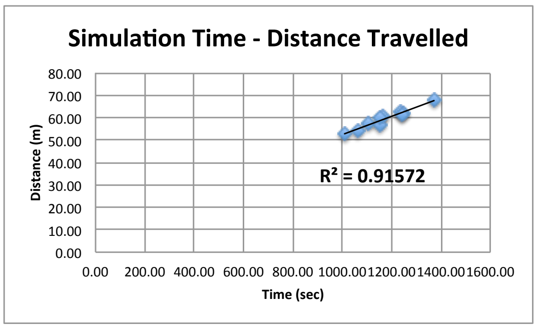


(c) The compatibility of distance-travelled and energy-expended with 12 robots to load then collect all containers in the arena.

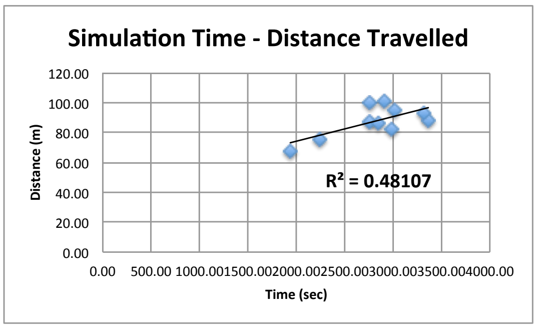


(f) The compatibility of distance-travelled and energy-expended with 12 robots, of which four are failed, to load then collect all containers.

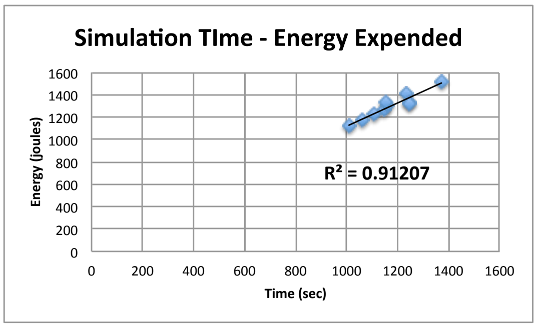
Figure 8.6: The correlation between the time, distance-travelled and energy-expended for a swarm size of 12 robots. Diagrams (a), (b) and (c) represents a swarm with 12 operational robots having no failures. Diagrams (d), (e) and (f) represents a swarm with 12 robots of which four have failures.



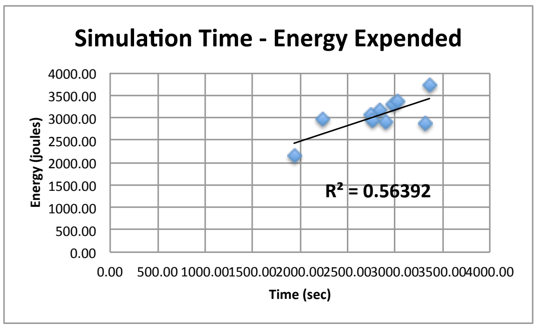
(a) The compatibility of time and distance-travelled with 12 robots to load then collect all containers in the arena.



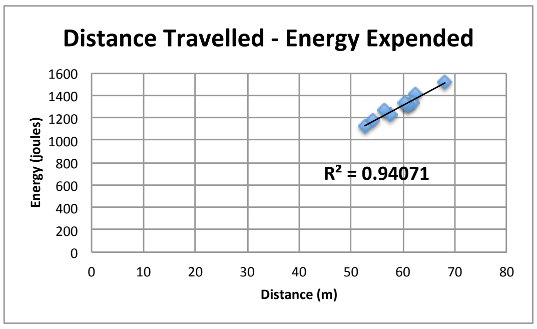
(d) The compatibility of time and distance-travelled with 12 robots, of which six are failed, to load then collect all containers.



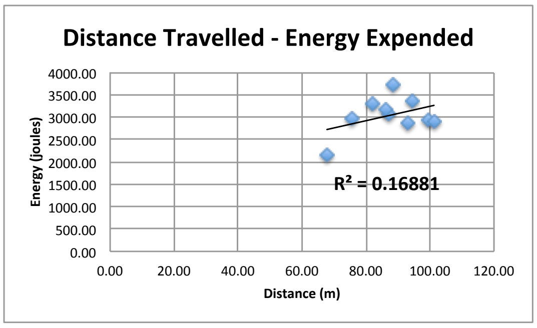
(b) The compatibility of time and energy-expended with 12 robots to load then collect all containers in the arena.



(e) The compatibility of time and energy-expended with 12 robots, of which six are failed, to load then collect all containers.

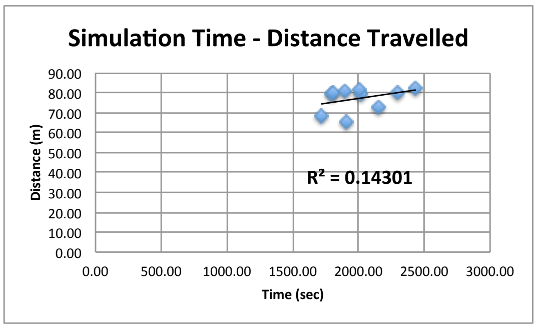


(c) The compatibility of distance-travelled and energy-expended with 12 robots to load then collect all containers in the arena.

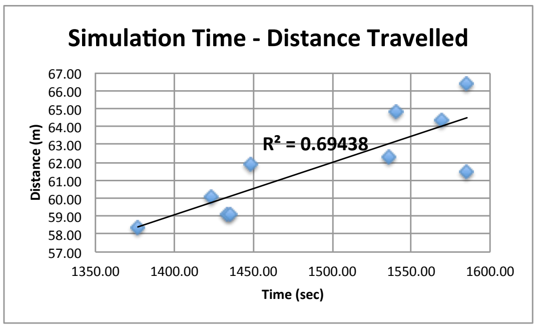


(f) The compatibility of distance-travelled and energy-expended with 12 robots, of which six are failed, to load then collect all containers.

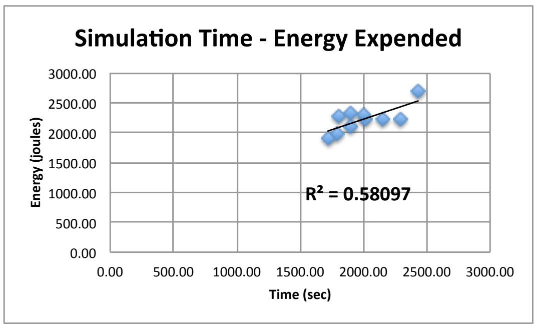
Figure 8.7: The correlation between the time, distance-travelled and energy-expended for a swarm size of 12 robots. Diagrams (a), (b) and (c) represents a swarm with 12 operational robots having no failures. Diagrams (d), (e) and (f) represents a swarm with 12 robots of which six have failures.



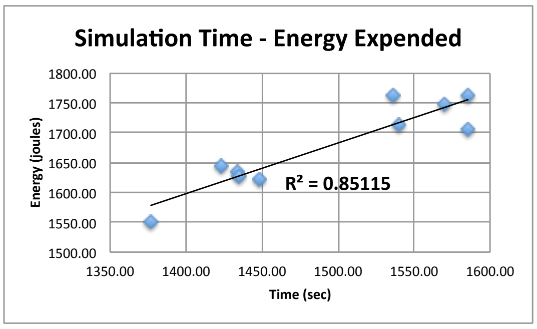
(a) The compatibility of time and distance-travelled with 12 robots, of which four are failed, to load then collect all containers.



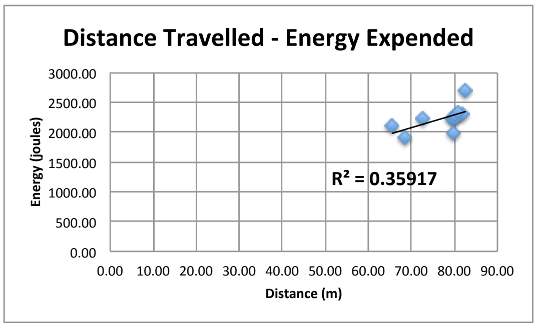
(d) The compatibility of time and distance-travelled with 12 robots, of which four failures are mitigated, to load then collect all containers.



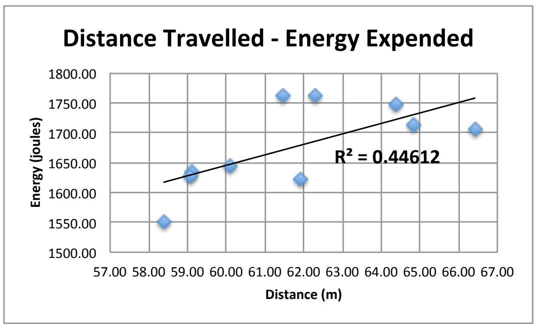
(b) The compatibility of time and energy-expended with 12 robots, of which four are failed, to load then collect all containers.



(e) The compatibility of time and energy-expended with 12 robots, of which four failures are mitigated, to load then collect all containers.

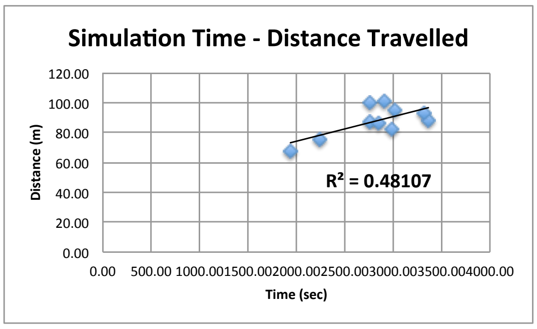


(c) The compatibility of distance-travelled and energy-expended with 12 robots, of which four are failed, to load then collect all containers.

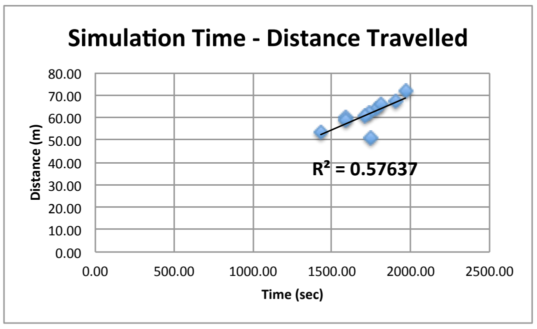


(f) The compatibility of distance-travelled and energy-expended with 12 robots, of which four failures are mitigated, to load then collect all containers.

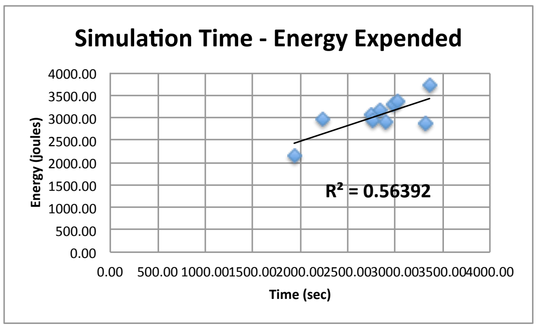
Figure 8.8: The correlation between the time, distance-travelled and energy-expended for a swarm size of 12 robots. Diagrams (a), (b) and (c) represents a swarm with 12 robots of which four have failures. Diagrams (d), (e) and (f) represents a swarm with 12 robots of which four failures are mitigated.



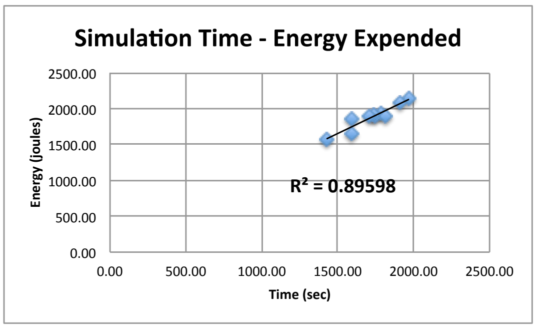
(a) The compatibility of time and distance-travelled with 12 robots, of which six are failed, to load then collect all containers.



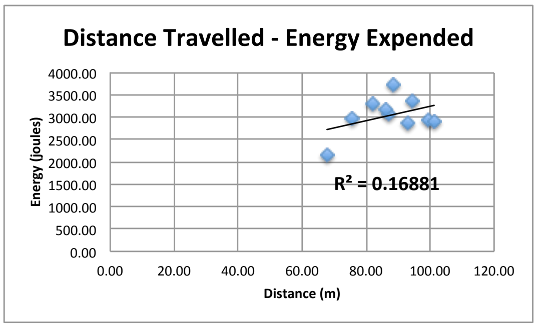
(d) The compatibility of time and distance-travelled with 12 robots, of which six failures are mitigated, to load then collect all containers.



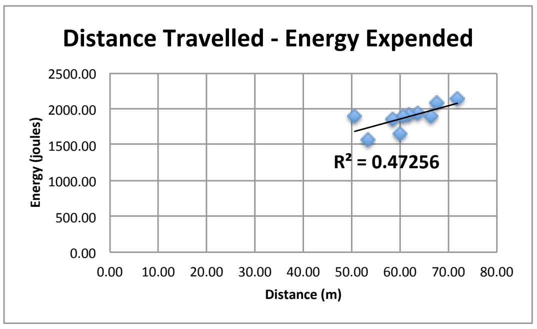
(b) The compatibility of time and energy-expended with 12 robots, of which six are failed, to load then collect all containers.



(e) The compatibility of time and energy-expended with 12 robots, of which six failures are mitigated, to load then collect all containers.



(c) The compatibility of distance-travelled and energy-expended with 12 robots, of which six are failed, to load then collect all containers.



(f) The compatibility of distance-travelled and energy-expended with 12 robots, of which six failures are mitigated, to load then collect all containers.

Figure 8.9: The correlation between the time, distance-travelled and energy-expended for a swarm size of 12 robots. Diagrams (a), (b) and (c) represents a swarm with 12 robots of which six have failures. Diagrams (d), (e) and (f) represents a swarm with 12 robots of which six failures are mitigated.

8.4 Summary

The methodology and the structure of the arena used during the observation in this chapter are exactly the same as in the previous case study was discussed in Chapter 6.

An experiment was undertaken, at the beginning of this chapter, which shows even when robots fail in the arena, there is still usable resources that could be useful to the entire simulation swarm robotics. One of these usable resources is the remaining energy in the faulty robots. The use of the remaining amount of resources was defined as the mitigation solution to the operational robots.

However, in this chapter, the proposed solution for the use of the remaining energy was expressed as building up a communication bridge based on these faulty robots, as they still have remaining energy. Therefore, there is no need for this additional energy to be transferred to operational robots.

