Multi-Operator Control of Connectivity-Preserving Robot Swarms

Genki Miyauchi

Supervisors: Dr Roderich Groß
Prof. Sanja Dogramadzi

Department of Automatic Control and Systems Engineering
The University of Sheffield

A Thesis Submitted for the degree of
Doctor of Philosophy

February 2024

Abstract

Robot swarms hold the potential to help humans accomplish many challenging problems. Prior research on human-swarm interaction systems has primarily focused on the interaction with a single human operator. This inability to collaborate with multiple humans may limit the swarm's use in practical scenarios. This thesis considers how multiple humans who are working alongside a robot swarm can effectively use the robots as a shared resource. In an environment that lacks global communication, we consider a scenario where humans freely explore the area to complete spatially distributed tasks. First, we present a framework based on formal languages that allow robots to help a pair of humans complete tasks while preserving the connectivity of the communication network between them. The framework supports automatic code generation to control the robots in a fully distributed manner. Next, we conducted a user study with 52 participants to investigate the usability of the framework under different communication constraints. The participants were each presented with a graphical user interface that provided only a local, first-person view of the environment. Finally, the framework is extended to support an arbitrary number of humans to share the control of a swarm. The framework is first verified in simulation and then through physical robot experiments to demonstrate their application in real life. Results show that the framework creates communication networks that can adapt to the movement of humans, enable humans to exchange robots depending on changing task demands, and are near-optimal with respect to network length and the number of robots required to maintain it. Moreover, they suggest that the ability to exchange robots in the swarm offers humans the flexibility to work independently or collaboratively depending on the current situation. This work can be seen as a step towards designing effective human-swarm teams.

Acknowledgements

I would like to begin by thanking my supervisor Dr. Roderich Groß for all his excellent support and encouragement throughout my PhD. His guidance and advice have been invaluable to my own development as a researcher. I am incredibly grateful for being able to work with him. I would also like to thank my second supervisor, Prof. Sanja Dogramadzi, for her support when needed. I am thankful to my department and the OpenSwarm project for their financial support.

I would like to thank Dr. Yuri Kaszbowski Lopes for his valuable knowledge and input. He has massively helped me understand and apply supervisory control theory during the early stages of my work.

I am grateful to those who have helped me prepare for the physical robot experiments. Thanks to Dr. Alan Millard for kindly sharing the Pi-pucks and the arena originally prepared for the SwarmHack 22' event, which has helped me easily verify the work in this thesis using real robots. Thanks to Garry Turner for helping me with all kinds of hardware-related issues in the lab.

Many thanks to all members of the Natural Robotics Lab—Anil, João, Matt Doyle, Yue, Matt Hall, Ed, Jahir, Zheshui, and Stanley—for the fruitful and exciting discussions we had on each other's works. Thanks to my office-mates—Fadl, Fethi, Pablo, Fuhai, and Rea—who I have shared the time with during my PhD journey. I would also like to thank everyone who has expressed their interest in swarm robotics and kindly participated in my user studies.

Finally, I am most grateful to my parents for their enduring support and encouragement throughout my entire life. Thank you for giving me all of the opportunities I've had and for always being there for me!

Table of contents

Chapter 1

Introduction

To improve the world that we live in, humans have continued to push the boundaries of technological development. The advancements have led to the birth of a wide variety of innovations, from the powerful machines seen in factories to the electronic devices featuring intelligent software we use every day. Autonomous robots are one of the technologies that can support humans in achieving a diverse set of challenging tasks. Robots can not only perform repetitive tasks with incredible speed and accuracy but have great potential to assist humans in achieving their goals, enhance or augment human abilities, and provide companionship. With such a wide variety of possible applications, intelligent autonomous robots that can team up and work alongside humans are highly anticipated.

Humans and many animals in nature often gather and work together to accomplish complex tasks that cannot be done individually. Similarly, a group of robots could also be made to self-organise and work together by coordinating their actions with one another. One way of achieving such coordination in robots can be found in the interaction observed in social animals, where the group exhibits a sort of collective intelligence (Bonabeau, 1999). Swarm robotics applies this idea of collective intelligence to allow large groups of robots to accomplish complex tasks using simple, local interactions in a distributed manner (Hamann, 2018). This makes the swarm highly scalable to different group sizes (Brambilla et al., 2013) and allows multiple operations to run in parallel (Dorigo et al., 2020), which can result in faster task completion and adaptation to dynamically changing environments and task requirements. Therefore, robot swarms are a promising technology for performing tasks that are scattered over large, unknown environments, such as environmental monitoring of forests (Tarapore

et al., 2020), ocean (Duarte et al., 2016), and disaster response (Carrillo-Zapata et al., 2020; Saez-Pons et al., 2010).

As the push to bring robot swarms into the real world continues, it is important to consider how we leverage the autonomy of robot swarms to support humans. In many real-world scenarios, tasks could have several constraints that are difficult for a robot swarm to detect but are either already known or easily identifiable to humans, such as inter-dependencies between tasks, changing priorities and deadlines, and the number of robots needed to complete each task. In such scenarios, it would be beneficial for a human operator to be able to make decisions and exert control or influence over the swarm (Kolling et al., 2016): for example, an operator could provide locations the swarm must attend to, and in what order they must visit these locations. Robot swarms that could be controlled intuitively by a human operator would further improve their usefulness in such scenarios. Recent advances in human-swarm interaction (HSI) have shown various ways a single human operator can control a swarm of robots, both during remote (Jang et al., 2021; Patel et al., 2019) and/or proximal (Gromov et al., 2016; Nagi et al., 2014) interactions depending on whether or not the operator is physically present within the environment.

Robot swarms are not restricted to interacting with a single human operator. A group of operators could potentially use a robot swarm more effectively than a single operator by exploiting the characteristics of robot swarms, such as scalability and flexibility (Patel et al., 2022). This gives rise to a human-swarm team that involves multiple human operators working together while interacting with a robot swarm. Such human-swarm teams could enable robot swarms to be deployed at a larger scale while also reducing the complexity experienced by individual human operators and making them applicable to an even wider range of scenarios. The work in this thesis considers such human-swarm teams that enable multiple human operators to exert shared control over a swarm of robots. By exploring new ways for multiple human operators to interact with robot swarms, the potential application areas of the technology could become almost limitless.

1.1 Motivation

One particularly important issue that has been overlooked in human-swarm interaction is the ability of multiple human operators to interact with a robot swarm. As a swarm grows in size, it may split into smaller subgroups to work on different tasks in parallel.

Managing an increasing number of subgroups quickly becomes difficult for a single human operator as the amount of information they receive is multiplied, ultimately overwhelming their capacity (Olsen and Wood, 2004). In such cases, the presence of an ineffective human operator becomes a hindrance for the robot swarm and may undermine its advantages and capabilities (Kapellmann-Zafra et al., 2016). Introducing multiple human operators could help solve this issue by distributing the overall workload experienced by each operator. Early works on multi-human multi-robot interaction have started to emerge in the past few years, yet research in this area is still very limited. There is a great need to explore different ways for such human-swarm teams to work together effectively.

An area where human-swarm teams would be highly beneficial is when tasks are scattered over large, unknown environments, such as those encountered in search and rescue (Delmerico et al., 2019). For example, a group of human operators may explore a site as rescuers to search for victims. The robots work alongside the operators to augment their abilities to carry out various tasks. Upon identifying tasks that must be completed, the operators could each focus on a different task using a specific subgroup of robots assigned to them. Furthermore, the operators could exploit the scalability and flexibility of robot swarms by sharing some of the robots they control with each other in response to changing task requirements. This would allow an operator with an excess number of robots in their group to offer some of their robots to other operators that require more to carry out their tasks. Such flexibility to dynamically share robots between operators and to adjust the size of subgroups would improve the effectiveness of the human-swarm team in a wide range of scenarios.

The ability to exchange information is often considered a prerequisite for humans to cooperate effectively (Marlow et al., 2018). However, exchanging information can be challenging in environments where global communication is not available. A promising approach for interconnecting separate locations is to have a robot swarm dynamically establish an ad-hoc network (Ghosh et al., 2019). Operators could use this network to exchange information with others to coordinate their actions, such as deciding the order of tasks they should attend to and exchanging robots among themselves to meet task requirements.

Another important aspect of human-swarm interaction is to make the swarm transparent and trustworthy (Wilson et al., 2023). This is especially true during proximal interaction where ensuring the safety of the humans working alongside the robot swarm is crucial. One way to help build trust is the use of formal methods when

designing the swarm controller. Formal methods are mathematical tools that rigorously verify the correctness of a system to guarantee that it exhibits the desired behaviours based on the specifications. They can also serve as documentation for the behaviours exhibited by individual robots, which could help improve both the trust of the swarm designer and the user towards the swarm (Lopes, 2016).

Designing the behaviour of a robot swarm is challenging due to its increasingly complex nature (Dorigo et al., 2020). Supervisory control theory (SCT) is a formal approach to designing the behaviour of swarm robotic systems (Lopes et al., 2016; Ramadge and Wonham, 1987). SCT assumes that the system to control can be represented as a discrete event system and uses formal languages to model the capabilities of the system. A designer would formally model the capabilities of and specifications for a swarm robotic system, which are then synthesised to produce the swarm controller. SCT has also been shown to support automatic code-generation, ensuring that the implementation accurately represents the models and reducing the amount of ad-hoc development required by the designer (Lopes et al., 2016). The synthesised controller provides guarantees that the system is non-blocking and will adhere to the designed specifications. Furthermore, the formalism introduced by SCT helps the swarm designer identify any contradictions or errors in the defined specifications at the design stage. This is useful as it helps reduce the amount of trial and error needed while implementing swarm behaviours in computer simulation or for real robots. However, SCT has not yet been applied to HSI systems.

1.2 Aims and Objectives

The aim of this thesis is to study how a robot swarm could be effectively shared among multiple human operators who are working on spatially distributed tasks. The robots must maintain a communication network between the operators while they freely explore the environment. Moreover, considering the deployment of the transparent and trustworthy HSI system in the real world, formal methods should be adopted for the design and development of the swarm.

The following objectives have been defined to fulfil this aim:

- To develop a framework for robot swarms that will enable multiple human operators to share the control of the robots. The robot swarm must autonomously

maintain connectivity between the operators, while also being available for the operators to perform tasks.

- To verify and analyse the performance of the framework with a large number of robots in simulation.
- To conduct a user study to assess the usability of the framework.
- To demonstrate how the framework could be applied in the real world using physical robotic platforms.

1.3 Preview of Contributions

The contributions of this thesis are as follows:

- The design and implementation of a framework for a robot swarm to help a pair of human operators complete spatially distributed tasks. The operators freely explore an environment to identify tasks, while the robot swarm help the operators complete the tasks and dynamically assign robots to preserve the connectivity of the communication network between the operators. A formal approach based on SCT is used to design the controllers to ensure the robots exhibit the desired behaviour based on the specifications.
- An extension of the aforementioned framework to allow human operators to dynamically share information and robots with each other. It allows an operator to send a request message via the communication network to the other operator when more robots are needed for a task. If the request is accepted by the receiving operator, robots are automatically dispatched from the team, which travel along the network to join the requesting operator's team. Simulation studies show that the ability to share robots between operators can help them complete tasks faster and reduce the total distance travelled by the swarm.
- A systematic investigation of the usability of the framework and the effects of communication on a pair of human operators sharing the control of a robot swarm. A user interface was developed to allow operators to search for tasks in a simulated environment and send robots to each other. Results from 52 participants show that the ability to share robots encourages operators to flexibly switch between working independently and collaboratively depending on the

situation and reduces the energy consumed by the swarm with the cost of a slight increase in the experienced workload.

- A further extension of the framework to support an arbitrary number of human operators to collaboratively work together. Connectivity maintenance between all operators is realised through robots that can dynamically create and reposition branches in the network. The quality of the networks is shown to have near-optimal lengths and number of robots. The framework scales well when the number of teams is increased as evidenced in a simulation study consisting of up to 6 operators and 144 robots. The network is also resilient against a large group of up to 40 robots travelling along the network.
- Verification of the framework using physical robotic platforms, demonstrating how it can provide a viable method for multiple human operators to share the control of a robot swarm. Experiments with physical Pi-puck robots controlled by simulated/real human operators demonstrate the applicability of the proposed framework in the real world.

1.4 Publications

This thesis contains the author's original work and has so far led to two peer-reviewed publications:

- **G. Miyauchi**, Y. K. Lopes, and R. Groß, "Multi-Operator Control of Connectivity-Preserving Robot Swarms Using Supervisory Control Theory," in *2022 IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 6889–6895.
- **G. Miyauchi**, Y. K. Lopes, and R. Groß, "Sharing the Control of Robot Swarms Among Multiple Human Operators: A User Study," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023.

These publications were presented orally by the author at their respective conferences, which were held in Philadelphia (USA) and Detroit (USA).

