Automatic Control & Systems Engineering.

Autonomous tool sharing between operators using multi-robot system

Sujith Kurian James

September 2022

Supervisor: Dr Roderich Groß

A dissertation submitted in partial fulfilment of the requirements for the degree of MSc in Robotics

EXECUTIVE SUMMARY

INTRODUCTION AND BACKGROUND

Human swarm interaction involves humans being part of a swarm system. It is often considered as one of the best solutions to solve several problems faced by swarm robotics. Although it has immense possibilities, human swarm interaction is a field that has been sparsely explored in the literature.

AIMS AND OBJECTIVES

The aim of the project was to develop such a system for autonomous tool sharing between human operators. The objective was to develop a swarm system that could carry and deliver heterogeneous sets of tools to human operators working in teams based on their requirements. This needed to be executed while maintaining connectivity between all the swarm robots and the human operators.

ACHIEVEMENTS

The developed system succeeded in meeting the set forward requirements. It was demonstrated that the system performed as expected in various scenarios. The system maintained connectivity in most of the scenarios and was able to successfully deliver the right tools to the operators. The effect of variations in different system parameters on the system performance was identified. The ideal number of robots to allocate to the human operators initially was also identified.

CONCLUSIONS AND RECOMMENDATIONS

It was established that the intended system could be developed. Some of the shortcomings of the system and solutions to rectify them were also identified. Further experiments need to be made to gauge the effectiveness of the proposed solutions. The system also needs to be implemented in real robots to evaluate real-world performance.

ABSTRACT

This dissertation contributes to the field of human swarm interaction by developing a system for autonomous tool sharing between human operators. Human swarm interaction has vast possibilities in many fields such as search and rescue missions. They can vastly improve the capabilities of a swarm system.

The idea of a swarm system that carries and shares tools among human operators working in teams has numerous benefits. If multiple tasks are spread around an area, it is a good idea to execute them parallelly in teams. But it has the shortcoming that there may not be enough tools for all the operators. Therefore, there needs to be a system to make sure that the right tools are available to the operator at the right time. This dissertation focuses on developing such a system.

The swarm system was modelled using Supervisory Control Theory (SCT). Modelling using SCT lets the model be used in any kind of robot as the SCT model is not specific to any robot type. The model was developed with heterogeneity in mind. It could handle up to 10 different types of robots each carrying a unique set of tools. The swarm system maintains connectivity between all the robots and the humans. This lets the human operators communicate with each other through the swarm robots. This is especially useful in scenarios where global communication does not exist. Robots can also be exchanged among the teams to deliver the right tools to the right operator.

The developed model was tested in multiple scenarios in simulation and was found to be working as intended. Furthermore, a study was conducted to determine the optimal number of robots of each type to be allocated to each team. Solutions to some of the possible limitations were also presented.

ACKNOWLEDGMENT

I would like to acknowledge and thank all the people who assisted and supported me during this dissertation.

First, I would like to thank my supervisor, Dr. Roderich Groß, who provided his guidance and expertise for this dissertation. I thank you for your time and supervision throughout the span of the project.

I would also like to express my gratitude to Mr. Genki Miyauchi, who provided invaluable assistance to me during the project and helped me to get up to speed with the technical side of this project.

I am also grateful to my friends and family for their continuous support and help.

Finally, I would like to thank everyone who contributed directly or indirectly towards the completion of this dissertation.

Table of Contents

1 Introduction 1

1.1 Background and motivation 1

1.2 Aims & objectives 2

1.2.1 Aims 2

1.2.2 Objectives 2

1.3 Project management 2

1.4 Outline 5

2 Background and related works 6

2.1 Design methods used in swarm robotics 6

2.1.1 Behaviour-based design methods 6

2.1.2 Automatic design methods 11

2.2 Connectivity preserving swarm robotics 12

2.3 Heterogeneous swarm robots 13

2.4 Human-Swarm Interaction (HSI) 15

2.5 Supervisory Control Theory (SCT) 16

2.5.1 Generators 17

2.5.2 Free behaviour models 17

2.5.3 Control specifications 18

2.5.4 Supervisor synthesis 18

3 Problem formulation and methodology 20

3.1 Problem formulation 20

3.2 Methodology 22

3.2.1 SCT model 26

3.2.2 Artificial potential field modelling 31

4 Simulation and results **33**

4.1 Simulation environment 33

4.1.1 Automatic sending and requesting of robots 35

4.2 Results and findings 35

4.2.1 Impact of number of followers and types of robots 39

4.2.2 Impact of stationary vs moving connector chain 39

4.2.3 Impact of obstacles 40

4.2.4 Impact of specification vector 40

4.2.5 Sample snapshots from trials 44

4.3 Limitations and solutions 48

4.4 Summary 51

4.5 Extension to real robots 51

5 Conclusions and future work **52**

5.1 Future work 52

References **53**

Appendix A Message structure used for communication **58**

List of figures

Figure 1: WBS diagram for the dissertation 3

Figure 2: Gantt chart with the original timeline (blue), delays (red) and uncompleted tasks (yellow) highlighted.....4

Figure 3: Example of free behaviour models (a and b) and control specification (c) for a simple robot. 18

Figure 4: Example for the environment in which the swarm system operates. 21

Figure 5: An instance prior (left) and after (right) a robot joins a chain at its tail.....24

Figure 6: An instance prior (left) and after (right) a robot leaves a chain and joins a team ...24

Figure 7: Original SCT models used by Miyauchi [3]. 27

Figure 8: Modified SCT model used in the dissertation. 28

Figure 9(a-c): Illustration of a shortcoming in the condition used by Miyauchi. 29

Figure 9(d-f): Illustration of a shortcoming in the condition used by Miyauchi. 30

Figure 10: The simulation environment.....34

Figure 11: Simulation environment for the trials to study the effect of specification vector 41

Figure 12: Box and whisker plot showing the task completion time for different ratios of number of robots of each type 42

Figure 13: Box and whisker plot showing the total distance travelled by followers for different ratios of number of robots of each type 43

Figure 14: Box and whisker plot showing the total distance travelled by travellers for different ratios of number of robots of each type 43

Figure 15: Snapshots from the simulation trial 5.....45

Figure 16: Snapshots from the simulation trial 32. 46

Figure 17: Snapshots from the simulation trial 41. 47

Figure 18: Example for a scenario in which the chain might break from the team. 49

Figure 19: Example for a situation in which an obstacle might block a traveller robot 50

Figure 20: Adding a redundant connector to make the system resistant to failure of a connector robot 50

List of tables

Table 1(a): Results from the simulation.....37

Table 1(b): Results from the simulation 38

Table 2: Summary of results from the trials to study the effect of specification vector with initial team robot ratio of 1:1 41

Table 3: Summary of results from the trials to study the effect of specification vector with initial team robot ratio of 1:3 42

Table 4: Contents of the original message structure and its description.....58

Table 5: Contents of the updated message structure and its description 59

Chapter 1

Introduction

1.1 Background and motivation

Swarm robotics is a domain that stemmed from the observation of social behaviours in nature. It is the ability of simple individual robots, to form groups and work collectively to perform complex tasks, that are impossible or difficult to be performed by an individual robot. Swarm robotics also brings along several advantages over using complex robots such as robustness, scalability, and flexibility [1].

Numerous research has been carried out in swarm robotics. Human swarm interaction is a relatively new discipline within swarm robotics that is currently gaining a lot of attention. Human swarm interactions can vastly improve the capabilities of a swarm system. Tasks such as search and rescue missions can be made more efficient if a human operator can dynamically influence the behaviour of the system based on the requirements. Another situation which can benefit from a human swarm interaction is autonomous tool sharing. Often a set of tasks can be completed faster if they are performed parallelly. However, the limited availability of tools can pose a barrier to this. An autonomous human swarm interaction system would be able to resolve these. The system can autonomously carry and deliver the required tools to the human operators based on their requirements. Furthermore, if they are operating in an environment without any means for global communication, the swarm system can also be employed as a communication network.

However, at present, research into human swarm interaction is sparse. This dissertation aims to contribute to this by developing an autonomous swarm system for sharing of tools between two human operators. The system is modelled using Supervisory Control Theory (SCT) and validated in simulation. An approach for implementing the system in a real robot system is also discussed. This dissertation expands on the work by Miyauchi et al. in [2], [3].

1.2 Aims & objectives

1.2.1 Aims

The aim of the project is to develop a mobile, ground-based, heterogeneous, swarm system to carry and deliver tools between two operators that move in an environment. The robots are autonomous and deliver the required tools to the operators based on their requests. In addition to this, the robots shall maintain a stable communication network between the operators by forming a dynamic chain. This will be formally modelled using supervisory control theory (SCT). The project is evaluated in simulation and a pathway to implement this in a real-world environment was also considered.

1.2.2 Objectives

Basic Objectives

1 Collect and study relevant literature on:
   a. Supervisory Control Theory (SCT)
   b. Human-Swarm Interaction (HSI)
   c. Heterogeneous swarm systems
   d. Communication methods between robots
2 Formulate the problem and establish metrics for performance evaluation
3 Create a simulation environment to test the robot system
4 Expand the algorithm developed by Miyauchi et al. [2], [3] to a heterogeneous system
5 Design a formal controller for tool exchange and connectivity preservation using SCT
6 Verify the developed system in simulation environment and analyse its performance

Advanced Objectives

7 Expand the algorithm to implement it in a real-world robot system

1.3 Project management

Efficient project management is key to the successful completion of the dissertation. At first, a Work Breakdown Structure (WBS) was generated based on the requirements. The WBS diagram is given in figure 1. This was followed by the creation of a Gantt chart. The Gantt chart is shown in figure 2.

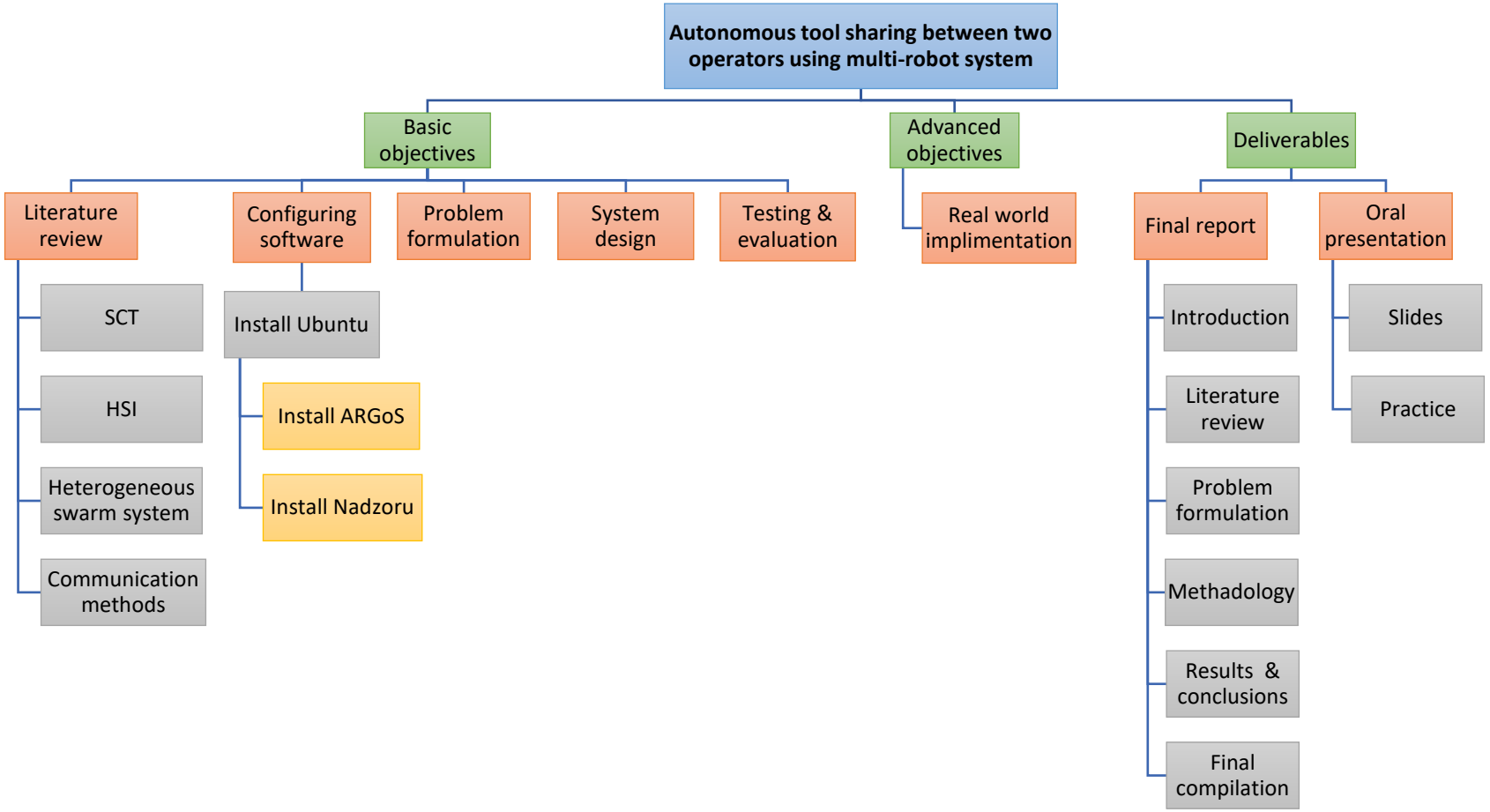


Figure 1: WBS diagram for the dissertation

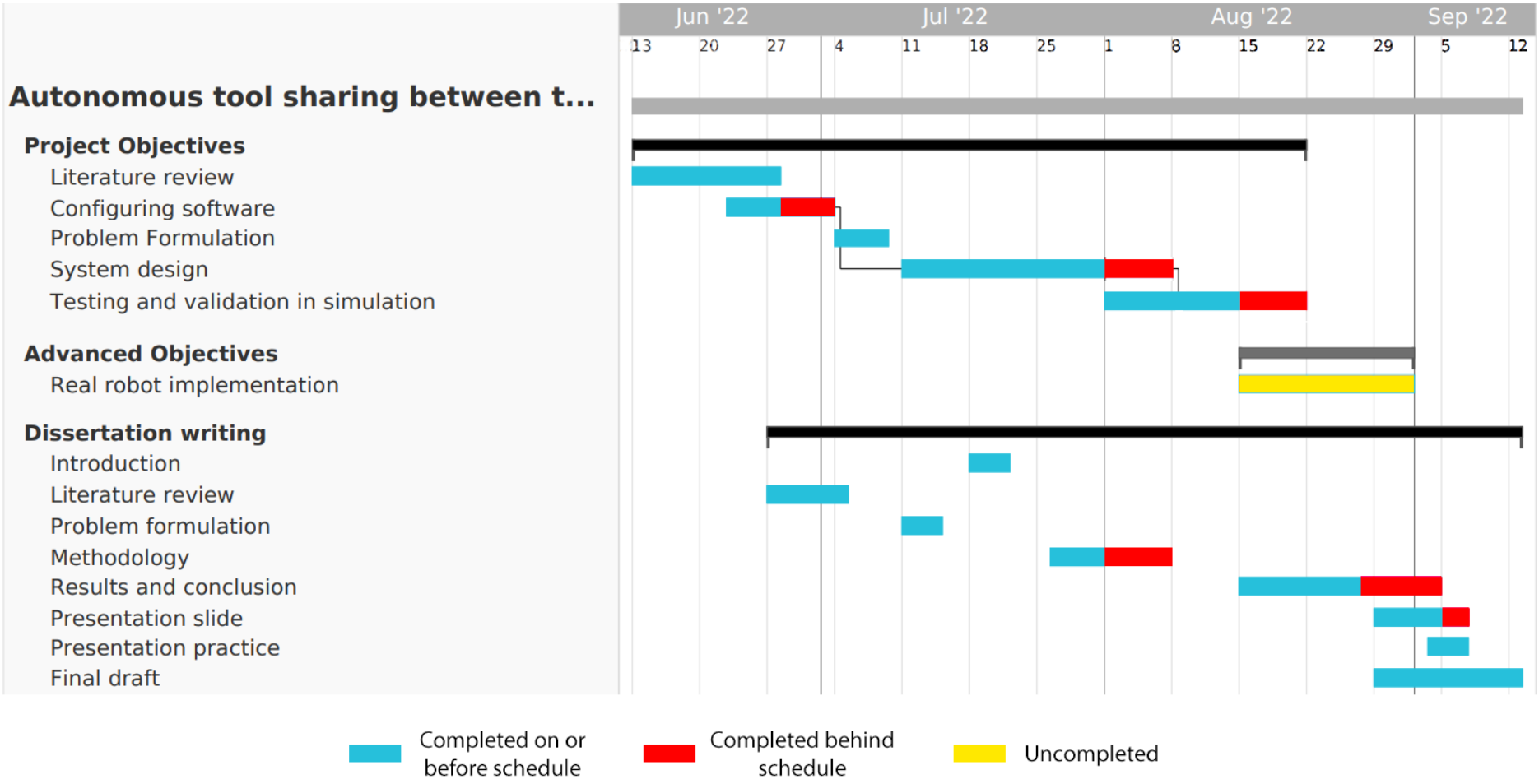


Figure 2: Gantt chart with the original timeline (blue), delays (red) and uncompleted tasks (yellow) highlighted

Due to specific circumstances, some of the tasks deviated from the original plan. These are highlighted in red colour in figure 2. One of the key complications faced during the project was the difficulty in configuring the required software. The git repository for one of the software was no longer available. Therefore, alternative methods had to be found to install it. Also, getting familiar with the software applications took more time than initially anticipated. This resulted in a delay to complete some tasks. To compensate for this, more time was spent on the project including weekends. As the project went ahead, it was realised that the implementation on real-life robots was more complicated than expected. This combined with the lack of time, resulted in the advanced objectives not being completed.

1.4 Outline

The structure of the report is as follows:

Chapter 2: Background and related works

Previous works related to the different methods for modelling a swarm system, the use of heterogeneous robots in swarm systems, connectivity preservation in swarm robotics, and human swarm interaction are presented in this section. The fundamental concepts of SCT are also presented.

Chapter 3: Problem formulation and methodology

This section formally defines the problem scenario, the methods used to address the problem, and the assumptions used. Details of the system design and algorithms used are also explained. Furthermore, several terminologies and symbols used in the dissertation are also introduced.

Chapter 4: Simulation and results

The results and observations from the simulations are presented in this section. The performance of the system in various scenarios and some of its limitations are analysed.

Chapter 5: Conclusions and future work

The overall findings and achievements of the dissertation are presented here. Potential directions in which the dissertation can be followed up are also discussed.

Chapter 2

Background and related works

This chapter provides insight into some of the theoretical background and previous works in swarm robotics. More specifically, it focuses on formal design methods used in swarm robotics, supervisory control theory and its application in swarm robotics, connectivity preserving swarm robots, heterogeneous robots in swarm robots and human-swarm interaction.

2.1 Design methods used in swarm robotics

Over the years, several approaches have been developed to model a swarm system. In [1], the authors broadly classify the different design methods as:

(1) Behaviour-based design methods
(2) Automatic design methods

2.1.1 Behaviour-based design methods

One of the first attempts of using a behaviour-based model in a multi-agent system is by Reynolds [4]. In his work, the motion of a flock of birds was simulated by programming a set of physics rules for each bird, which determines its motion and interaction with other birds. In a behaviour-based design for swarm robotics, the controller is designed to achieve a defined behaviour of the swarm system at a microscopic or macroscopic level. In the microscopic level modelling, the behaviour of the individual robots is modelled such that a desired collective behaviour of the swarm can be achieved. On the other hand, in the macroscopic approach, the behaviour of the entire system is modelled. Some of the methods used in behaviour-based modelling are as follows:

Probabilistic Finite State Machine (PFSM): In a Finite State Machine (FSM), all the possible states of the robot are defined as a finite number of states with a state transition function which dictates how states evolve in the system. The transition function depends on an input data, usually a sensory input. But in a PFSM, in addition to the sensor input, the state transition function also depends on probability. This probability can be a constant or evolve with time. In [5], the authors compared three mathematical model based multi-target search algorithms using swarm robots with an optimal PFSM based strategy. They found the PFSM based model to be more efficient, stable and easy to implement across all different comparison criteria.