The results show that the proposed mitigation solution has an effect for enhancing the performance of the simulated robotic swarm with stationary failures.

Chapter 9

Research Discussion

The work represented in this thesis discusses the importance of applying a reliability approach to swarm robotics; an approach for better failures detection and solutions in distributed robotic systems.

By implementing the proposed exogenous failure detection and diagnosis approach, robots are able to share and diagnose their processing data and mitigating complex problems while they are still independent.

The proposed exogenous failure detection model was inspired by the concepts from fireflies, (Christensen et al., 2009) and the robot internal simulation approach, (Millard et al., 2013).

The discussion in the thesis has been supported with case studies with the collection of containers, once by an individual robot then through collaboration with a partner robot.

9.1 Research Contributions

The reviewed literature shows that the majority of researcher agree that swarm robots hardware failures are quite common. Although failures can exist in swarm robotics, researchers claim that swarm robotics is still scalable. That means, whenever there is failure exist it is possible to add more robots to cover up the failure gap. However, adding more robots to cover up the failures will discard an important property which is the robustness property of swarm robotics, because the failed robots remain in the task area and will interfere with the operational robots.

Referring to **RQ1**, due to the limited reliability, particularly in the robot's hardware level, a number of failure detection methods and algorithms have been listed throughout the literature review in this thesis.

The main principles required for diagnosis the swarm failures lies on the identity of communicated robots in the swarm. These principles will build up the idea of which robot is operational and which has the failure. Then comes the importance of transferred data, including the location, the sensor readings and the actuators values.

The proposed exogenous failure detection model have the ability of not just detect the failed robot, but also can trace down the exact location and type of failures in a failed robot. Another principle has been covered in the proposed exogenous failure detection model for identifying an exact failure within the failed robot was inspired from Fault Tree Analysis (FTA) technique. The concept of Tree Analysis (FTA) was analysing failures attempts to identify all of the potential causes of the event and the logical sequence of secondary events that were leading up to the failure.

Moreover, the exogenous failure detection model works by identifying the type of failures immediately, despite techniques that have been reviewed previously. The failure analysis and diagnosis in the proposed model of this research are explained through the next sections of this Chapter.

Following up to the concept of the algorithm of the immune systems in the Granuloma Formation, the principles from the two failure detection approaches have been combined to come up with the proposed exogenous failure detection model in this thesis. The fireflies approach requires an existence-communication between individual robots by monitoring and synchronising a flashing LED of two robots. On the other hand, the internal simulator approaches which require an individual robot to predict the health status of its partner, then compare its computational values with the real received values from the partner.

Again, principles from the reviewed approaches have been combined and customised to form the proposed exogenous failure detection model to be applied to swarm robotics and it could also be implemented to multi-robot systems.

The work resolved the issues discussed in **RQ2**. The capability of the proposed exogenous fault detection model was designed to repeatedly diagnose swarm robot's health statuses and undertake an action towards mitigating the faulty robots. Although the structure of the exogenous failure detection model makes it capable of dealing with several different failures, further simulated experimentation is needed to be investigated.

From the collected results, after applying the exogenous failure detection model to the simulation swarm robotics, the simulation swarm robotics shows noticeable differences in the behaviour of robots especially after using the mitigation procedures. The exogenous failure detection model was implemented within the E-Puck robot's controller. Along with the part that is responsible for doing the swarm task, the exogenous failure detection part in the controller did not make a big different. The exogenous failure detection and the mitigation process plays a job for bringing down the cause of failures to make the swarm robotics as close in the behaviour as to a swarm robotics with no failures.

As the robot's controller plays both parts, achieving the swarm task and diagnosis other robot's health status, the controller did not show a noticeable defective. The mean time for the proposed exogenous failure detection model to immediate react toward detecting the failure was 0.2 seconds. This reaction result may not be compared to other fault detection models as it was not mentioned, and so there was no particular model for detecting the response time for detecting the failure. Moreover, the approaches from other failure detection models depend mostly on a present of list of failures of which requires a robot's controller to go through a list of possible failures every controller cycle, or the failure detection model need a prediction of failures in the future which may take time for detecting the failures.

With the existence of failures, swarm robotics tend to spend more time for finishing their task. That is due to the dynamic obstacle that is generated from faulty robots when they suddenly become faulty and shutdown. These faulty robots become stationary in the way of operational robots in the arena, which forces operational robots to take extra iteration and turn around these faulty ones to avoid them. Also, it becomes more difficult for the operational robots while they are cooperating with another operational robot and performing a task. That causes cooperative robots to take extra moves to turn around obstacles by keeping on transporting the container.

Therefore, by applying mitigation to the swarm robotics, the simulation shows a tremendous enhancement to the overall performance and efficiency, and hence addresses **RQ3**

During the represented work in this thesis, mitigation procedures moved from mitigating a swarm with failures while achieving a simple task. When one robot was required to transport the container to the destination location, and mitigation the swarm while cooperating with a partner and achieving a complex task, by transporting the container to the collecting location after passing the loading location in the arena. The collected results show how the time, the distance-travelled and the energy expended from the simulation has been enhanced compared to the collected results for a swarm with non-mitigated failures.

Without applying the mitigation procedures, the swarm could tend to run for a long time, that could affect the entire swarm task when all the robot's battery ran out of energy. Also, this might end up the simulation with an uncompleted task.

Another form of mitigation was represented which allow the swarm to reuse faulty robots to become a communication bridge to handle communications over the long distance between operational robots. Throughout the simulation experiments in this thesis, this particular mitigation procedure did show quite the same performance of swarm with failures compared to the behaviour of swarm after pushing faulty robots to the side of the arena. But at least, failed robots did help to bring operational robot over a far distance to cooperate with another operational robot that was waiting for assistance.

Achieve the mitigation and determine which mitigation would be required for a particular swarm task by diagnosing and identifying the failure. As failures could affect the swarm robots at the hardware, a brief discussion in this Chapter (9) explains how a particular hardware failure could be diagnosis and defined, also the provided discussions answers to **RQ4**. The software failure will require another research effort that could be justified by more experiments. The failure diagnosis in the robots' hardware requires diagnosing each device individually. The confirmation of the validity of one device helps to diagnose the other device in the E-Puck robot. This step is needed to diagnosis the failure and in which device exists, are going to be explained more during this Chapter. The swarm robotics software failure requires further investigation. Therefore, the software failures have been moved to the future work in this thesis.

9.2 Identifying Threats to Swarm Robotics

The failures could threaten the robot at the hardware and/or software level. Also the swarm task could be threatened by an environment disturbance. Figure 2.4 illustrates a number of threats that could harm the swarm robotics.

During the work in this thesis, the failure affected the swarm robot at the hardware level. Most importantly, noise was implemented at the navigation unit, to simulate the failure, in the E-Puck robots at different times during the simulation experiments.

The idea behind selecting the navigation unit to be determined whither it has been manipulated with a failure is to trace down any cause of failure in different devices in case the navigation unit is operational. The mechanism for diagnosis of all devices on E-Puck robots would require a deep investigation and more experiments. Therefore, the experiments for all devices' failures was proposed to be for future work. Also, that includes an investigation into the failures at the software level.

The E-Puck robot is equipped with a number of devices including proximity sensors, ground sensors, actuators and a communication unit. Based on the proposed exogenous failure detection model, the validity of one of these devices leads toward diagnosis of the next device and check if it has a failure or not. If all devices are validated, then the E-Puck robot will be considered operational.

However, this chapter will deliver details about how the proposed exogenous failure detector could identify hardware failures, not just failures in the navigation unit, but also it will extend that to the rest of most common E-Puck robot devices, including the compass and the actuator motors. Apart from the navigation device, detecting failures in the detail algorithms provided in the next subsections are based on logical concepts.

Never the less, these algorithms were built at the same time along side with the proposed exogenous failure detection model but have never tested. Running experiments to test these algorithms will requires more time to justify its functionality. That is due to the fact a failure detection for more other devices is required, like the proximity sensors and the ground sensors and the LEDs in case the onboard camera is required for the swarm task. The outcome of the provided algorithms for detecting failures in the compass and actuator motors could be rather similar to the outcome from the proposed failure detection in the navigation device in the thesis.

9.2.1 Diagnosis Failures in the E-Puck Navigation Device

The diagnosis of failures that were previously represented in Section 7.4 will be extended to explain more details of how the proposed exogenous failure detection model was structured in order to detect failures occurring in the E-Puck robot devices. Diagnosis and identifying the exact device that has the failure could help with mitigation and recovering the swarm robotics and enhance its reliability.

During the use of the proposed exogenous failure detection model, the communication unit is assumed be fully operational with no failure at any time. This assumption helps to diagnose and track down any existing failures in the other devices. The diagnosis of the E-Puck robot devices needs to apply to each device, step by step. That will be explained later in this section.

Following the description that is related to Figure 7.6, the extension of the diagnosis to identifying the location of the failure is demonstrated in Figure 9.1. As the communication unit is assumed operational, the simulation experiments in Section 7.3 experience a failure in the navigation device only. By ensuring that the navigation device is valid and operational, the diagnosis process then moves on to check and validate the other devices in the E-Puck robots.

To diagnose the navigation device, at the beginning, the robot broadcasts a message to the swarm. The message includes the ID from of the robots, the position (X and Z) and the rest of the input and output data. Then, the robot listens to the received data. Whenever the robots receive data from another robot, the controller on the robot computes the distance, $D_{navigation}$, between the two robots based on calculating the difference of both robot's locations. As the robots' locations are shared and it has not been generated by one robot, Equation 5.1 is applied to the robot. Also, the robot computes the distance based on the signal strength, E_{signal}, of the received data, Equation 5.2. If the result from $D_{navigation}$ is matched with the result from E_{signal}, then both robots are considered operational. Of course this procedure is counterproductive, that means both robots have to diagnose and analyse its own received data.

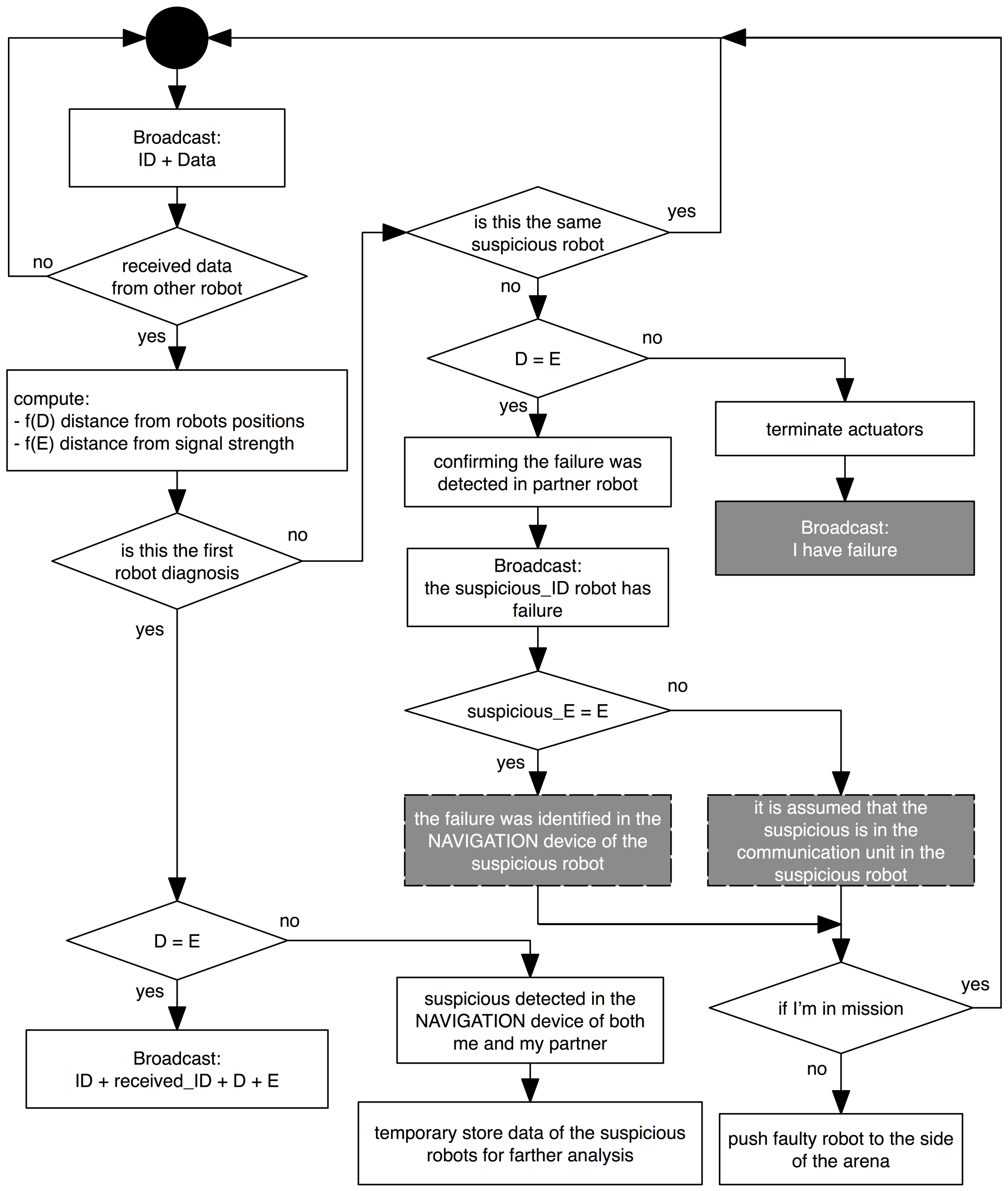

Broadcast:
ID + Data
received data from other robot
no
yes
compute:
- f(D) distance from robots positions
- f(E) distance from signal strength
is this the first robot diagnosis
no
yes
is this the same suspicious robot
yes
no
D = E
no
yes
terminate actuators
Broadcast:
I have failure
confirming the failure was detected in partner robot
Broadcast:
the suspicious_ID robot has failure
suspicious_E = E
no
yes
the failure was identified in the NAVIGATION device of the suspicious robot
it is assumed that the suspicious is in the communication unit in the suspicious robot
D = E
no
yes
suspicious detected in the NAVIGATION device of both me and my partner
if I'm in mission
yes
no
Broadcast:
ID + received_ID + D + E
temporary store data of the suspicious robots for farther analysis
push faulty robot to the side of the arena

Figure 9.1: The E-Puck controller behaviour for the cooperation between robots with the extra feature for diagnosis of the navigation device from the exogenous failure detection model. The symbol D represents the computed distance from the position of two E-Puck robots. The symbol E represents the computed distance from the signal strength of the receiver. The black circle refers back to the rest of the E-Puck controller including the robot task part.