The author also contributed to another project that is not featured in this thesis, which has led to the following publication:

- F. Pratissoli, A. Reina, Y. K. Lopes, C. Pinciroli, **G. Miyauchi**, L. Sabattini, and R. Groß, "Coherent Movement of Error-Prone Individuals Through Mechanical Coupling," *Nature Communications*, vol. 14, no. 1, p. 4063, 2023.

1.5 Thesis Outline

The remainder of this thesis is organised as follows:

- Chapter 2 presents the background and related work for this thesis. Section 2.1 begins the chapter with an overview of the field of swarm robotics. Current research areas in human-swarm interaction are described in Section 2.2, followed by a review of the existing work that has considered multi-human swarm interaction in Section 2.3. Existing formal design methods in swarm robotics are discussed in Section 2.4. Section 2.5 introduces the concepts of SCT used to design the robot controllers in this thesis. Section 2.6 concludes the chapter.

- Chapter 3 presents a framework for a swarm of robots to maintain connectivity and enable the sharing of robots between two human operators. Section 3.1 provides a formal description of the problem. The framework is then presented in Section 3.2, which describes how the swarm maintains connectivity and how the robot's control logic is designed using SCT. Simulation studies are performed in Section 3.3 to demonstrate the performance of the proposed method. Section 3.4 concludes the chapter.

- Chapter 4 presents a user study conducted to investigate the usability of the framework. Section 4.1 describes the scenario considered in this work, the system and user interface developed for the study, and the experiment design and procedure. Section 4.2 presents the experiment and questionnaire results obtained from a total of 52 participants, which is analysed in Section 4.3. Section 4.4 concludes the chapter.

- Chapter 5 presents an extension of the framework for sharing the control of robot swarms and maintaining connectivity between an arbitrary number of human operators. Section 5.1 provides a formal description of the problem. Section 5.2 presents the robot controller that dynamically branch and optimise the structure of the network. Simulation studies are performed in Section 5.3, which includes analyses on scalability and congestion. Section 5.4 concludes the chapter.

- Chapter 6 presents a validation of the multi-human-swarm interaction system using physical robotic platforms. The design of the system architecture, the arena, and the robot platform used are described in Section 6.1. Real robot experiments with simulated and human operators are presented in Section 6.2. Section 6.3 concludes the chapter.
- Chapter 7 concludes this thesis, highlighting the significance of the findings. Finally, Section 7.1 discusses potential directions for future work.

Chapter 2

Background

This chapter reviews the existing literature related to the work presented in this thesis. Section 2.1 provides a brief introduction to swarm robotics and how they have been applied to solve spatially distributed tasks. Section 2.2 provides an overview of human-swarm interaction systems explored by researchers for a single human operator to control a robot swarm and design considerations for such interaction systems. Section 2.3 further extends this discussion to multiple human operators, particularly for swarms that may split into smaller subswarms, which is the main interest of this thesis. Section 2.4, investigates a range of formal methods applied to the design of robot swarms. Section 2.5 introduce preliminaries for supervisory control theory, which is the formal approach used for designing the robot controllers throughout this thesis.

2.1 Swarm Robotics

Swarm robotics is a subfield of multi-robot systems (MRS) that investigates the coordination of a large number of autonomous robots that accomplish tasks by locally interacting with one another and the surrounding environment. These systems are characterised by their distributed sensing capabilities and decentralised algorithms that give rise to the emergence of global behaviour.

Potential advantages of swarm robotics systems over a single robot are their high robustness, scalability, and flexibility (Şahin, 2004). Robustness is the ability to continue performing a task despite the failure of a few individuals. Scalability is the ability to operate under largely different group sizes. Flexibility is the ability to adapt to a wide range of tasks and environments. These characteristics are ideal

for simultaneously performing many tasks in large, unstructured, and unpredictable environments, where it may be difficult or even impossible for a single robot to complete.

This thesis is mainly concerned with human operators utilising a swarm of robots to perform spatially distributed tasks in environments that lack the infrastructure to facilitate global communication. For the remainder of this section, we focus our review on how robot swarms are able to achieve division of labour and connectivity maintenance to complete these tasks.

2.1.1 Task Allocation

Multiple tasks can be executed simultaneously by allocating one or more robots to work on each task. The high scalability of robot swarms makes them suitable for completing multiple tasks as they can split into smaller "subswarms". The number of robots in a subswarm can be modified dynamically to adapt to different task requirements, which may also change over time. However, the formation of such task-oriented groups in a swarm of robots is non-trivial.

In MRS, the assignment of a set of robots to discrete tasks is known as the multi-robot task allocation (MRTA) problem (Gerkey and Matarić, 2004). As the coordination between robots also needs to be taken into account, finding an optimal allocation for MRTA is known to be NP-hard (G. Lagoudakis et al., 2005). A well-studied method for finding a near-optimal solution to the allocation problem is to use auction mechanisms (Kalra et al., 2005; Smith, 1980). In these methods, an auctioneer agent announces a task to all robots, while the robots in turn each respond with a bid that represents their own self-assessed suitability for the task (Koenig et al., 2006). Given the received bids, the auctioneer agent decides which robots are assigned to which tasks. This allows the computation to be distributed among all the robots involved in the auction. However, most auction mechanisms assume bids to be exchanged via global communication and are directly sent or relayed to a central auctioneer agent. This makes them prone to scalability issues and difficult to directly apply in swarm robotics. Furthermore, a lot of focus is placed on the allocation problem of tasks that can be carried out by a single robot, while robot swarms usually require multiple robots to perform a task (also known as *multi-robot* tasks (Gerkey and Matarić, 2004; Schneider et al., 2016)).

A more swarm-like solution to such "division of labour" is using response thresholds (Bonabeau, 1999). Kanakia et al. (2016) used response thresholds to recruit an approximate number of desired robots for a given task. Each robot's "willingness" to

work on the task is determined by a sigmoid function that takes the estimated number of robots at the task location as input. Hsieh et al. (2008) proposed a control policy that increased the probability of a robot to transition to a neighbouring position in a network of nodes depending on the current number of robots. A similar method of swarm redistribution was demonstrated by Zahadat and Schmickl (2016) with underwater robots.

When different types of tasks are present in the same area, robots could assume different roles rather than splitting into physically separated groups. Pini et al. (2009) examined task allocation in a foraging problem that involved task allocation and role switching. The robots assumed the role of either a harvester or a storer to collect or store a resource in the nest, respectively. Once a harvester collected a resource, it carried the resource near the nest and waited for a storer, who then retrieved the resource and stored it in the nest. A robot switched between the harvester and storer roles if they waited to exchange a resource longer than a predefined time, as it meant there was an insufficient number of robots in the other role. This allowed the robots to dynamically adapt their numbers in each role depending on the number of resources in the environment at that time. Sequential task execution in a foraging problem was later shown by Garattoni and Birattari (2018).

For robot swarms to perform more complex tasks, some level of hierarchy is likely to be needed to manage the swarm (Dorigo et al., 2020). One option would be to use informed leaders who could recruit robots that are required to perform a specific task. Pinciroli et al. (2009, 2010) developed a recruitment system for heterogeneous swarms. Each aerial robot acted as an informed leader that recruited wheeled robots moving randomly on the floor. A wheeled robot under an aerial robot's range of influence had a probability to join or leave the aerial robot's group. This enabled the aerial robots to guide the required number of wheeled robots to each task location. While this approach successfully managed to gather and allocate robots to the tasks, robot recruitment was performed using probabilities, causing perturbations during subgroup allocation. The informed leaders were also required to travel back to the central recruitment area whenever new robots were needed.

2.1.2 Connectivity Maintenance

In many scenarios, we require subswarms that are working on different parts of the environment to maintain connectivity among themselves. For example, it enables the

subswarms to share task-related information, exchange robots, or merge together to form a larger subswarm if necessary.

Sharing task-related information can be observed in nature. For example, ants use pheromone trails to create a path between their nest and valuable food resources. The pheromone trails allow the ants to communicate the presence of resources and serve as a guide for others to navigate the environment to reach them (Deneubourg et al., 1990). The idea of modifying the environment has inspired the design of many foraging strategies in swarm robotics. Mayet et al. (2010) used robots that could draw glow paint containing small grains of phosphorescent material on the ground, which was then detected by other robots using ultraviolet light to navigate between the nest and a food resource. Artificially depositing chemicals to the environment, however, is challenging to achieve with existing technologies and may not be suitable or possible to modify the environment in many situations. Instead, methods that emulate pheromones by broadcasting virtual pheromones to neighbouring robots using local short-range communication have been proposed (Payton et al., 2001; Talamali et al., 2020). While pheromone-based methods allow robots with extremely limited capabilities to navigate between two distant locations, they cannot be used to share task-related information in real-time as the information obtained from the deposited pheromone may be outdated.

A popular approach to maintain connectivity is to organise the robots into a chain-like formation to create an ad-hoc network. Nouyan et al. (2008, 2009, 2006) proposed a method for a group of s-bots (Dorigo et al., 2005) in a foraging task to autonomously align themselves into a chain-like structure that extends from the nest towards a target object. Robots in the chain emit either a blue, green, or yellow colour using the LEDs surrounding their bodies. These three colour sequences are repeated over the chain to provide a sense of direction, which encourages other robots to move towards the open end of the chain and extend it further by adding themself to the chain. Once the chain reached the target object, the chain was used as a path to guide the robots carrying the object back to the nest. By using the robots' bodies as landmarks, the chain enabled robots that lack global navigation capabilities to coordinate their behaviours over long distances. Similar chain formation using heterogeneous robot swarms was demonstrated by Dorigo et al. (2013) to form a wireless network that connected a base station and a task location.

Robots forming a chain can also create branches to form a larger network that connects more than two locations. The global maintenance of such networks has

been an important topic in MRS due to its high relevance in real-world applications. Majcherczyk et al. (2018) proposed a robot swarm that maintained connectivity by extending the network to form a tree-like structure. The robots' objective was to allow the task robots to reach multiple spatially distributed task locations while maintaining connectivity between them. Similar approaches have also been demonstrated by others (Chandarana et al., 2018; Molins et al., 2019). Lee et al. (2022) applied tree-like networks to robots equipped with a conveyor belt. The robots autonomously extended the network and aligned their conveyor belts so that resources could be carried back to the base station.

When robots do not have prior knowledge of the environment, they must dynamically build and maintain the network throughout the mission. One method to extend the network is using distance-based methods, where new relay robots are added to the network when the distance between robots exceeds a safety threshold. Varadharajan et al. (2019, 2020) proposed an algorithm for robots to progressively form a chain connecting a base station and a leader robot that is moving towards a target location. New robots extended the chain whenever the leader robot's distance to the chain exceeded a certain threshold. An alternative method is to use the received signal strength indicator, which is a measurement of how well a robot can receive a signal from another robot in the network. Abu-Aisheh et al. (2022) proposed an algorithm that utilised signal strength to determine whether new relay robots were needed to expand the swarm's coverage. By measuring the signal strength from each robot, an orchestrator chose a robot to become a relay robot at a specific location.

Another important aspect is the capability to dynamically shorten the branches in the network when they are no longer needed or modify the network topology to adapt to the changing environment. This helps in tackling a common challenge in connectivity maintenance; to minimise the number of relay robots involved in maintaining the network. Yi et al. (2021) proposed a topology correction controller, where relay robots actively inserted themselves in the network to maximise the number of task robots that can reach their respective task locations. The idea of topology correction was further extended by Soma et al. (2023), who showed a complete set of local topology modifications that robots can apply in sequence to transform from any given tree structure to another. The approach is distributed and requires each robot to have access to neighbours that are only two hops away. The works by Lin et al. (2021); Luo et al. (2020a,b) demonstrated robots that preserve global algebraic connectivity instead of local connectivity of individual robots using control barrier functions, though it has only been shown using centralised approaches.

All of the above examples considered the maintenance of global connectivity among fully autonomous robots, but none of them considered the involvement of human operators. Such involvement would be particularly useful in scenarios where human operators could use the network or influence the topology of the network to accomplish their desired tasks. For example, a swarm of robots could be used to maintain the connectivity between a human rescuer and a base station during a search-and-rescue mission and dynamically adapt its topology to the human who is freely exploring the environment (Min et al., 2018). While human involvement has often been shown to improve the performance of robot swarms, very few have considered the involvement of human operators that interact with connectivity-preserving robot swarms.

2.2 Human-Swarm Interaction

In order to deploy robot swarms in the real world, it is important to consider how humans can supervise and interact with them when necessary. However, the distributed nature of robot swarms can pose a number of challenges for a human operator. This has led human-swarm interaction (HSI) to become an active research topic in recent years.