PFSM was also used in the chain formation problem in swarm robotics in [6]. Nouyan et al. used two probability parameters and studied their impact to design controllers to form (i) a linear chain and (ii) and tree-like structured chain. They also found that the generated controllers had good scalability, robustness, and fault tolerance. PFSM was also employed by Liu et al. in [7]. They developed a macroscopic level probabilistic model of adaptive collective forage in a swarm of robots using PFSM and a set of difference equations. They also introduced a concept called 'sub-PFSM' to extend the model to a system of heterogeneous robots. In [8], the authors successfully proved that a macroscopic model developed for self-organized aggregation in swarm robotics using PFSM can correctly predict the final aggregate distribution of the swarm system.

Artificial physics: Artificial physics methods uses virtual forces that follow basic laws of physics to model the controllers. It computes virtual attractive and repulsive forces on the robot. The parameters of the robot such as velocity, the direction of motion, and acceleration are calculated based on the net force on a robot. Artificial physics-based methods are primarily used to model swarms which require formation such as aggregation, and pattern formation. One of the first applications of the physics-based approach was by Khatib in [9]. He used the concept of an artificial potential field to control the motion of a robot manipulator. He defined the desired end-effector position as an attractive pole and the obstacles as a repulsive pole. Later Reif and Wang [10] expanded this concept to a Very Large Scale Robotic (VLSR). This method was called '*social potential'* fields. Similar to [9] the authors used attractive and repulsive forces. Each individual robot calculates the attractive and repulsive forces detected by it and computes the forces and performs actions based on this force. The actions performed by each robot are independent of the other. In [11] the authors developed a new framework called '*physicomimetics'* by extending this concept and introducing a new virtual frictional force called a viscous force to improve the stability of the system. They successfully demonstrated the concept in simulation and a team of seven real robots.

Protoswarm: Protoswarm is a scripting language based on a language called 'proto' [12]. Proto lets the user program the global behaviour of a system and automatically compiles codes to run on the individual robots that result in the desired global behaviour being followed. The global behaviour is transferred into discrete local robots depending on the communication capabilities at the local level. However, the 'proto' language is designed for non-mobile systems. Though, Bachrach et al. [13] extended this concept to form Protoswarm by adding motion capability to the robots. The concept was demonstrated both in simulation and using a group of 40 robots.

Property-driven design: In [14] the authors proposed a new method called 'property-driven design' to model a swarm system. This follows a top-down approach as opposed to the usual bottom-up approach as it models the system as a whole by describing the swarm with a set of desired properties. This was implemented in four phases. In phase (i), the desired properties of the system are defined. In phase (ii), a model of the system was developed iteratively till it satisfies all the desired properties. In phase (iii), the model is used to implement the system in simulation. Observations from the simulation are fed back into the model to improve it. In phase (iv), the model is implemented in real robots. In the paper, the authors used the method to create an aggregation behaviour. The system was successfully implemented in a real system made up of 10 e-puck robots.

Neural networks: In [15] a novel method for foraging with swarm robotics is proposed. The authors combined a virtual pheromone concept ([16], [17], [18]) with a neural network. When a robot produces a pheromone, a neuron is activated based on the density of the pheromone. The neuron then diffuses its output to its neighbouring neurons. Once a pheromone is produced, it evaporates (intensity reduces over time), thus reducing its density and updating the neural network. The model was tested in simulation using a group of 20 robots.

Unified Modelling Language (UML): UML is a modelling language primarily used for object-oriented design. It is a graphical software that lets the user model the structural (static) and behavioural (dynamic) properties of the system [19]. One can visualise the various interactions, the state transitions, and so on using UML, thus helping one to create a more accurate model of the system. The usage of UML in swarm robotics is sparse. Although, the authors of [20] used UML to model 'granuloma formation', an immune system response. Their aim was to analyse how the immune system adapts when the individual immune cells die. Based on their observations, they developed two strategies to address the situation when the battery of a robot runs out, thus introducing fault tolerance in the swarm system.

Behaviour-Data Relations Modelling Language (BDRML): In complex swarm systems, UML is unable to fully express the interactions between the robots and the environment. To address this, Pitonakova et al. [21] proposed a new modelling language called BDRML. In BDRML the behaviours of the robots, the data that the robots use, and the interactions between them are modelled and represented explicitly. This enables more control over the communication between the robots and the environment. The model can be represented both visually and textually. The authors also gave an example of BDRML application by using it to

develop robot control algorithms for collective foraging and task allocation in a swarm robotic system.

Petri Net (PN): PN is a formal method to model the flow of information and control in a system primarily used to model and simulate Discrete Event Systems (DES). PNs have a larger representational power than FSMs as they can represent languages that cannot be represented by FSMs. Pereira et al. [22] expanded the generic PN by taking into account the robot actions and sensor readings and called it the 'Executable Petri Nets' (EPN). The authors demonstrated the capabilities of EPN with two case studies: (i) A wireless connected swarm and (ii) transportation of a virtual payload between two rooms connected by a corridor.

Supervisory Control Theory (SCT): SCT, introduced by Ramadge and Wonham [23], is a method used to automatically create supervisors for a controller. In SCT the robotic system as a whole is considered as a discrete event system. In SCT, all the capabilities of the system and the desired system behaviours are specified formally and the SCT algorithm formulates a supervisor that completely or mostly satisfies the required behavioural properties. Lopes et al. in [24] adapted SCT to model a controller for segregation, aggregation, object clustering, and group formation case in swarm robotics. The authors also mentioned about the synthesis of three types of supervisors (monolithic, modular, and local modular) that can be modelled using SCT. They demonstrated the controller by implementing it in real swarm robots using e-puck and kilobot robots.

Mirzaei et al. in [25] further improved the capability of the SCT framework developed in [24] by adding probability and time constraints to the events and called it probabilistic timed Supervisory Control Theory (ptSCT). They developed a software tool to design the SCT model that can automatically generate controller code that can be used in ARGoS [26] platform. They validated it by conducting a foraging case study in simulation using the ARGoS and evaluated its performance against various test conditions.

Jayasiri et al. [27] developed an SCT model for a fussy DES to address the uncertainties associated with real-world DES. In this, the events were modified to incorporate the fuzzy nature of the system. They implemented the system in simulation and real-world systems for a navigational problem where the robot had to navigate to a goal position through several obstacles. The system performed satisfactorily against the defined performance metrics. However, the system only used a single pioneer 3 DX robot for the experiments.

In [28], Dulce-Galindo et al. used SCT to model an autonomous navigation system for multiple robots. The system had an open-loop behaviour implemented by 8 automata. 6 supervisors were synthesised from these. Each robot had access to the environment map with respect to its coordinates and boundaries. However, the positions of the obstacles were unknown. They were also able to relay their global position in the environment. The authors also proposed a centralised scheduler that can autonomously allocate tasks between the robots.

The authors in [29] proposed an online supervisory control for DES using a limited lookahead policy for a multi-robot warehouse automation problem. An online supervisory control using a lookahead policy [30] can incorporate a large and time-varying DES. In a lookahead policy, the controller generates a limited-step-ahead projection of the system's behaviour after each occurrence of an event and selects the appropriate control action based on the projected behaviour. They demonstrated this method in simulation and in a real-world experiment. However, the authors did not provide enough details on how the simulation or the experiment was conducted.

Chanyoung yu et al. [31] proposed a method to model a heterogeneous multi-robot system for use in agricultural fields using SCT. The heterogeneous system consisted of a leader UGV (Unmanned Ground Vehicle) and three follower UAVs (Unmanned Aerial Vehicles). The controller model incorporated path following, formation maintenance, obstacle avoidance and movement capabilities. A mission was given to the leader. While the leader performs the mission, the followers form a formation and move along with the leader. When the leader encounters an obstacle, obstacle avoidance was performed while path following and formation maintenance was given a lower priority. The authors validated the experiment in a simulation environment and a real environment.

In [2], Miyauchi et al. used SCT to synthesise a controller for a human-swarm interaction system. The system consisted of two teams each with a human operator accompanied by a set of swarm robots. They moved around the simulation environment performing various tasks. As the operators moved around, a connectivity-preserving chain was formed by the swarm robots between the two operators. The system could also share messages and robots between the two teams. The authors tested their model successfully in the ARGoS simulation platform using 40 e-puck robots.

2.1.2 Automatic design methods

An automatic design method aims to generate the behaviour and/or properties of the swarm automatically with little or no intervention from the developer. This enables faster development of models and more generalised versions of it. Some of the methods used in automatic model generation are given below:

Reinforced learning (RL): RL is a trial and error-based method of designing a swarm model. It employs a system of rewards and penalties [32]. Whenever a robot makes a correct decision which brings it closer to the desired properties, it is given a positive reward. When it deviates from the required properties, it gets penalised. The idea is that through multiple iterations, the system can converge to an optimal design model that satisfies the requirements. Even though extensive theoretical studies can be found about the use of RL in multi-robot systems, the real-life application of RL in swarm robotics is limited. One such study by Mataric [33] formulated an RL-based model for foraging using a multi-robot system. The model was validated using a group of four mobile robots.

Evolutionary Robotics (ER): ER [34] uses evolutionary computation techniques [35], [36] which are based on Darvin's concepts of evolution. This is also an iterative process. In the first step, a set of behaviours (generated randomly or based on prior knowledge) is evaluated against a fitness function. The top behaviours are then selected and some mutation is applied to them. These behaviours are then again evaluated against the fitness function and the process is repeated. In each iteration, only a set of the best behaviours are carried on to the next population. Over time, a consensus might be achieved on the behaviour. These are usually executed at the individual robot level. In the case of a swarm system, the learning can be shared among all the robots and evaluated against a common fitness function. Several studies can be found in the literature employing ER.

Baldassarre et al. in [37] designed a controller through 30 evolutionary runs that successfully achieved coordinated motion in a linear, physically connected robot system. At first, the controller was evolved in simulation and the best evolved controller was tested in four real linearly connected s-bots. The s-bots were able to complete the tasks without the need for any modifications to the controller. In [38] the authors employed a two-phase evolutionary approach to develop a neural network-based model to create individual and social learning mechanisms for robots. In [39] Groß et al. used evolutionary learning to produce solitary and group transport behaviours in a swarm robot system through repeated simulation runs.

ALLIANCE: ALLIANCE is a behaviour-based architecture developed by Parker in [40] and further improved in [41]. ALLIANCE is a modelling method that focuses on fault tolerance and adaptivity. It can be employed in large-scale heterogeneous multi-robot systems. The robots are designed individually using a behaviour-based method with several task-achieving behaviours (motivational behaviours) per robot. The behaviours can be modelled as low level or high level. The low-level behaviours correspond to survival behaviours such as obstacle avoidance while high-level behaviours are related to goals such as exploration. A low-level behaviour can be suppressed by a high-level behaviour depending on the situation. Fault tolerance is achieved through the interaction of motivational behaviour among the robots. This allows a robot to take over tasks from other robots in case of a failure. In [41] Parker extended this further and introduced a new architecture called L-ALLIANCE by introducing a method for automated learning for the control parameter for the motivational behaviour. To validate the method introduced in [41], the control strategy developed using L-ALLIANCE was implemented in a heterogeneous multi-robot system made up of two R-2 robots and one Genghis-II robot.

Learning momentum: Learning momentum is a method developed by Clark et al [42] to introduce online learning ability to adapt the behaviour parameters in a reactive robotic control system. In [43] the authors integrated this concept to a software architecture called 'MissionLab' and studied the effect of implementing learning momentum in simulated and real robot systems. However, they found out that. even though the robots performed the tasks successfully in both simulation and real world, it took longer to complete the tasks compared to conventional strategies used at the time. The authors further extended the learning momentum framework in [44] to multi-robot system and studied its effects through simulation.

2.2 Connectivity preserving swarm robotics

Connectivity-preserving swarm robotics is a well-studied topic in the literature. Some of the benefits of connectivity preserving swarm robots include sharing of information, path formation, and reducing travel costs. Some of the studies into connectivity-preserving swarm robotics are given in the following:

[45] formulates a method for connectivity preservation among homogeneous multi-robot systems without the need for explicit communication between them. The authors implemented this by creating a sensing area for each robot, beyond which the robot is not able to detect another robot. The motion of each robot is modelled in such a way that, each robot will have

at least one robot in its sensing range. However, this has a major drawback. If one of the robots fails, then the movement of the swarm will be halted. Also, even though the authors claim that this method avoids any deadlock in a real-life scenario, they haven't proved this using real robots.

In [46] the authors present an approach for deploying a large-scale swarm robot system with connectivity preservation. The authors modelled the swarm to form a logical tree topology and prevent the splitting of the tree. The tasks are spaced over an area and the swarm robots have to navigate to the area to perform the tasks while maintaining the tree network. They proposed two different algorithms and compared them in simulation. They also validated the solutions using a group of 9 Khepera IV robots.

A few more studies [47], [48], [49], and [2] dealing with connectivity-preserving robots are given in more detail in the following sections.

2.3 Heterogeneous swarm robots

The majority of the literature on swarm robotics is based on homogeneous robots. However, employing heterogeneous robots can massively increase the capabilities of the swarm system. For example, the swarm system could be made of different kinds of robots which can perform different unique tasks. Also, the robots can be of different make and from different manufacturers. This section provides some insight into heterogeneous swarm robots.

The advantages of using a heterogeneous swarm system were demonstrated in [50]. However, the authors used homogeneous robots and gave them heterogeneous behaviours. They ran simulation experiments for an aggregation case study and compared the performance of the homogeneous and heterogeneous robots. They concluded that, even in the extreme scenario, the heterogeneous swarm outperformed the homogeneous swarms.

'Swarmanoid' [51] is a project aimed at developing heterogeneous swarm robots. It consists of about 60 autonomous robots of three different types: foot-bots, hand-bots, and eye-bots. The primary objective of the project is to design, implement and control the distributed robotic system with robustness and scalability. This is also one of the first studies to design a swarm system operating in a 3-dimensional environment.

Many architectural frameworks have been proposed for designing swarm robotic systems. ALLIANCE [40], is one such framework aimed at building swarm systems with fault tolerance

and cooperative control (see section 2.1.2 for more details). ACTRESS [52] is another such framework. Although not exclusively built for heterogeneous systems, the generalised architecture of the frameworks enables any kind of robot to be added to the system, if a proper communication method can be devised for the robot.

In [53], the authors developed a behaviour-based swarm system made of heterogeneous robots. The aim was to develop an efficient task allocation strategy for the swarm robots. They used an auction-based task assignment strategy for the swarm. They demonstrated it in a multi-robot scenario where the robots were tasked with cleaning an environment. Both simple and complex tasks were studied. The same algorithm performed all the tasks in both cases. The task allocation strategy also had a linear computational cost with respect to the number of robots in the system.

Prorok et al [54] proposed a method for fast and efficient distribution of robots in a heterogeneous swarm system to perform certain tasks. They modelled each robot as different species each with different capabilities. Robots with different capabilities are required to complete each task. The controller was designed through the optimisation of an analytical function.

[47] describes a navigation method for a heterogeneous swarm. The heterogeneity is defined as each robot having a different velocity, acceleration, sensing distance and sensing region. The system is also able to maintain connectivity. Some constraints were made to the capabilities of the leader depending on the capabilities of the followers. This is to ensure that a spanning tree with the leader as the root can be constructed and maintained, thus maintaining connectivity. The authors validated the method in simulation and with real robots.

In [48], the authors introduced a new approach for a heterogeneous system to dynamically deploy robots to complete multiple tasks. The heterogeneity is characterised by each robot having a unique capability and different capacities for a particular capability. The system was also modelled to maintain connectivity between the robots. Different to [54], the tasks are dynamic and are not known in advance. The leader obtains details about the task only when they physically reach the task location. This requires dynamic reallocation of the robots to complete the task while still maintaining connectivity.

2.4 Human-Swarm Interaction (HSI)

Human-Swarm Interaction refers to the cooperation between one or more human operators and a swarm robotic system. Involving a human operator in a swarm system can have many benefits such as addressing the limitations of the swarm system, providing additional information to the swarm, or modifying the target tasks [55]. However, this is a domain that has not been studied well.

A detailed survey on HIS is provided in [55]. Particularly, the authors describe how the human operator can interact with the swarm robots and how the operator can influence the swarm. They identified two ways in which the operator can control the swarm; (i) remote interaction and (ii) proximal interaction. In remote interaction, the operator controls the whole swarm (one-to-many), or a selected number of individuals (one-to-one) via wireless communication methods. The primary constraints for this are the bandwidth, latency, and asynchrony limitations of the current swarm systems. An example of remote interaction with a swarm system can be seen in [56], In proximal interaction, the operator shares the environment with the swarm and controls the swarm through methods such as gesture, haptics, or voice commands. However, this capability is limited by the poor sensing capabilities and low computational power of the swarm robots. [57] provides an example of using gestures to control a swarm of Unmanned Arial Vehicles (UAVs).

A description about the ways in which an operator can control a swarm system was presented in [55]. These include (i) controlling the swarm algorithms to change the swarm behaviour, (ii) controlling the swarm parameter setting, (iii) creating virtual or physical modifications to the environment, and (iv) controlling the swarm through selected swarm leaders.

In [58], the authors compared the effect of controlling a swarm system by changing the swarm behaviour and by influencing the environment for a foraging mission with different complexity. Even though they were not able to make a definite comparison between the two methods in terms of performance, they found that control by behaviour modification was easier to be adapted by inexperienced operators.

In [59], a swarm system was designed and optimised to perform a search and localization task in a simulation environment. However, an HSI feature was implemented into the system in which an operator can take control of the robot and control it via teleoperation. A user study

was conducted and found that and system with HSI was able to achieve the task faster compared to the system only employing autonomous abilities.

Walker et al [60] introduced a method to dynamically select leaders from a swarm for human control in a flocking task with varying goal regions. The dynamic selection of a leader, as opposed to the static, one-time, selection of leaders, proved to be more effective in such situations. The user controls the leader using a virtual joystick. The leaders were chosen based on a modified version of the Random Competition Clustering (RCC) algorithm [61].

One potential problem of using a teleoperated control of a leader in a connectivity-preserving swarm system is that the operator may move the leader beyond the sensing range of the follower robots, thus breaking the connectivity. Yang et al addressed this problem in [49] by introducing a method to dynamically modulate the interconnections between the robots. They used a custom potential function to represent the energy excreted on a robot. The authors found that, by setting an upper limit to this energy, local connectivity can be guaranteed. They used this to limit the energy injected by the operator to the leader to prevent disconnection in the swarm system. The system was implemented and validated on a group of real robots made up of 1 leader and 3 slaves.

An idea to form a dynamic chain of connectivity-preserving robots between two human operators is presented in [2]. Two operators (considered as *leaders*) move around in an environment to goal regions to perform a task independently. A chain of robots is formed between the operators as they move around the area. The robots can join or break up from the robot chain. The framework also provides the ability for robots to move along the robot chain from one operator to another operator. Each task requires a certain number of robots to perform it. Therefore, the exchange of robots between tasks was vital. The controllers were modelled using SCT and run in ARGoS simulation package with 40 e-puck robots.

2.5 Supervisory Control Theory (SCT)

This section provides the necessary theoretical background required to understand the concepts of SCT.