However, if the results from $D_{navigation}$ and E_{signal} are not matching, then both robots will suspect a failure in the navigation device. At this moment, they temporarily store the data after diagnosis to be used later for identifying the failed device and which robot.

Next, one or both suspicious robots follow the same previous steps of diagnosis with a third robot. If the suspicious behaviour is still present even after diagnosis with the third robot, they then check with a fourth robot. However, if the third or fourth robot is operational, then this robot will be announced faulty by broadcasting its information to the rest of the swarm robots. The mitigation procedure is then applied and the faulty robot is pushed to the side of the arena.

Both, the operational and the faulty robot, are able to identify the faulty device in the robot. Yet the robot with the failure is trustable anymore, so one should announce the identity of the failed device is the operational robot. The operational robot first compute the distance E_{signal} from the signal strength, and uses the E_{signal} that was received from the faulty robot at the beginning. Then, the operational robot checks whether both E_{signal} are matched. If so, that means the navigation device in the faulty robot is generating false data. Therefore, the failure was identified as the navigation device.

Depending on the swarm robotics task, robots can decide if this failure needs to be mitigated or should the faulty robot be left with no action taken. The part regarding undertaking a decision by the swarm members could be part of another research contribution and needs more investigation.

After the failure diagnosis, if there is no suspicious behaviour in the computed data, $D_{navigation}$ and E_{signal}, then it is time for the controller to move on and diagnose the next device.

9.2.2 Diagnosis Failures in the E-Puck Compass

The E-Puck robot is equipped with a compass device that is used to determine the robot's absolute orientation from the north.

The deployed experiments in this thesis relied on the compass device, which was used as part of the navigation purposes.

Since the compass is not an integrated unit with the navigation device in the E-Puck robot, there is still a need to diagnose and analyse the generated values from the compass device. Validating the compass in the E-Puck will lead to diagnoses of other devices and check if one has a failure that could lead to harm the swarm robotics.

As far the navigation device is diagnosed and validated as operational, the next step is to diagnose the compass device. In this step, diagnosis of the compass device may not need another robot to diagnose it, but the reason there is a need to collaborate with another partner is to reduce the doubts that the robot itself had another failure at the software level, or the robot could be categorised as suspicious because it did not yet find an operational robot. Also, even if the robot itself is operational, still other robots have suspicions and need assistance from operational robots.

Figure 9.2 addresses the compass diagnosis procedure. At the beginning, the E-Puck robot needs to diagnose itself to confirm it can diagnose the received data from the other robots. The diagnosis procedure starts after the controller computes the orientation from the confirmed validated navigation device (coordinates ΔX and ΔZ) and compares the results with the values from the compass. The coordinates are defined as the difference between a previous location of the robot and the current location after a period of time, T, $\Delta X = (X_2 - X_1)$ and $\Delta Z = (Z_2 - Z_1)$. Equation 9.1 is applied to compute the direction, Θ. If both values are matched, then the robot is operational, and can diagnose partner robots in the field.

$$\Theta = \tan^{-1}\left(\frac{\Delta Z}{\Delta X}\right) \tag{9.1}$$

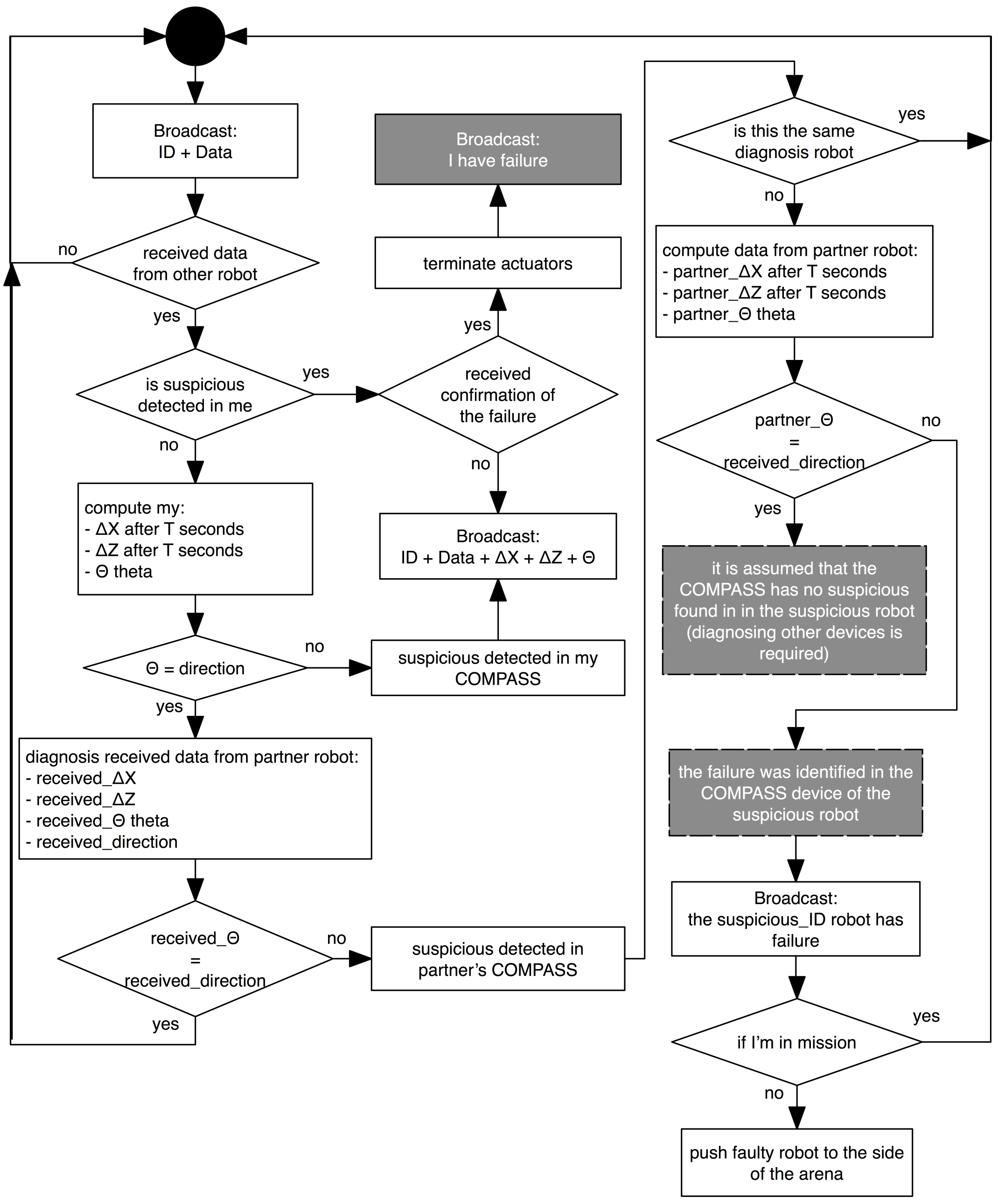
The operational robot diagnoses the suspicious partner robots, based on two measurements. It firstly diagnoses the exact received computed values from the suspicious partner to confirm the suspiciousness in that robot. If the suspicious behaviour is precise in the partner robot, then the operational robot runs its own diagnosis procedure based on the received data, that includes the coordinates, X and Z, and the direction from the suspicious robots. If the new computed Θ results of the suspicious partner is matched with the received direction from the compass, then the suspicious partner is announced as not failed, and a confirmation message is circulated and broadcasted to be received by the suspicious robot.

Depending on the swarm robotics task, the robots can decide if this failure needs to be mitigated or should the faulty robot be left with no action taken.

9.2.3 Diagnosis Failures in the E-Puck Actuator Motors

The E-Puck model is a differential wheeled robot. The travel speed of the E-Puck is $13cm/s$, (Cyberbotics, 2013; Mondada and Michael, 2007). Based on the fact that the navigation device is operational along with its compass, it is possible to diagnose the predicted coordinates of the E-Puck robot from the current time, T and the time in the future, ΔT. The computation for this diagnosis lies in the calculation of the distance-travelled too, by applying Equation **??**. The outcome is then compared to the coordinates that were first predicted at ΔT.

Figure 9.3 covers the idea of actuator motors diagnosis. As part of the diagnosis process of the compass, the E-Puck robot first diagnoses its own actuator motors. The diagnosis computation function in the robot controller gets the values that were sent to the actuator motors while the robot is wandering. If the robot's actuator motors are confirmed operational, then this robot will be susceptive to diagnosis other robots in the swarm. When an operational robot receives data from a partner robot, the controller in the



Broadcast: ID + Data
received data from other robot
no
yes
is suspicious detected in me
yes
no
compute my:
- ΔX after T seconds
- ΔZ after T seconds
- Θ theta
Θ = direction
no
yes
diagnosis received data from partner robot:
- received_ΔX
- received_ΔZ
- received_Θ theta
- received_direction
received_Θ = received_direction
no
yes
Broadcast: I have failure
terminate actuators
yes
received confirmation of the failure
no
Broadcast: ID + Data + ΔX + ΔZ + Θ
suspicious detected in my COMPASS
suspicious detected in partner's COMPASS
is this the same diagnosis robot
yes
no
compute data from partner robot:
- partner_ΔX after T seconds
- partner_ΔZ after T seconds
- partner_Θ theta
partner_Θ = received_direction
no
yes
it is assumed that the COMPASS has no suspicious found in in the suspicious robot (diagnosing other devices is required)
the failure was identified in the COMPASS device of the suspicious robot
Broadcast: the suspicious_ID robot has failure
if I'm in mission
yes
no
push faulty robot to the side of the arena

Figure 9.2: The E-Puck controller behaviour for the cooperation between robots with the extra feature for diagnosing the compass device from the exogenous failure detection model. The Θ is computed from the coordinates, ΔX and ΔZ after T seconds. The coordinates are defined as the difference between a previous location of the robot and the current location after a period of time, T, $\Delta X = (X_2 - X_1)$ and $\Delta Z = (Z_2 - Z_1)$. The black circle refers back to the rest of the E-Puck controller including the robot task part.

operational robot runs the same process by applying the received data and diagnoses it. Whenever the diagnosis data are not matched with the predicted data, the suspicious robot will be announced as having a failure in the actuators, therefore, the robot failed.

As far as the E-Puck robot diagnoses partner robots, still it keeps a diagnosis of itself

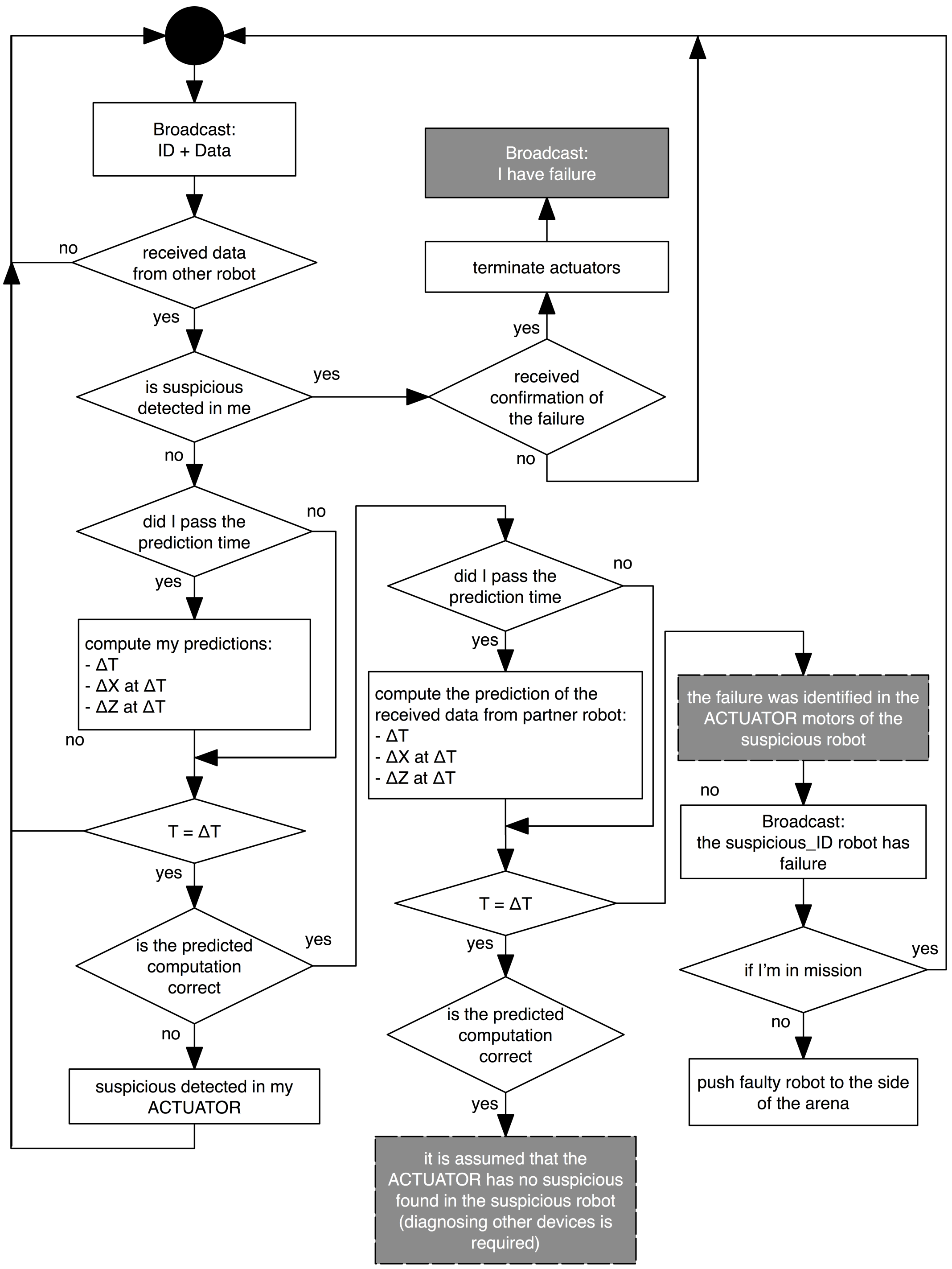

Broadcast:
ID + Data
received data from other robot
no
yes
is suspicious detected in me
yes
no
Broadcast:
I have failure
terminate actuators
yes
received confirmation of the failure
no
did I pass the prediction time
no
yes
compute my predictions:
- ΔT
- ΔX at ΔT
- ΔZ at ΔT
no
T = ΔT
yes
is the predicted computation correct
yes
no
suspicious detected in my ACTUATOR
did I pass the prediction time
no
yes
compute the prediction of the received data from partner robot:
- ΔT
- ΔX at ΔT
- ΔZ at ΔT
T = ΔT
yes
is the predicted computation correct
yes
it is assumed that the ACTUATOR has no suspicious found in the suspicious robot (diagnosing other devices is required)
the failure was identified in the ACTUATOR motors of the suspicious robot
no
Broadcast:
the suspicious_ID robot has failure
if I'm in mission
yes
no
push faulty robot to the side of the arena

Figure 9.3: The E-Puck controller behaviour for the cooperation between robots with the extra feature for diagnosing the actuator motors from the exogenous failure detection model. The coordinates, ΔX and ΔZ are the predicted cimputed values at ΔT. The current time is represented as T The black circle refers back to the rest of the E-Puck controller including the robot task part.

at all times. However, the decision to undertake the mitigation procedure depends on the swarm robotics task whether the mitigation is required for the task or not.