2.2.1 Cognitive Complexity

Cognitive complexity is an operator's "cognitive effort" to manage a group of robots performing independent tasks (Kolling et al., 2016). The concept originates from *computational complexity* in computer science, where it is used to define the time required for an algorithm to solve a problem that scales as the input size increases. A cognitive complexity of $O(1)$ represents a human operator interacting with a single robot. A complexity of $O(n)$ represents an operator interacting with n robots, so the time an operator spends on each robot reduces as n becomes larger. $O(> n)$ defines the extreme case, where the operator must also handle the interactions between individual robots. This is problematic for human operators interacting with robot swarms since the number of robots involved can range anywhere between a few dozen to over a hundred; it is not realistic for an operator to interact with every robot in the swarm without being overwhelmed. The fan-out model is a related concept, which was proposed by Olsen and Wood (2004) to describe and predict the number of robots a single operator can control simultaneously. The presence of a large number of robots

can also provoke an operator's psychophysiological state to cause stress and anxiety (Podevijn et al., 2016), further increasing the likelihood of being overwhelmed by the swarm.

Instead of having to interact with individual robots, a human operator's cognitive complexity can be significantly decreased if the interaction is abstracted to the swarm-wide level. This allows a human operator to consider the swarm as a single entity, which reduces the cognitive complexity to $O(1)$. However, when the robot swarm splits into multiple subswarms, a similar situation arises where the human operator is again faced with the problem of interacting with potentially many subswarms (Kolling et al., 2016).

So far, we have assumed that the robots' productivity increases when the human operator is able to spend more attention on each robot and decreases when the robots receive less attention. This concept is known as *neglect tolerance* (Olsen and Goodrich, 2003). On the other hand, robot swarms exhibit emergent behaviour, which usually requires some time before it can be observed. *Neglect benevolence* is the idea that some robot swarms may perform better when they are neglected (i.e. no interventions) by the operator for a certain period of time (Walker et al., 2012). Nagavalli et al. (2015) showed that upon observing a swarm's behaviour, human operators can learn to identify the best moment to send an input to the swarm. While this highlights the importance of operators to interact with robot swarms at the most effective moments, it also suggests that it is not necessary to be able to interact with every robot in order to effectively work with swarms.

2.2.2 Situational Awareness

Situational awareness refers to a human operator's comprehension of the current situation and projection of possible future outcomes (Drury et al., 2003; Endsley, 1995). It is influenced by how information is presented to a human operator and how the operator makes use of the obtained information to make assessments of the current situation. While maintaining a good level of situational awareness is important, unlike traditional robotic systems, it is often difficult to achieve this when interacting with robot swarms due to their highly distributed nature.

When the global state of the swarm is accessible, interfaces that can show the predicted behaviour of the swarm can improve the human operator's situational awareness (Walker et al., 2012). However, it may not always be possible to obtain

such global state information when deploying robot swarms in real-world environments (e.g. a global map showing the positions of the entire swarm). Kapellmann-Zafra et al. (2016) studied whether a human operator can aggregate robots when provided access to only a local camera view and proximity sensor readings from a single randomly selected robot. Under the effects of such limited situational awareness constraints, operators struggled to aggregate the robots effectively, even when they had previously observed their global dynamics. However, upon receiving sufficient training, operators became familiar with the dynamics of the robot swarm and increased their performance under the presence of the aforementioned situational awareness constraints.

Level of Autonomy (LoA) is one factor that affects a human operator's situational awareness. LoA is a 10-point scale that defines a range of autonomous behaviours that a system could exhibit during human interaction (Sheridan, 2002). In swarm robotics, a low LoA would mean the robots often wait for inputs from the operator, resulting in an increase in the workload experienced by the operator. A high LoA would mean the robots are fully autonomous and require minimal input from the operator. While a high LoA may reduce the operator's workload, the operator gets detached from the rest of the system, which can potentially lead to the degradation of the operator's knowledge and experience related to the task. Instead of locking the swarm to a fixed LoA, Hussein and Abbass (2018) proposed a mixed-initiative system, which allows the swarm to flexibly change its LoA depending on the current task and human preference.

It is important for swarm designers to consider how and what information should be presented to the human operator and what LoA the swarm should exhibit depending on the specific use case they are being used in. Interface designs that control the type of information displayed to an operator are discussed in Section 2.2.4.

2.2.3 Control Methods

In recent years, several methods have been proposed for dynamically controlling robot swarms. These methods aim to minimise the complexity required to control a large number of robots by departing from the traditional one-to-one teleoperation style of control (Lewis et al., 2006). Kolling et al. (2016) suggested four categories for classifying existing control methods, namely behaviour selection, leader influence, environment modification, and parameter setting.

- **Behaviour selection.** This method provides an operator with a library of swarm behaviours to choose from. The operator controls the swarm by switching

between the behaviours to achieve the desired outcome. Kolling et al. (2012, 2013) investigated an operator choosing from a set of discrete behaviours (e.g. Stop, Random, Rendezvous) to perform a foraging task. The behaviours had to be given to robots in the swarm at the right moment to maintain effective coverage over an area. Different sets of behaviours were examined by Bashyal and Venayagamoorthy (2008) and combining simple behaviours into *plays* by Miller et al. (2005). Behaviour selection has also been shown to control hub-based swarms (Crandall et al., 2017). Behaviour selection is one of the simplest methods to control a robot swarm. However, it requires the operator to have a thorough understanding of the set of available behaviours and their expected outcomes.

- **Leader influence.** The operator directly controls one or more leader robots, which affects the behaviour of other neighbouring robots. Walker et al. (2013) examined the leader-follower model based on Couzin's flocking model (Couzin et al., 2002), where the followers moved synchronously with the leader. The use of multiple dynamic leaders showed even greater influence over the robot swarm (Walker et al., 2014a,b). Instead of attracting follower robots, leaders can also act as repulsive forces, as seen in the predator approach investigated by Goodrich et al. (2013). Therein, the operator switched between attractive and repulsive forces to navigate the swarm. During proximal interaction, a human could directly be perceived as a leader by the robots (Nagi et al., 2014). Leader influence has also been considered a simple method to control swarms due to its similarities with traditional teleoperation-style control. An important point to note is that the controllability over the swarm is limited by the range of influence of the leader robot.

- **Environment modification.** The operator indirectly controls the behaviour of robots by modifying the environment. An example of this type of control can be seen using real and virtual objects as cues. Obstacles (Patel et al., 2019) and attractive/repulsive beacons (Kolling et al., 2013) were placed in the environment by a human to influence any robots that came close to them. If the robots are capable of sensing virtual cues in the environment, virtual walls (Jang et al., 2021) and artificial potential fields (Prabhakar et al., 2020) may also be modified to influence the robots as well. Environment modification is highly scalable since it does not depend on the size of the swarm, though the indirect influence on robots makes the interaction less intuitive for operators to control.

- **Parameter setting.** This method allows an operator to modify a set of parameters that govern the behaviour of a swarm, such as thresholds or probability values that trigger a change in the robot's behaviour. For example, to encourage more exploration during a foraging task, an operator may decide to decrease the parameter value that controls the likeliness of all robots to move in the same direction (Crandall et al., 2017). Instead of using raw numeric values, a set of parameters can also be used to exhibit different types of social-like behaviours indicating the robots' *Sociability*, *Conformity*, *Commitment*, and *Disposition* (Hexmoor et al., 2005). Among the four classes of control, parameter setting is the least explored control method due to its less intuitive nature of having to set parameters; the operator must understand not only the effects of modifying a single parameter but also the interplay between them.

Some human-swarm interaction systems may have two or more modes of control, which lets the human operator pick their preferred type of control (Patel et al., 2019). When multiple human operators interact with a robot swarm, interference with each other's control should be kept to a minimum. Leader influence and environment modification usually have a limited range of influence and thus, can be considered as appropriate control methods for multi-human-swarm interaction.

2.2.4 Interface

Due to the distributed nature of robot swarms, understanding the state of the swarm and the progress of the current task can be difficult for a human operator who is simply observing the motion of individual robots. An important factor when designing a complex system is to improve its *transparency*, which is to provide useful information in a way that is easily understood by the operator interacting with it (Roundtree et al., 2021). Interface designs that effectively display information about the swarm can help improve an operator's situational awareness, which is important for gaining human trust in the swarm system (Wilson et al., 2023).

Interfaces that provide a global view of the environment displaying the terrain, the position of all robots and points of interest have been the most commonly explored designs in HSI. This type of interface that provides a "bird's-eye view" was shown to be effective for interacting with robot swarms (Kolling et al., 2013). More recently, digital twin technologies have been used to interact with swarms. Jang et al. (2021) created a digital twin of the swarm and allowed an operator equipped with a virtual

reality headset to view and interact with the swarm from any preferred angle. The operator was able to pick and place a robot in the digital twin, which was translated into a command to move the robot to the specified position. While interfaces that provide a global view can give a complete picture of the current situation, many assume perfect communication between the interface and every robot, which may not always be possible to achieve in challenging environments.

Instead of simply displaying every robot in the swarm, various visualisation techniques have been explored to represent swarms differently. Previous interfaces have represented a swarm as an amorphous blob (Walker et al., 2016), an arrow that represents the direction and velocity of the swarm (Roundtree et al., 2018), and a heatmap that represents the swarm's area coverage (Divband Soorati et al., 2021). The progress of tasks can also be overlaid within the interface (Christensen et al., 2022; Roundtree et al., 2021). Many of these visualisations are intended to highlight certain properties of the swarm or task that are found to be useful for the operator. The choice of the type of visualisation depends on the type of information available and the context in which the swarm is being used.

Interfaces that assume a local perspective, for example, from the view of a single robot, are less commonly explored. Operators engaging in proximal interaction (Gromov et al., 2016; Nagi et al., 2014; Rockbach and Bennewitz, 2022) can also be considered to have a local perspective, as their view of the swarm is restricted to their physical position in the environment. As situational awareness was shown to be lower when operators were restricted access to a global view (Kapellmann-Zafra et al., 2016), there is a need to explore how operators with a local perspective of the environment can effectively utilise a swarm that may travel beyond their field of view.

2.2.5 Modalities

Since the early days of human-swarm interaction research, many works have explored the use of different modalities other than using a monitor, mouse and keyboard to control and receive feedback from robot swarms. In this section, we look at two of the most actively studied topics; gesture- and haptic-based interfaces.

Gesture is used commonly by humans to express or reinforce their intentions (Nehaniv et al., 2005). A range of hand signals have been used to signal the swarm to perform various behaviours, such as to split, merge, and steer the swarm (Alonso-Mora et al., 2015; Kim et al., 2020; Podevijn et al., 2014; Serpiva et al., 2021). Podevijn

et al. (2014) used a Microsoft Kinect to detect user gestures and mapped them to specific commands for the swarm. Experimental tests demonstrated that a user was able to split a large swarm into smaller groups and guide each subswarm to different goal locations. Pointing gestures have also been used to instruct one or more robots to move from one location to another (Gromov et al., 2016; Nagi et al., 2014). In the work by Gromov et al. (2016), a user was equipped with two armbands, each consisting of eight electromyography sensors, to extract the 3-D vector represented by the configuration of the arm and to recognise the current hand gesture. The overall pointing gesture was combined with voice commands, which confirmed which robots the user wanted to control. Gesture-based interfaces have shown to be very intuitive even for inexperienced users and are an effective mode of control, especially during proximal interaction.

Haptic feedback creates a sense of touch by applying force or vibration to the user. Nunnally et al. (2013) was one of the first to make use of haptics in HSI. A Phantom Omni device was used by the user to broadcast a force vector to influence the behaviour of the robots. The operator in turn experienced force via the Phantom device when there were obstacles near the robots. Tsykunov et al. (2018) presented SwarmTouch, which informed the change to the formation of several drones using a haptic glove that vibrated the user's fingertips. Haptic feedback can also be provided to other parts of the body, such as the work by Abdi and Paley (2023) where an operator felt the swarm's centre of mass through a haptic vest. Zooids (Le Goc et al., 2016) and Ubiswarm (Kim and Follmer, 2017) are table-top swarms that can be considered to use haptic feedback since a user directly picks up one or more robots to interact with them. In these systems, the user physically interacts with the robots rather than experiencing a device that emulates the sensation. These works show that haptic-based interfaces are well-suited for quickly providing feedback about the swarm as they can be "felt" by the user directly, though the user must be aware of the meanings of the different haptic signals.

The two modalities discussed in this section are both intuitive to humans as they are often based on or inspired by human-human interaction. Voice is another popular modality investigated in human-robot interaction, which is often included in multimodal interfaces that combines a number of different modalities (Chandarana et al., 2017; Pourmehr et al., 2013). More recent studies have proposed to use swarms as embodied extensions to enhance human abilities (Hasbach and Bennewitz, 2022; Rockbach and Bennewitz, 2022). Other modalities explored in HSI include face engagement (Pourmehr

et al., 2013; Zhang and Vaughan, 2016) and brain-machine interfaces (BMI) (Karavas et al., 2017).