SCT is a formal framework to automatically synthesise controllers for Discrete Event Systems (DES). A DES is made up of discrete states and events. Events cause the system to change from one state to the other. The events can be either controllable or uncontrollable. Controllable

events are the command signals produced by the controller such as 'move forward'. Uncontrollable events are the feedback signals generated in the system such as a sensor reading.

While modelling a system through SCT, at first, the *'free behaviour models'* and *'control specifications'* are defined. Free behaviour models represent the full capabilities of the system, i.e., it defines what the system can do. Control specifications define what the system should do. SCT generates *supervisors* that restrict the free behaviour of the system according to the control specifications. The free behaviour models and control specifications are expressed using regular languages. Each event in the DES is represented by a symbol. A finite sequence of such symbols (events) is called a string and a language is a subset of all such strings in a subsystem. A *generator* is used to produce the free behaviour model and control specifications.

2.5.1 Generators

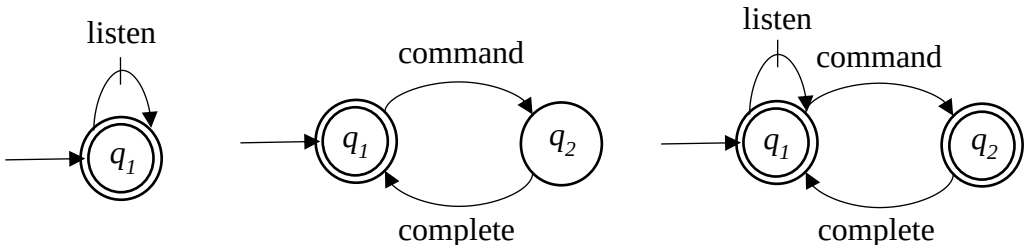
A generator outputs the words that belong to a language. A generator G is defined as a 5-tuple: $G = \{Q, \Sigma, \delta, q_o, Q_m\}$. Where Q is the finite set of all the states, Σ is the finite set of all the events, δ is the partial transition function $\delta: Q \times \Sigma \rightarrow Q$, $q_o \in Q$ is the initial state, and $Q_m \subseteq Q$ is the set of marked states. Events (Σ) can be controllable (Σ_c) or uncontrollable (Σ_u). Therefore, $\Sigma = \Sigma_c \cup \Sigma_u$. δ is called a partial transition function because it may not exist for all $q \in Q$. Marked states represent stable or safe states of the system, typically representing the end of a task.

2.5.2 Free behaviour models

Free behaviour models of a system represent what the system is able to do. i.e., its full capabilities. A system can be formed of m independent free behaviour models that describe a particular system capability. G_i is the generator used to produce free behaviour models where $i \in \{1, 2, \ldots, m\}$. An example of free behaviour models for a simple robot is given in figures 3(a) and 3(b). G_1 represents the ability of the robot to 'listen' (a controllable event) to a signal from an operator. G_2 represents the capability of the robot to perform a task. In state 1 (q_1), the robot is stationary. When it receives a signal from the operator, the event 'command' (an uncontrollable event) gets activated, and the robot goes into state 2 (q_2) where it performs a task. When the task is over, the event 'complete' (an uncontrollable event) is triggered and the system goes back to the initial state (q_1). The states are denoted by circles and the double circle indicates a marked state. The unlabelled arrow represents the initial states. Arrows with a strike on them represent controllable events and arrows without the strike denote an uncontrollable event. (Note that the state q_1 in G_1 does not have to be the same as q_1 in G_2).

2.5.3 Control specifications

Control specifications are used to dictate the intended behaviour of the system. The control specifications determine the controllable events that can be allowed in each state. It can be defined as *n* independent specifications for the entire system. It is produced by the generator E_j where j ∈ {1, 2, …, *n*}. An example of control specification (E_1) for the above-mentioned system is given in figure 3(c), At state 1 (q_1), the robot can 'listen' to the operator signal. When it receives a signal, 'command' is triggered, and it moves into state 2 (q_2). In q_2 the robot cannot 'listen' to the operator signal. It can 'listen' to the operator signal only after the uncontrollable event 'timeout' is triggered and the system goes back to the initial state q_1. Thus, the control specification restricts the robot from listening to an operator signal when it's performing a task.



(a) G_1 (b) G_2 (c) E_1

Figure 3: Example of free behaviour models (a and b) and control specification (c) for a simple robot.

2.5.4 Supervisor synthesis

A supervisor enforces the desired behaviour of the system. It ensures that the system only performs the actions that meet the behavioural requirements at any given time. It does this by limiting the capabilities of the system, defined in the free behaviour models, using the control specifications. The first step in synthesising the supervisor is to combine the generators of free behaviour models and control specification through an operation called synchronous composition (represented by ||). The synchronous composition of two generators G_a and G_b is defined as:

$$G_a \,||\, G_b = (Q_a \times Q_b,\ \Sigma_a \cup \Sigma_b, \delta_{a\,||\,b}, (q_{0_a}, q_{0_b}),\ Q_{m_a} \times Q_{m_b})$$

where

$$\delta_{a\,||\,b}((q_a, q_b), e) = \begin{cases} (\delta_a(q_a, e), \delta_b(q_b, e)) & \text{if } (\delta_a(q_a, e)! \wedge \delta_b(q_b, e)! \wedge e \in \Sigma_a \wedge e \in \Sigma_b \\ (\delta_a(q_a, e), q_b) & \text{if } (\delta_a(q_a, e)! \wedge e \in \Sigma_a \wedge e \notin \Sigma_b \\ (q_a, \delta_b(q_b, e)) & \text{if } (\delta_b(q_b, e)! \wedge e \notin \Sigma_a \wedge e \in \Sigma_b \\ \text{undefined} & \text{otherwise} \end{cases}$$

and $\delta(x, y)!$ implies that δ is defined for the input (x , y). This makes sure that events that are present in both Σ_a and Σ_b occur synchronously and events that are not common to them can occur asynchronously.

If *K* is the language produced by the synchronous composition of the free behaviour model (*G*) and control specifications (*E*), then *K* is defined as:

$$K = G \parallel E$$

There is no guarantee that *K* is controllable. To make *K* controllable, all the bad states in *K* need to be removed. A bad state is defined as a state, in which an uncontrollable event is denied by the specification but enabled in the free behaviour model. This is unable to be disabled by the supervisor. Any state that has an uncontrollable path to a bad state also has to be removed. The next step is to make sure that *K* is non-blocking. Non-blocking means that all the states of the generator must be reachable and co-reachable. Therefore, all non-reachable and non-co-reachable states must be removed from *K*. This along with the removal of bad states can be done through the *SupC* operator. Therefore, the final controllable, accessible, and co-accessible supervisor *S* is obtained as:

$$S = SupC\ (G, K)$$

There are three ways to synthesise a supervisor: (i) monolithic, (ii) modular, and (iii) local modular.

In a monolithic supervisor, all the free behaviour models are combined into a single generator using synchronous composition. The control specifications are also combined into a single generator. Language *K* is then produced by the synchronous composition of the above two generators. The supervisor is produced from *K*.

In a modular supervisor, each specification is combined with all the free behaviour models i.e., there will be *n* number of supervisors (*n* is the total number of control specifications).

In the local modular supervisor, a control specification is combined with only the free behaviour models that share the same event in the control specification. Here too, the total number of supervisors is the same as the total number of control specifications (*n*).

Lopes et al. [24] compared the three methods and found that the local modular supervisor had the least number of states and transitions and hence, was the most memory efficient of the three. Therefore, the local modular supervisor is used in this study.

Chapter 3

Problem formulation and methodology

This section describes the problem scenario for this project followed by the methodology used to solve the problem. Section 3.1 formally describes the problem, and introduces some symbols and keywords used in the project. Section 3.2 presents the approach used to address the problem and the various assumptions used. The problem formulation and methodology used here are chiefly adopted from [2], [3].

3.1 Problem formulation

The scenario consists of a 2D bounded environment (represented as $\mathcal{W} \subset \mathbb{R}^2$) which contains obstacles (represented as $\mathcal{O} \subset \mathcal{W}$), human operators and N robots. The boundary is also considered an obstacle. Let the free space $\mathcal{W}_{\text{free}} = \mathcal{W} \backslash \mathcal{O}$.

The operators and robots are organised into 2 teams. Each team is comprised of one operator and a subset of the robots. At any given time, a robot can only be part of a maximum of one team. The robots are called *workers*. The robots are of different types each carrying a unique set of tools. Let there be k types of robots, i.e., there are k sets of tools. If N_i, $i \in \{1, 2, \ldots, k\}$ is the total number of robots of type i, then $\sum_{i=1}^{k} N_i = N$.

There are multiple tasks spread across the environment. The tasks are defined as a tuple ($\mathcal{A}$, $\mathbf{m}_i$, t). $\mathcal{A} \subset \mathcal{W}_{\text{free}}$ represents the *task region*. $\mathbf{m}_i \in \mathbb{N}^k$ is a vector that specifies the number of robots of each type required to perform the task. For example, in a system with 3 types of robots, A, B, and C, one task might require 2 robots of type A, 3 robots of type B, and 1 robot of type C. Then $\mathbf{m}_i = (2, 3, 1)^T$. $\mathbf{m}_i$ is also called *specification vector*. The task starts as soon as all the required workers are placed within the task region. t represents the *demand* of the task i.e., the duration (in time steps) required to complete the task when sufficient robots are available in the task region. The demand t decreases monotonically with time while the task is being executed. The operator and the required robots must be always present within the task region while executing the task. In case a robot or the operator leaves the task region, the task execution stops. The remaining demand of the task was not reset and the task continues from its previous demand level once the conditions are met again. An increase in the number of

available robots does not affect the time steps t required to complete the task. $\mathcal{A}$, $\mathbf{m}_i$ and t are together called *task requirements* and they are static for a given task.

An example of the environment is provided in figure 4. The black lines represent the boundaries and walls (obstacles). The red sections represent the task region. The texts inside the task region represent the number of robots, of each type, required for each task (within the brackets) and the time demand of the task. For example, the task on the top left corner in figure 4 needs 1 robot of type A, 2 robots of type B, and 1 robot of type C. The demand is 10.

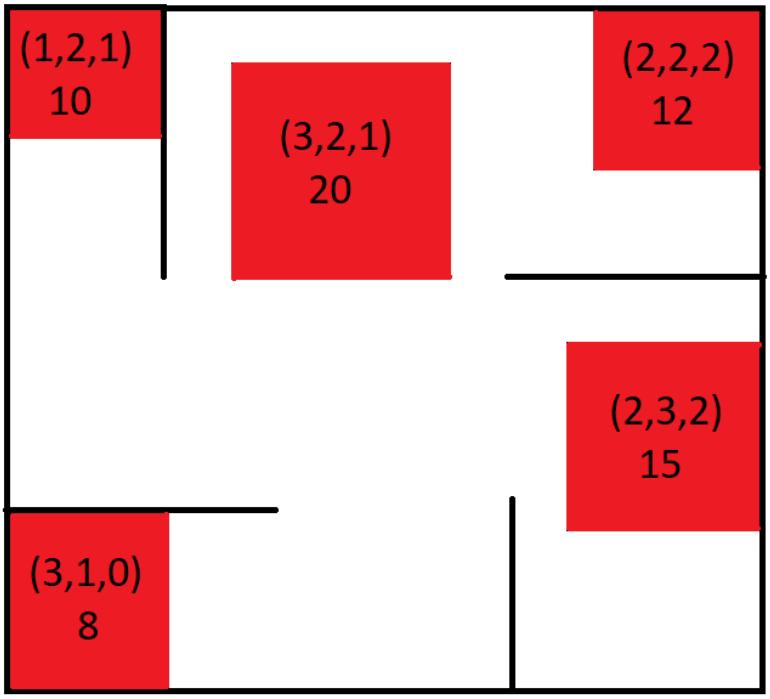


Figure 4: Example for the environment in which the swarm system operates.

The presence of the human operator in the team was realised using a *lead agent*. The lead agent was assumed to be a device held by the operator which lets the operator interact with the swarm or a robot that is teleoperated by a human operator. The operator can use this device or robot to give instructions to the swarm system such as requesting workers of a certain type. This enables the operator to passively control the swarm system. Once all the workers with the required tools reach the task location, the operator can either use the tools to carry out the tasks, or the robots can perform the tasks themselves under the supervision of the operator.

In theory, a team can have more than one operator. There could be a lead operator who controls and interacts with the swarm robots, while the other operators can use the tools to execute tasks or assist with the monitoring of the robots performing the tasks. However, this scenario is not explicitly considered in this project.

All the robots can communicate with other robots that are within their communication range, r_{com}. The distance is measured from the centre of a robot to the centre of the next robot. For this project, r_{com} was set to 80cm. The robots within a distance r_{com} from a given robot are called the *neighbours* of that robot. The communication network formed by the robots is an undirected graph. The nodes of the graph are the individual robots and the edges of the graph are formed by pair of robots that can communicate which each other. The system must remain connected at all times. As long as all the nodes of the communication network graph are connected, the connectivity is preserved. In other words, any given robot should have at least one neighbour to preserve connectivity. The communication network is assumed to be connected at the beginning.

In the beginning, two teams are present with a certain number of robots of each type allocated to them. The robots belonging to the team leaders are called the *followers* of that team. The lead agents may move within the environment and visit task regions accompanied by their followers. The locations of the task regions are known to the operator (the lead agent) in advance. However, the specification vector and demands of the task are revealed only once the lead agent enters the task region.

The objective is to complete all the tasks while preserving the connectivity of the swarm system. The tasks should be completed as fast as possible while reducing the distance the robots need to travel.

3.2 Methodology

The methodology and controller design used for this dissertation is based on the methods developed by Miyauchi [2], [3]. However, several modifications were made to improve the system and adapt it to the heterogeneous system.

The system consists of 2 teams. At any given time, the workers in the system belong to either a '*follower'*, '*connector'*, or a '*traveller'* mode. The followers are the workers that are part of a team. Therefore, the followers belong to either one of the teams. They provide the operator with the necessary tools. The connector robots form a dynamic chain between the two teams to preserve connectivity. The connectors can be made to either remain stationary or move. In the case where the connector is allowed to move, they move in such a manner to make the connector chain a straight line. The connector chain also facilitates the exchange of messages between the two teams. The connector that is the closest to a team is called the *tail connector*

of the chain. Lastly, the travellers are the robots that are chosen to be sent to the other team. They traverse through the connector chain to join the other team based on the request from leaders.

As the teams move around, more connectors might need to be added to the chain to maintain connectivity. Similarly, there might also be a situation where a connector is redundant. In this case, the redundant connectors need to be added to a team as followers. If there is only one follower left in a team and the follower is more than the safe distance (r_{safe}) from the tail connector, the team can no longer move further away from the tail connector as that would break the connectivity.

For a follower i to become a connector, the following two conditions need to be met [2]. This is also illustrated in figure 5.

- C1: There should be at least one connector directly connected to the follower i (tail connector) and the distance to all the connectors is greater than a safety limit $r_{safe} + \varepsilon$ ($r_{safe} + \varepsilon < r_{com}$).
- C2: The follower i should be closer to the tail end connector than all other team members directly connected to the tail connector.

Where r_{safe} is the minimum distance between the follower and a tail connector to form a new connector and ε is the tolerance. r_{safe} was set to 50cm and ε was 5% of r_{safe}, i.e, 2.5cm

When it is necessary for a follower to become a connector and if it satisfies conditions C1 and C2, it sends a request to the tail connector to join the chain. If there are no connectors present, the request is sent to the lead agent. If the request is to the tail connector, it sends an acceptance signal to the follower robot, letting it join the connector chain and become the new tail connector. If the request is sent to the lead robot, it accepts the request if it has at least one other follower. If the lead agent or the tail connector receives multiple such requests, then the request that was received first is accepted and the other requests are rejected.

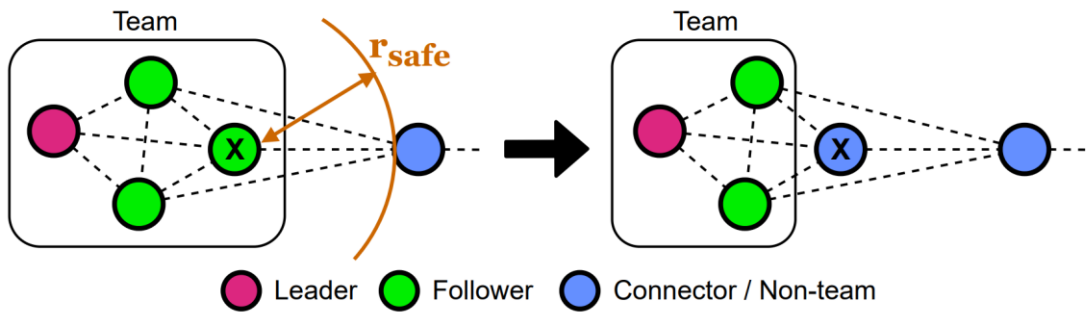


Figure 5: An instance prior (left) and after (right) a robot joins a chain at its tail. The team is represented by the box and dashed lines represent the connections between robots. The robot marked X was sufficiently far away from the connector (condition C1) and was closer to the connector than the other followers (condition C2). It sends a request to leave the team to become a connector, which is approved by the blue connector robot. Image reprinted from [2].

If a connector i needs to become a follower, the following two conditions need to be met [2]. This is also illustrated in figure 6.

- F1: Connector i is a tail connector
- F2: There exists at least one follower in the team that is closer to the successor of connector i than the safety limit $r_{join} = r_{safe} - \varepsilon$.

When a connector satisfies conditions F1 and F2, it leaves the connector chain and join the closest team. Its direct successor becomes the next tail connector.

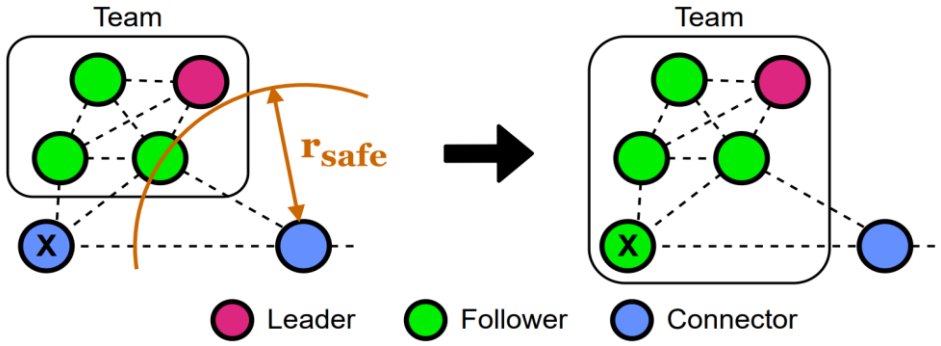


Figure 6: An instance prior (left) and after (right) a robot leaves a chain and joins a team. The robot marked X was a tail connector of the chain (condition F1) and the team was sufficiently close to the preceding connector (condition F2), making it redundant. Therefore, it leaves the chain to join the nearby team. Image reprinted from [2].

To complete certain tasks, there might arise the need to exchange robots between teams. For example, if a task of team 1 requires, 2 robots of type C, but it is unavailable, it can request,

the required number of robots of type C from team 2. This request is passed along the connector chain till it reaches the lead agent of the other team. If team 2 have the requested number of robots of type C, it can send them to team 1.

The robots communicate with each other through messages. They enable the system to function as intended. Each robot publishes messages that can be received by the robots within its communication range (i.e., its neighbours). This also lets the leader know about the details and types of its followers. The message system is similar to the one used in [2] and [3] but has been modified to meet the requirements of the heterogeneous system. Two vital parts of the message are 'heartbeat' and hop count. These are described in the following paragraphs. A detailed description for each of the individual contents of the message along with the differences from the one used in [2] and [3] is given in Appendix A.