9.3 Summary

The failure diagnosis in the proposed exogenous failure detection model was inspired by the concepts from fireflies and the robot internal simulation approach. The idea of having quite a realistic way of interacting between individual robots is similar to that observed with a real life consultation between the doctor and the patient at the hospital. When a patient consults a doctor with a question and answer session at the end that may lead to a proper solution(s).

Diagnosing devices in E-Puck robots requires analysing each device individually. The diagnosis process works sequentially, starting with diagnosis of the navigation device and ending with the actuator motors. Therefore, the diagnosis of a certain device requires diagnosis of the previous device in that sequence.

It is worth mentioning that the diagnosis provided in this chapter is logically correct and achievable. The observation can be extended to cover the diagnosis of devices of other types of swarm robot models and hence other large scale multi-agent systems.

Chapter 10

Conclusions

Swarm robotics depends on cooperation between robots, so the failure of an individual robot may have an impact on the performance of the entire swarm behaviour.

It is considered necessary for swarm robotics to be able to deal with different types of failures that could constitute a threat to the swarm robotics task. Achieving reliability requires the identification and addressing of different failures and their effects on individual robots, as well as the entire swarm. The objective of this achievement lies in detection mechanisms that identify and address the fault in the swarm.

The fault detection approach is based on relevant exogenous fault diagnosis systems. There are two relevant approaches that deal with detecting swarm robotics failures, namely the Fireflies approach and the Robot Internal Simulator. The investigation of these two approaches revealed a number of issues including the unsynchronized diagnosis between robots. However, with a combination of both approaches, with an applied modification to one model, and by allowing robots to analyse their neighbours' data, the resulting exogenous fault detection model was perfectly convincing, especially using direct communication amongst the simulated swarm robots. The proposed exogenous fault detection model allows the robots to broadcast their own health status to each other for consultation and feedback.

Simulated experiments were set in the Webots simulation tool. The measurement metrics used for the evaluation were the time, distance-travelled and energy-expended by all the robots in the simulated swarm.

The first experiment was run to evaluate the best swarm size to use for the research experiments. This experiment showed that when the swarm size increased, then the simulated swarm performance got better. The results showed a slight decrease in the overall simulated swarm robotic performance between swarm size 10 and 20 robots, where the significant decrease was observed between 1 and 10 robots. Although the real time to finish the simulated swarm task is better with the swarm size of 10 robots, 10 robots was chosen as the perfect swarm size.

The next experiments conducted were intended to evaluate the exogenous fault detection model and mitigate the impact of failure on the simulated swarm. The results of the experiments with the exogenous fault detection model were fairly successful.

Another level of the simulation experiment was provoked to observe the exogenous failure detection model and a couple of proposed mitigation procedures with a complex swarm task. The complex task required two E-Puck robots to cooperate, in order to transport the container to the loading location, then to the collection location. This required an increase in the arena size to 5×5 meter square. This time, failures occurred during the mid-time after the starting of the simulation, but not applied at the very beginning.

Even implementing the complex task, the exogenous failure detection model approves to be remarkably success for detecting the failures in a short time. In addition, the proposed mitigation procedures have shown an enhancement to the total time from the simulation, the distance-travelled and the energy-expended.

The simulated experiments with the proposed exogenous fault detection model were based on the evaluation of one hardware failure in the navigation device. There is still a need for testing of more hardware failures, including compass, actuator motors and proximity sensors. An elaborated observation were undertaken toward diagnosis different types of devices failure in the E-Puck robot. The observation can be extended to cover the diagnosis of devices of other types of swarm robot models and hence other large scale multi-agent systems.

In addition, there is a need for more investigation at the swarm robotics software level. This will be added to the current exogenous fault detection model in the future.

10.1 Future Work

Fault detection and recovery mechanisms will be investigated further in this research. Moreover, examining a wide variety of swarm robotics failures through the simulation system is very important. This is because there is a need to explore the effects of different failures in order to observe the swarm system entities, then to bring them to the next level where recovery and mitigation take place in the swarm. The success of the simulation experiment and the convincing results in this research will lead the investigation towards more recovery solutions.

The next step now is to diagnose more failures in different devices including the proximity sensors and actuators. With the same diagnosis concept, swarm robots are able to detect a fault in one of these devices after making sure that the navigation and the communication devices are operational.

In the current experiments, the navigation device's validity was investigated along with the assumption of the validity of communication between robots. Now, after ensuring that the navigation device has no fault, the values provided could be compared against other hardware devices to check their validity too.

The robot controller sends the moving values to the actuators according to the present obstacle. At the same time, the controller receives a navigation value based on the movement of the robot. If the computed values do not match, then the robot acknowledges a fault in its actuators.

The validity of the actuators will require another device to check this. Proximity sensor values are important for robot movement. The robot controller will continually check with both the navigation device and actuator values to determine if the robot is moving as claimed. If the proximity sensors do not receive any signal indicating an obstacle, but the navigation values are acknowledging that the robot is static, and if the controller knows that both navigation device and actuators are operational, then a failure will be acknowledged in the proximity sensors.

Future work will include investigating the impact of the probability of failures in actuators and proximity sensors.

There is still a need to consider the limited comparability between the configurability of simulation tools and real world robots. This is due to the hardware capabilities, which cannot be compared with those in real robots. The consequence could be demonstrated with the navigation device, because in real world robots, the GPS will have an overset accuracy which is different from the simulation tool. If it was required to have an accurate navigation device during this simulated research experiment, there are alternative devices that could be used for robot navigation.

There are two types of robot navigation concepts: indoor and outdoor. The outdoor systems need to be navigated through a GPS device, while indoor systems use a variety of devices for navigation, including WiFi localisation, vision-based navigation and preset guide maps. SLAM technology, Simultaneous Localisation and Mapping (Choset and Nagatani, 2001), is another method for robot navigation. SLAM could be determined for both indoor and outdoor swarm robotics applications.

In this research, the navigation device used was set to a hundred percent accuracy which means the values obtained are totally accurate. Therefore, the next experiments could be undertaken with alternative ways to guarantee swarm robots navigation having a realistic accuracy.

Appendix A

Experiments for E-Puck Validity in Webots

An additional experiments and details related to the validity of E-Puck in Webots that were undertaken during the work in Chapter 3.

A.1 Initial E-Puck Robots Positions in the Simulated Arena

In addition to initial positions of the 10 E-Puck robots were represented in Figure 3.8 for most of the research simulated experiments were undertaken in Chapters 4 and 5, Figure A.1 presents the initial positions of all the 20 E-Puck robots. Table A.1 shows a descriptive initial positions and orientations of all the 20 E-Puck robots.

Additionally, the initial positions of E-Puck robots in experiments undertaken in Chapters 6, 7 and 8 are represented in Figure A.2, with E-Puck robots position details illustrated in Table A.2.

A.2 Initial Containers Positions in the Simulated Arena

Table A.3 illustrates the positions on containers from the loading and collecting task in Chapters 6, 7 and 8, Figure A.2.

A.3 Validating the Energy-Expended Sensor

The energy expended has been discussed about through the previous three simulated experiments. The main goal of this section is to validate the energy expended in E-Puck robot and check wither the robot is expending energy as the robot supposed to do.

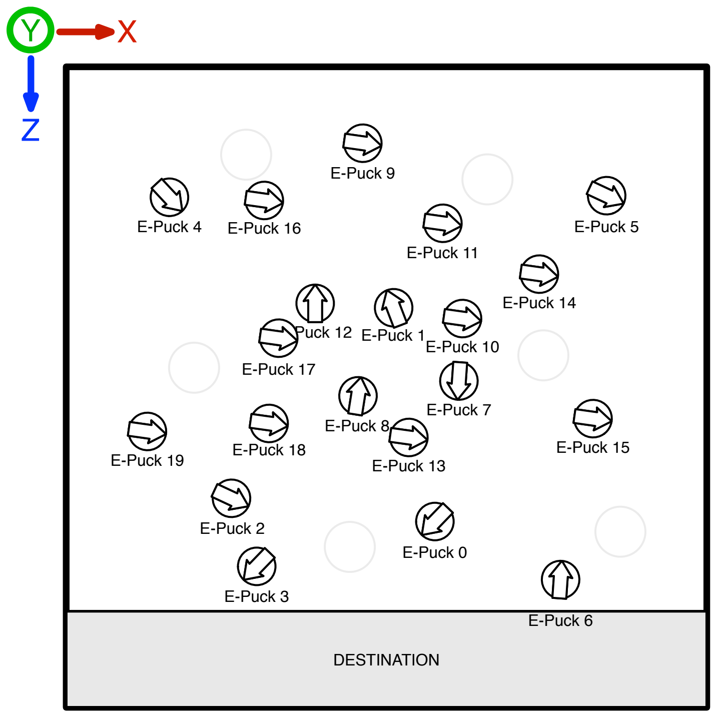


Figure A.1: The 20 E-Puck robots location in the arena for foraging task. The circle represent the robot and the arrow represents the orientation. The measurement is in meter. The grey area at the bottom represents the destination to which the objects must be moved. The X, Y and Z axes at the top left corner represent the orientation of the arena in the VRML standard. This digram is set to scale.

It has been observed from the previous experiments that the energy are not expending as it should be. Briefly, while robot is not distracted by any obstacle or force that add-on more torque to its devices (e.g. the actuators), the energy is linearly consumption. On another hand, when the robot's devices are overload forced to a different limit, the energy expended must be dropped down quicker. However, the Webots is not measuring the energy expended correctly. For this reason, it is necessarily to implement a function tool to simulate the real behaviour of the energy consumption. Otherwise, contraction with the Webots team support to fix this as soon as possible in order to carry on the coming experiments where it is necessarily to detect any failure in the battery sensor that check the energy level.

In the E-Puck prototype file, the Webots developer claimed that the energy should consumed based on the use of the robot's CPU and motors. Figure A.3 shows an extra experiment for testing the energy expended in one E-Puck robot. The experiment was set to rune for ten seconds. In the first testing, all devices has been terminated except the CPU, Figure A.3(a). The second test were taking while the CPU and motors are all operational, Figure A.3(b). The last test were taking while all devices on the robot are operational, that includes the CPU, motors, proximity sensors, LEDs and the navigation device.

Table A.1: The 20 E-Puck robots positions in the arena for foraging task. The orientation relative to the -Z direction.

Robot Name	**X-Axis(m)**	**Z-Axis(m)**	**Orientation(degree)**
E-Puck 0	0.87	1.09	226
E-Puck 1	0.77	0.56	339
E-Puck 2	0.37	1.03	110
E-Puck 3	0.43	1.20	226
E-Puck 4	0.21	0.28	144
E-Puck 5	1.30	0.28	110
E-Puck 6	1.19	1.23	6
E-Puck 7	0.93	0.74	184
E-Puck 8	0.68	0.78	11
E-Puck 9	0.69	0.14	92
E-Puck 10	0.94	0.58	98
E-Puck 11	0.89	0.34	323
E-Puck 12	0.58	0.54	68
E-Puck 13	0.81	0.88	248
E-Puck 14	1.14	0.47	278
E-Puck 15	1.27	0.83	8
E-Puck 16	0.45	0.29	128
E-Puck 17	0.48	0.63	83
E-Puck 18	0.46	0.84	8
E-Puck 19	0.16	0.86	23

Results are expressed in these experiments explain that the energy expended are totally linear with the time. This confirms what has been previously reported during the above experiments that either counting on time or energy are providing the same concept.

A.4 Checking Validity of the E-Puck Proximity Sensors

One E-Puck robot and one obstacle were tested and evaluated in Webots in order to observe the behaviour of the E-Puck avoidance sensors. An E-Puck and a cylinder obstacle were placed at the same axis (z-axis), but away from each other (Figure A.4). While the E-Puck was facing the obstacle, and when the simulation was running, the individual value of each simulated experiment run was recorded (Table A.4).

Through this simulation experiment, it was observed that E-Puck tends to randomly avoid the obstacle whenever the robot gets close to the obstacle, i.e., the E- Puck tends to turn to either the right or the left in no particular order. This is due to the detected noise caused by the sensors. More clearly, the motors rely on avoidance sensors through the *Braitenberg* algorithm. In this algorithm, the controller computes the summation values of the proximity sensor on the right side of the E-Puck, $\sum PS_{Right} \epsilon \{0, 1\}$, and

Table A.2: The 20 E-Puck robots positions in the arena for the loading and collecting task. The orientation of all E-Puck robots initiated facing the north, which is the -Z direction.

Robot Name	**X-Axis(m)**	**Z-Axis(m)**
E-Puck 0	0.2	4.8
E-Puck 1	0.35	4.8
E-Puck 2	0.5	4.8
E-Puck 3	0.65	4.8
E-Puck 4	0.8	4.8
E-Puck 5	0.2	4.6
E-Puck 6	0.35	4.6
E-Puck 7	0.5	4.6
E-Puck 8	0.65	4.6
E-Puck 9	0.8	4.6
E-Puck 10	0.2	4.4
E-Puck 11	0.35	4.4
E-Puck 12	0.5	4.4
E-Puck 13	0.65	4.4
E-Puck 14	0.8	4.4
E-Puck 15	0.2	4.2
E-Puck 16	0.35	4.2
E-Puck 17	0.5	4.2
E-Puck 18	0.65	4.2
E-Puck 19	0.8	4.2

Table A.3: The 8 containers positions in the arena for the loading and collecting task.