2.3 Multi-Human-Swarm Interaction

This thesis considers how multiple human operators can share the control of a robot swarm. As discussed in Section 2.2.1, the cognitive complexity can be overwhelming for a single operator when interacting with an increasing number of robots or subswarms. Furthermore, an operator engaging in proximal interaction cannot supervise all subswarms as they may need to work at different locations simultaneously. These problems could be addressed by having multiple operators who each interact with a different subswarm to distribute the cognitive load, which could enable them to accomplish more complex tasks and improve productivity (Whetten et al., 2010). This section reviews existing research on multi-human-swarm interaction and some of the challenges that remain to be addressed.

2.3.1 Communication

Communication is often considered a prerequisite for a group of people to cooperate effectively. This also applies to a team of human operators that uses a robot swarm to complete a common goal. For example, operators may need to negotiate with each other to avoid or resolve potential conflicts in trying to control the same robot (Patel et al., 2022). Poor communication quality can hinder cooperation as it becomes difficult to maintain a shared mental model of the situation among the operators. In addition, excessive communication can become an overhead for the operators, resulting in increased cognitive workload (MacMillan et al., 2004). One way of avoiding communication overhead is to make use of implicit communication. Rico et al. (2008) reported that implicit or indirect communication, such as the use of short phrases agreed between the team members prior to the mission, can be effective when undertaking highly interdependent tasks, provided that the members have a sufficient understanding of the situation.

Patel et al. (2022) investigated the effects of different inter-human communication levels between two operators who were tasked to remotely supervise and control a group of simulated mobile robots. Communication levels considered were no communication, direct communication (i.e. via verbal communication), indirect communication (i.e.

via the interface), and a combination of direct and indirect communication. Their user study with participants showed that those involving verbal communication allowed operators to perform at their best. The authors also highlighted the usefulness of indirect communication, which was superior in conveying the state of robots, which was difficult to convey verbally.

Alhafnawi et al. (2022) took a novel perspective on inter-human communication, exploring multi-human-swarm interaction in a social setting. They developed MOSAIX, a platform consisting of up to 100 tile-shaped robots, each equipped with a display facing upwards. MOSAIX was tested in a public setting, where the member of the public would input their opinion on an issue to one of the robots. These robots then moved around displaying the chosen opinion, which allowed other people to see them and sparked conversions on the issue in a more appealing way compared to traditional methods using electronic devices.

2.3.2 Robot Sharing

When multiple operators interact with a robot swarm, one factor that needs to be considered is the relationship between humans and robots (Yanco and Drury, 2004), in other words, how should the robots be assigned to the humans? Prior research on multi-operator control of robot swarms can often be categorised into either (1) statically allocated robots (i.e. a subset of robots are assigned to each operator) or (2) shared robots (i.e. all operators can interact with all robots at any time) (Nagavalli et al., 2017). Statically allocated robots define a strict mapping for each robot to one of the operators and can only be controlled by the dedicated operator. On the other hand, shared robots treat the robots as a shared resource that can be accessed by any operator.

Lewis et al. (2010) compared whether controlling robots freely among two operators would improve the performance of 24 robots in a search and rescue task as opposed to assigning half of the robots to each operator. Against initial predictions, the shared control setup did not perform as well as the fixed assignment of robots to operators. The authors called this phenomenon *diffusion of responsibility*, as the operators often assumed a robot that was not being used to be the other operator's responsibility. This reveals the fact that some level of responsibility separation is necessary when sharing large groups of robots among multiple operators. Similar studies were later

conducted, which showed operators using shared control experiencing lower levels of workload (Gao et al., 2014).

Patel and Pinciroli (2020) argued that assigning fixed responsibilities to operators may not always be the best solution. Through a box-pushing experiment, they demonstrated that sharing robots among operators that have identical authority over the robots resulted in an increase in situational awareness for both operators. The operators were more engaged with the robots and with the mission when tasks were shared between the operators, whereas assigning specialised responsibilities resulted in operators getting distracted once their assigned tasks were fulfilled.

The previous works assumed the operators to have equal capabilities. Nagavalli et al. (2017) investigated role specialisation among operators. Gesture-based interfaces were developed to navigate five mobile robots to a goal location. One operator had the ability to control a leader robot, while the second operator controlled the behaviour of the follower robots. This separation by functionality was inspired by the shepherding of sheep, where a shepherd would determine the direction of travel and a sheepdog ensures the sheep remain clustered. Despite the expected overhead between operators, it was reported that robots arrived at the goal faster when two operators were each responsible for controlling different aspects of the swarm compared to a single operator. Other examples of differing human roles can be seen in hierarchical teams, where one of the operators acts as a commander to oversee and support other operators (Grosh and Goodrich, 2020).

These works have shown that both methods for sharing robots among operators have their strengths and weaknesses. Statically allocated robots are beneficial when the tasks can be completed by the operators independently but do not have the flexibility to swap robots between the initially allocated teams. Shared robots give individual operators some flexibility on how many robots to use on a given task, but operators may struggle from "diffusion of responsibility" due to all the robots being shared. A hybrid approach that exhibits both methods or a robot swarm with the ability to flexibly switch between the two would be beneficial.

2.4 Formal Design Methods

Designing the behaviour of a robot swarm is challenging because changes made in the robot controller not only affect the actions of the particular robot but also the emergent

behaviour of the entire swarm. The process of developing swarm behaviours usually requires trial and error, where the designer uses their intuition and experience to make modifications until the desired behaviours are achieved (Bozhinoski and Birattari, 2023; Hamann, 2018). However, as future swarms become more complex, they cannot be designed solely by such methods; there is a need for a systematic approach to support the swarm designer during the development of robot swarms (Dorigo et al., 2020).

Formal methods are system design techniques that can help reduce the amount of ad-hoc development. It involves creating a model of the system to be controlled and a set of specifications that the system must meet. The resulting model can be analysed by a variety of existing model checking and verification tools (Lopes et al., 2016). The explicit representation of capabilities increases the designer's confidence in the correctness of the system (Webster et al., 2020). In this section, we review previous work on formal methods applied to the development of robot swarms.

Model checking is a method that uses temporal logic to analyse whether a given model matches the specifications (Clarke, 1999). One of the first works that applied formal methods to swarms was by Winfield et al. (2005), which showed how swarm connectivity could be maintained using temporal logic by specifying the desired properties of the swarm. Chen et al. (2020) used integer programming alongside linear temporal logic to perform subswarm assignment on 20 ground robots, although this was based on a centralised approach.

Brambilla et al. (2014) proposed *property-driven design*, a top-down approach to swarm development based on describing swarm properties and model checking. It consists of four steps, (i) formally define the desired properties of the swarm, (ii) design and verify a prescriptive model using Markov chains, (iii) implement the prescriptive model and (iv) implement and test the swarm. A swarm designer following these steps would be able to concentrate on the design of the global swarm behaviour first, before focusing on the implementation details. While this approach offers methods to check whether certain properties can be met even before running tests using simulated or real robots, the automatic generation of source code has yet to be shown; implementing the model still requires creative insight from the designer.

Finite-state machines are commonly used to define robot controllers as they are generally easily readable to humans. Francesca et al. (2014) proposed *AutoMoDe-Vanilla*, an automatic design method for robot swarms. First, a designer defines a set of atomic behaviours called modules, which may include parameters. This is followed by an optimisation process that creates a controller represented as a probabilistic

finite state machine. Experiments showed that the automatically designed controller performed better in an aggregation task than the controllers manually designed by human experts. The AutoMoDe method has since been extended to a class of automatic modular design methods (Francesca et al., 2015; Garzón Ramos and Birattari, 2020; Hasselmann and Birattari, 2020; Kuckling et al., 2020; Ligot et al., 2020; Salman et al., 2019).

Behaviour trees have also been used to model the controllers for robot swarms. The tree defines a sequence of actions and conditions that a robot performs. While being easily readable to humans, behaviour trees are hierarchical so it is possible to detach a subtree and attach it to a different part of the behaviour tree, providing modularity and reusability of a particular behaviour (Jones et al., 2018). This can be useful when evolving a behaviour tree since the designer can read to understand the evolved behaviour and make modifications if necessary (Hogg et al., 2020). Recently, AutoMoDe has also been extended to use behaviour trees (Kuckling et al., 2021).

Supervisory control theory (SCT) is a formal method proposed by Ramadge and Wonham (1987). SCT is used to restrict the behaviour of a controller (also known as a supervisor) so that the given logical control specifications are satisfied. A unique aspect of SCT is its explicit representation of uncontrollable events; events that cannot be disabled by the controller. Lopes et al. (2016) applied SCT for the control of robot swarms. Through four case studies, they demonstrated how state-of-the-art swarm algorithms can be represented using SCT. The approach involved the automatic generation of control code from the developed supervisor. Further extensions have been proposed for modelling probabilistic decision making (Lopes et al., 2017) and communication between robots (Lopes et al., 2020).

Formal methods can help reduce the complexity involved in the swarm development cycle. This is especially needed for swarms that are intended to interact with multiple humans as they further complicate the interaction. However, the application of formal methods focusing on HSI is currently very limited. Providing certain formal guarantees on a swarm's capabilities during proximal interaction could ensure the safety of humans working near them.

2.5 Supervisory Control Theory Preliminaries

The SCT framework is used to control systems that can be represented as a discrete event system (DES) (Ramadge and Wonham, 1987). In other words, there is a discrete set of states that can express the state of the system at any time. In DES, transitioning between states is triggered by *events*. Events in SCT are classified as either an *uncontrollable* or *controllable*. Uncontrollable events can occur at any time (e.g. sensor inputs), while controllable events only occur when triggered by the controller (e.g. actions initiated by a robot). When designing a controller in SCT, the designer must define formal languages that express (i) what the robot can do in principle (called *free behaviour models*) and (ii) how it should behave (called *control specifications*). The corresponding languages are then automatically synthesised into a coherent language to produce the controller (called the *supervisor*) that is executed on each robot. The supervisor guarantees that, at any time, only valid sequence of events can occur. This is realised by restricting the set of controllable events that a robot can choose from at each state.

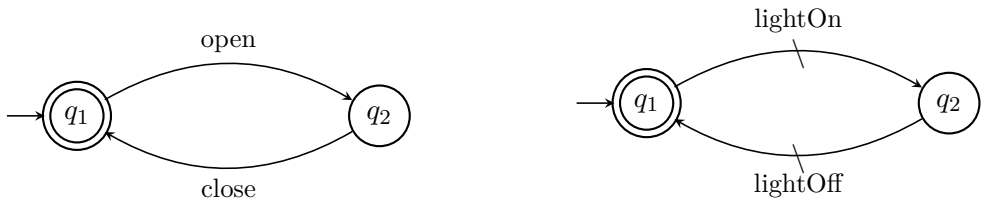
This thesis uses SCT to model the behaviour of robots based on a class of approaches proposed by Lopes (2016); Lopes et al. (2016). The remainder of this section provides an overview of how models can be designed in SCT using an example system and how a swarm designer would apply the approach in practice.

2.5.1 Generators

In SCT, regular languages are often used to represent the capabilities and control specifications of a system. Languages are expressed as generators, which are similar to automata. The difference is that automata checks whether a given word is part of a language, while generators produce words that are part of the language. Generator G is defined as the following 5-tuple:

$$G = (Q, \Sigma, \delta, q_0, Q_m) \tag{2.1}$$

where Q is the finite set of states, Σ is the finite set of controllable and uncontrollable events, $\delta : Q \times \Sigma \to Q$ is the partial transition function, $q_0 \in Q$ is the initial state, and $Q_m \subseteq Q$ is the set of marked states. Marked states are states that are considered to be safe and are usually used to represent a state after a process has finished or an idle



(a) G_1 **(b)** G_2

Figure 2.1. Example free behaviour models for a fridge representing **(a)** a sensor to detect the state of the fridge door; **(b)** the capability to control the light inside the fridge.

state of the system. Generators are used to represent both free behaviour models and control specifications, which are discussed in the following sections.

2.5.2 Free Behaviour Models

Free behaviour models define the capabilities of a system. In other words, it describes all the possible actions that a system *could* perform. A system can be composed of multiple free behaviour models. Each model represents a certain functionality of the system. The models essentially act as the building blocks of the system and are assumed to be independent of one another. This means new functionalities can be added without affecting the existing models.

Figure 2.1 shows two free behaviour models for controlling the light inside a fridge. G_1 represents the state of the fridge door. It indicates that events *open* and *close* occur alternatively. G_2 represents the capability to control the light inside the fridge. The initial state is represented by an unlabelled arrow. An arrow with and without a stroke each represents a controllable and uncontrollable event, respectively. A double-circled or single-circled state each represents marked and non-marked states, respectively. In the given example, the marked state is defined to be when the fridge door is closed in G_1 and when the light inside the fridge is turned off in G_2.