The 'heartbeat' is a message that is constantly sent from one leader to the other leader every 10 time-steps. This ensures that the system connectivity is preserved. The information about the number of robots of each type the leader has and the minimum number of robots of each type required for the current task is passed on to the other team through the heartbeat message. This enables a team to know about the details of the other team. The ability to transfer data about each type of robot was an addition made to the message structure used by Miyauchi [2], [3]. The information about sending robots or requesting robots is also transferred through the heartbeat message. Requests for multiple types of robots can be sent through a single request. This was not possible in the original message structure developed by Miyauchi [2], [3].

The relative position of a follower in the team or the position of a connector in the chain relative to a team was realised through a message called 'hop count'. It indicates the position of the robots in the communication network graph. A follower that can directly sense the lead agent has a hop count of 1. A follower that cannot directly sense the leader but can sense a robot with a hop count of 1 gets a hop count of 2 and so on. Similarly, a connector which is in direct communication with a team has a hop count of 1. the next connector in line has a hop count of 2, and so on. The connectors have two sets of hop count, one relative to each of the two teams.

The methodology used to select the robots (*candidate robots*) to send over to the other team during an exchange was different to the one used by Miyauchi [2], [3]. In his work, the first follower in the team that receives a message from the tail connector i.e., the follower that was closest to the tail connector was sent over to the other team. However, this was not possible in the case of a heterogeneous system. For this project, the candidate robots of each type were

those robots which were the furthest from the leader. This ensures that, whenever possible, the candidate robot is in the outer vicinity of the team. This makes its navigation to the connector chain easier.

When a robot is sent to the other team, it travels along the connector chain. The algorithm for travel is different from the one used by Miyauchi [2], [3]. The robot uses its sensors to detect the distance and bearing of the next connector in the chain and moves to the left side of the connector (offset by a certain distance). Once it reaches that connector, it looks for the next connector and travels towards it. The direction in which to travel through the chain is determined using the hop count of the connectors. The traveller moves from a connector with a higher hop count (to the team to join) to one with a lower hop count. This continues till it reaches the other team. They travel along the left side of the chain. When it senses a robot from the team to join, the traveller moves towards it. Once the robot is substantially closer to the team and this distance is less than the distance between itself and the nearest tail connector (i.e., it is closer to the team than the connector), it joins the team and becomes a follower.

At the start, if a traveller does not detect a connector, due to it being out of the communication distance, it moves toward the leader. It is assumed that the leader is able to directly sense the connector. Therefore, once it reaches the leader, it can sense and travel towards the tail connector and start its travel towards the other team. If there are no connectors (the teams are close), the connector travels to the followers with the highest hop count to the leader (furthest away from the leader) and keeps moving from one follower to the other till it senses a robot from the other team. When it detects the team to join, it moves towards it and joins when it is closer than a pre-set distance.

The behaviour of the system was formally modelled using SCT. The SCT model was developed with the software ‘Nadzoru’ [62]. The modelled system was tested and evaluated using the physics-based simulation software ‘ARGoS’ [26]. Several cases were studied using the simulation environment.

3.2.1 SCT model

The SCT models used in the project are mostly unchanged from the ones used in [2] and [3]. However, one key change was made to one of the operational procedures to adapt it to the

heterogeneous robot problem. Only the portion that had been altered is discussed here. For the full SCT model and explanations, refer to [2] and [3].

Modifications were made to the free behaviour model G_{12}^R and the control specification E_7^R in [3]. The original models are given in figure 7 and the modified ones are in figure 8 (changes highlighted in red text). These determine the behaviour of the robot when it is in traveller mode. The uncontrollable events 'nearLF' and 'notNearLF' were changed to 'closerLF' and 'notCloserLF' respectively. The event 'nearLF' was triggered when the traveller was near to the leader or one of the followers of the team it needs to join. 'notNearLF' was triggered when this condition was not met. The distance from the leader or the follower to the traveller to trigger the event 'nearLF', r_{team}, was set to 40cm.

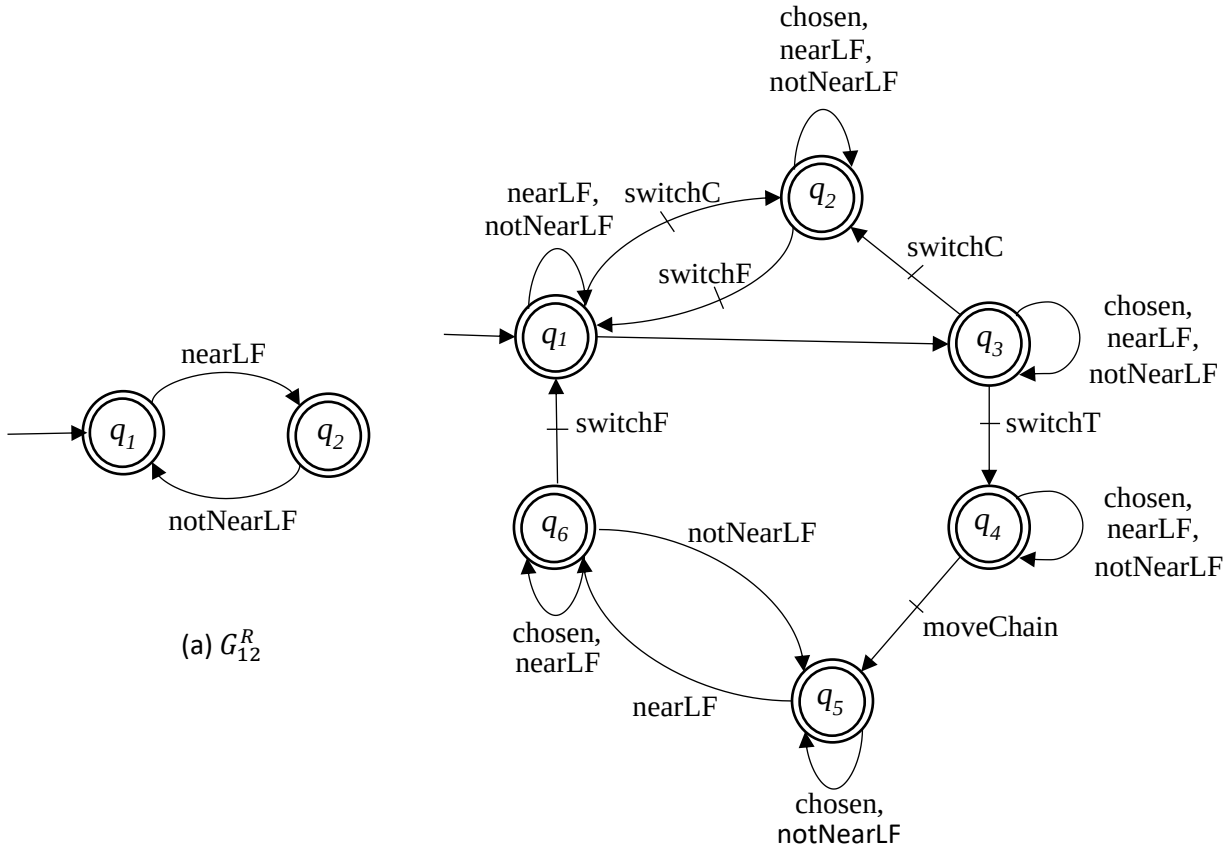


Figure 7: Original SCT models used by Miyauchi [3]. (a) Free behaviour model to detect nearby team members, (b) control specification for a traveller to become a follower.

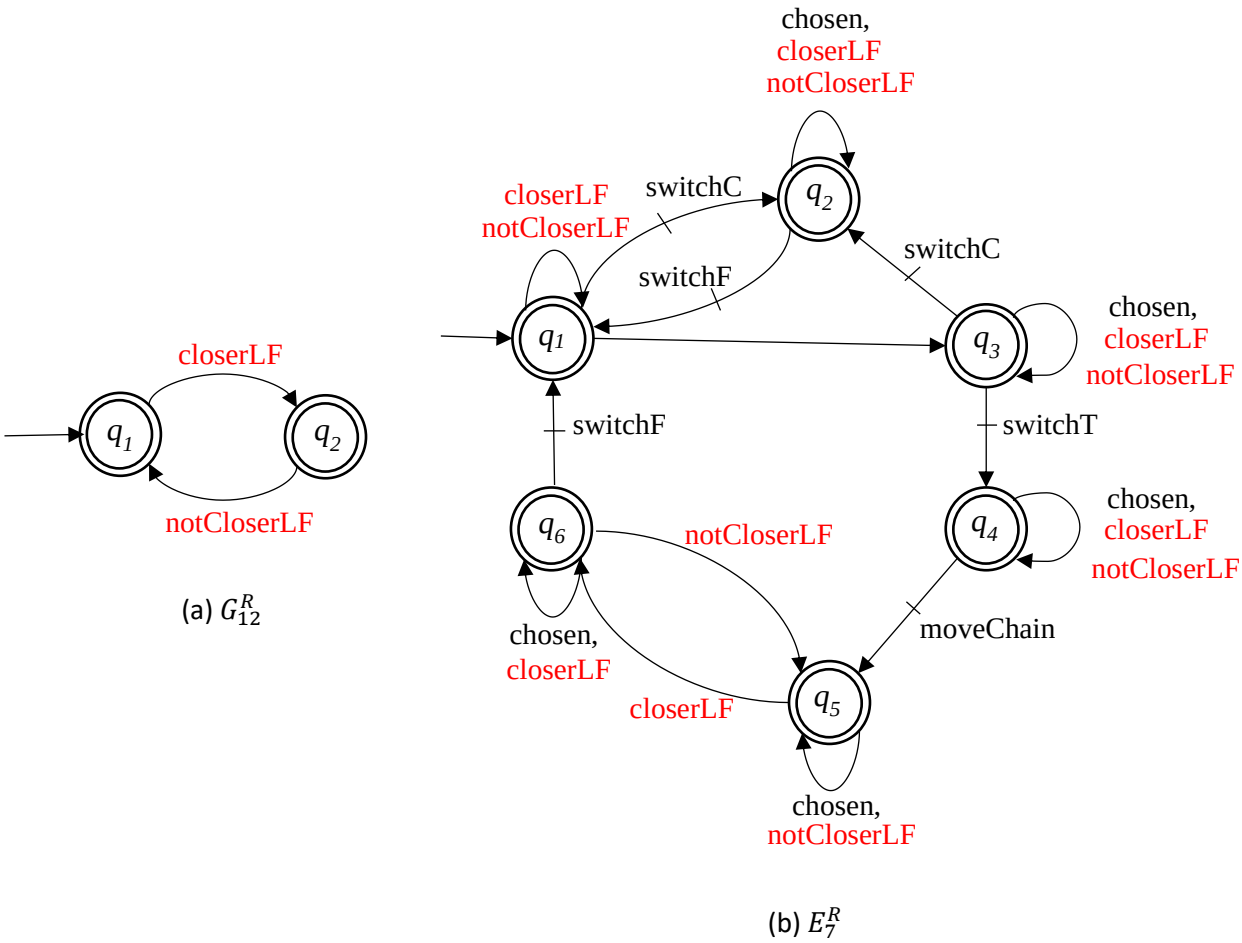
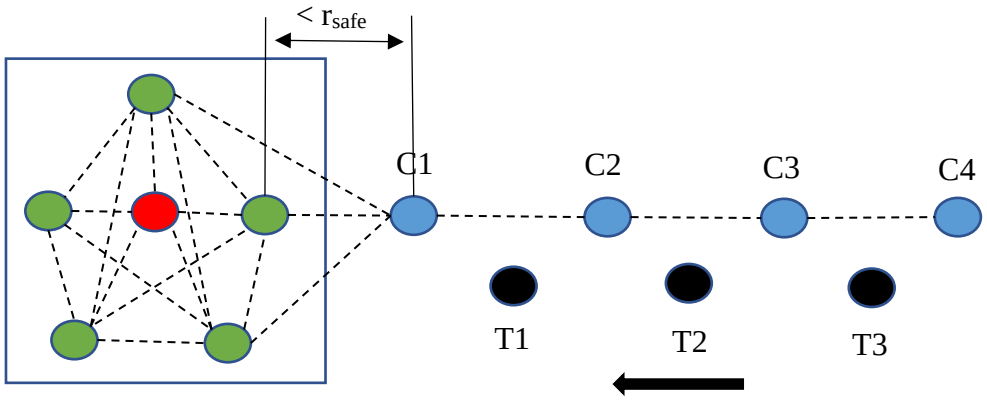


Figure 8: Modified SCT model used in the dissertation. (a) Free behaviour model to check if traveller is closer to the team than the tail connector, (b) control specification for a traveller to become a follower.

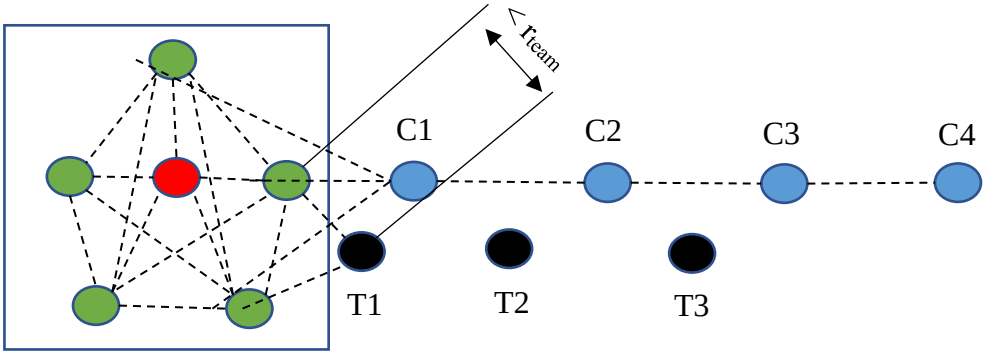
When a follower (state q_1) gets selected for exchange (state q_3) and becomes a traveller (state q_4), it starts moving through the chain to the other team (state q_5). When the event 'nearLF' was triggered (state q_6), the traveller would join the team and switch to follower mode (state q_1). However, this methodology led to an issue when multiple traveller robots travelled together close to each other. This is illustrated in figure 9(a-f).

In figure 9(a), 3 travellers T1, T2, and T3 moves towards the team. The distance between the tail connector C1 and its closest team member is less than r_{safe}. As the travelers comes closer to the team (figure 9(b)), the distance between T1 and its closest team member becomes less than r_{team}. Hence the event 'nearLF' gets triggered and T1 joins the team as a follower (figure 9(c)). As it joins the team, distance between T1 and T2 becomes less than r_{team}, prompting T2 to become a follower too (figure 9(d)). When this happens, it is possible that the distance between T2 and C2 is less than r_{join}. This makes the connector C1 redundant, triggering conditions F1 and F2, and it becomes a follower too (figure 9(e)). But as C1 and T2 moves

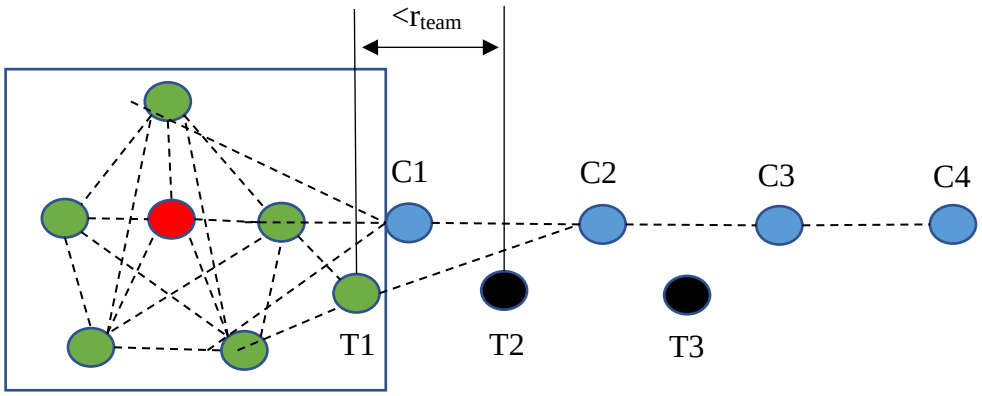
closer to the team leader (due to flocking behaviour, refer section 3.2.2), the distance between T2 and C2 becomes more than r_{safe}. This triggers conditions C1 and C2 and forces T2 to become a connector (figure 9(f)). Therefore, instead of turning to a follower, the exchanged robot becomes a connector.



(a)

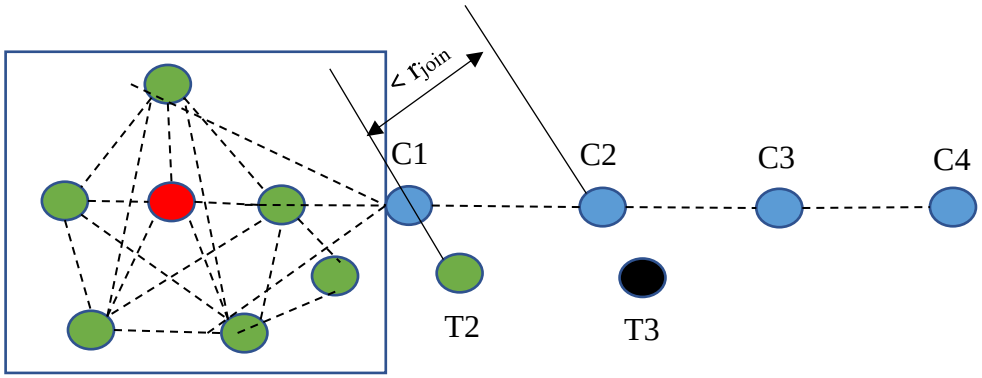


(b)

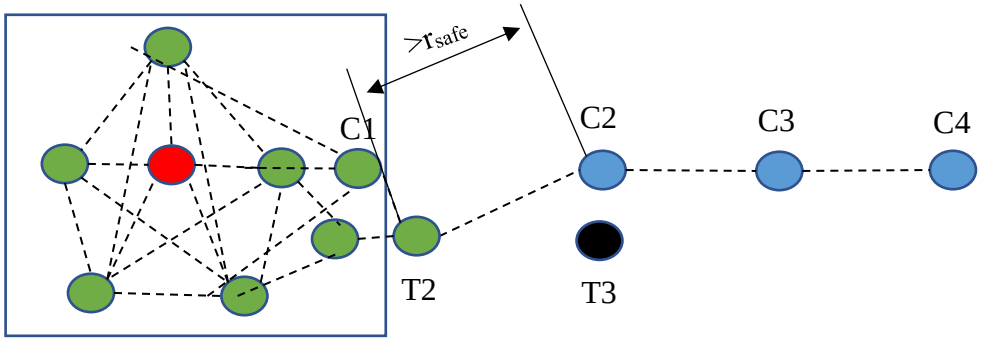


(c)

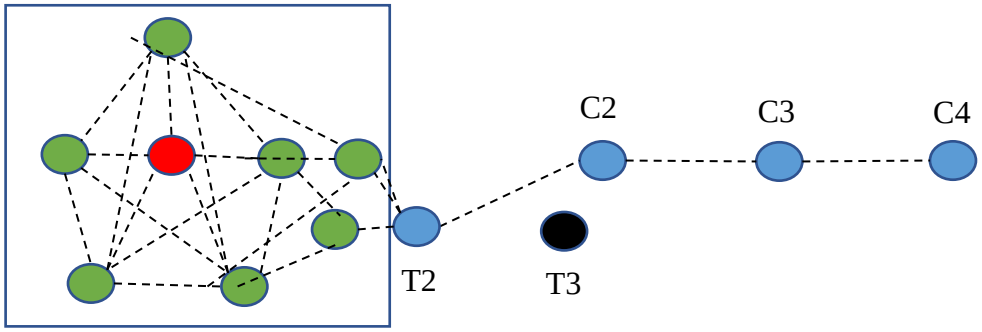
Figure 9(a-c): Illustration of a shortcoming in the condition used by Miyauchi. (a) Travellers travelling towards the team, (b) T1 within r_{team}, (c) T2 within r_{team}.