Robot Name	**X-Axis(m)**	**Z-Axis(m)**
Container 0	2.8	2.8
Container 1	2.8	2.2
Container 2	2.2	2.8
Container 3	2.2	2.2
Container 4	2.5	2.9
Container 5	2.5	2.1
Container 6	2.9	2.5
Container 7	2.1	2.5

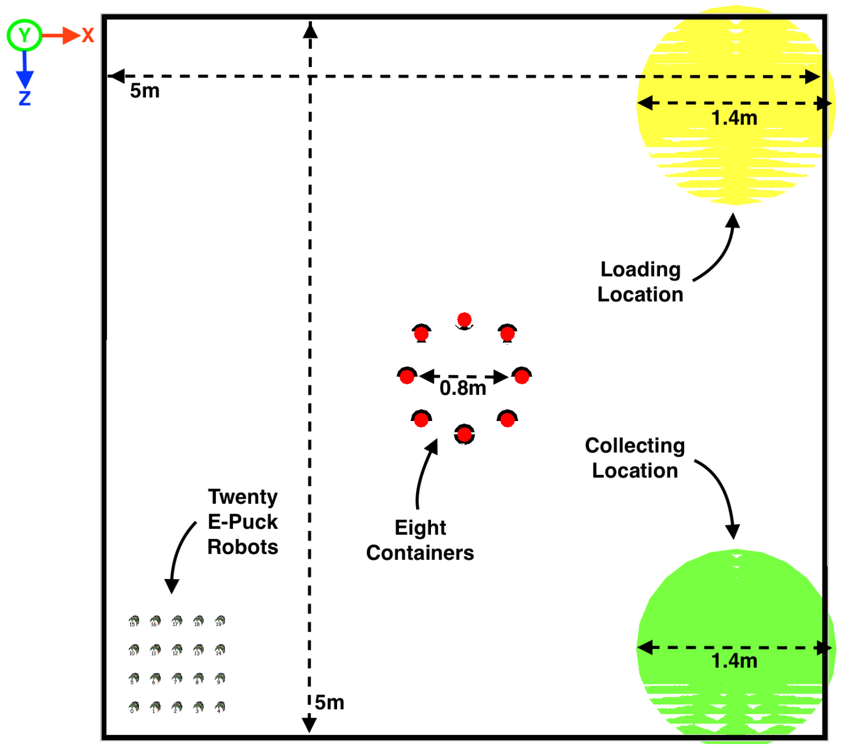


Figure A.2: The 20 E-Puck robots location in the arena for the loading and collecting task. The two circles at the top and bottom right corners represents the loading and collecting location for the swarm task. Containers are located in the middle of the arena. The measurement is in meter. The X, Y and Z axes at the top left corner represent the orientation of the arena in the VRML standard. From the Webots simulation platform.

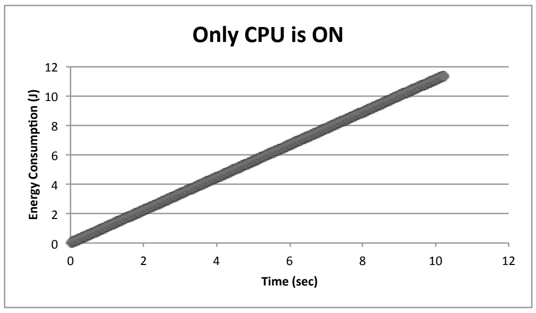
Table A.4: E-Puck proximity sensor simulation experiment. This shows how many times the E-Puck avoids the obstacle to either the right or left side. The simulation experiment sample was based on 100 running times.

Side	**Times**
Right Side	44 times out of 100
Left Side	56 times out of 100

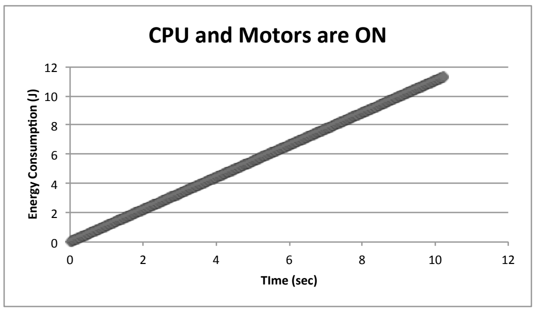
the left side, $\sum PS_{Left} \epsilon \{0,1\}$. Then they are multiplied with the motor $MaxSpeed$ (1000 units). As the output is going to be less than the max speed, an $Offset$ value will be added to each side. This offset value is measured as 500 units that balance the robot speed before and after detecting an obstacle and keep the robot at max speed as long as possible. The output values are then passed on to the corresponding motor on the other side of the robot, M_{Right} and M_{Left}, ,as shown in Equation A.1 and Equation A.2.

$$M_{Right} = Offset + \left(\sum PS_{Left} \times MaxSpeed\right) \tag{A.1}$$

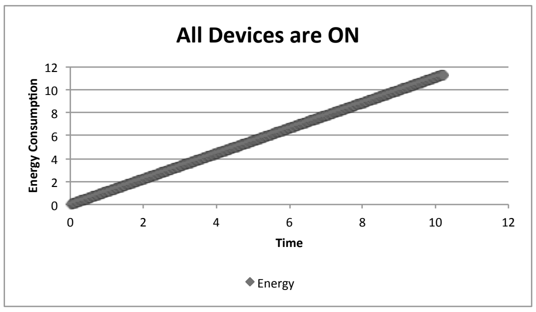
$$M_{Left} = Offset + \left(\sum PS_{Right} \times MaxSpeed\right) \tag{A.2}$$



(a) The energy expended of the robot's CPU over ten seconds time.



(b) The energy expended while the CPU and motors are powered-on over ten seconds time.



(c) The energy expended while all devices are powered-on over ten seconds time.

Figure A.3: Validating robot's energy expended while the CPU, motors and all devices are powered-on Respectively.

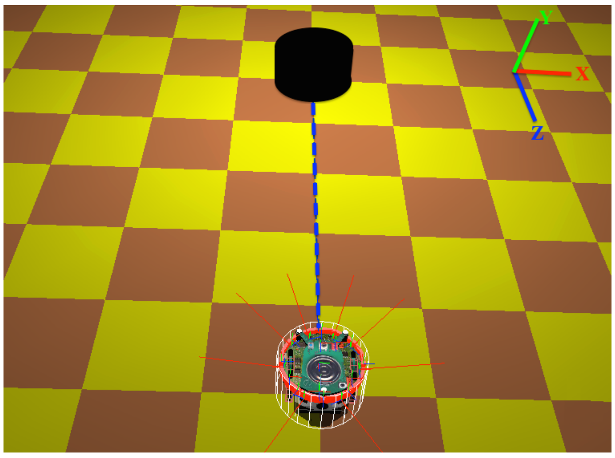


Figure A.4: E-Puck robot placed at the same orientation as the cylindrical obstacle. Red lines represent the sensors' orientation. The X, Y and Z axis at the top right corner represent the orientation of the arena in the VRML standard, as described above.

With the existence of such noise in each sensor (Figure A.5), each motor receives a different value. This causes the robot to demonstrate a sort of randomness which persistently manoeuvres the movement of the robot slightly to the left and right. This is what causes the robot to randomly decide which side it has to turn to in order to avoid the obstacle when it reaches it.

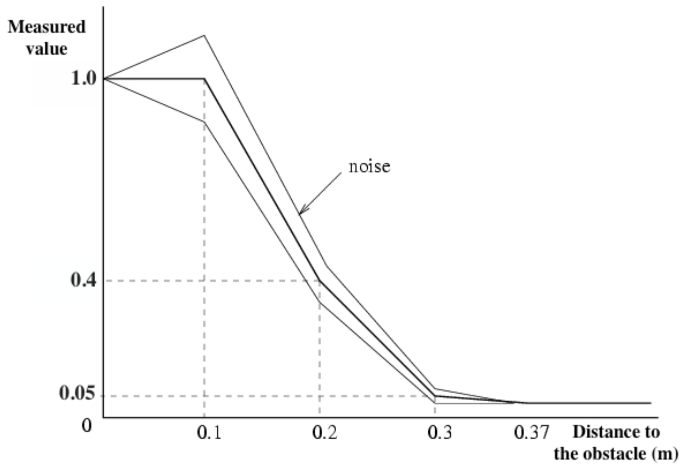


Figure A.5: The E-Puck sensor responses against wall/obstacles with respect to the noise. The response value is scaled between 0.0 and 1.0, where 1.0 represents something to avoid and 0.0 represents nothing to avoid. Adapted from Cyberbotics (2013).

A.5 Checking Validity of the Navigation Device

This experiment was set to check the validity of the navigation device accuracy. A couple of tests were observed, each consisting of one stationary E-Puck robot. The simulation was set to run for ten seconds and then the results were observed, and showed that, in this test, ten seconds were enough to give convincing results. Otherwise, more observations will be considered in the future.

During the first test, the navigation device was set to hundred percent accuracy (no offset), which means that the navigation readings are totally accurate. For the duration of ten seconds, the results observed that the navigation device readings for the x-axis and z-axis are compatible with the time, where the coordinations of the robot are indicating to a stationary location as it supposed to be during this test.

In the second test, the accuracy of the navigation device was set to one metre noise. The choice of the noise is due to the simulated size of the arena, 1.5×1.5 meter square, where that distinguish wither the robot is in the centare or the side of the arena. The outcome of the robot's controller reading of axis-x and axis-z after the noise was injected are represented in Figure A.6.

The existence of this noise on the robot's x and z locations, while the robot is stationary, causes error in the controller while computing the robot's coordination. That means the robot could be within a radius of one meter away from its exact position, Figure A.7. By considering the size of the arena, robots will have a problem while implementing the exogenous fault detection technique for validating their health status, therefore a problem for undertake their task.

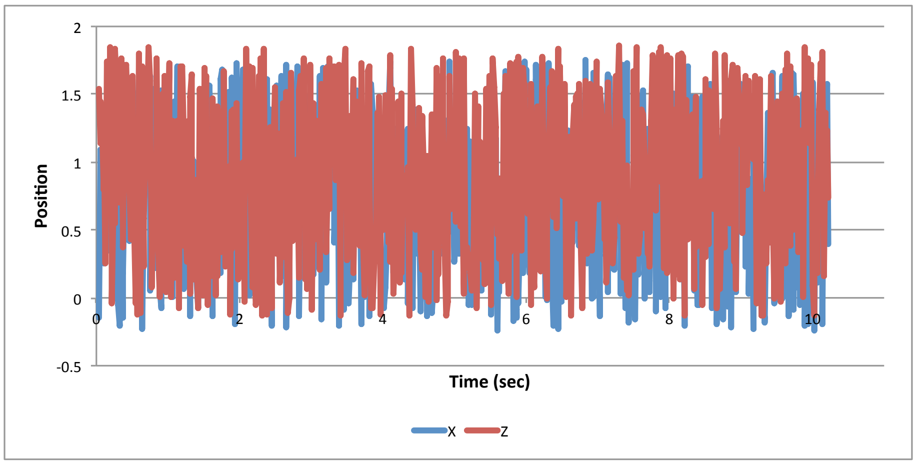


Figure A.6: The second test for checking the validity of the navigation device in the E-Puck robot. It shows that while the robot is stationary, the navigation device readings for the x-axis and z-axis are providing noises. This is due to the injected noise to the navigation device. The back line represents the navigation reading of x-axis and the front line represents the navigation reading of z-axis.

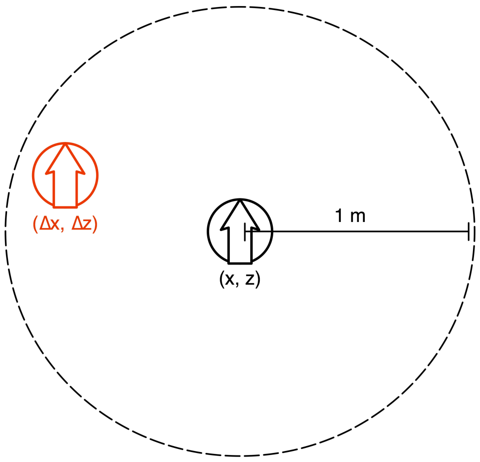


Figure A.7: The possible location of the robot after the noise was injected in the navigation device. The small circle with arrow represents the robot and its orientation. The dot circle shows the robot's noise range. Axes x and z represents the expected position of the robot in the arena. The Δx and Δz represents a possible location within one meter radius away from the exact position. The location of the navigation device was set in the middle of the robot's body, therefore the position was measured from the centre of the robot.

During the research simulation experiments, a simulation failure in the navigation device will be considered as changing the accuracy of the navigation reading by injecting a noise to the navigation device in some robots.

Appendix B

Swarm Size Evaluation Experiments for Forging Task

Additional results are summarised here from the simulated experiments, for evaluating swarm size, that has been collected during experiments in Chapter 4.

B.1 Results from Swarm Size Evaluation for Forging Task

This section shows detailed results from the simulation experiments in Chapter 4 that are related to the swarm size evaluation experiment. Table B.1, Table B.2, Table B.3, Table C.2, Table B.5, Table B.6, Table B.7, Table B.8, Table B.9, Table B.10, Table B.11, Table B.12 and Table B.13 summarises results of the time, distance-travelled and energy-expended with different swarm sizes, once per robot then followed with per swarm run, in the simulation experiments.

B.2 Results from a Previous Work an Earlier Version of Webots

These results has been observed from the first evaluation simulated experiment of swarm sizes on a previous Webots version released (v7.4.1). The different is that they enhanced the multi-threading in their software that causes a quick processing, therefore, a fast robots movement.

It was noted from the first run that the total time spent by the swarm robotic took around five minutes to finish the task. During this period, swarm robotic have consumed over two thousands joules, in addition to that, they have travelled about eighty meters

Table B.1: Results from the swarm with one robot for foraging task per robot. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	417.33	45.30	460.89
2	332.65	33.88	367.42
3	371.96	38.46	410.82
4	187.50	16.57	207.22
5	431.43	44.88	476.47
6	367.93	38.23	406.36
7	405.22	42.47	447.53
8	431.43	45.55	476.45
9	238.90	22.82	263.95
10	265.12	27.13	292.88

Table B.2: Results from the swarm with 6 robots for foraging task per robot. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	80.65	8.14	89.07
2	54.44	5.42	60.13
3	135.08	12.91	149.18
4	66.54	6.84	73.49
5	149.19	16.15	164.73
6	49.40	4.90	54.57
7	47.38	4.66	52.35
8	125.00	12.05	138.06
9	67.54	6.86	74.60
10	80.65	7.92	89.07

in total. Table B.14 shows all the results from different runs with the swarm sizes of 6, 10 and 20 robots.

However, during multiple runs with different swarm size, it was observed that whenever the swarm size get larger, the swarm task finish faster.