2.5.3 Control Specifications

Control specifications define the desired behaviour of a system. In other words, it describes the actions that the system *should* do. To do so, we disable some controllable events from occurring in certain states to achieve the desired behaviour of the system.

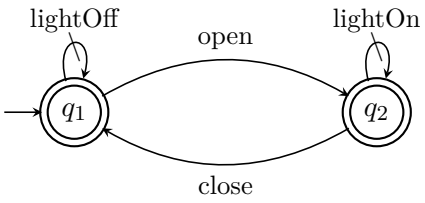


Figure 2.2. Example control specification to turn the light inside the fridge on when the door opens and off when it is closed.

By using the events defined in the free behaviour models, we can define control specifications by creating a relationship between the independent components of the system.

Using the free behaviour models G_1 and G_2 from Figure 2.1, we can define a simple behaviour that turns on the light inside the fridge when the door opens and turns off when it is closed. Figure 2.2 presents an example control specification. In the initial state q_1, the only action allowed is *lightOff*. If the door opens, the state transitions to q_2. The only action allowed in q_2 is *lightOn*. It returns to the initial state when the door closes, where *lightOff* is again the only allowed action that can be triggered by the system. All states in control specifications are generally labelled as marked states unless it is used to represent a system with a buffer that needs to reach a state where the buffer is empty.

2.5.4 Supervisor Synthesis

So far, we have seen that the free behaviour models define all possible events that can occur and the control specifications restrict controllable events from being triggered in certain states. The supervisor represents the control logic of the system, which can be obtained from the free behaviour and control specification models through an automatic process that synthesises them together (Lopes et al., 2016).

The synthesised supervisor guarantees the following properties.

- **Controllable.** There are no outgoing uncontrollable events in any state that are disabled by the control specification.
- **Accessible.** Every state can be reached from the initial state.
- **Coaccessible.** Every state can reach at least one marked state.

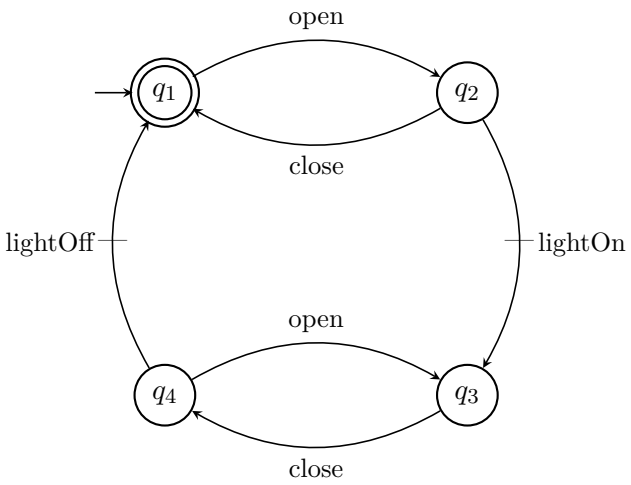


Figure 2.3. The synthesised supervisor for controlling the light inside a fridge.

These properties guarantee that the supervisor is non-blocking, in other words, there will be no deadlocks. During supervisor synthesis, any states that are non-accessible, non-coaccessible, or have an event that makes the supervisor uncontrollable are called *bad* states and are removed from the supervisor. The removal of these states is performed iteratively since removing one state could turn others into bad states, which are also removed. An advantage of the synthesis process is that when there is an error in the designed automata, it usually results in the final supervisor having a significantly small number of states or it may be missing certain events introduced in the free behaviour models. This is because models that conflict with each other result in the creation of more bad states, which are entirely removed during the synthesis. This is useful for the designer as any error in the logic can be detected at an early stage of the design process even before testing the system in simulation.

Figure 2.3 shows the supervisor that was synthesised using the models from Figure 2.1 and 2.2. The controllable events *lightOn* and *lightOff* are only triggered when uncontrollable events open and close occur, respectively. Note that the event *close* from state q_2 to q_1 is not disabled as it is an uncontrollable event. This covers the situation when the door opens but immediately closes before the system is able to trigger *lightOn*, in which case the light will not be turned on. An analogous situation is also covered by event *open* from state q_4 to q_3 when the door is opened immediately after it has been closed.

Designing free behaviour models and control specifications, and synthesising the models to produce the supervisor can be done using the openly available software tool Nadzoru[1] (Pinheiro et al., 2015).

When using the supervisors on a real system, it is necessary to link the events to the actual actions that are executed by the system. To do so, the SCT framework defines the *operational procedures* layer. Here, the designer programs a set of callback functions, which implements the behaviour of the robot when each event is triggered. While some programming is required, this is usually less than the amount of code that has to be written otherwise as the hardware-specific functionalities are the only parts that need to be coded; the control logic is already defined by the supervisors.

2.5.5 Public Events

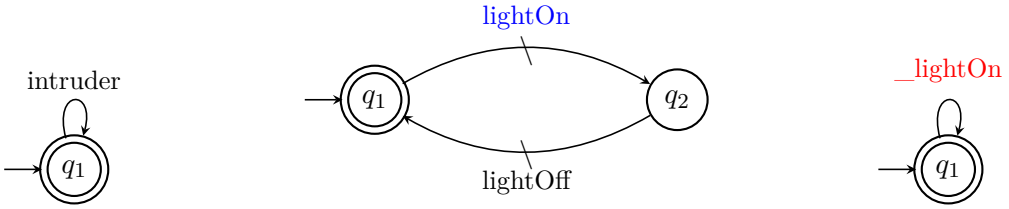
The original SCT framework assumes only the events that are observable by each robot. These events are referred to as *private* events. Lopes et al. (2020) proposed incorporating *public* events; an extension to SCT that allows multiple robots to share the occurrence of events that were previously private. In this extension, both uncontrollable and controllable events can be either private or public. This allows communication among robots to be modelled within SCT. Generator G with public events is defined as the following 7-tuple:

$$G = (Q, \Sigma, \delta, q_0, Q_m, \Sigma^{ext}_{u,pub}, M) \tag{2.2}$$

where Σ comprises private uncontrollable and controllable events $\Sigma_{u,priv}$, $\Sigma_{c,priv}$ as well as public uncontrollable and controllable events $\Sigma_{u,pub}$, $\Sigma_{c,pub}$. $\Sigma^{ext}_{u,pub}$ is the set of public uncontrollable events in G and other generators. Finally, $M : \Sigma_{c,pub} \rightarrow \Sigma^{ext}_{u,pub}$ is a mapping from public controllable events to the related public uncontrollable events. Q, δ, q_0 and Q_m are identical to Equation (2.1).

For example, consider a swarm of robots that are tasked to find an intruder. We would like only the last robot that has spotted the intruder to turn its LEDs on. Figure 2.4 shows the free behaviour models for these robots. G_1 represents a sensor capable of detecting an intruder. G_2 represents the robot's ability to turn its LEDs on and off. Event *lightOn* is specified as a public controllable event, so it will inform other robots that it has turned its lights on. G_3 represents the corresponding public

[1] `https://gitlab.com/genki_miyauchi/nadzoru`



(a) G_1 **(b)** G_2 **(c)** G_3

Figure 2.4. Free behaviour models for last spotted location. **(a)** A sensor to detect an intruder. **(b)** The capability to control the robot's own LEDs. **(c)** A receiver to detect the occurrence of another robot turning their LEDs on. Public uncontrollable and controllable events are shown in red and blue, respectively.

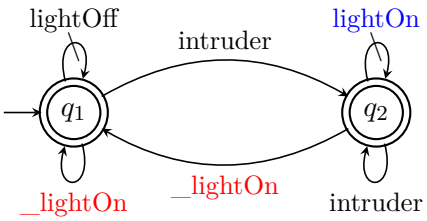


Figure 2.5. Control specifications for last spotted location to turn its own LEDs on when it detects an intruder and turn it off when another robot turns its LEDs on.

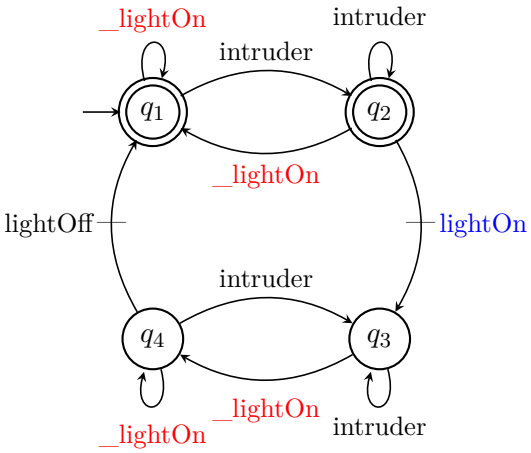


Figure 2.6. The synthesised supervisor for detecting an intruder using public events.

uncontrollable event *_lightOn* which is triggered when it detects that another robot has turned its lights on.

Figure 2.5 defines the control specification. It defines that when an intruder is detected (event *intruder*), the robot will turn its LEDs on (event *lightOn*) which is communicated to all other robots. If it detects that another robot has turned its lights on (event *_lightOn*), the robot will turn off its own lights (event *lightOff*). By using public events, it is possible to model the behaviour of robots when they receive signals from other robots. The synthesised supervisor is shown in Figure 2.6.

2.6 Summary

In this chapter, an overview of the field of swarm robotics and human-swarm interaction were provided. First, the properties that make swarms of robots useful for carrying out "division of labour", such as their capabilities to autonomously switch roles, form subgroups and maintain global connectivity, were shown. This was followed by exploring a wide range of factors that shape the interaction between a human and a swarm. The increasing interest in human-swarm interaction in the past decade has led to a large body of work in this field, which has helped us build our understanding of how a single human can control or influence the behaviour of a swarm. Yet, there has so far been little work that examined how the interaction can be scaled to a group of humans. This thesis is interested in such human-swarm teams. The majority of recent work in multi-human-swarm interaction has assumed that at any time, any human can interact with any individual robot in the swarm. Such ideal conditions may not be possible during remote interaction in environments with restricted global communication or during proximal interaction when the humans are physically present at different locations. Instead, humans may each be assigned a subset of the swarm to complete tasks. This motivates the need to consider a group of humans who are primarily focused on completing their individual tasks but are still capable of flexibly sharing their resources (i.e. robots) with each other depending on task requirements at the time.

For programming robot swarms, a wide range of formal approaches have been proposed to support the swarm designer in order to minimise the amount of ad-hoc development. A systematic approach could make the design of swarms verifiable and reproducible, which could help improve their transparency and trustworthiness, and bring them closer to being deployed in the real world. We presented an overview of

SCT for the design of robot swarms, which has the added benefit of automatically generating source code to ensure that the implementation matches the model and potentially reducing the overall development time.

Chapter 3

Connectivity-Preserving Robot Swarms Using Supervisory Control Theory

Involving multiple human operators to support a robot swarm can be beneficial in complex scenarios as presented in the previous chapter. While the ability to share a swarm has been shown to contribute positively towards task performance (Patel and Pinciroli, 2020), the ability to exchange information between the operators is often considered a prerequisite for them to cooperate effectively (MacMillan et al., 2004). The shared control of robot swarms between multiple operators working at separate locations remains a challenge, especially when global communication is not available.

A promising approach for inter-connecting separate locations are robot chains, that is, sequences of robots positioned in such a way that they form a linear communication network between two points of interest. Robot chains can be established dynamically by a swarm of robots (Nouyan et al., 2009). They can be used to guide robots to points of interest (Dorigo et al., 2013) or preserve connectivity among two or more points of interest to relay information (Majcherczyk et al., 2018). Garattoni and Birattari (2018) represented a sequence of tasks as the points of interest and used robot chains to guide other robots between them. However, most works do not consider the involvement of humans using these chains. To our knowledge, the problem of forming robot chains between two (mobile) operators has not yet been explored in the literature.

This chapter considers the problem of forming a dynamic chain of robots connecting two operators that freely move within an environment. The swarm assists a pair of

operators in performing spatially distributed tasks, while also maintaining a communication network between them. The operators are either physically present in the environment, in which case they each carry a portable device enabling them to interact with nearby robots, or otherwise, they each remotely operate a lead robot instead. The robot chain enables the operators to share information and robots among themselves.

Designing a robot swarm is challenging due to the difficulty in defining the collective behaviour that emerges from the local interactions between individuals (Birattari et al., 2019). A formal framework based on *supervisory control theory* (SCT) (Ramadge and Wonham, 1987) is used to model the behaviour of the robots, which has been applied for controlling groups of autonomous robots though without the involvement of humans (Dulce-Galindo et al., 2019; Lopes, 2016). In SCT, the capabilities of the system (here, the robots) as well as their specifications are expressed using formal languages. The control logic is then obtained from these languages in an automatic fashion. Based on SCT, this work proposes a distributed solution which formally guarantees that the deployed robot controllers match the modelled specifications. This is in contrast to existing works on robot chain formation, which, although building on formal representations, implemented the control logic in an ad-hoc manner.