(d)



(e)



(f)

Figure 9(d-f): Illustration of a shortcoming in the condition used by Miyauchi. (d) T2 becomes a follower, (e) C1 becomes a follower, (f) T2 becomes a connector

This did not, however, cause a problem in Miyauchi's case. This was because in that case, only the number of robots mattered. Therefore, even if the traveller becomes a connector, and an existing connector joins the team, the team still gets an additional robot. However, for this project, as the type of robot matters, this could not be allowed to happen.

To prevent this, the new conditions 'closerLF' and 'notCloserLF' were used. 'closerLF' was triggered only when the distance from the leader or the follower to the traveller is less than r_{team} and the distance between the traveller and the tail connector is greater than $r_{team} + \Delta$. r_{team} was set as 20cm and Δ as 25% of r_{team} i.e., 5cm. This ensures that the travellers did not become a connector during the process of joining the team. 'notCloserLF' was triggered when 'closerLF' was not met.

3.2.2 Artificial potential field modelling

The worker robots use the concept of artificial potential force to move around [9]. The virtual force is made of three components; one attractive force towards the goal, one repulsive force from nearby robots, and one repulsive force from obstacles and other robots when they are too close. These are similar to the ones used in [2] except for the changes made to the *weights* for each force component.

The robots used in the simulation were e-puck robots. They are equipped with 8 proximity sensors which can be used to detect the bearing of other robots and their distance, up to 10cm accurately. In addition to this, a range and bearing system is assumed to be available providing the relative positions of nearby robots. Let $\mathbf{p}_{ij}$ be the position of robot j with respect to robot i.

The virtual force on robot i ($\mathbf{u}_i$) is given by:

$$\mathbf{u}_i = \alpha \mathbf{u}_i^{attract} + \beta \mathbf{u}_i^{repulse1} + \gamma \mathbf{u}_i^{repulse2} \quad (1)$$

where α, β, and γ are positive scalar constants used to control the effect of each individual force on the net force. These are also called *weights*.

In the case of followers, the attractive force $\mathbf{u}_i^{attract}$ is used to make the follower follow the lead agent. It is defined as:

$$\mathbf{u}_i^{attract} = \frac{1}{|N_i^{attract}|} \sum_{j \in N_i^{attract}} \frac{\mathbf{p}ij}{\|P_{ij}\|} \quad (2)$$

where $N_i^{attract}$ denotes the set of neighbours of robot i that are either the lead agent or followers of the same team that are closer to the lead agent than robot i.

In the case of travellers, the attractive force $\mathbf{u}_i^{attract}$ is a vector to its next goal point. For connectors (In the case it is allowed to move), it is the attractive force towards its two immediate neighbouring connectors. For a tail connector, the attraction is calculated to its team and the next connector in the chain or to the two teams if it is the only connector in the chain.

The force component $\mathbf{u}_i^{repulse1}$ is used to repel robot i from all its neighbouring robots. It is defined as:

$$\mathbf{u}_i^{repulse1} = \frac{1}{|N_i|} \sum_{j \in N_i} \frac{1}{d_{ij}^3} \frac{\mathbf{pij}}{\|P_{ij}\|} \quad (3)$$

where $N_i$is the set of neighbours of robot i, d_{ij} is the distance between robots i and j.

$\mathbf{u}_i^{repulse2}$ is the force component used to prevent collisions if robot i comes too close to other robots or obstacles. It is defined as:

$$\mathbf{u}_i^{repulse2} = -\frac{1}{8d_{max}} \sum_{j=1}^{8} (d_{max} - d_j)\, \hat{v}_J \quad (4)$$

where d_{max} = 10 cm is the maximum range of the proximity sensors, d_j is the distance acquired from the j[th] proximity sensor, and $\hat{v}_J$ is the unit vector pointing from the robot's centre to the j[th] sensor. When the j[th] sensor does not detect an object, $d_j = d_{max}$.

The weights assigned to followers, travellers, and connectors are as follows:

$\alpha_F = 1, \beta_F = 3, \gamma_F = 6$

$\alpha_T = 1, \beta_T = 0.5, \gamma_T = 3$

$\alpha_C = 2, \beta_C = 40, \gamma_C = 15$

Where the subscripts F, T, and C represent the weights for follower, traveller, and connector respectively.

The weights were derived based on trial and error. The lower repulsion weights (β and γ) for the traveller enables it to travel closer to a follower when it is travelling from the team to the connector chain. As the traveller comes close, the higher weight of the follower makes it move away from the traveller and thus does not disrupt the motion of the traveller. This was especially useful in the scenario where a robot close to the leader and surrounded by many followers was required to travel to the other team. If the follower and travellers used the same weights, the traveller may not be able to travel to the chain due to the repulsion from the surrounding followers.

Chapter 4

Simulation and results

This chapter discusses how the system was implemented and tested in the simulation and presents the results and findings from the simulation. The simulation layout and working are explained in section 4.1. Section 4.2 describes the results and various analyses from the simulation trials. Section 4.3 gives a summary of the results and discusses about of the limitations of the system. Finally, a pathway to implement this system in a real robot is deliberated in section 4.4.

4.1 Simulation environment

The ARGoS simulation platform [26], was used to simulate and test the system. ARGoS platform was used primarily since the same was used in the studies by Miyauchi [2], which was the foundation for this dissertation. The biggest advantage of ARGoS compared to most other simulators such as Webots, V-REP, or Enki is the ease of implementing heterogeneous robots. It can also handle a large number of swarm robots and process them faster than in real-time. E-puck robots were used for the simulation. The programming was implemented in C++ programming language.

The trials were run in a 4m x 4m arena. 4 or 6 tasks were distributed within the arena each with its task requirements. A snapshot of the simulation environment is given in figure 10. The e-puck robots were illustrated by cylinders with 3.5cm diameter. Initially, the leaders, along with their followers, were placed close to each other so that no connectors were needed. The two leaders were represented by ‘L1’ and ‘L2’, for the first and second team respectively. The type of robot was represented using the English alphabet. For example, ‘A’ represented a type of robot carrying a particular set of tools, ‘B’ represents another type of robot and so on. To uniquely identify each robot, numbers were added to each robot type. For instance, robot ‘A1’ or ‘C15’. The names of the robots were colour coded for easier identification based on their type. The system was designed to handle up to 10 robot types. The cyan-coloured lines between the robots represent the communication links, i.e., the robots that are directly connected to each other.

The brown square boxes represent the task region. The side length of the task region varied from 1m to 1.2m. The black bold number in the task region denotes the demand for each task. The numbers to the left of the square box represent the current number of robots of each type present in the task region and the required number of robots to perform the task (current number of robots / required number of robots). The colour of these matches the colour used to represent the name of the corresponding type of robot. For example, for the task on the top left-hand side, the task demand was 120 (time-steps). The tool requirements were 4 robots of type A, 2 robots of type B, 4 robots of type C, 6 robots of type D and 2 robots of type E. In the given instance, no robot was present in the task region. The task begins execution once the required tools (robots) are present within this task region.

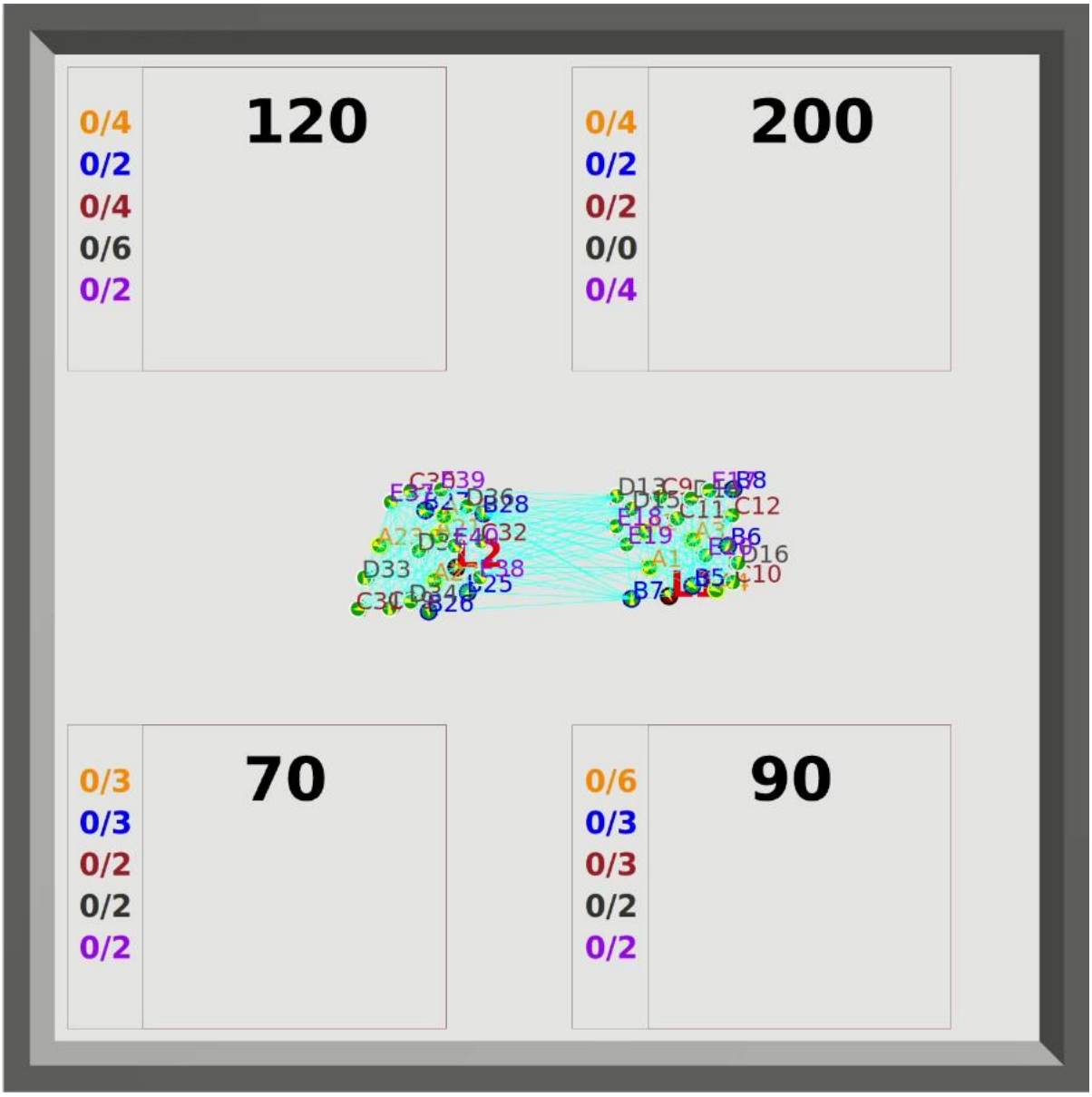


Figure 10: The simulation environment

Each of the leaders were given the location of their allotted task regions. The leaders travel to them one by one using a proportional controller. Once a task is completed, it moves to the next task region. When all the tasks were completed, the leader stops moving.

Each time-step in the simulation denoted 0.1s of real-time. Therefore, the experiment was updated every 0.1s. ARGoS enables the user to speed up the experiment. For the video recordings of simulation trials, the experiment was fast-forwarded to 5 times the real-time speed. The video recordings for all the trials can be found in [63]. The codes used in the dissertation can be obtained from the git repository [64].

4.1.1 Automatic sending and requesting of robots

In a real-life scenario, the operator can request or send robots at any given time. The operator can also decide how many robots need to be sent to the other team when a request comes or choose not to send them at all. However, for the simulation, this was not possible. Therefore, a procedure for the automatic sending and requesting of robots was created for the simulation.

If the required number of robots for any type were not available to the leader to perform a task, it was programmed to send a request to the other team for the required number of robots of each type through the heartbeat message. When a leader receives such a request, it checks the number of robots it can send to the other team. If it has surplus robots, it sends the requested numbers of robots. If it cannot send the requested number of robots, it sends the maximum number of robots it can after ensuring that it has at least one follower left. If the leader that received a request, is performing a task, it waits until the current task is over before sending the robots. In the scenario where both the leaders are waiting to receive robots to start the task, the leader of team 2 was hard coded to send the requested robots to team 1. The leader that needs the robots, resends the request every 900 time-steps if the requirement was not yet met.

4.2 Results and findings

The primary aim of the simulation was to verify whether the system worked as intended in all situations. To test the robustness of the system, simulation trials were carried out for a wide range of parameters. The number of followers in the teams, the number of tasks, the task requirements, and the number of types of robots were varied for the trials. The trials were conducted for both cases of the connector chain (remaining stationary and moving). In addition to this, the initial location of the worker robots around the leader was randomised in each trial, thus resulting in different results each time.

During the trials, it was observed whether the connectivity was maintained and the exchange of robots was executed correctly. Furthermore, performance parameters such as the time taken to complete all the tasks, the number of robots exchanged, the maximum number of connectors in the chain, and the distance travelled by the workers and leaders were also recorded. The results of these trials are given in tables 1(a) and 1(b).

The tasks were ordered as follows: the first task was team one's first task, the second task was team two's first task, the third task was team one's second task, the fourth task was team two's second task, and so on. The task demands are given in the same order as the task order, i.e., the first number indicates the demand for the first task, the second number indicates the demand for the second task and so forth. The same applies to the specification vector. The numbers in the first bracket indicate the specification vector ($\mathbf{m}_i$) for the first task, followed by the second task and so on.

The maximum number of connectors refers to the highest number of connectors found in the chain during the entire trial. The average distance travelled per worker was calculated by dividing the total distance travelled by the workers by the total number of workers in the simulation (same as the total number of followers initially).

In addition to this, two sets of simulation trials were taken to study the impact of the specification vector on system performance by keeping all the parameters the same except the specification vector. These are detailed in section 4.2.4.

Table 1(a): Results from the simulation

Trial No	No of types of robots	No of tasks	Task demands	Specification vector	No of followers (team 1 + team 2)	Chain type	No of times exchange requested	No of robots exchanged	Max no of connectors	Total distance travelled (metres)		Time to complete tasks	Average distance travelled per worker
										Leader	Workers	timesteps	metres
1	5	4	150,70 120,50	$(6,9,3,6,9)^T$ $(10,5,4,3,2)^T$ $(2,3,5,3,2)^T$ $(3,2,2,4,1)^T$	30+30	Moving	3	15	4	7.48	462.3	2565	7.7
2							3	11	3	7.6	303.07	2050	5.1
3						Stationary	3	13	7	7.62	408.65	3264	6.8
4							2	11	6	7.64	287.53	2097	4.8
5			90,200 70,120	$(6,3,3,2,2)^T$ $(3,3,2,2,2)^T$ $(4,2,2,0,4)^T$ $(4,2,4,6,2)^T$	20+20	Stationary	3	10	10	8.97	251	2339	6.3
6							3	11	8	9.08	282	2540	7.05
7							3	11	8	9.59	273	2557	6.8
8							2	6	7	9.92	244	2038	6.1
9		6	150,70,80 120,50,200	$(2,3,2,2,3)^T$ $(5,3,2,2,2)^T$ $(5,5,3,2,2)^T$ $(2,2,2,1,3)^T$ $(0,0,3,3,2)^T$ $(2,3,0,5,4)^T$	20+20	Moving	3	10	5	10.37	248	3488	6.2
10							2	7	5	10.2	224.8	2073	5.6
11						Stationary	3	9	8	10.2	221	3017	5.5
12							3	11	7	10.15	231	2747	5.8
13			150,70,80 120,50,200	$(6,2,2,4,2)^T$ $(2,2,4,4,2)^T$ $(2,2,2,4,2)^T$ $(2,0,2,4,2)^T$ $(2,2,2,0,2)^T$ $(4,4,4,0,2)^T$	20+20	Moving	4	9	5	9.95	390	3418	9.8
14							5	10	6	9.89	420	3718	10.5
15							4	9	5	10.16	330	2691	8.3
16						Stationary	5	13	9	9.8	372	4008	9.3
17							3	6	8	10.53	315	2958	7.9
18							2	7	7	10.07	255	1928	6.4
19			150,70,80 120,50,200	$(6,2,2,4,2)^T$ $(2,2,4,4,2)^T$ $(2,2,2,4,2)^T$ $(2,0,2,4,2)^T$ $(2,2,2,0,2)^T$ $(4,0,4,0,2)^T$	20+20	Moving	3	4	5	9.82	307.5	2492	7.2
20							4	9	5	9.77	335	2684	8.4
21							4	7	5	10.07	309.5	2349	7.7
22						Stationary	3	5	7	9.9	286	2494	7.2
23							3	8	7	10	290	2515	7.3
24			150,70,80 120,50,200	$(3,1,1,2,1)^T$ $(1,1,2,2,1)^T$ $(1,1,1,2,1)^T$ $(1,0,1,2,1)^T$ $(1,1,1,0,1)^T$ $(2,0,2,0,1)^T$	10+10	Moving	3	4	3	9.69	107	2091	5.4
25							3	4	4	9.57	112.9	2360	5.6
26						Stationary	3	4	7	9.94	94.4	2632	4.7
27							4	7	7	9.99	102.3	3487	5.1

Table 1(b): Results from the simulation

Trial No	No of types of robots	No of tasks	Task demands	Specification vector	No of followers (team 1 + team 2)	Chain type	No of times exchange requested	No of robots exchanged	Max no of connectors	Total distance travelled		Time to complete tasks	Average distance travelled per worker
										Leader	Workers		
28	5	6	150,70,80 120,50,200	$(3,2,2,4,2)^T$ $(2,2,3,3,2)^T$ $(2,2,2,2,2)^T$ $(2,1,2,2,2)^T$ $(2,2,2,0,2)^T$ $(3,2,4,2,2)^T$	15+15	Moving	3	4	5	10.43	199	2215	6.6
29							3	5	6	10.08	208	2654	6.9
30						Stationary	3	6	8	10.37	171	2233	5.7
31							3	3	8	10.14	179	2838	6.0
32			120,90,200 80,70,120	$(3,2,2,4,2)^T$ $(2,2,3,3,2)^T$ $(4,2,3,2,1)^T$ $(2,1,2,2,2)^T$ $(2,2,2,0,2)^T$ $(3,2,4,2,2)^T$	15+15	Moving	5	7	6	10.12	232.5	2704	7.75
33							4	8	6	10.42	222.5	2594	7.4
34							3	6	6	10.25	237.5	2861	7.9
35							4	12	6	10.37	244	2886	8.1
36		6 (with obstacle)	150,70,80 120,50,200	$(3,2,2,4,2)^T$ $(2,2,3,3,2)^T$ $(2,2,2,2,2)^T$ $(2,1,2,2,2)^T$ $(2,2,2,0,2)^T$ $(3,2,4,1,2)^T$	15+15	Stationary	5	9	8	10.4	212.5	3280	7.08
37							3	6	7	10.2	216	4013	7.2
38							6	9	8	10.3	229	4072	7.63
39			120,90,200 80,70,120	$(3,2,2,4,2)^T$ $(2,2,3,3,2)^T$ $(4,2,3,2,1)^T$ $(2,1,2,2,2)^T$ $(2,2,2,0,2)^T$ $(3,2,4,2,2)^T$	15+15	Stationary	3	9	8	10.07	212	2837	7.05
40							5	9	8	10.45	223.5	3642	7.45
41							3	5	7	10.63	207	2879	6.9
42	3	6	150,70,80 120,50,200	$(6,6,6)^T$ $(4,4,4)^T$ $(4,4,8)^T$ $(4,4,4)^T$ $(8,4,2)^T$ $(2,4,8)^T$	18+18	Moving	4	10	6	9.93	264	3601	7.3
43							5	14	7	9.84	277.7	3542	7.7
44						Stationary	5	13	8	10.3	315	3722	8.8
45							5	16	9	10.27	315	3984	8.8
46	1		150,70,80 120,50,200	$(6)^T$ $(4)^T$ $(4)^T$ $(5)^T$ $(3)^T$ $(5)^T$	7+7	Moving	3	5	7	9.88	87.33	2899	6.2
47							3	3	6	9.7	83.25	2367	5.9
48						Stationary	5	7	8	10.2	70.27	3307	5.0
49							4	8	9	10.03	73	3183	5.2

4.2.1 Impact of number of followers and types of robots

In trials 1 to 4, it was evaluated whether the system was capable of handling a large number of robots. For this, the system was tested with 30 followers in each team for a total of 60 workers. Due to the large size of the team, the task regions had to be made bigger. Thus, only 4 tasks were used. The leaders were able to complete all 4 tasks. On average, 11-15 robots were exchanged successfully through 2-3 requests.