Table B.3: Results from the swarm with 6 robots for foraging task per swarm run. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	80.65	48.83	534.40
2	54.44	32.49	360.78
3	135.08	77.47	895.09
4	66.54	41.04	41.04
5	149.19	96.90	988.36
6	49.40	29.38	327.42
7	47.38	27.95	314.07
8	125.00	72.30	828.34
9	67.54	41.14	447.60
10	80.65	47.53	534.44

Table B.4: Results from the swarm with 10 robots for foraging task per robot. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	44.05	4.49	48.65
2	39.95	3.84	44.13
3	87.06	9.31	96.13
4	39.95	3.84	44.13
5	56.34	5.97	62.21
6	71.70	7.68	79.17
7	53.26	5.44	58.83
8	50.19	5.23	55.43
9	39.95	3.87	44.13
10	67.60	7.03	74.65

Table B.5: Results from the swarm with 10 robots for foraging task per swarm run. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	44.05	44.90	486.49
2	39.95	38.39	441.31
3	87.06	93.12	961.26
4	39.95	38.39	441.31
5	56.34	59.71	622.11
6	71.70	76.77	791.68
7	53.26	54.42	588.26
8	50.19	52.31	554.29
9	39.95	38.73	441.31
10	67.60	70.32	746.52

Table B.6: Results from the swarm with 14 robots for foraging task per robot. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	30.25	2.82	33.41
2	53.43	5.38	9.00
3	47.38	4.84	52.33
4	34.28	3.51	37.86
5	61.50	6.16	67.91
6	65.53	6.70	72.35
7	59.48	6.18	65.68
8	43.35	4.31	47.87
9	43.35	4.42	47.87
10	42.34	4.34	46.76

Table B.7: Results from the swarm with 14 robots for foraging task per swarm run. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	30.25	39.42	467.75
2	53.43	75.33	826.01
3	47.38	67.72	732.56
4	34.28	49.10	530.02
5	61.50	86.28	950.69
6	65.53	93.81	1012.95
7	59.48	86.57	919.49
8	43.35	60.33	670.21
9	43.35	61.86	670.23
10	42.34	60.80	654.63

Table B.8: Results from the swarm with 15 robots for foraging task per robot. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	26.64	2.53	29.43
2	27.66	2.74	30.56
3	40.98	3.82	45.26
4	33.81	3.36	37.34
5	44.05	4.42	48.65
6	45.07	4.30	49.78
7	35.86	3.54	39.60
8	45.07	4.59	49.78
9	36.88	3.74	40.73
10	26.64	2.50	29.43

Table B.9: Results from the swarm with 15 robots for foraging task per swarm run. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	26.64	38.00	441.41
2	27.66	41.10	458.37
3	40.98	57.28	678.97
4	33.81	50.35	560.06
5	44.05	66.32	729.75
6	45.07	64.44	746.77
7	35.86	53.12	594.02
8	45.07	68.85	746.67
9	36.88	56.17	610.95
10	26.64	37.48	441.43

Table B.10: Results from the swarm with 16 robots for foraging task per robot. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	35.86	3.60	39.60
2	30.74	3.11	33.95
3	31.76	3.11	35.08
4	35.86	3.62	39.53
5	31.76	3.11	35.08
6	30.74	3.04	33.95
7	32.78	3.03	36.21
8	27.66	2.63	30.56
9	30.74	3.09	33.95
10	37.90	3.74	41.86

Table B.11: Results from the swarm with 16 robots for foraging task per swarm run. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	35.86	57.63	633.58
2	30.74	49.82	543.15
3	31.76	49.74	561.25
4	35.86	57.93	632.47
5	31.76	49.74	561.25
6	30.74	48.63	543.15
7	32.78	48.51	579.41
8	27.66	42.00	488.98
9	30.74	49.40	543.16
10	37.90	59.78	669.77

Table B.12: Results from the swarm with 20 robots for foraging task per robot. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	36.30	3.52	40.08
2	31.26	3.06	34.52
3	34.28	3.42	37.86
4	31.26	3.07	34.52
5	34.28	3.41	37.86
6	31.26	3.16	34.52
7	34.28	3.47	37.85
8	37.30	3.64	41.19
9	30.25	2.97	33.41
10	29.24	2.56	32.30

Table B.13: Results from the swarm with 20 robots for foraging task per swarm run. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	36.30	70.44	801.66
2	31.26	61.13	690.34
3	34.28	68.31	757.14
4	31.26	58.39	690.40
5	34.28	68.20	757.13
6	31.26	63.14	690.36
7	34.28	69.42	757.09
8	37.30	72.83	823.85
9	30.25	59.35	668.10
10	29.24	51.10	645.94

Table B.14: Results shows the total time (seconds), total energy-expended (joules) and total distance-travelled (meters) to clear the arena of 6 objects. In all runs objects were located at identical position.

Swarm Size	**Run**	**Simulation Time**	**Energy Expended**	**Distance Travelled**
6	1	304.22	2022.12	79.35
	2	305.89	2033.3	77.22
	3	175.9	1169.34	42.82
	4	466.77	3102.32	125.45
	5	367.7	2444.57	84.38
	6	293.57	1951.28	76.58
	7	343.2	2281.16	89.83
	8	547.58	3639.43	147.79
	9	276.5	1837.81	71.98
	10	222.61	1479.7	56.5
10	1	228.34	2529.57	97.63
	2	221.66	2455.65	94.85
	3	392.51	4348.09	173.58
	4	136.29	1509.82	58.43
	5	152.86	1693.52	64.32
	6	238.32	2640.26	100.27
	7	133.87	1483.11	56.35
	8	173.36	1920.61	72.48
	9	182.85	2025.62	78.66
	10	119.42	1323.11	49.11
20	1	135.71	3007.63	101.43
	2	103.86	2301.51	79.99
	3	97.36	2157.28	80.49
	4	119.36	2644.99	94.19
	5	129.6	2872.15	94.79
	6	111.25	2465.14	88.5
	7	209.68	4646.3	168.11
	8	135.71	3007.63	101.43
	9	103.86	2301.51	79.99
	10	97.36	2157.28	80.49

Appendix C

Swarm Foraging Task with Failures

Additional results are summarised here from the simulated experiments, after implementing failures to individual robots, and after applying the exogenous failure detection model and mitigation procedures, that has been collected during experiments in Chapter 5.

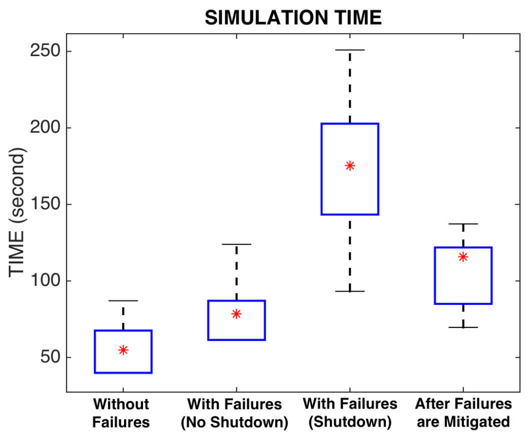
C.1 Results Collected Before and After Shutdown Faulty Robots

The demonstrated experiments in Section 5.3 and Section 5.4 experienced the simulation swarm before and after injecting multiple swarm robots with failures, then shutdown the faulty robots. However, although leaving these faulty robots operational (without shut them down) could cause a farther harm to the swarm system, the observed results shows a significant increases in the distance-travelled results, Figure C.1. The collected results is summarised in Table C.1. The distance-travelled was computed based on the robot's position from the navigation device (not a modulator sensor). Therefore, the distance-travelled that was collected was to high.

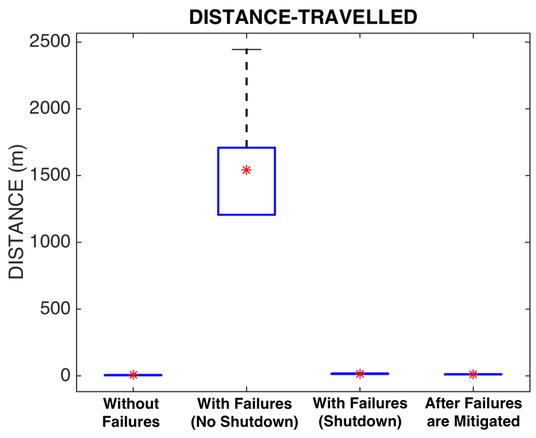
Table C.2 summarises results for a swarm with 10 operational robots with no failures. Table C.3 summarises the results after 3 robots been injected with failures but they are still moving (no failure was shutdown). Table C.4 summarises results after the 3 failed robots are shutdown and become stationary in the arena. Finally, Table C.5 summarises results after mitigating the 3 faulty robots by shut them down and push them to the side of the arena.

Table C.1: The performance of the simulation time, distance-travelled and energy-expended of the simulation swarm after with 3 failures. The collected results represents purpose for shutdown failed robots, compared to the swarm behaviour with wandering faulty robots. In addition to the behaviour of the swarm after mitigating failures by pushing faulty robots to the side of the arena. The simulation experiment was set to run for 10 times.

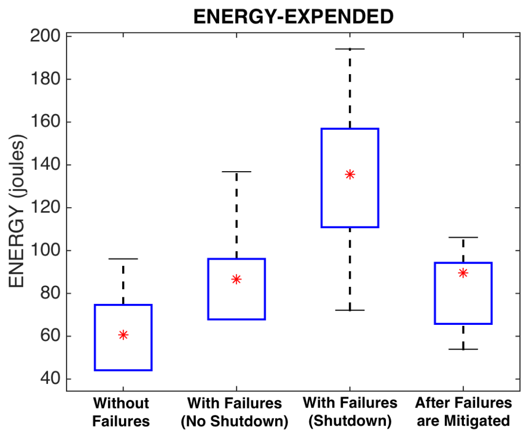
Metric	10 Robots No Failures No Mitigation	10 Robots 3 Failures No Shutdown	10 Robots 3 Failures 3 Shutdown	10 Robots 3 Failures 3 Mitigated
Time from simulation (Seconds)	55.01	78.35	175.34	115.85
Distance-Travelled (Meter)	5.67	1540.92	15.93	12.65
Energy-Expended (Joules)	60.75	86.52	135.64	89.61



(a) The impact of failures on the simulation time.



(b) The impact of failures on the distance-travelled.



(c) The impact of failures on the energy-expended.

Figure C.1: Box plots showing a comparison in the performance of the simulated swarm without failure, before shutdown the faulty robots, after shutdown faulty robots and after mitigating failures. The data represented in these diagrams correspond to the simulation time, distance-travelled and energy-expended. The box represents the degree of dispersion (spread) and skewness in the data. Stars represent the mean, the top line represents the longest running time and the bottom line represents the shortest running time during this experiment.

Table C.2: Results from the swarm with 10 robots for foraging task per robot. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	44.05	4.49	48.65
2	39.95	3.84	44.13
3	87.06	9.31	96.13
4	39.95	3.84	44.13
5	56.34	5.97	62.21
6	71.70	7.68	79.17
7	53.26	5.44	58.83
8	50.19	5.23	55.43
9	39.95	3.87	44.13
10	67.60	7.03	74.65

Table C.3: Results from the swarm with 10 robots for foraging task per robot. A number of 3 robots has failures in the navigation device, and they are not shutdown. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	61.46	1206.46	67.86
2	61.46	1206.46	67.86
3	63.50	1255.56	70.13
4	79.89	1575.09	88.21
5	105.49	2063.33	116.47
6	61.46	1206.46	67.86
7	123.92	2444.50	136.82
8	87.06	1709.01	96.12
9	61.46	1206.46	67.86
10	77.84	1535.84	85.96

Table C.4: Results from the swarm with 10 robots for foraging task per robot. A number of 3 robots has failures in the navigation device, and they been shutdown. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	159.78	12.04	123.63
2	176.16	18.31	136.25
3	217.12	21.39	167.88
4	124.96	13.44	96.67
5	250.91	18.33	194.13
6	193.57	15.29	149.76
7	191.52	17.88	148.11
8	143.39	15.20	110.92
9	93.22	10.00	72.15
10	202.78	17.47	156.89

Table C.5: Results from the swarm with 10 robots for foraging task per robot. A number of 3 robots has failures in the navigation device, and they been mitigated. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	72.74	9.91	56.30
2	85.02	10.61	65.80
3	137.25	14.29	106.15
4	240.67	20.71	186.09
5	109.60	11.93	84.79
6	69.66	9.33	53.93
7	121.89	12.91	94.28
8	92.19	11.36	71.33
9	113.70	12.33	87.95
10	115.74	13.08	89.53

Appendix D

Swarm Size Evaluation for Loading and Collecting Task

Additional results are summarised here from the simulated experiments, for evaluating swarm size, that has been collected during experiments in Chapter 6.

D.1 Results from Swarm Size Evaluation for Loading and Collecting Task

This section shows detailed results from the simulation experiments in Chapter 6 that are related to the swarm size evaluation experiment for loading and collecting containers task. Table D.1, Table D.2, Table D.3, Table D.4, Table D.5, Table D.6, Table D.7, Table D.8, Table D.9, Table D.10, Table D.11, Table D.12, Table D.13, Table D.14, Table D.15, Table D.16. summarises results of the time, distance-travelled and energy-expended with different swarm sizes, once per robot then followed with per swarm run, in the simulation experiments.