This chapter is organised as follows. Section 3.1 formally defines the problem considered in this chapter. Section 3.2 describes the proposed methodology to maintain connectivity between the operators, how the behaviour of the robots is modelled using SCT, and the flocking behaviour of robots to follow a leader. Section 3.3 describes the simulation performed with groups of up to 40 mobile robots in an environment with obstacles and presents the results obtained. Finally, Section 3.4 summarises the chapter.

The work in this chapter is based on the author's publication (Miyauchi et al., 2022) and has been adapted for this thesis.

3.1 Problem Formulation

The problem scenario involves two human operators with shared control over a group of robots. Their objective is to complete a set of tasks that are scattered in the environment. The robots, also called *workers*, can perform all of these tasks.

The environment is represented as $\mathcal{W} = \mathbb{R}^2$. It may contain simple obstacles, $\mathcal{O} \subset \mathcal{W}$, such as walls. Let $\mathcal{W}_{\text{free}} = \mathcal{W} \setminus \mathcal{O}$. Each task is defined by the tuple

$(A, w, n^{\min})$. $A \subset \mathcal{W}_{\text{free}}$ represents the region from which the workers can perform the task. Each region is assumed to be connected, and their sets are disjoint. The quantity of work to be performed is discrete, with $w \in \mathbb{N}_0$ denoting the initial task *demand*. The quantity of work monotonically decreases provided that $n^{\min}$ or more workers perform that task.

Formally,

$$w[k+1] = \begin{cases} \max(0, w[k] - n[k]), & \text{if } n[k] \geq n^{\min} \\ w[k], & \text{otherwise} \end{cases} \tag{3.1}$$

where $n[k]$ is the number of workers performing the task at time k.

Each operator is assisted by a dedicated *lead agent*. The lead agent is unable to perform the tasks. The operator can provide instructions to and obtain information from their agent at all times. The lead agent in turn can provide instructions to and obtain information from any worker or the other lead agent, provided they reside within its local communication range. By doing so, the operators can indirectly exert control over the group of robots. For example, they can request the workers to start or stop performing tasks. For a task to be performed (by any workers), a lead agent has to be present within the same task region.

Two scenarios are considered. In the first one, the operators are physically present in the environment, and the lead agents are portable devices that the operators have with them, as they move around. In the second scenario, the operators are not physically present in the environment and the lead agents are mobile robots. The operators remotely control the movements of the lead agents. In principle, a combination of these scenarios where one operator is physically present, while the other is not, could be considered. This chapter specifically focuses on the scenario where each operator remotely controls a lead agent.

We assume that all robots (i.e. workers and lead agents) are represented by disks. The robots' communication network is an undirected graph. The nodes of this graph are the robots. An edge of this graph represents a pair of robots that can communicate. To do so, the distance between the two (measured from the centres of the corresponding disks) must not exceed the communication range, r_{com}. Moreover, there must be a direct line of sight between the two (i.e. no obstacles or other robots are present in between). All robots that a given robot is able to communicate with are referred to as that robot's *neighbours*.

We assume that all task specifications $(A, w, n^{\min})$ are known only to the operators. We assume the communication network to be connected at the start of the mission. The objective of the operators is to minimise the time to complete all tasks. The objective of the robots (i.e., workers and lead agents) is to maximise the number of workers that are performing tasks, or are available to do so while maintaining global connectivity in the communication network at all times.

3.2 Methodology

We use a team-centred approach to address the overall problem. At any moment in time, each worker is either (i) in team 1, which is led by the first lead agent, (ii) in team 2, which is led by the second lead agent, or (iii) part of a chain that shall connect the two teams. Where a worker is part of a team, we refer to it as a *follower*. Followers accompany their lead agent and execute task-performing behaviours when signalled to do so by the operator. Where a worker is part of a chain, we refer to it as a *connector*.

The connectors do not move. This facilitates connections to neighbouring connectors to be preserved. However, as the lead agents move, it will be necessary to make modifications to the chain. Two types of modifications are considered: adding or removing a chain member.

Figure 3.1 illustrates the situation when a new chain member is added. In other words, a follower of a given team must switch to the connector role, thereby extending the chain. The new connector allows the team at the *tail* of its chain to explore the environment further without becoming disconnected.

For follower i to become a connector, the following conditions with a non-team neighbour j have to be met:

- $C1$: The distance between follower i and a non-team neighbour j is greater than the safety limit $r_{\text{safe}} + \varepsilon$.
- $C2$: The distance between follower i and a non-team neighbour j is the shortest among all distances between other followers in i's team and the non-team neighbour j[1].

[1]Note that although follower i can establish the relative distances between its neighbours, it cannot detect whether there are any obstacles in between them. To establish whether two neighbours are directly connected, it needs to communicate with one of them to confirm they can detect each other.

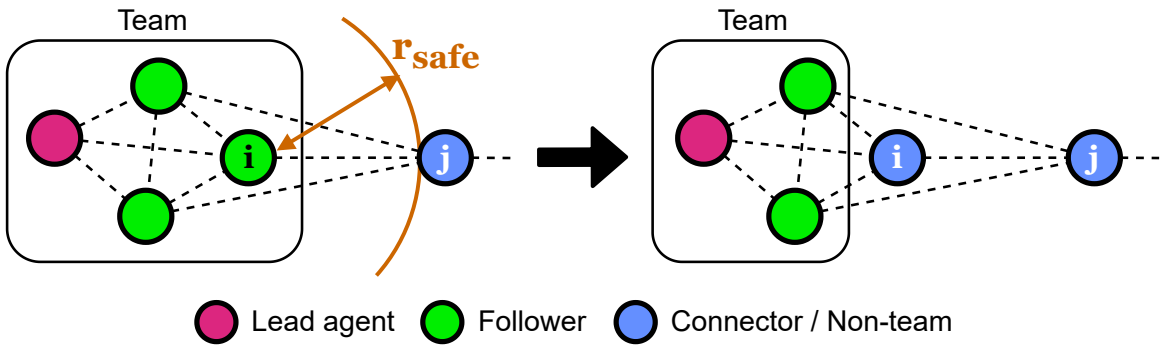


Figure 3.1. A situation prior (left) and after (right) a robot joins a chain at its tail, thereby extending it. The box represents a team. Dashed lines represent connections between robots. Follower i is sufficiently far away from the connector j (condition $C1$) and observes that it is closer to connector j than its peers (condition $C2$). It requests leaving its team to become a connector, which is approved by connector j.

It is possible for more than one follower to satisfy conditions $C1$ and $C2$. To ensure that only one follower becomes a connector, each follower satisfying both conditions sends a *request* message to the lead agent (or equivalent connector at the tail of the chain, if one has already been established). The agent (or connector) picks the first request received from a follower and sends a message allowing this follower to leave the team. Flooding of messages is used to share request and accept messages throughout the team.

When two connectors have a direct connection between each other, they are said to be *adjacent*. Figure 3.2 illustrates the situation where a member of the chain is removed. In other words, a connector switches to the follower role, thereby shrinking an unnecessarily long chain. The robot joins the closest team when it becomes a follower.

Given connector i has an adjacent connector j, the conditions for connector i to become a follower of a nearby team u are as follows:

- $F1$: Connector i is a tail connector of team u.
- $F2$: The distance between a follower of team u and an adjacent connector j is less than the safety limit $r_{\text{safe}} - \varepsilon$.

Table 3.1 summarises the actors involved and the roles that they can assume.

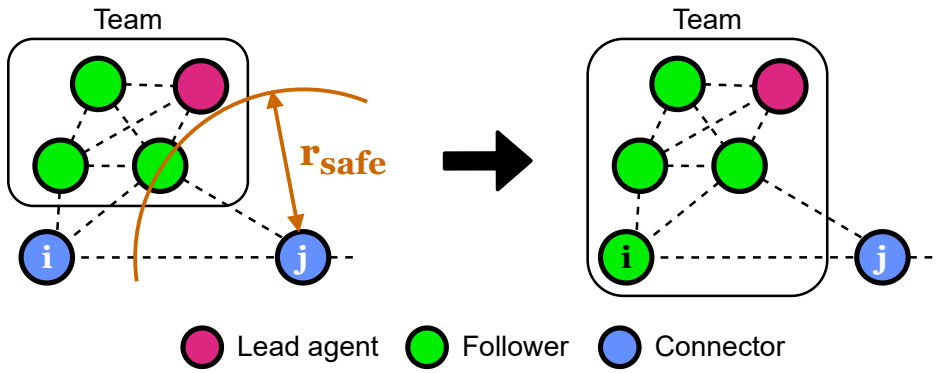


Figure 3.2. A situation prior (left) and after (right) a robot leaves a chain, thereby shrinking it. Connector i is at the tail of the chain (condition $F1$) and observes that a team is sufficiently close to an adjacent connector j (condition $F2$), rendering itself redundant. It leaves the chain to join the nearby team.

3.2.1 SCT Formulation

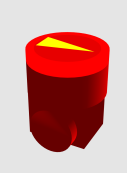
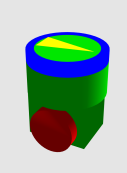
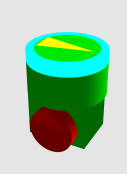
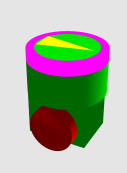
This section describes how we model the aforementioned chain management behaviour using SCT. The robot's state evolves via uncontrollable and controllable events. Uncontrollable events represent control inputs, for example, the instructions that operators may issue at any time. Controllable events represent actions that can be triggered by the robot's controller itself, such as performing the locally available task. Each event is either private or public (Lopes et al., 2020). In this work, we use public events to formally model how messages impact the robots as they propagate through the network. If a robot's controllable event is public, it triggers a corresponding uncontrollable event in neighbouring robots. Further details of the SCT framework can be found in Section 2.5.

The following section describes the SCT models used by the lead agents and workers, which are labelled $\theta \in \{l, w\}$. l refers to the models designed for the lead agents, while w refers to the models for the workers. G_i^θ denotes the ith free behaviour model and E_j^θ denotes the jth control specification.

Lead agent models

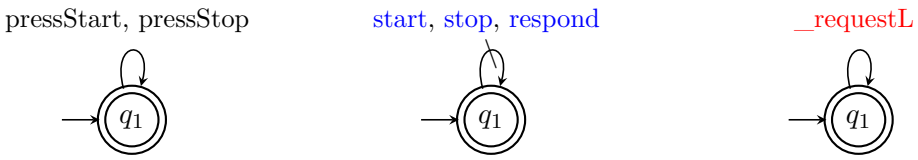
We assume that the lead agent is a robot that is remotely controlled by a human operator. As the role of the lead agent is to instruct the followers' supervisors through public events, only communications (but not its movement) are modelled. Figure

Table 3.1. Summary of actors and roles. The *traveller* role is discussed in Section 3.2.3.

Actors	Role	Appearance	Description
Operator	—	—	A human operator whose objective is to complete all tasks by guiding the workers to the task locations via the lead agent.
Lead agent	—		A robot controlled by an operator. The operator can provide instructions to and obtain information from workers via the lead agent.
Worker	Follower		A robot that is following a lead agent. It has the capability to execute task-performing behaviours.
Worker	Connector		A robot that is part of the network maintaining the connectivity between multiple teams.
Worker	Traveller		A robot that is moving along the network to switch from one team to another.

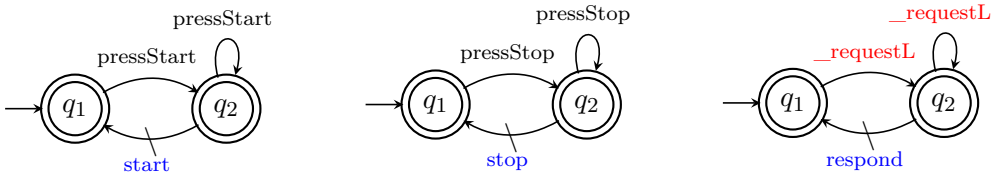
3.3 shows the free behaviour models of the lead agent. G_1^l represents the operator input device used to control the lead agent. The events correspond to the operator pressing a start (event *pressStart*) or stop (event *pressStop*) button to indicate whether followers should execute a task. G_2^l represents the lead agent's ability to send messages to neighbouring robots. The lead agent can send task-related messages (events *start* and *stop*) and respond to requests received from followers to become a connector (event *respond*). G_3^l represents the lead agent's ability to receive such requests (event *_requestL*) from workers to become a connector.