In trials involving 40 followers or less, the chain connectivity was always preserved. However, in the trials with 60 followers, the team got disconnected from the chain in few of the trials (explanation provided in point 2 in section 4.3). It was also noticed that the movement of the leader slowed down as the number of followers in a team increased. This was because of the repulsion force from the followers as they got too close to the leader.

Trials were also taken with 3 and 1 type of robots. The system did not exhibit any behavioural changes when the number of types of robots varied. This implied that the system remained robust with a varying number of types of robots.

4.2.2 Impact of stationary vs moving connector chain

The effect of keeping the connector chain stationary versus making it move to form a straight line was also studied. On average, it was observed that the maximum number of connectors in the chain was higher in the case of the connectors remaining stationary. Furthermore, the number of robots exchanged in the case of the moving chain was, at worst, comparable to the stationary chain. Therefore, on average, the moving chain required lesser number of robots to be exchanged. This was expected as fewer followers were available to each team in the case of the stationary chain due to it needing more robots in the chain. The time it took for the travellers to reach the other team was also marginally higher in the stationary chain as they had to travel a longer distance. Nevertheless, the time it took to complete the tasks was not affected significantly. This was because the number of times an exchange was requested remained fairly the same in both cases. However, this depends on the specification vector. Since more robots are available per team in the moving chain, the chance of the need to exchange robots was less. Therefore, the moving chain had the potential to complete the tasks faster than the stationary chain. However, such a scenario was not tested.

The total distance travelled by the followers was higher in the case of the chain moving. In situations where the robot uses a lot of energy to move, this is a shortcoming as the battery of

the workers can drain faster. Also, the straight-line formation of the chain does not work in the presence of obstacles (refer to section 4.2.3 for more details).

4.2.3 Impact of obstacles

The system was also tested in the presence of obstacles in trials 36-41. As the connectors need to be in the Line of Sight (LOS) of their adjacent connectors/team, the algorithm to make the connector chain a straight line failed in the presence of obstacles. The chain got disconnected when an obstacle blocked the LOS between two connectors. Therefore, the connectors were made stationary. It is also possible for the chain to break in the case of the connectors being stationary due to obstacles blocking LOS although less likely. In the trials, the system performed as expected. However, in certain scenarios, the travellers got stuck while travelling through the chain as there was not enough space between the chain and the obstacle (explanation provided in point 3 in section 4.3).

The time taken to complete the tasks increased noticeably due to obstacles. As the robots only had a limited sensing range, an obstacle was detected only when the robot comes close to it. Therefore, the leaders were unable to anticipate the obstacle in advance and adjust its path. This meant the leaders took more time to navigate to the task regions.

4.2.4 Impact of specification vector

The effect of change in the specification vector was also studied. For these trials, only 2 types of robots were considered. 4 tasks were present in the arena, each with the same demand and specification vector. There were 24 followers in each team (12 robots of each type). Therefore, the ratio of the number of robots of type A to type B was 1:1. Trials were taken for different ratios for the number of robots required for the task (specification vector), namely $(0,20)^T$, $(2,18)^T$, $(4,16)^T$, $(6,14)^T$, $(8,12)^T$, and $(10,10)^T$. The connector chain was made stationary. The simulation environment is given in figure 11. The same experiment was repeated with 6 robots of type A and 18 robots of type B in each team (1:3 ratio).

30 trials were taken for each of the cases. The time taken to complete the task, and the distance travelled by the followers and travellers were measured for each trial. The number of messages sent and received by the leaders was also measured. A summary of the results is given in tables 2 (for ratio 1:1) and 3 (for ratio 1:3). Box and whisker plots for the trial results are provided in figures 12, 13, and 14.

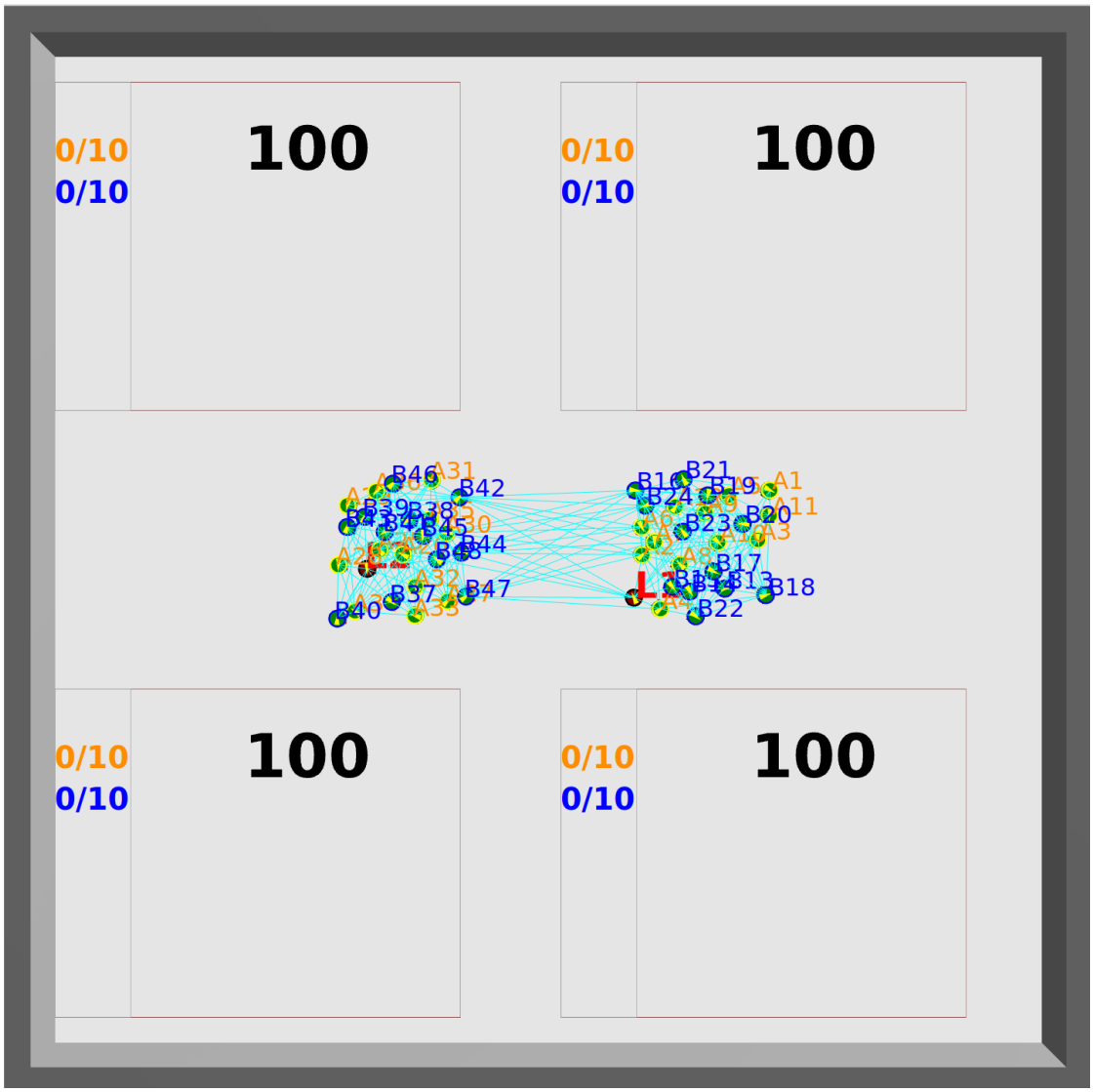


Figure 11: Simulation environment for the trials to study the effect of specification vector

Table 2: Summary of results from the trials to study the effect of specification vector with initial team robot ratio of 1:1

Ratio of robots of each type	Average task completion time (seconds)	Average distance travelled – Follower (metres)	Average distance travelled – Traveller (metres)	Successful messages (%)
0/20	587.83	634.97	266.5	100
2/18	499.83	553.47	227.67	100
4/16	466.8	539.7	158.17	100
6/14	428.43	509.9	91.93	100
8/12	220.07	332.82	11.9	100
10/10	116.87	246.48	0.71	100

Table 3: Summary of results from the trials to study the effect of specification vector with initial team robot ratio of 1:3

Ratio of robots of each type	Average task completion time (seconds)	Average distance travelled – Follower (metres)	Average distance travelled – Traveller (metres)	Successful messages (%)
0/20	459.33	476.76	165.99	100
2/18	427.1	452.28	86.48	100
4/16	205.07	289.77	11.05	100
6/14	284.6	346.18	21.61	100
8/12	457.63	480.47	84.64	100
10/10	498.37	529.67	117.97	100

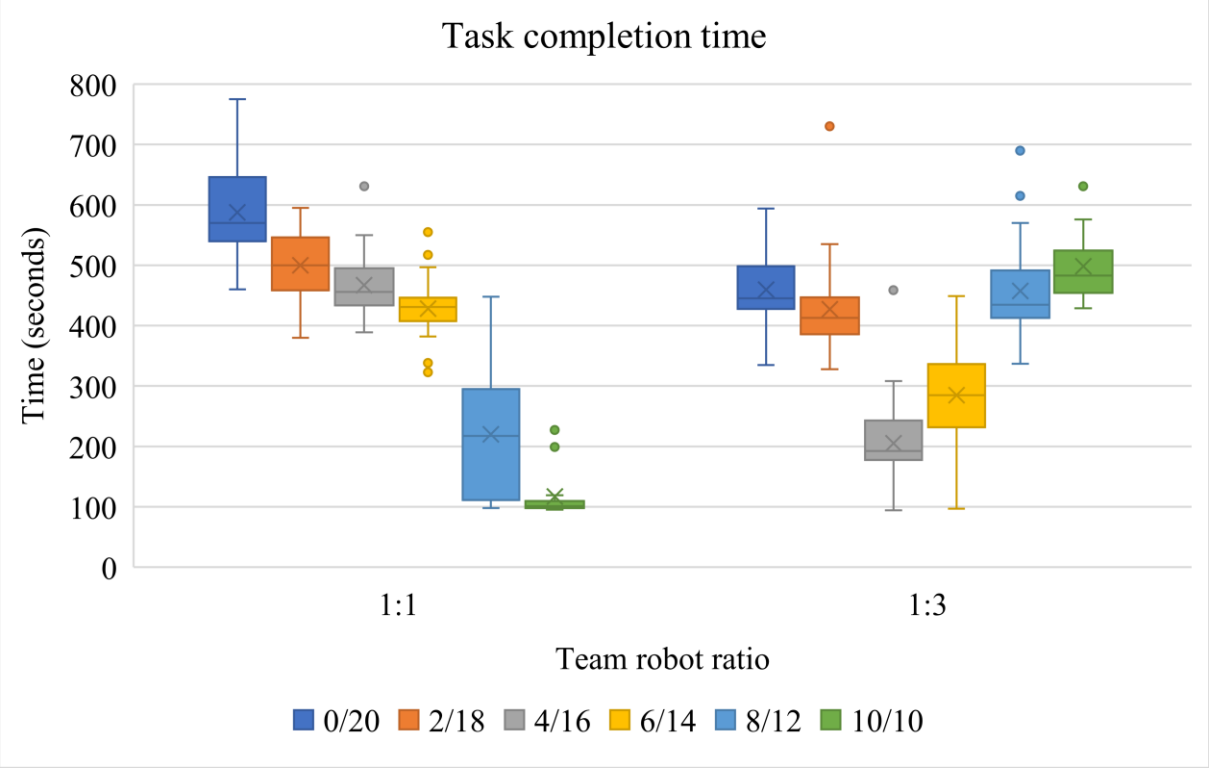


Figure 12: Box and whisker plot showing the task completion time for different ratios of number of robots of each type required for the task

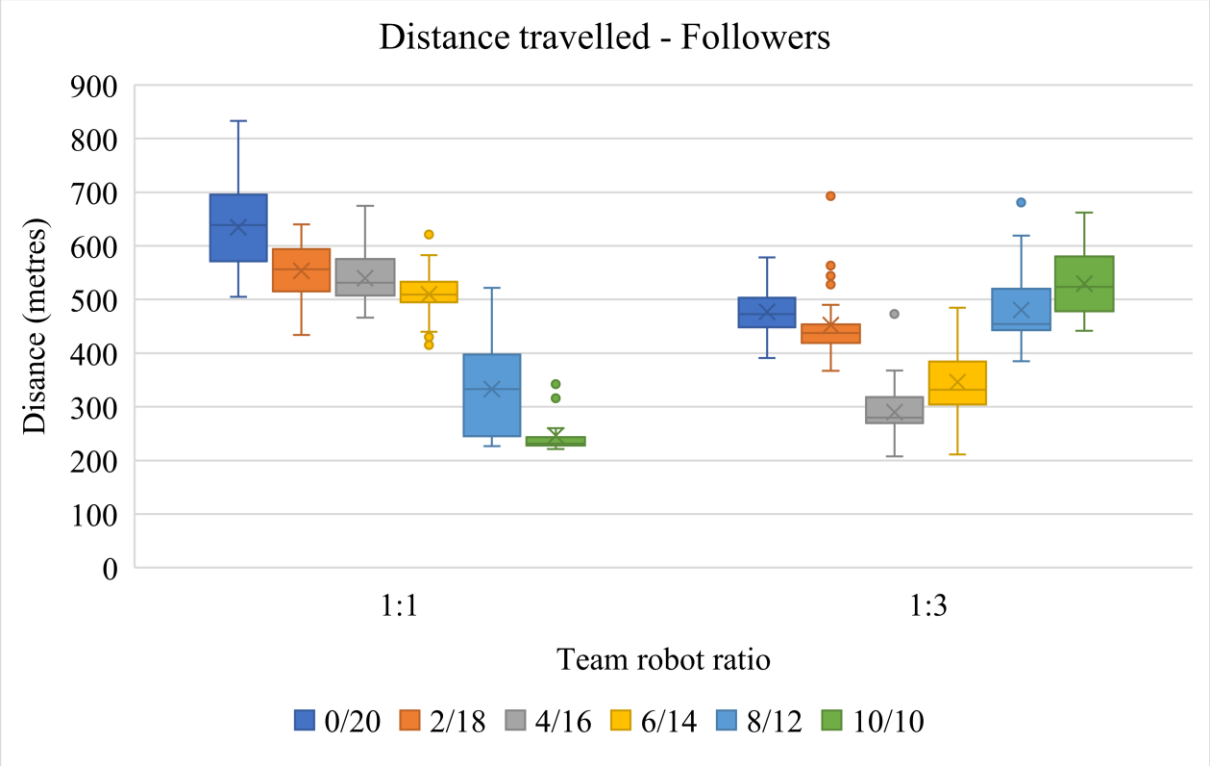


Figure 13: Box and whisker plot showing the total distance travelled by followers for different ratios of number of robots of each type required for the task

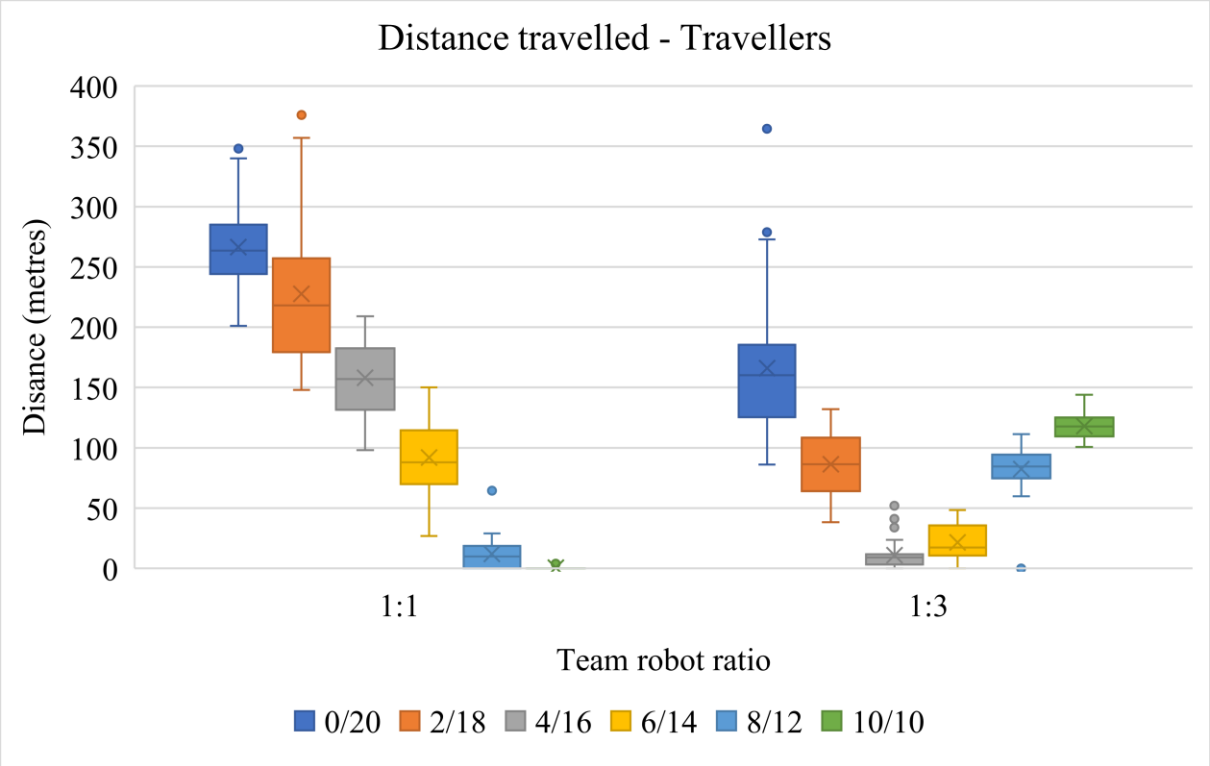


Figure 14: Box and whisker plot showing the total distance travelled by travellers for different ratios of number of robots of each type required for the task