Table D.1: Results from the swarm with 4 robots for transporting containers to loading then collecting locations per robot.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	3828	189	4238
2	4155	209	4601
3	3192	167	3534
4	4019	203	4450
5	5504	278	6094
6	5210	217	5521
7	4707	218	4803
8	3737	233	4705
9	3690	225	5228
10	5103	237	5238

Table D.2: Results from the swarm with 4 robots for transporting containers to loading then collecting locations per swarm run.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	3828	757	16953
2	4155	835	18404
3	3192	668	14135
4	4019	811	17800
5	5504	1111	24377
6	5210	866	22083
7	4707	872	19212
8	3737	933	18819
9	3690	898	20910
10	5103	949	20950

Table D.3: Results from the swarm with 6 robots for transporting containers to loading then collecting locations per robot.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	2377	112	2632
2	2395	118	2652
3	2625	134	2906
4	2442	123	2704
5	2333	117	2583
6	2524	131	2862
7	2417	125	2772
8	2608	136	2912
9	2589	134	2826
10	2557	132	2835

Table D.4: Results from the swarm with 6 robots for transporting containers to loading then collecting locations per swarm run.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	2377	673	15792
2	2395	708	15912
3	2625	802	17436
4	2442	737	16221
5	2333	701	15497
6	2524	783	17169
7	2417	748	16630
8	2608	815	17474
9	2589	804	16954
10	2557	792	17011

Table D.5: Results from the swarm with 8 robots for transporting containers to loading then collecting locations per robot.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	1044	57	1155
2	1228	60	1360
3	1516	76	1678
4	1278	68	1415
5	1401	70	1551
6	1104	62	1461
7	1063	57	1159
8	1181	60	1352
9	1152	60	1342
10	1453	71	1653

Table D.6: Results from the swarm with 8 robots for transporting containers to loading then collecting locations per swarm run.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	1044	454	9242
2	1228	480	10882
3	1516	604	13425
4	1278	548	11317
5	1401	564	12411
6	1104	496	11686
7	1063	452	9270
8	1181	481	10814
9	1152	480	10736
10	1453	569	13224

Table D.7: Results from the swarm with 10 robots for transporting containers to loading then collecting locations per robot.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	1250	62	1384
2	1240	65	1373
3	1451	69	1606
4	1438	71	1593
5	979	49	1084
6	980	57	1312
7	1195	63	1377
8	1331	62	1371
9	1417	60	1237
10	1190	60	1397

Table D.8: Results from the swarm with 10 robots for transporting containers to loading then collecting locations per swarm run.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	1250	616	13843
2	1240	647	13729
3	1451	695	16063
4	1438	710	15926
5	979	492	10839
6	980	572	13122
7	1195	633	13767
8	1331	618	13710
9	1417	596	12367
10	1190	596	13972

Table D.9: Results from the swarm with 12 robots for transporting containers to loading then collecting locations per robot.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	1107	58	1225
2	1009	53	1117
3	1372	68	1519
4	1063	54	1177
5	1150	57	1273
6	1234	62	1408
7	1162	61	1302
8	1249	62	1337
9	1245	61	1315
10	1152	60	1327

Table D.10: Results from the swarm with 12 robots for transporting containers to loading then collecting locations per swarm run.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	1107	692	14705
2	1009	632	13409
3	1372	817	18226
4	1063	651	14118
5	1150	678	15281
6	1234	749	16898
7	1162	732	15629
8	1249	742	16043
9	1245	733	15783
10	1152	724	15919

Table D.11: Results from the swarm with 14 robots for transporting containers to loading then collecting locations per robot.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	1337	64	1481
2	1162	56	1287
3	1138	56	1260
4	1253	62	1388
5	1311	63	1452
6	1241	60	1397
7	1287	63	1375
8	1301	60	1343
9	1335	60	1357
10	1228	61	1380

Table D.12: Results from the swarm with 14 robots for transporting containers to loading then collecting locations per swarm run.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	1337	902	20732
2	1162	787	18013
3	1138	785	17641
4	1253	866	19431
5	1311	886	20324
6	1241	835	19562
7	1287	876	19256
8	1301	834	18803
9	1335	842	19000
10	1228	847	19325

Table D.13: Results from the swarm with 16 robots for transporting containers to loading then collecting locations per robot.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	1346	66	1491
2	867	43	960
3	1443	71	1597
4	1745	82	1932
5	1447	71	1602
6	1337	65	1452
7	1009	68	1426
8	1522	62	1348
9	1643	63	1414
10	1008	60	1601

Table D.14: Results from the swarm with 16 robots for transporting containers to loading then collecting locations per swarm run.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	1346	1048	23850
2	867	686	15357
3	1443	1130	25558
4	1745	1312	30916
5	1447	1131	25632
6	1337	1036	23228
7	1009	1095	22812
8	1522	985	21562
9	1643	1000	22621
10	1008	965	25615

Table D.15: Results from the swarm with 20 robots for transporting containers to loading then collecting locations per robot.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	2032	88	2250
2	1759	81	1948
3	1359	62	1504
4	2126	100	2355
5	1581	75	1751
6	1736	79	2033
7	1793	84	1885
8	1447	86	1987
9	1626	82	1911
10	1402	87	1931

Table D.16: Results from the swarm with 20 robots for transporting containers to loading then collecting locations per swarm run.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	2032	1764	44997
2	1759	1625	38964
3	1359	1250	30086
4	2126	2001	47094
5	1581	1505	35020
6	1736	1575	40662
7	1793	1679	37705
8	1447	1715	39737
9	1626	1639	38219
10	1402	1732	38626

Appendix E

Swarm Experiment for Loading and Collecting Task with Failures

Additional results are summarised here from the simulated experiments, for loading and collecting containers, that has been collected during experiments in Chapter 7.

E.1 Results from Swarm with Loading and Collecting Task without Failures

This section shows detailed results from the simulation experiments in Chapter 7 that are related to the swarm behaviour without failures for loading and collecting containers task. These results are evaluated and compared to the results that has been collected and demonstrated in the next section after the simulation swarm was injected with failures.

Table E.1 summarises results of the time, distance-travelled and energy-expended for a swarm with 12 operational robots for loading and collecting containers, in the simulation experiments.

Table E.2 summarises results of the time, distance-travelled and energy-expended for a swarm with 8 operational robots for loading and collecting containers, in the simulation experiments.

Table E.3 summarises results of the time, distance-travelled and energy-expended for a swarm with 6 operational robots for loading and collecting containers, in the simulation experiments.

Table E.1: Results from the swarm with 12 operational robots for loading and collecting containers task (per robot). The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	1107	58	1225
2	1009	53	1117
3	1372	68	1519
4	1063	54	1177
5	1150	57	1273
6	1234	62	1408
7	1162	61	1302
8	1249	62	1337
9	1245	61	1315
10	1152	60	1327

Table E.2: Results from the swarm with 8 operational robots for loading and collecting containers task (per robot). The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	1044	57	1155
2	1228	60	1360
3	1516	76	1678
4	1278	69	1415
5	1401	70	1551
6	1104	62	1461
7	1063	57	1159
8	1181	60	1352
9	1152	60	1342
10	1453	71	1653

E.2 Results from Swarm with Loading and Collecting Task with Failures

This section shows detailed results from the simulation experiments in Chapter 7 that are related to the swarm behaviour with the existence failures for loading and collecting containers task. The swarm behaviour represented in the following results are corresponding to the changed of the number of failures. Where the swarm once was injected with 4 failures and with 6 failures respectively.

Table E.4 summarises results of the time, distance-travelled and energy-expended for a swarm with 12 robots of which 4 robots have failure in the navigation device, for loading and collecting containers task in the simulation experiments.

Table E.3: Results from the swarm with 6 operational robots for loading and collecting containers task (per robot). The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	2377	112	2632
2	2395	118	2652
3	2625	134	2906
4	2442	123	2704
5	2333	117	2583
6	2524	131	2862
7	2417	125	2772
8	2608	136	2912
9	2589	134	2826
10	2557	132	2835

Table E.5 summarises results of the time, distance-travelled and energy-expended for a swarm with 12 robots of which 6 robots have failure in the navigation device, for loading and collecting containers task in the simulation experiments.

Table E.4: Results from the swarm with 12 robots for loading and collecting containers task. A number of 4 robots have failures in the navigation device, and they been shutdown. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**	**Average Time Before Failures**
1	1720	69	1905	721
2	2020	80	2238	823
3	2431	83	2694	728
4	1901	66	2106	395
5	1795	80	1988	846
6	2006	82	2299	931
7	2151	73	2232	809
8	2298	80	2226	889
9	1808	80	2265	634
10	1893	81	2320	977

E.3 Results from Swarm with Loading and Collecting Task with Failure Mitigation

This section shows detailed results from the simulation experiments in Chapter 7 that are related to the swarm behaviour with the existence failures for loading and collecting containers task. The swarm behaviour represented in the following results are corresponding to the changes in the swarm after mitigating faulty robots.

Table E.5: Results from the swarm with 12 robots for loading and collecting containers task. A number of 6 robots have failures in the navigation device, and they been shutdown. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**	**Average Time Before Failures**
1	3364	89	3729	770
2	1938	68	2147	770
3	2757	87	3055	889
4	2984	82	3307	736
5	3023	95	3350	879
6	2847	86	3155	925
7	3323	93	2867	919
8	2763	100	2940	965
9	2905	101	2913	977
10	2239	76	2967	785

Table E.6 summarises results of the time, distance-travelled and energy-expended for a swarm with 12 robots of which 4 faulty robots are mitigated, for loading and collecting containers task in the simulation experiments.

Table E.7 summarises results of the time, distance-travelled and energy-expended for a swarm with 12 robots of which 6 faulty robots are mitigated, for loading and collecting containers task in the simulation experiments.

Table E.6: Results from the swarm with 12 robots for loading and collecting containers task. A number of 4 faulty robots have been mitigated and pushed down to the side of the arena. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	1306	61	1672
2	1295	63	1664
3	1176	57	1576
4	1416	66	1828
5	1465	64	1846
6	1188	58	1567
7	1347	63	1685
8	1303	61	1643
9	1278	63	1645
10	1219	61	1636

Table E.7: Results from the swarm with 12 robots for loading and collecting containers task. A number of 6 faulty robots have been mitigated and pushed down to the side of the arena. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	1985	66	2171
2	1978	70	2136
3	1445	57	1633
4	1561	54	1711
5	1758	67	1926
6	1681	58	1807
7	1532	54	1692
8	1902	65	2136
9	1908	67	2156
10	1473	58	1604

Appendix F

Swarm Experiment from Reusing Faulty robots

Additional results are summarised here from the simulated experiments, for loading and collecting containers, that has been collected during experiments in Chapter 8.

F.1 Results from Swarm with Loading and Collecting Task without Failures

This section shows detailed results from the simulation experiments in Chapter 7 that are related to the swarm behaviour without failures for loading and collecting containers task. These results are evaluated and compared to the results that has been collected and demonstrated in the next section after the simulation swarm was injected with failures.

Table F.1 summarises results of the time, distance-travelled and energy-expended for a swarm with 12 operational robots for loading and collecting containers, in the simulation experiments.

Table F.2 summarises results of the time, distance-travelled and energy-expended for a swarm with 8 operational robots for loading and collecting containers, in the simulation experiments.

Table F.3 summarises results of the time, distance-travelled and energy-expended for a swarm with 6 operational robots for loading and collecting containers, in the simulation experiments.

Table F.1: Results from the swarm with 12 operational robots for loading and collecting containers task (per robot). The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	1107	58	1225
2	1009	53	1117
3	1372	68	1519
4	1063	54	1177
5	1150	57	1273
6	1234	62	1408
7	1162	61	1302
8	1249	62	1337
9	1245	61	1315
10	1152	60	1327

Table F.2: Results from the swarm with 8 operational robots for loading and collecting containers task (per robot). The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	1044	57	1155
2	1228	60	1360
3	1516	76	1678
4	1278	69	1415
5	1401	70	1551
6	1104	62	1461
7	1063	57	1159
8	1181	60	1352
9	1152	60	1342
10	1453	71	1653

F.2 Results from Swarm with Loading and Collecting Task with Failures

This section shows detailed results from the simulation experiments in Chapter 7 that are related to the swarm behaviour with the existence failures for loading and collecting containers task. The swarm behaviour represented in the following results are corresponding to the changed of the number of failures. Where the swarm once was injected with 4 failures and with 6 failures respectively.

Table F.4 summarises results of the time, distance-travelled and energy-expended for a swarm with 12 robots of which 4 robots have failure in the navigation device, for loading and collecting containers task in the simulation experiments.

Table F.3: Results from the swarm with 6 operational robots for loading and collecting containers task (per robot). The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	2377	112	2632
2	2395	118	2652
3	2625	134	2906
4	2442	123	2704
5	2333	117	2583
6	2524	131	2863
7	2417	125	2772
8	2608	136	2912
9	2589	134	2826
10	2557	132	2835

Table F.5 summarises results of the time, distance-travelled and energy-expended for a swarm with 12 robots of which 6 robots have failure in the navigation device, for loading and collecting containers task in the simulation experiments.

Table F.4: Results from the swarm with 12 robots for loading and collecting containers task. A number of 4 robots have failures in the navigation device, and they been shutdown. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**	**Average Time Before Failures**
1	1720	69	1905	721
2	2020	80	2238	823
3	2431	83	2694	728
4	1901	66	2106	395
5	1795	80	1988	846
6	2006	82	2299	931
7	2151	73	2232	809
8	2298	80	2226	889
9	1808	80	2265	634
10	1893	81	2320	977

F.3 Results from Swarm with Loading and Collecting Task with Failure Mitigation

This section shows detailed results from the simulation experiments in Chapter 7 that are related to the swarm behaviour with the existence failures for loading and collecting containers task. The swarm behaviour represented in the following results are corresponding to the changes in the swarm after mitigating faulty robots.

Table F.5: Results from the swarm with 12 robots for loading and collecting containers task. A number of 6 robots have failures in the navigation device, and they been shutdown. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	Simulation Time	Distance Travelled	Energy Expended	Average Time Before Failures
1	3364	89	3729	770
2	1938	68	2147	770
3	2757	87	3055	889
4	2984	82	3307	736
5	3023	95	3350	879
6	2847	86	3155	925
7	3323	93	2867	919
8	2763	100	2940	965
9	2905	101	2913	977
10	2239	76	2967	785

Table F.6 summarises results of the time, distance-travelled and energy-expended for a swarm with 12 robots of which 4 faulty robots are mitigated, for loading and collecting containers task in the simulation experiments.

Table F.7 summarises results of the time, distance-travelled and energy-expended for a swarm with 12 robots of which 6 faulty robots are mitigated, for loading and collecting containers task in the simulation experiments.

Table F.6: Results from the swarm with 12 robots for loading and collecting containers task. A number of 4 faulty robots have been mitigated and used as communication bridge. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	Simulation Time	Distance Travelled	Energy Expended
1	1434	59	1633
2	1377	58	1552
3	1570	64	1747
4	1585	62	1762
5	1536	62	1763
6	1448	62	1622
7	1585	66	1705
8	1435	59	1627
9	1423	60	1645
10	1540	65	1714

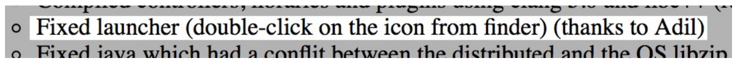
Table F.7: Results from the swarm with 12 robots for loading and collecting containers task. A number of 6 faulty robots have been mitigated and used as communication bridge. The simulation time is in seconds, the distance-travelled is in meters and the energy-expended is in joules.