Figure 3.4 shows the control specifications of the lead agents. E_1^l and E_2^l define the public controllable events that are enabled once the corresponding operator input is



(a) G_1^l **(b)** G_2^l **(c)** G_3^l

Figure 3.3. Free behaviour models for the lead agents representing their ability **(a)** to receive control inputs from the operator, **(b)** send messages to the workers, **(c)** and receive messages from them. States are represented by circles. The initial state is indicated by an unlabelled arrow. Marked states are represented by double-line circles. Transitions and associated events are shown as labelled arrows. Arrows with a stroke relate to controllable events, and arrows without a stroke relate to uncontrollable events. Public controllable events and public uncontrollable events are shown in blue and red respectively.



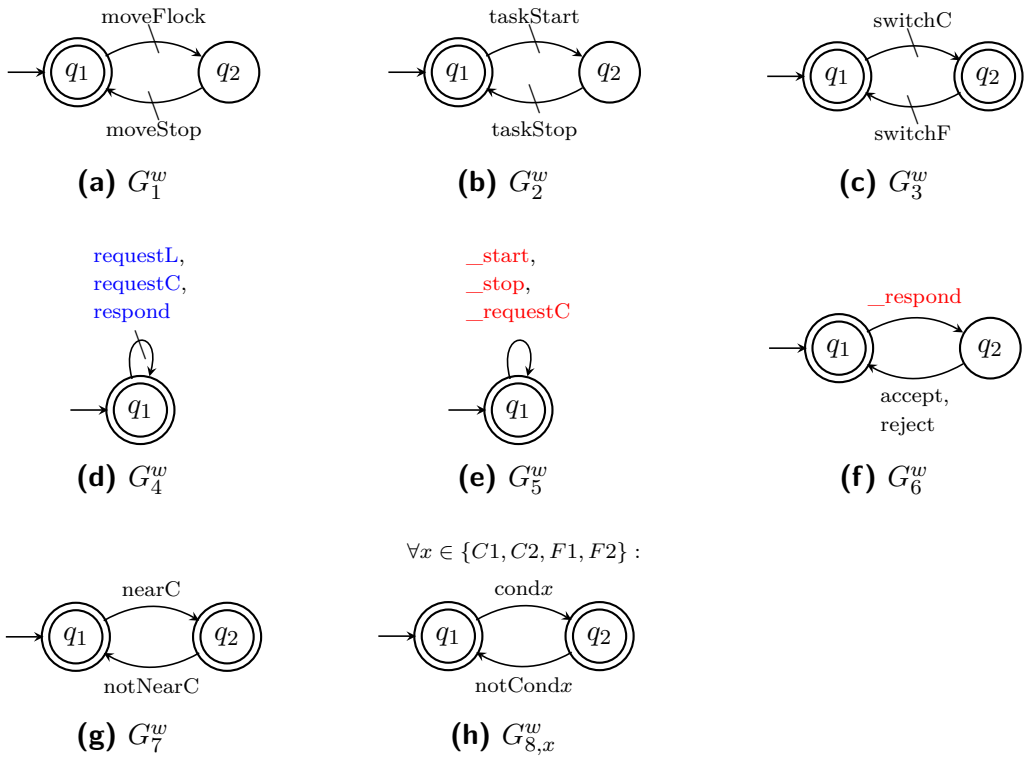
(a) E_1^l **(b)** E_2^l **(c)** E_3^l

Figure 3.4. Control specification for the lead agents allowing them **(a–b)** to transmit a signal upon receiving the corresponding operator input, and **(c)** to send a response when a request from a follower to become a connector is received.

received. E_3^l defines the message a lead agent sends upon receiving a request from a follower that wishes to become a connector. Multiple such requests can be received, and the decision to accept or reject these are handled by the operational procedures of the lead agent (i.e. within the callback function of the corresponding event). Only the first request is accepted (for any period without connectors). The operational procedures also ensure that request and respond messages include identifying information, helping to respond positively to specific robots.

Worker models

Figure 3.5 shows the free behaviour models of the workers. G_1^w represents robot motion. It allows the robot to either remain stationary (state q_1) or to flock with its team (state q_2). Event *moveFlock* enables the flocking behaviour, whereas event *moveStop* enables the robot to stop. The actual flocking behaviour is realised in the



(a) G_1^w **(b)** G_2^w **(c)** G_3^w

(d) G_4^w **(e)** G_5^w **(f)** G_6^w

(g) G_7^w **(h)** $G_{8,x}^w$

Figure 3.5. Free behaviour models for the worker robots representing their ability **(a)** to either flock or remain stationary, **(b)** to work on a task at their current location, **(c)** to switch between the follower and connector roles, **(d)** to transmit and **(e)** receive messages, **(f)** to process the responses received, **(g)** to detect nearby connectors, and **(h)** to determine whether the conditions $C1$, $C2$, $F1$, $F2$ are satisfied.

operational procedures (Section 3.2.4). This abstraction allows the reuse of the same SCT models on multiple platforms (e.g. legged and wheeled robots), provided that a connectivity-preserving flocking behaviour can be implemented on such platform. G_2^w represents the worker's ability to execute a task-performing behaviour. Event *taskStart* and *taskStop* start and suspend task execution, respectively. G_3^w represents the ability to change the role to follower (event *switchF*) and connector (event *switchC*). A worker is initially a follower (state q_1). G_4^w and G_5^w represent the worker's ability to send and receive messages from other neighbouring robots. The worker's request to become a connector is represented as *requestL* and *requestC*, which are sent to a lead agent or a connector, respectively. G_6^w represents the worker's ability to process the response to a request it has previously sent. When the event *_respond* is received, the operational procedures internally triggers either event *accept* or *reject*, depending on whether the robot matches the identifying information that was appended with event

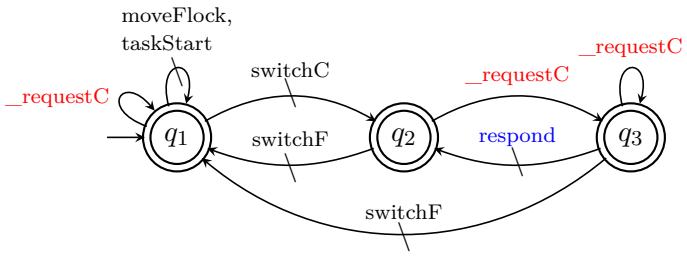
_respond. G_7^w represents whether a worker detects a connector (event *nearC*) or not (event *notNearC*). Finally, G_8^w represents whether the conditions we have defined for switching between roles are satisfied. $G_{8,C1}^w$ and $G_{8,C2}^w$ monitor the conditions to switch to a connector, whereas $G_{8,F1}^w$ and $G_{8,F2}^w$ monitor the conditions to switch to a follower.

Figure 3.6 shows the control specifications of the worker robots. E_1^w specifies the actions that are enabled in the follower and connector state, respectively. As a follower (state q_1), the robot is allowed to start tasks and flock. As a connector (state q_2 and q_3), it is allowed to respond to requests from followers. While being at the tail of the chain, it accepts the first such request. E_2^w ensures a worker starts or stops working on tasks when a corresponding signal was received from the lead agent. E_3^w specifies that if a worker detects a connector (event *nearC*), the request must be sent to the connector. Otherwise, the request is sent to the lead agent. $E_{4,C1}^w$ and $E_{4,C2}^w$ ensure together that a request to become a connector can be made only when conditions $C1$ and $C2$ are both satisfied. Similarly, $E_{5,F1}^w$ and $E_{5,F2}^w$ together allow a robot to switch to a follower only when conditions $F1$ and $F2$ are both satisfied. E_6^w describes the process for a follower to become a connector. The first action a follower must perform is to flock with the team (state q_2). If the follower decides to send a request to become a connector (state q_3), it must stop moving and wait for a reply (state q_4). The follower will only switch to a connector if the request was accepted.

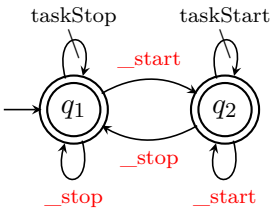
We synchronise the free behaviour models and specifications using local modular synthesis (de Queiroz and Cury, 2002; Lopes et al., 2020). After synchronisation, the local modular supervisors of lead agents have a total of 6 states and 25 transitions (sum of 3 supervisors), whereas the worker robots have 46 states and 196 transitions (sum of 8 supervisors). The next two sections present the extensions that enable lead agents to relay messages along the network and exchange robots with each other.

3.2.2 Message Relaying

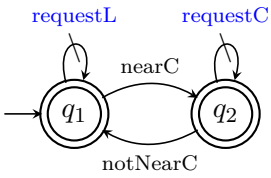
The first extension is to enable messages to be relayed between the operators. This can be realised using public events. When an operator chooses to send a message to the other operator, its lead agent broadcasts the message to its neighbouring robot. By comparing their hop counts (which is realised via the operational procedures), the workers establish whether they are closer or farther away from the particular team the message is being sent. This allows messages to be relayed until they reach the other lead agent.



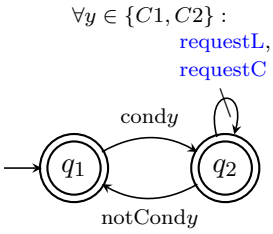
(a) E_1^w



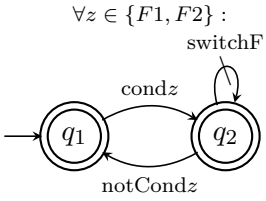
(b) E_2^w



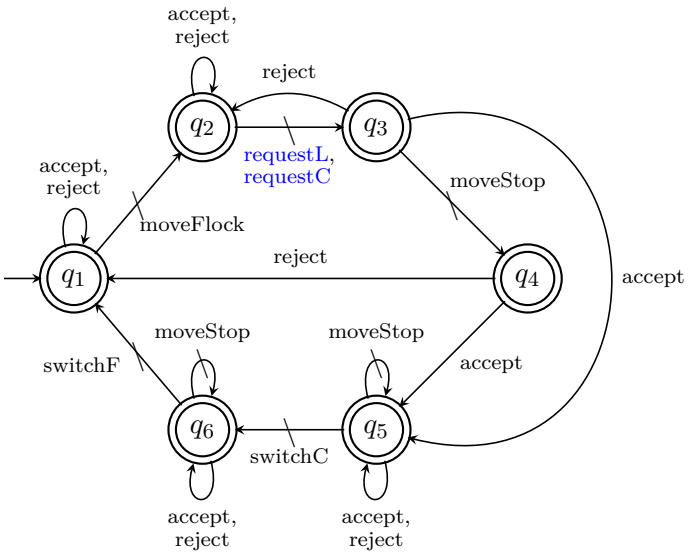
(c) E_3^w



(d) $E_{4,y}^w$



(e) $E_{5,z}^w$



(f) E_6^w

Figure 3.6. Control specifications for worker robots allowing them **(a)** to perform certain actions depending on their role, as a follower to start tasks and flock, as a connector, to send a response when a request from a follower to become a connector is received, **(b)** to start or stop performing tasks when the corresponding signals are received by the lead agent, **(c)** to determine whom request messages should be sent to, **(d)** to send request messages for becoming connectors when the conditions are satisfied, **(e)** to become a follower when the conditions are satisfied, and **(f)** to become a connector when the received response was accepting them.

(a) G_4^l **(b)** G_5^l

Figure 3.7. Free behaviour models for the lead agents representing their ability **(a)** to receive inputs from the operator in the form of messages intended for the other operator, or a signal to make a worker switch its team, and **(b)** to send a message that needs to be relayed to the other lead agent or inform a worker to join the other team.

(a) E_4^l **(b)** E_5^l

Figure 3.8. Control specifications for the lead agent allowing them to transmit a signal upon receiving the corresponding operator input.

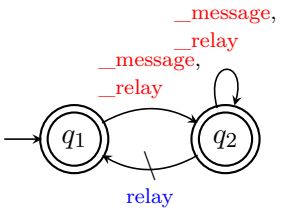
Figure 3.7 and Figure 3.8a show the additional free behaviour models and control specification of the lead agent to relay a message to the other operator. Event *inputMessage* in G_4^l and event *message* in G_5^l each represents the lead agent's ability to receive a message from its operator that needs to be relayed to the other operator and to broadcast the message to its followers, respectively. E_4^l defines that when the lead agent receives a message from the operator, it must broadcast the message to the workers in its team.

Figure 3.9 shows the additional free behaviour models of the workers for relaying a message. G_9^w represents the worker's ability to transmit a message to other neighbouring robots. G_{10}^w represents the worker's ability to receive a message from a lead agent (event *_message*) or another worker (event *_relay*).

Figure 3.10 shows the additional control specification for the workers. E_8^w defines that when a worker receives a message from other robots, it will broadcast the same message. To avoid messages from being relayed between the workers forever, the

(a) G_9^w **(b)** G_{10}^w

Figure 3.9. Free behaviour models for the worker robots representing their ability **(a)** to transmit to and **(b)** receive messages from neighbouring lead agents and workers.