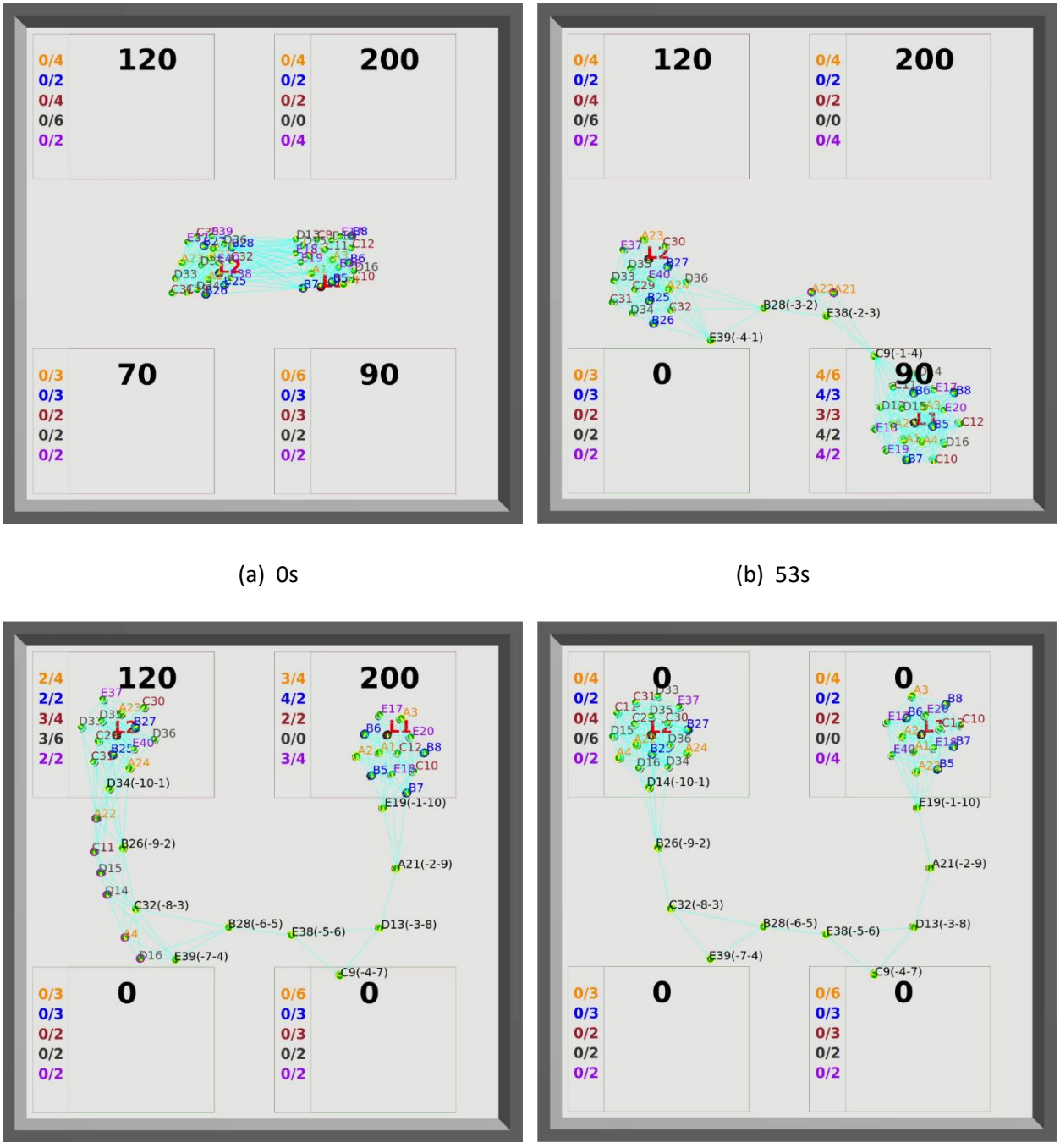
From the graphs, it can be seen that, for the first case with a robot ratio of 1:1, the fastest task completion time was noticed for the specification vector $(10,10)^T$, which was equivalent to 1:1. For the second case with a robot ratio of 1:3, the fastest completion time was for the specification vectors $(4,16)^T$ and $(6,14)^T$, which translated roughly to 1:3. The robot exchange was also minimal for these specification vectors as evident from the shorter travel distance for the travellers (figure 14). This contributed to faster task completion. This also reduced the follower travel distance since the more time the followers spend on tasks, the more it moves around due to their constant motion owing to the flocking behaviour. As the same pattern was observed in both sets of experiments, it can be concluded that the system performed best when the ratio of the number of robots of each type in the specification vector matches the ratio of the number of robots of each type in the team.

This can also be deducted rationally. The number of robots of each type forming a connector is proportional to the ratio of the number of robots of each type in the team. For example, if the ratio is 1:1, and if there are 4 connectors in the chain, on average 2 of them would be of type A and the remaining 2 of type B. Similarly, if the ratio was 1:3, then it would be 1 connector of type A and 3 connectors of type B. Therefore, as the team moves around, the ratio of the robots of each type in it remains the same. Thus, if the task specification vector had the same ratio, the less likely that an exchange is required.

Therefore, if prior information is available about the specification vector for the tasks, the number of robots of each type in the teams could be altered based on it to have the best performance.

4.2.5 Sample snapshots from trials

Figure 15(a-d) shows some snapshots at various time steps from the trial 5. The connector chain was stationary in this trial. There were 4 tasks and 5 robot types with 20 followers per team. Team 1 (L1) performed the two tasks on the right-hand side starting with the bottom one. Similarly, team 2 (L2) moves to the two tasks on the left-hand side in the same order. At the initial state (0s) both the teams were approximately at the centre. At 53s, team 2 finished its first task and is on the way to its second task. The robots ‘A21’ and ‘A22’ can be seen travelling from team 2 to team 1 through the connector chain as team 1 is 2 robots short of type A. At time 129s, 6 robots (2 of type A, 1 of type C, and 3 of type D) were travelling from team 1 to team 2. The tasks were completed at 234s.

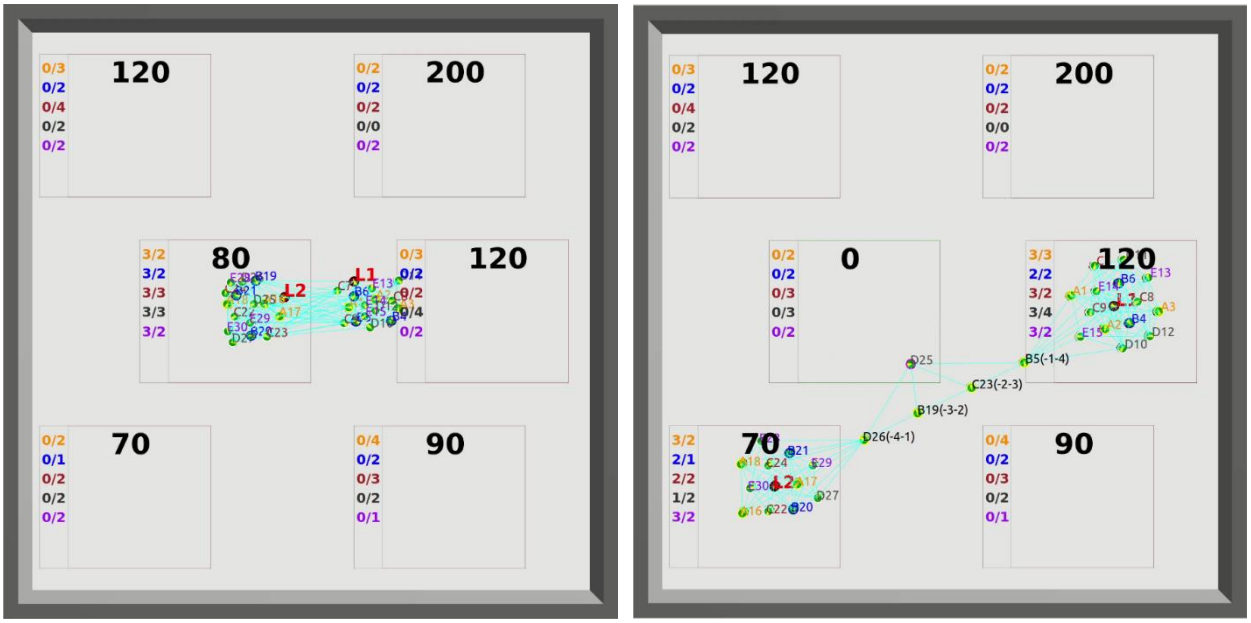


(a) 0s (b) 53s

(c) 129s (d) 234s

Figure 15: Snapshots from the simulation trial 5. (a) Simulation in the initial state (0s). (b-c) Simulation after 53 and 129 seconds. (d) Simulation after the tasks were over (234s).

Figure 16(a-d) shows instances from the trial 32. In this case, the connector chain was allowed to move to form a straight-line. There were 6 tasks in total and 15 followers per team. Team 1 executed the 3 tasks on the right-hand side, starting with the middle, then the bottom, and lastly the top. Team 2 performed the 3 tasks on the left-hand side in the same order.

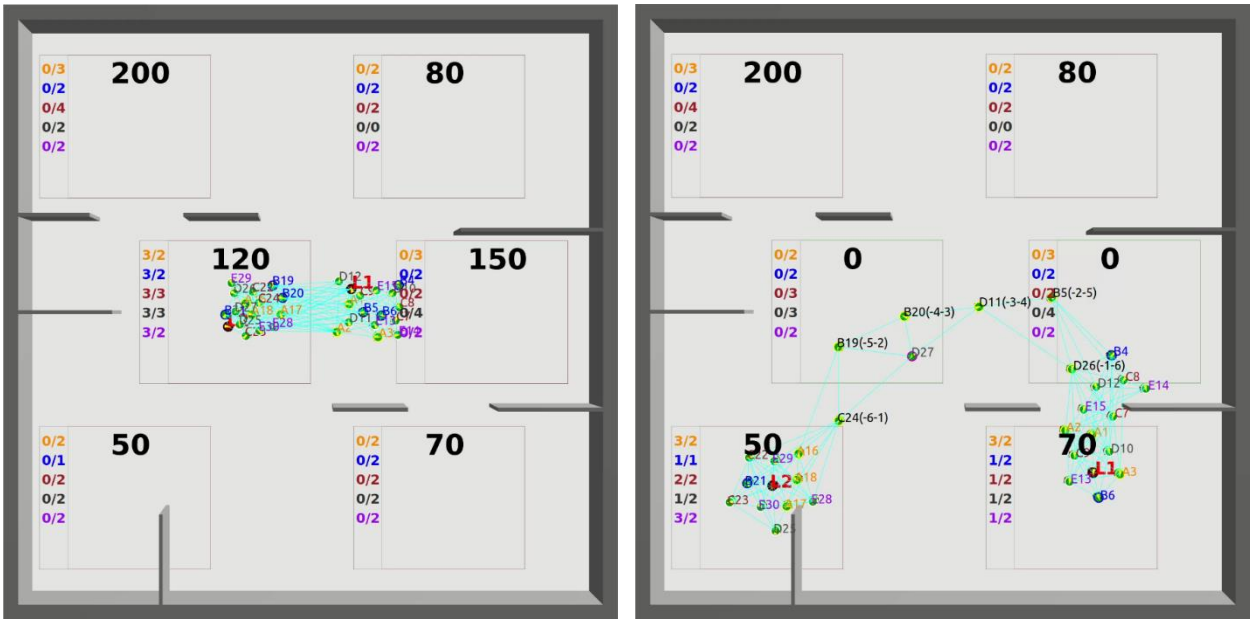


(a) 0s (b) 46s

(c) 196s (d) 270s

Figure 16: Snapshots from the simulation trial 32. (a) Simulation in the initial state (0s). (b-c) Simulation after 46 and 196 seconds. (d) Simulation after the tasks were over (270s).

Figure 17(a-d) show simulation trial 41 with obstacles in it. There were 6 tasks in total with 15 followers per team. The teams were able to successfully navigate to the task regions and exchange robots. Connector chain was made stationary.



(a) 0s (b) 110s

(c) 164s (d) 288s

Figure 17: Snapshots from the simulation trial 41. (a) Simulation in the initial state (0s). (b-c) Simulation after 110 and 164 seconds. (d) Simulation after the tasks were over (288s)

4.3 Limitations and solutions

During the trials in simulation, some limitations were noticed. A few other limitations that were not observed but possible to occur were also identified. These, along with some possible solutions are presented below:

1. When a large number of followers surrounded the leader, the leader was slow to move towards the task region as it was constantly getting blocked by the followers.

Solution: An extra condition can be introduced to the followers which prevent them from getting too close to the leader.

2. In a few trials (not listed in the table), the connector chain got disconnected from the team. This occurred when all the followers directly connected to the tail connector were chosen to be sent to the other team. When they become travellers, there were no other followers that were in direct contact with the tail connector, hence the team got separated from the chain. An example of such a possible scenario is given in figure 18. In the figure, the tail connector D33 is in direct contact with only the 4 followers A4, D14, D16, and E37. If all these 4 followers were exchanged, it will cause the team to break away from the chain.

Solution: An algorithm can be developed to check whether there exists at least one follower among the remaining ones after selection (for exchange) that is in direct contact with the tail connector, before sending the robots to the other team. If there exists no such follower, then at least one of the robots should not be sent to the other team and a different set of followers needs to be selected to send, if they are available.

3. Sometimes, the travellers get blocked by obstacles as they traverse along the left side of the chain, preventing them from joining the required team. An example situation is given in figure 19. If the connector C22 was closer to the obstacle wall on its left side, the travellers B6 and C7 would not be able to reach team 2 as there would be no space between the connector and the wall.

Solution: A new logic can be implemented for chain travel. When a traveller encounters an obstacle on the way, they could switch from the left side to the right side to avoid the obstacle.

4. Failure of a robot: If one of the followers fails, the system won't be affected if there is at least one other follower in the team. If it's the only follower, the system would be still

connected but the leader won't be able to move any further. If one of the connectors fails, the chain would break.

Solution: The distance between two connectors could be reduced such that any given connector is in direct contact with 2 other connectors in one direction, instead of one. In this case, even if one of the connectors failed, the system would remain connected. This is illustrated in figure 20. Consider connector D13. It is only directly connected to E39 and E18. But if it was also connected to B7 and A22, the system would remain connected even if any one of them fails. However, in the event one of the leaders fails, that team won't be able to function anymore. The other team would still work, but it won't be able to request robots.

5. Moving chain cant be used in the presence of obstacles.

Solution: A new logic could be introduced to form the moving chain in such a manner that no obstacle comes in between two connectors.

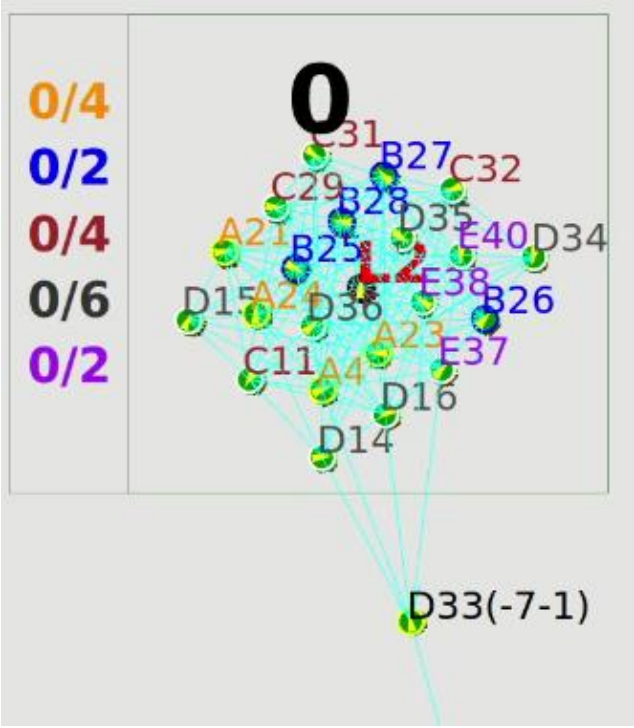


Figure 18: Example for a scenario in which the chain might break from the team.

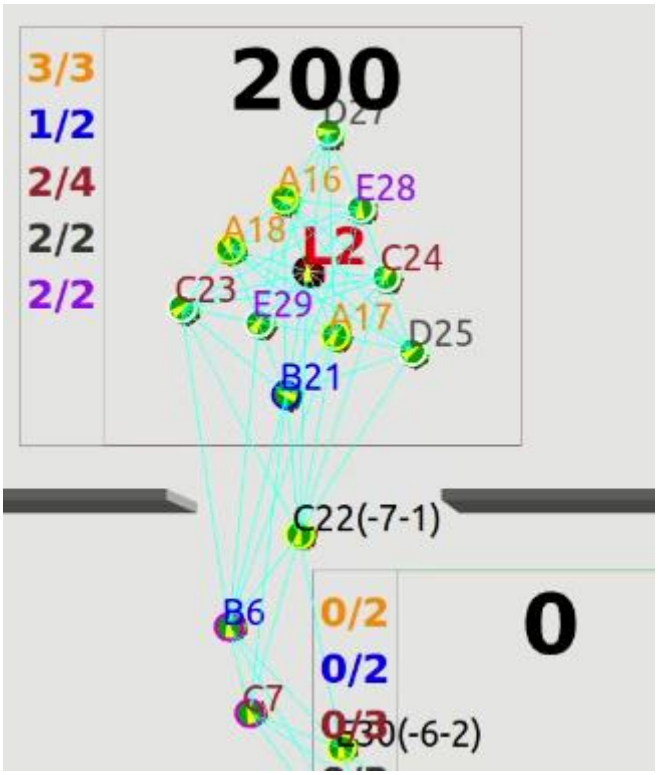


Figure 19: Example for a situation in which an obstacle might block a traveller robot

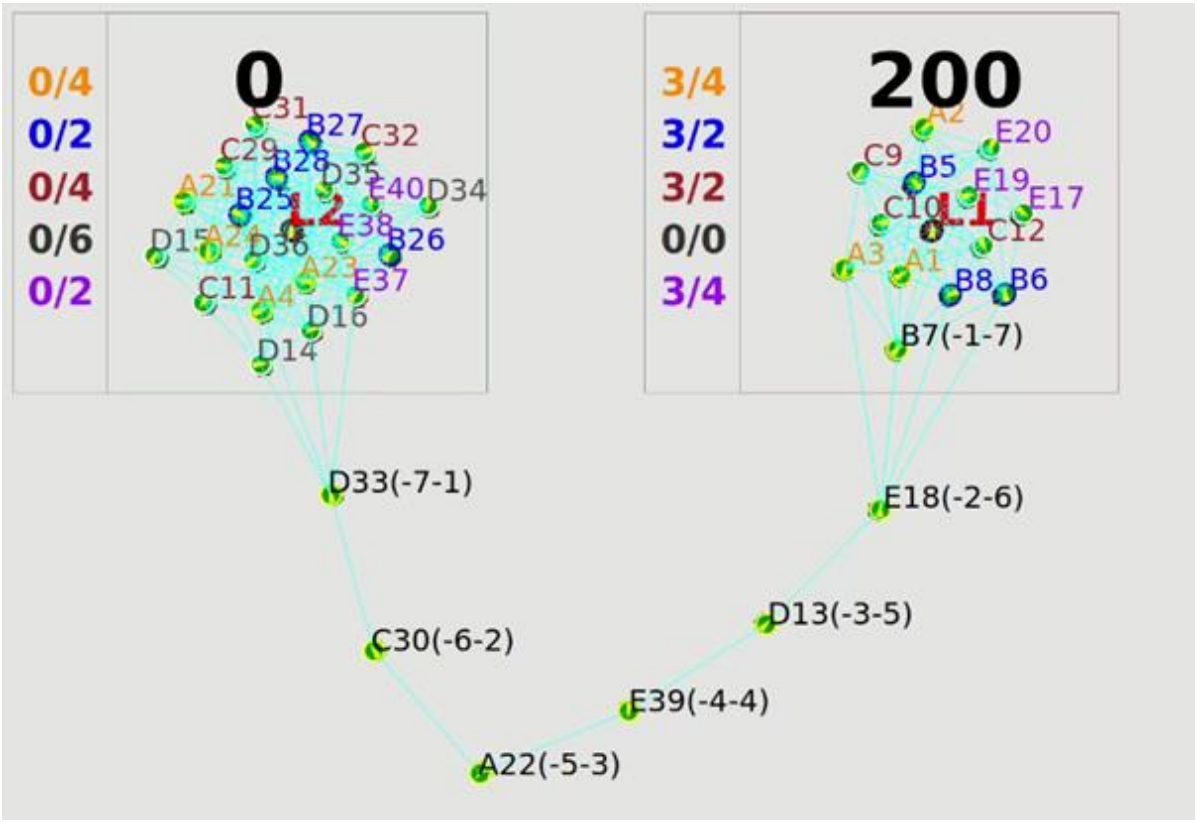


Figure 20: Adding a redundant connector to make the system resistant to failure of a connector robot

4.4 Summary

From the various trials, it was found that the system worked as intended in most of the scenarios. With the currently implemented algorithms, there was a risk of the connector chain breaking when dealing with a large number of followers. An additional algorithm was discussed which could possibly prevent the breaking of the chain. But the connectivity and exchange performed as expected when 40 or fewer followers were used. Another problem faced was the presence of obstacles. In certain cases, obstacles caused the chain to break. In some other cases, it prevented the exchange of workers from one team to the other. It was also unable to implement the algorithm to form a reasonable straight-line chain in the presence of obstacles. Possible solutions to address these were discussed in the previous section.