Run	**Simulation Time**	**Distance Travelled**	**Energy Expended**
1	1966	72	2139
2	1909	68	2083
3	1784	64	1938
4	1745	51	1908
5	1430	53	1573
6	1738	62	1908
7	1816	66	1907
8	1708	61	1896
9	1589	60	1646
10	1588	59	1852

Appendix G

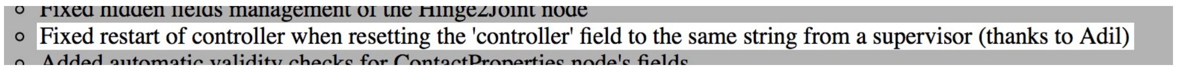
A Contribution to Solving Issues In Webots

This is the authors contribution for solving and helping detecting a couple of problems in Webots through previous releases, that been discussed in Section 3.4.



- Compiled controllers, libraries and plugins using clang
- Fixed launcher (double-click on the icon from finder) (thanks to Adil)
- Fixed java which had a conflit between the distributed and the OS libzip

Figure G.1: A contribution for solving an issue in Webots v7.3.0. From the ChangeLog page in Cyberbotics (2013).



- Fixed hidden fields management of the Hinge2Joint node
- Fixed restart of controller when resetting the 'controller' field to the same string from a supervisor (thanks to Adil)
- Added automatic validity checks for ContactProperties node's fields

Figure G.2: A contribution for solving an issue in Webots v7.4.0. From the ChangeLog page in Cyberbotics (2013).

References

Adams, D. (1976). The granulomatous inflammatory response. a review. *The American Journal of Pathology*, 84(1):164.

Avižienis, A. (1967). Design of fault-tolerant computers. In *Proceedings of the November 14-16, 1967, Fall Joint Computer Conference*, pages 733–743. ACM.

Balch, T. and Hybinette, M. (2000). Social potentials for scalable multi-robot formations. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 73–80. IEEE.

Bayindir, L. and Şahin, E. (2007). A review of studies in swarm robotics. *Turkish Journal of Electrical Engineering*, 15(2):115–147.

Bell, G., Parisi, A., and Pesce, M. (1995). The virtual reality modeling language: version 1.0 specification, http://vrml.wired.com/vrml.tech/vrml10-3.html.

Beni, G. and Wang, J. (1993). Swarm intelligence in cellular robotic systems. In *Robots and Biological Systems: Towards a New Bionics?*, pages 703–712. Springer.

Bjerknes, J. D. and Winfield, A. F. (2013). On fault tolerance and scalability of swarm robotic systems. In *Distributed Autonomous Robotic Systems*, pages 431–444. Springer.

Bloch, H. P. and Geitner, F. K. (1990). *An introduction to machinery reliability assessment*. Van Nostrand Reinhold New York.

Bonabeau, E., Dorigo, M., and Theraulaz, G. (1999). *Swarm Intelligence - From Natural to Artificial Systems*. Number 1. Oxford University Press, USA.

Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2012). Swarm robotics: A review from the swarm engineering perspective. *IRIDIA, CoDE, Université Libre de Bruxelles, Brussels, Belgium, Tech. Rep. TR2012-014*.

Brutschy, A., Garattoni, L., Brambilla, M., Francesca, G., Pini, G., Dorigo, M., and Birattari, M. (2014). The tam: abstracting complex tasks in swarm robotics research. *Swarm Intelligence*, pages 1–22.

Choset, H. and Nagatani, K. (2001). Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *IEEE Transactions on robotics and automation*, 17(2):125–137.

Christensen, A. L., O'Grady, R., and Dorigo, M. (2009). From fireflies to fault-tolerant swarms of robots. *IEEE Transactions on Evolutionary Computation*, 13(4):754–766.

Chtanova, T., Schaeffer, M., Han, S.-J., van Dooren, G. G., Nollmann, M., Herzmark, P., Chan, S. W., Satija, H., Camfield, K., Aaron, H., et al. (2008). Dynamics of neutrophil migration in lymph nodes during infection. *Immunity*, 29(3):487–496.

Cianci, C. M., Raemy, X., Pugh, J., and Martinoli, A. (2007). Communication in a swarm of miniature robots: The e-puck as an educational tool for swarm robotics. In *Swarm Robotics*, pages 103–115. Springer.

Cyberbotics (2013). www.cyberbotics.com.

Desjeux, P. (2004). Leishmaniasis: current situation and new perspectives. In *Comparative immunology, microbiology and infectious diseases*, volume 27, pages 305–318. Elsevier.

Dorigo, M., Floreano, D., Gambardella, L., Mondada, F., Nolfi, S., Baaboura, T., Birattari, M., Bonani, M., Brambilla, M., Brutschy, A., et al. (2012). Swarmanoid: a novel concept for the study of heterogeneous robotic swarms. *IEEE Robotics & Automation Magazine*.

Flugge, A. J., Timmis, J., Andrews, P., Moore, J., and Kaye, P. (2009). Modelling and simulation of granuloma formation in visceral leishmaniasis. In *IEEE Congress on Evolutionary Computation, 2009. CEC'09*, pages 3052–3059. IEEE.

Halavati, R., Shouraki, S. B., Heravi, M. J., and Jashmi, B. J. (2007). An artificial immune system with partially specified antibodies. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, pages 57–62. ACM.

Higgins, F., Tomlinson, A., and Martin, K. M. (2009a). Survey on security challenges for swarm robotics. In *Fifth International Conference on Autonomic and Autonomous Systems, 2009. ICAS'09.*, pages 307–312. IEEE.

Higgins, F., Tomlinson, A., and Martin, K. M. (2009b). Threats to the swarm: Security considerations for swarm robotics. *International Journal On Advances in Security*, 2(2 and 3):288–297.

Hinchey, M. G., Sterritt, R., and Rouff, C. (2007). Swarms and swarm intelligence. *IEEE Computer*, 40(4):111–113.

Humza, R., Scholz, O., Mokhtar, M., Timmis, J., and Tyrrell, A. (2009). Towards energy homeostasis in an autonomous self-reconfigurable modular robotic organism.

In *Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009. COMPUTATIONWORLD'09. Computation World:*, pages 21–26. IEEE.

Ismail, A. R., Bjerknes, J. D., Timmis, J., and Winfield, A. (2015). An artificial immune system for self-healing in swarm robotic systems. In *Information Processing in Cells and Tissues*, pages 61–74. Springer.

Ismail, A. R. and Timmis, J. (2009). Aggregation of swarms for fault tolerance in swarm robotics using an immuno-engineering approach. In *Proceedings of the 9th Annual Workshop on Computational Intelligence (UKCI 2009).*

Ismail, A. R. and Timmis, J. (2010). Towards self-healing swarm robotic systems inspired by granuloma formation. In *Engineering of Complex Computer Systems (ICECCS), 2010 15th IEEE International Conference on*, pages 313–314. IEEE.

Jeanne, R. L. (1986). The evolution of the organization of work in social insects. *Monitore Zoologico Italiano-Italian Journal of Zoology*, 20(2):119–133.

Jennings, N. R. and Wooldridge, M. (1998). *Applications of intelligent agents.* Springer.

Kannan, B. and Parker, L. E. (2007). Metrics for quantifying system performance in intelligent, fault-tolerant multi-robot teams. In *Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on*, pages 951–958. IEEE.

Kernbach, S., Häbe, D., Kernbach, O., Thenius, R., Radspieler, G., Kimura, T., and Schmickl, T. (2013). Adaptive collective decision-making in limited robot swarms without communication. *The International Journal of Robotics Research*, 32(1):35–55.

Khadidos, A., Crowder, R. M., and Chappell, P. H. (2015a). Enhancing exogenous fault detection in swarm robotics by analysing transferable data. In Alford, N., FrÃ, J., et al., editors, *Proceedings of the Eighth Saudi Students Conference in the UK*, pages 385–395. World Scientific.

Khadidos, A., Crowder, R. M., and Chappell, P. H. (2015b). Exogenous fault detection and recovery for swarm robotics. *IFAC-PapersOnLine*, 48(3):2405–2410.

Kitano, H. (2007). Towards a theory of biological robustness. *Molecular systems biology*, 3(1).

Krieger, M. J., Billeter, J.-B., and Keller, L. (2000). Ant-like task allocation and recruitment in cooperative robots. *Nature*, 406(6799):992–995.

Lau, H., Bate, I., Cairns, P., and Timmis, J. (2011). Adaptive data-driven error detection in swarm robotics with statistical classifiers. *Robotics and Autonomous Systems*, 59(12):1021–1035.

Lau, H., Bate, I., and Timmis, J. (2009). An immuno-engineering approach for anomaly detection in swarm robotics. In *Artificial Immune Systems*, pages 136–150. Springer.

Liu, W., Winfield, A., Sa, J., Chen, J., and Dou, L. (2006). Strategies for energy optimisation in a swarm of foraging robots. In *International Workshop on Swarm Robotics*, pages 14–26. Springer.

Liu, W. and Winfield, A. F. (2010). Modeling and optimization of adaptive foraging in swarm robotic systems. *The International Journal of Robotics Research*, 29(14):1743–1760.

Magnenat, S., Rétornaz, P., Noris, B., and Mondada, F. (2008). Scripting the swarm: event-based control of microcontroller-based robots. In *SIMPAR 2008 Workshop Proceedings*.

Martinoli, A., Mondada, F., Correll, N., Mermoud, G., Egerstedt, M., Hsieh, M. A., Parker, L. E., and Støy, K. (2012). *Distributed Autonomous Robotic Systems: The 10th International Symposium*, volume 83. Springer.

Mathews, E., Graf, T., and Kulathunga, K. S. (2012). A bio-inspired coverage and connectivity maintenance algorithm. In *Bio-Inspired Models of Networks, Information, and Computing Systems*, pages 149–154. Springer.

Michel, O. (2004). Webotstm: Professional mobile robot simulation. *arXiv preprint cs/0412052*.

Millard, A. G., Timmis, J., and Winfield, A. F. (2013). Towards exogenous fault detection in swarm robotic systems.

Millard, A. G., Timmis, J., and Winfield, A. F. (2014). Run-time detection of faults in autonomous mobile robots based on the comparison of simulated and real robot behaviour. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pages 3720–3725. IEEE.

Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C., Floreano, D., and Martinoli, A. (2009). The e-puck, a robot designed for education in engineering. In *Proceedings of the 9th conference on autonomous robot systems and competitions*, volume 1, pages 59–65.

Mondada, F. and Michael, B. (2007). E-puck-epfl education robot. *EPFL, Ecublens janvier*.

Murray, L., Liu, W., Winfield, A., Timmis, J., and Tyrrell, A. (2012). Analysing the reliability of a self-reconfigurable modular robotic system. In *Bio-Inspired Models of Networks, Information, and Computing Systems*, pages 44–58. Springer.

Newborough, J. and Stepney, S. (2005). A generic framework for population-based algorithms implemented on multiple fpgas. In *Artificial Immune Systems*, pages 43–55. Springer.

O'Dowd, P. J., Studley, M., and Winfield, A. F. (2014). The distributed co-evolution of an on-board simulator and controller for swarm robot behaviours. *Evolutionary Intelligence*, 7(2):95–106.

Parker, L. E. (1998). Alliance: An architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240.

Parker, L. E. (2001). Evaluating success in autonomous multi-robot teams: experiences from alliance architecture implementations. *Journal of Experimental & Theoretical Artificial Intelligence*, 13(2):95–98.

Parker, L. E. (2012). Reliability and fault tolerance in collective robot systems. *Pan Stanford Publishing.*

Parker, L. E. and Kannan, B. (2006). Adaptive causal models for fault diagnosis and recovery in multi-robot teams. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2703–2710. IEEE.

Patterson, M. R. and Patterson, S. J. (2010). Unmanned systems: An emerging threat to waterside security: Bad robots are coming. In *International Waterside Security Conference (WSS)*, pages 1–7. IEEE.

Peng, J., Wen, M.-f., Xie, G.-q., Zhang, X.-y., and Lin, K.-c. (2012). Coordinated dynamic mission planning scheme for intelligent multi-agent systems. *Journal of Central South University*, 19(11):3170–3179.

Pini, G., Brutschy, A., Scheidler, A., Dorigo, M., and Birattari, M. (2014). Task partitioning in a robot swarm: Object retrieval as a sequence of subtasks with direct object transfer. *Artificial life*, 20(3):291–317.

Ren, H. and Tse, Z. (2012). Investigation of optimal deployment problem in three-dimensional space coverage for swarm robotic system. *Social Robotics*, pages 468–474.

Şahin, E. (2005). Swarm robotics: From sources of inspiration to domains of application. *Swarm Robotics*, pages 10–20.

Şahin, E., Girgin, S., Bayindir, L., and Turgut, A. E. (2008). Swarm robotics. *Swarm Intelligence: Introduction and Applications*, 1(87).

Schut, M. C. (2010). On model design for simulation of collective intelligence. *Information Sciences*, 180(1):132–155.

Shi, Z., Tu, J., Zhang, Q., Liu, L., and Wei, J. (2012). A survey of swarm robotics system. *Advances in Swarm Intelligence*, pages 564–572.

Sooktip, T., Wattanapongsakorn, N., and Coit, D. (2011). System reliability optimization with constraint k-out-of-n redundancies. In *IEEE International Conference on Quality and Reliability (ICQR)*, pages 216–220. IEEE.

Stancliff, S. B., Dolan, J. M., and Trebi-Ollennu, A. (2006). Mission reliability estimation for multirobot team design. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2206–2211. IEEE.

Timmis, J., Andrews, P., and Hart, E. (2010). On artificial immune systems and swarm intelligence. *Swarm Intelligence*, 4(4):247–273.

Verma, V. and Simmons, R. (2006). Scalable robot fault detection and identification. *Robotics and Autonomous Systems*, 54(2):184–191.

Visinsky, M. L., Cavallaro, J. R., and Walker, I. D. (1994). Robotic fault detection and fault tolerance: A survey. *Reliability Engineering & System Safety*, 46(2):139–158.

Webb, B. (2000). What does robotics offer animal behaviour? *Animal Behaviour*, 60(5):545–558.

Winfield, A. F. (2009). Foraging robots. In *Encyclopedia of Complexity and Systems Science*, pages 3682–3700. Springer.

Winfield, A. F. and Nembrini, J. (2006). Safety in numbers: fault-tolerance in robot swarms. *International Journal of Modelling, Identification and Control*, 1(1):30–37.

Yuan, K., Li, Y., and Fang, L. (2007). Multiple mobile robot systems: a survey of recent work. *Acta Automatica Sinica*, 33(8):785.

Zhu, A. and Yang, S. X. (2010). A survey on intelligent interaction and cooperative control of multi-robot systems. In *Control and Automation (ICCA), 2010 8th IEEE International Conference on*, pages 1812–1817. IEEE.