Additionally, it was realised that the system performs best when the ratio of robots of each type in the team is close to the ratio of robots of each type needed for each activity.

4.5 Extension to real robots

It was unable to test the system in actual robots since the fundamental goals took longer than anticipated. However, with a few adjustments, the created controllers could be uploaded to an e-puck robot. The leader can be an e-puck robot controlled manually via a computer. Provision can also be made to enable the sending and requesting of robots (through a computer application). The e-puck robots can be divided into several types. The same types of e-puck robots can turn on their LEDs in the same colour for simpler identification.

Often the system behaviour in simulation and real life can vary significantly. One possible issue that can arise is the noise in the sensor data. The noise can cause the system to behave unexpectedly. It is also possible for the e-puck to run out of memory while storing the messages from other robots. Another possible issue might be that there may be not enough bandwidth available for all the e-puck to communicate individually. Due to these possible issues, implementing the system in real e-puck robots could take up a significant amount of time.

Chapter 5

Conclusions and future work

This dissertation aimed to build on the work by Miyachi [2] and extend it to a heterogeneous system. Numerous modifications were done to the existing model and several new algorithms were developed to adapt it to the heterogeneous system. The new model was validated extensively in simulation. The system was tested in various scenarios to ensure its robustness. The system was found to perform as intended in most scenarios. The connectivity was preserved in nearly all the cases and the exchange of the robots was executed correctly. Experiments were also conducted to identify the ideal ratio of robots of each type in the team to have the best performance. Some of the limitations of the system and possible solutions were also reviewed. A brief look into the real robot implementation and possible constraints were discussed. Overall, the aims and basic objectives of the dissertation were met.

5.1 Future work

Some of the future works can be aimed at implementing the solutions for the limitations discussed in section 4.3. The solutions could then be validated in the simulation. The effect of the failure of a robot could also be tested and verified. More complicated obstacle shapes could be tested as well.

All the simulations were carried out in the same arena size of 4m x 4m. The system could be tested in arenas of different sizes. The size and shape of the task region could also be varied. More types of robots can also be tested. The model was designed to handle up to 10 robot types, though only up to 5 robot types were tested.

Furthermore, the model could be adapted to real robots and tested in real-world situations. This would help to understand how well the model can be transferred to a real system. Further changes to the model could be carried out based on real-world implementation.

References

[1] M. Brambilla, E. Ferrante, M. Birattari and M. Dorigo, “Swarm robotics: a review from the swarm engineering perspective,” *Swarm Intelligence,* no. 7, pp. 1-41, 2013.

[2] G. Miyauchi, Y. K. Lopes and R. Groß, “Multi-Operator Control of Connectivity-Preserving Robot Swarms Using Supervisory Control Theory,” in *IEEE International Conference on Robotics and Automation (ICRA)*, Philadelphia, USA, 2022.

[3] G. Miyauchi, Y. K. Lopes and R. Groß, “Multi-Operator Control of Connectivity-Preserving Robot Swarms Using Supervisory Control Theory: Supplementary Materia,” 2021. [Online]. Available: https://doi.org/10.15131/shef.data.16608421.

[4] C. W. Reynolds, “Flocks, herds and schools: A distributed behavioral model,” *ACM SIGGRAPH Computer Graphics,* vol. 21, no. 4, pp. 25-34, 1987.

[5] J. Li and Y. Tan, “A probabilistic finite state machine based strategy for multi-target search using swarm robotics,” *Applied soft computing journal,* no. 77, pp. 467-483, 2019.

[6] S. Nouyan, A. Campo and M. Dorigo, “Path formation in a robot swarm: Self-organized strategies to find your way home,” *Swarm Intell,* no. 2, pp. 1-23, 2008.

[7] W. Liu and A. F. T. Winfield, “Modeling and Optimization of Adaptive Foraging in Swarm Robotic Systems,” *The International Journal of Robotics Reasearch,* vol. 29, no. 14, pp. 1743-1760, 2010.

[8] O. Soysal and E. Şahin, “A Macroscopic Model for Self-organized Aggregation in Swarm Robotic Systems,” in *Lecture Notes in Computer Science*, 2006, pp. 27-42.

[9] O. Khatib, “Real-time obstacle avoidance for manipulators and mobile robots,” in *IEEE International Conference on Robotics and Automation*, St. Louis, MO, USA, 1985.

[10] J. H. Reif and H. Wang, “Social potential fields: A distributed behavioral control for autonomous robots,” *Robotics and Autonomous Systems,* no. 27, pp. 171-194, 1999.

[11] W. M. Spears, D. F. Spears, J. C. Hamann and R. Heil, “Distributed, Physics-Based Control of Swarms of Vehicles,” *Autonomous Robots,* no. 17, pp. 137-162, 2004.

[12] J. Beal and J. Bachrach, “Infrastructure for engineered emergence in sensor/actuator networks,” *IEEE Intelligent Systems,* vol. 21, no. 2, pp. 10-19, 2006.

[13] J. Bachrach, J. Mclurkin and A. Grue, “Protoswarm: A language for programming multi-robot systems using the amorphous medium abstraction,” in *7th International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Estoril, Portugal, 2008.

[14] M. Brambilla, C. Pinciroli, M. Birattari and M. Dorigo, “Property-driven design for swarm robotics,” in *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2012)*, Valencia, Spain, 2012.

[15] Y. Song, X. Fang, B. Liu, C. Li, Y. Li and S. X. Yang, "A novel foraging algorithm for swarm robotics based on virtual pheromones and neural network," *Applied Soft Computing Journal,* vol. 90, 2020.

[16] S. A., R. S, K. M and T. B, "Efficient spatial coverage by a robot swarm based on an ant foraging model and the Levy distribution," *Swarm Intelligence,* vol. 11, pp. 39-69, 2017.

[17] K. T, T. I and S. K, "Superadditive effect of multi-robot coordination in the exploration of unknown environments via stigmergy," *Neurocomputing,* vol. 148, no. 19, pp. 83-90, 2015.

[18] W. G.F, T. S, C. M, R. Z and G. J.W, "Ratiometric decoding of pheromones for a biomimetic infochemical communication system," *Sensors-Basel,* vol. 17, no. 11, pp. 1-16, 2017.

[19] A. Clark and A. Evans, "Foundations of the Unified Modeling Language," in *2nd BCS-FACS Northern Formal Methods Workshop*, Ilkley, UK, 1997.

[20] M. A. Haek, A. R. Ismail, A. Nordin, S. Sulaiman and H. Lau, "Modelling immune systems responses for the development of energy sharing strategies for swarm robotic systems," in *International Conference on computational science and technology (ICCST)* , Kota Kinabalu, Sabah, Malaysia, 2014.

[21] L. Pitonakova, R. Crowder and S. Bullock, "Behaviour-Data Relations Modelling Language For Multi-Robot Control Algorithms," in *International Conference on Intelligent Robots and Systems (IROS)*, Vancouver, BC, Canada, 2017.

[22] J. N. Pereira, P. Silva, P. U. Lima and A. Martinoli, "Formalizing institutions as executable petri nets for distributed robotic systems," in *Artificial Life Conference Proceedings*, 2011.

[23] P. J. Ramadge and W. M. Wonham, "Supervisory control of a class of discrete event processes," *Society for Industrial and Applied Mathematics,* vol. 25, no. 1, 1987.

[24] Y. K. Lopes, S. M. Trenkwalder, A. B. Leal, T. J. Dodd and R. Groß, "Supervisory control theory applied to swarm robotics," *Swarm Intelligence,* vol. 10, pp. 65-97, 2016.

[25] F. Mirzaei, A. A. Pouyan and M. Biglari, "Automatic controller code generation for swarm robotics using probabilistic timed supervisory control theory (ptSCT)," *Journal of Intelligent & Robotics Systems,* vol. 100, pp. 729-750, 2020.

[26] C. Pinciroli, V. Trianni, R. O'Grady, G. Pini, A. Brutschy, M. Brambilla, N. Mathews, E. Ferrante, G. D. Caro, F. Ducatelle, T. Stirling, A. Gutierrez, L. M. Gambardella and M. Dorigo, "ARGoS: a Modular, Multi-Engine Simulator for Heterogeneous Swarm Robotics," in *International Conference on Intelligent Robots and Systems (IROS)*, San Francisco, CA, USA, 2011.

[27] A. Jayasiri, G. K. I. Mann and R. G. Gosine, "Behaviour coordination of mobile robotics using supervisory control of fuzzy discrete event systems," *IEEE Transactions on systems, man, and cybernetics - Part B,* vol. 41, no. 5, pp. 1224-1238, 2011.

[28] J. Dulce-Galindo, M. A. Santos, G. V. Raffo and p. N. Pena, "Autonomous navigation of multiple robots using supervisory control theory," in *European Control Conference (ECC)*, Napoli Italy, 2019.

[29] Y. Tatsumoto, M. Shiraishi, K. Cai and Z. Lin, “Application of online supervisory control of discrete-event systems to multi-robot warehouse automation,” *Control engineering practice,* vol. 81, pp. 97-104, 2018.

[30] S.-L. Chung and S. Lafortune, “Limited Lookahead Policies in Supervisory Control of Discrete Event Systems,” *IEEE Transactions on Automatic Control,* vol. 37, no. 12, pp. 1921-1935, 1992.

[31] C. Ju and H. I. Son, “Modeling and Control of Heterogeneous Agricultural Robots Based on Ramadge–Wonham Theory,” *IEEE robotics and automation letters,* vol. 5, no. 1, pp. 48-55, 2020.

[32] L. P. Kaelbling, M. L. Littman and A. W. Moore, “Reinforcement Learning: A Survey,” *Journal of Arti,* vol. 4, pp. 237-285, 1996.

[33] M. J. Mataric, “Reinforcement Learning in the Multi-Robot Domain,” *Autonomous Robots,* vol. 4, p. 73–83, 1997.

[34] D. Floreano and S. Nolfi, Evolutionary Robotics, MIT Press.

[35] J. H. Holland, Adaptation in Natural and Artificial Systems, MIT Press.

[36] D. E. Goldberg, Genetic algorithms in search, optimization, and machine learning, Addison-Wesley, 1989.

[37] G. Baldassarre, V. Trianni, M. Bonani, F. Mondada, M. Dorigo and S. Nolfi, “Self-Organized Coordinated Motion in Groups of Physically Connected Robots - Part B,” *Transactions on systems, man, and cyberneticsTRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS,* vol. 37, no. 1, pp. 224-239, 2007.

[38] G. Pini and E. Tuci, “On the design of neuro-controllers for individual and social learning behaviour in autonomous robots: an evolutionary approach,” *Connection Science,* vol. 20, no. 2-3, pp. 211-230, 2008.

[39] R. Groß and M. Dorigo, “Evolution of solitary and group transport behaviors for autonomous robots,” *Adaptive Behavior,* vol. 16, no. 5, p. 285–305, 208.

[40] L. E. Parker, “ALLIANCE: An architecture for fault tolerant, cooperative control of heterogeneous mobile robots,” in *International Conference on Intelligent Robots and Systems (IROS)*, Munich, Germany, 1994.

[41] L. E. Parker, “Task-Oriented Multi-Robot Learning in Behavior-Based Systems,” in *International Conference on Intelligent Robots and Systems. (IROS)*, Osaka, Japan, 1996.

[42] R. J. Clark, R. C. Arkin and A. Ram, “Learning Momentum: On-line Performance Enhancement for Reactive Systems,” Georgia Institute of Technology, Atlanta, GA, 1992.

[43] J. B. Lee and R. C. Arkin, “Learning Momentum: Integration and Experimentation,” in *IEEE International Conference on Robotics 8 Automation*, Seoul, Korea, 2001.

[44] J. Lee and R. Arkin, “Adaptive multi-robot behavior via learning momentum,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Las Vegas, NV, USA, 2003.

[45] A. Cezayirli and F. Keresteciogl, “Navigation of non-communicating autonomous mobile robots with guaranteed connectivity,” *Robotica,* vol. 31, no. 5, pp. 767-776, 2013.

[46] N. Majcherczyk, A. Jayabalan, G. Beltrame and C. Pinciroli, “Decentralized Connectivity-Preserving Deployment of Large-Scale Robot Swarms,” in *International Conference on Intelligent Robots and Systems (IROS)*, Madrid, Spain, 2018.

[47] R. Maeda, T. Endo and F. Matsuno, “Decentralized Navigation for Heterogeneous Swarm Robots With Limited Field of View,” *IEEE Robotics and Automation Letters,* vol. 2, no. 2, 2017.

[48] C. Lin, W. Luo and K. Sycara, “Online Connectivity-aware Dynamic Deployment for Heterogeneous Multi-Robot Systems,” in *IEEE International Conference on Robotics and Automation (ICRA)*, Xi'an, China, 2021.

[49] Y. Yang, D. Constantinescu and Y. Shi, “Connectivity-preserving swarm teleoperation with a tree network,” in *International Conference on Intelligent Robots and Systems (IROS)*, Macau, China, 2019.

[50] D. Kengyel, H. Hamann, P. Zahadat, G. Radspieler, F. Wotawa and T. Schmickl, “Potential of Heterogeneity in Collective Behaviors: A Case Study on Heterogeneous Swarms,” in *PRIMA 2015: Principles and Practice of Multi-Agent Systems*, Bertinoro, Italy, 2015.

[51] M. Dorigo and e. al., “Swarmanoid: a novel concept for the study of heterogeneous,” *IEEE Robotics & Automation Magazine,* vol. 20, no. 4, pp. 60-71, 2013.

[52] H. Asama, A. Matsumoto and Y. Ishida, “Design of an autonomous and distributed robot system : ACTRESS,” in *International Workshop on Intelligent Robots and Systems (IROS)*, Tsukuba, Japan, 1989.

[53] P. Garcia, P. Caamaño, R. J. Duro and F. Bellas, “Scalable Task Assignment for Heterogeneous Multi-Robot Teams,” *International Journal of Advanced Robotic Systems,* vol. 10, 2013.

[54] A. Prorok, M. A. Hsieh and V. Kumar, “Fast Redistribution of a Swarm of Heterogeneous Robots,” in *EAI International Conference on Bio-inspired Information and Communications Technologies (BICT)*, New York City, US, 2015 .

[55] A. Kolling, P. Walker, N. Chakraborty, K. Sycara and M. Lewis, “Human Interaction With Robot Swarms: A Survey,” *IEEE transactions on human-machine systems,* vol. 46, no. 1, pp. 9-26, 2016.

[56] J. McLurkin, J. Smith, J. Frankel, D. Sotkowitz, D. Blau and B. Schmidt, “Speaking swarmish: Human-robot interface design for large swarms of autonomous mobile robots,” *AAAI Spring symposium,* pp. 72-75, 2006.

[57] J. Nagi, A. Giusti, L. M. Gambardella and G. A. D. Caro, “Human-Swarm Interaction Using Spatial Gestures,” in *International Conference on Intelligent Robots and Systems (IROS)*, Chicago, IL, USA, 2014.

[58] A. Kolling, K. Sycara, S. Nunnally and M. Lewis, “Human-swarm interaction: an experimental study of two types of interaction with foraging swarms,” *Journal of Human-Robot Interaction,* vol. 2, no. 2, pp. 103-129, 2013.

[59] S. Bashyal and G. K. Venayagamoorthy, “Human Swarm Interaction for Radiation Source Search and Localization,” in *IEEE Swarm Intelligence Symposium*, St. Louis MO, USA, 2008.

[60] P. Walker, S. A. Amraii, N. Chakraborty, M. Lewis and K. Sycara, “Human Control of Robot Swarms with Dynamic Leader,” in *International Conference on ntelligent Robots and Systems (IROS)*, Chicago, IL, USA, 2014.

[61] K. Xu and M. Gerla, “A heterogeneous routing protocol based on a new stable clustering scheme,” *MILCOM proceedings,* vol. 2, pp. 838-843, 2002.

[62] [Online]. Available: https://github.com/kaszubowski/nadzoru.git.

[63] [Online]. Available: https://drive.google.com/drive/folders/1qBljRXJ9TvxjLCjvg24fgDWlBPVm5Mxl?usp=sharing.

[64] [Online]. Available: https://gitlab.com/genki_miyauchi/argos-sct-swap.git.

Appendix A

Message structure used for communication

The structure of the message used in [2] along with its description is as follows:

Table 4: Contents of the original message structure and its description

CONTENT		DESCRIPTION
State		State of the robot (leader, follower, connector, follower)
ID		ID of the robot
Team ID		Team ID of the robot
Leader signal		Signal from the leader about task (start or stop)
Team Switch	• Robot to switch • Team to Join	• The ID of robot to exchange • The ID of team to join
Hop count	• Prefix • Team ID • Count • ID	• Number of messages • Team ID of the robot • Hop count • ID of robot with lower hop count
Connection message	• Prefix • Type • From • To • To team	• Number of messages • Message type (Request/Accept) • ID of robot the message if from • ID of robot the message is to • ID of the team the message is to
Shared message	• Share to leader • Share to team • Share distance	• ID of tail connector send to leader • ID of tail connector send to team • Distance to tail connector
Teams nearby	• Prefix • Team ID	• Number of messages • ID of the nearby team
Relay message	• Prefix • Type • From • Time • First follower • Follower number • Task minimum number • Robot number	• Number of messages • Message type • ID of the leader the message is from • Time at which message is send • First follower to receive message from chain • Number of followers in the team • Minimum number of robots required for task • The number of robots to request
Connections		ID of the nearby robots

The structure of the message used in this dissertation is as follows (changes highlighted in red):

Table 5: Contents of the updated message structure and its description

CONTENT	DESCRIPTION
State	State of the robot (leader, follower, connector, follower)
ID	ID of the robot
Team ID	Team ID of the robot
Leader signal	Signal from the leader about task (start or stop)
Team Switch	
• Robot to switch	• The ID of robot to exchange
• Team to Join	• The ID of team to join
Hop count	
• Prefix	• Number of messages
• Team ID	• Team ID of the robot
• Count	• Hop count
• ID	• ID of robot with lower hop count
Connection message	
• Prefix	• Number of messages
• Type	• Message type (Request/Accept)
• From	• ID of robot the message if from
• To	• ID of robot the message is to
• To team	• ID of the team the message is to
Shared message	
• Share to leader	• ID of tail connector send to leader
• Share to team	• ID of tail connector send to team
• Share distance	• Distance to tail connector
Teams nearby	
• Prefix	• Number of messages
• Team ID	• ID of the nearby team
Relay Message (Heartbeat)	
• Prefix	• Number of messages
• Type	• Message type (heartbeat, request, response)
• From	• ID of the leader the message is from
• Time	• Time at which message is send
• Follower number per type	• Number of followers in the team of each type
• Task minimum number per type	• Minimum number of robots required for the task of each type
• Robot number per type	• The number of robots to request for each type
Connections	ID of the nearby robots
Robot type	Type of the robot

The ‘first follower’ in the relay message was not required. Therefore, it was removed. The ‘follower number’, ‘task minimum number’, and ‘robot number’ were modified to incorporate details for up to 10 robot types. A new message content ‘robot type’ was added to identify the type of each of the robots.