(a) E_8^w

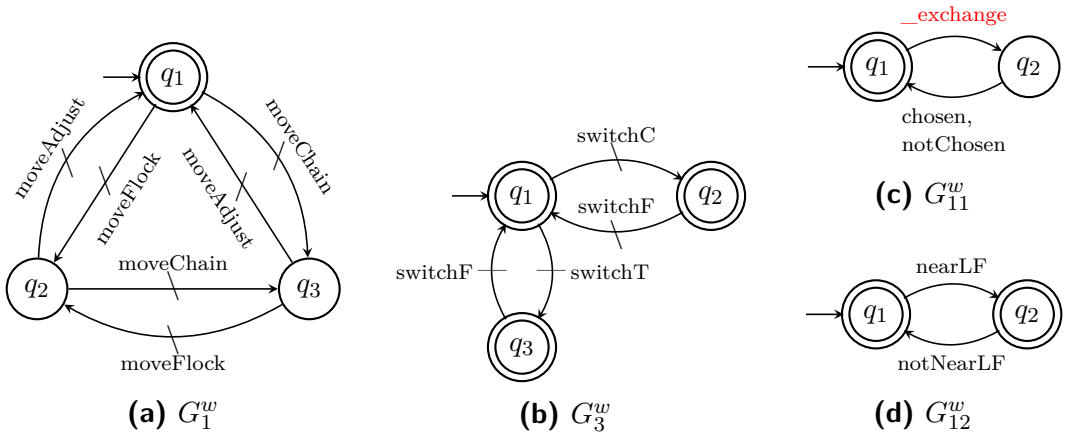
Figure 3.10. Control specification for the worker robots allowing them to relay operator messages it has received.

workers compare their hop count towards each lead agent to ensure the message is relayed towards the correct lead agent, which essentially acts as the sink node.

3.2.3 Robot Exchange

The second extension is to enable operators to dynamically exchange workers via the chain. The request for additional workers is sent to the other lead agent using the message relaying extension described in the previous section. To define the behaviour of a worker to travel along the chain, we introduce a new role called *traveller*. Upon receiving a signal from the lead agent, a follower switches to a traveller and starts moving along the chain to join the other team. A traveller moves towards a connector within its communication range that has the smallest hop count to the other team; as new connectors in the chain become visible, the traveller updates the target connector it move towards. Once the traveller detects a robot from the team to join, it becomes a follower of that team.

Figure 3.7 and Figure 3.8b show the additional free behaviour models and control specification of the lead agent for exchanging robots. Event *inputExchange* in G_4^l and



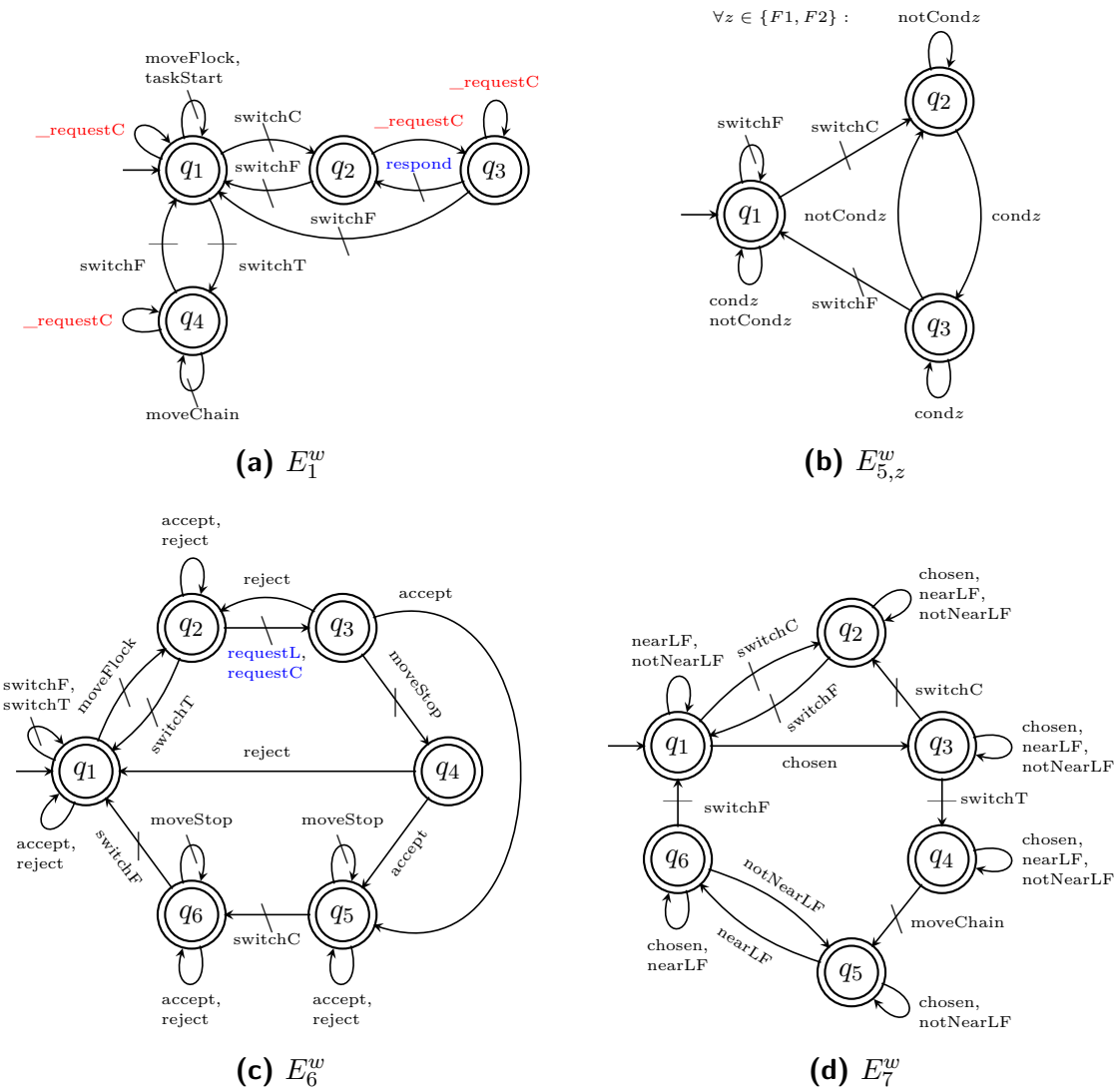
(a) G_1^w **(b)** G_3^w **(c)** G_{11}^w **(d)** G_{12}^w

Figure 3.11. Free behaviour models for the worker robots representing their ability **(a)** to either flock, move along the chain, or remain stationary, **(b)** to switch between the follower, connector, and traveller roles, **(c)** to process whether it has been chosen to move to the other team, and **(d)** to detect nearby team members.

event *exchange* in G_5^l each represents the lead agent's ability to detect the remaining number of robots that need to be sent to the other operator and to signal a follower in its team to join the other operator, respectively. E_5^l defines that when there are robots that still need to be sent, it must broadcast a message to a worker in its team to join the other team. By triggering event *exchange*, the lead agent informs one of its followers to join the other team and decrements the remaining number of robots to send. The lead agent only sends one robot at a time, so if the operator had specified to send multiple robots, the lead agent will eventually trigger the event the same number of times as the number of robots that must be sent to the other team.

Figure 3.11 shows the new free behaviour models of the workers for exchanging robots. G_1^w and G_3^w are modified versions of the free behaviour models presented earlier. These extensions allow a worker to become a traveller (event *switchT* in G_3^w) and move between teams by travelling alongside the chain of robots (event *moveChain* in G_1^w). G_{11}^w defines the worker's ability to determine whether it was chosen by the lead agent to join the other team or not. G_{12}^w allows a traveller to detect whether it has arrived near the team it was instructed to join.

Figure 3.12 defines the new control specifications for the workers. E_1^w, $E_{5,F1}^w$, $E_{5,F2}^w$ and E_6^w are modified versions of the specifications presented earlier, which adds the new events related to the traveller. E_7^w describes the process for a worker to switch its role between a follower and a traveller. Upon receiving a signal from the lead agent



(a) E_1^w **(b)** $E_{5,z}^w$

(c) E_6^w **(d)** E_7^w

Figure 3.12. Control specifications for the worker robots allowing them **(a)** to perform certain actions depending on their role, as a follower to start tasks and flock, as a connector, to send a response when a request from a follower to become a connector is received, as a traveller, to move along the chain to the other team, **(b)** to switch from a connector to a follower when the conditions are satisfied, **(c)** to become a connector when the received response was accepting it to switch roles, and **(d)** to become a traveller when it was chosen by the lead agent.

to join the other team (state q_3), the worker becomes a traveller (state q_4) and starts moving along the chain (state q_5) until it detects a member from the other team (state q_6), at which point it switches to a follower of that team (state q_1).

Table 3.2 summarises the definition of events used by the lead agents and workers. After synchronisation using local modular synthesis, the local modular supervisors of lead agents have a total of 10 states and 39 transitions (sum of 5 supervisors), whereas the worker robots have 170 states and 814 transitions (sum of 10 supervisors).

3.2.4 Flocking Behaviour

Flocking is a research topic of its own (Turgut et al., 2008; Vásárhelyi et al., 2018). In this work, it enables robots in the follower state to accompany the lead agent, as the latter moves within the environment. It is not used by robots in any other state.

The robot platform used in this study is the e-puck (Mondada et al., 2009), which is a mobile differential-wheeled robot. The e-puck has a circular body of diameter 7 cm. Its body is equipped with eight proximity sensors, which are distributed along its perimeter. We assume a range-and-bearing system to provide the relative positions of nearby agents. We use $\mathbf{p}_{ij}$ to denote robot j's position in the local coordinate system of robot i.

The robot uses virtual forces to determine its direction of movement similar to the works by Majcherczyk et al. (2018); Turgut et al. (2008). The virtual force of robot i is given by

$$\mathbf{u}_i = \alpha \mathbf{u}_i^{\mathrm{a}} + \beta \mathbf{u}_i^{\mathrm{rn}} + \gamma \mathbf{u}_i^{\mathrm{ro}} \tag{3.2}$$

where α, β and γ are positive scalars to weigh the influence of the force components.

Force component $\mathbf{u}_i^{\mathrm{a}}$ causes robot i to follow the lead agent. It is defined as

$$\mathbf{u}_i^{\mathrm{a}} = \frac{1}{|\mathcal{N}_i^{\mathrm{a}}|} \sum_{j \in \mathcal{N}_i^{\mathrm{a}}} \mathbf{p}_{ij} \tag{3.3}$$

where $\mathcal{N}_i^{\mathrm{a}}$ denotes the set of neighbours of robot i that are either the lead agent or members of the same team that are closer to the lead agent than robot i (based on hop count).

Table 3.2. Summary of events' description used in this study. Private controllable, private uncontrollable, public controllable and public uncontrollable events are labelled $\mathcal{C}$, $\mathcal{U}$, $Pub\mathcal{C}$ and $Pub\mathcal{U}$, respectively.

Event	Type	Used In	Description
moveFlock	$\mathcal{C}$	w	Robot flocks with the lead agent.
moveChain	$\mathcal{C}$	w	Robot moves along the chain to the other team.
moveStop	$\mathcal{C}$	w	Robot stops moving.
taskStart, taskStop	$\mathcal{C}$	w	Robot starts or stops working on a task.
switchC, switchF, switchT	$\mathcal{C}$	w	Robot switches to the connector, follower, or traveller role.
pressStart, pressStop, inputMessage, inputExchange	$\mathcal{U}$	l	Robot detects an operator input.
start, stop	$Pub\mathcal{C}$	l	Robot sends a signal to start or stop working on a task.
_start, _stop	$Pub\mathcal{U}$	w	Robot received a signal from the lead agent to start or stop working on a task.
requestL, requestC	$Pub\mathcal{C}$	w	Robot sends a request message to a lead agent or a connector.
_requestL, _requestC	$Pub\mathcal{U}$	l, w	Robot received a request from a worker.
respond	$Pub\mathcal{C}$	l, w	Robot sends a reply to the request received.
_respond	$Pub\mathcal{U}$	w	Robot received a response to the request it made to switch to a connector.
accept, reject	$\mathcal{U}$	w	Process the response to determine whether its request to switch to a connector was accepted or rejected.
nearC, notNearC	$\mathcal{U}$	w	Robot determines whether a connector was detected or not.
cond$x$, notCond$x$	$\mathcal{U}$	w	Robot determines whether condition $x \in \{C1, C2, F1, F2\}$ was satisfied or not.
message	$Pub\mathcal{C}$	l	Robot sends a message to the other lead agent.
relay	$Pub\mathcal{C}$	w	Robot relays a message to the target lead agent.
_message, _relay	$Pub\mathcal{U}$	l, w	Robot received a message that needs to be relayed.
exchange	$Pub\mathcal{C}$	l	Robot sends a message that specifies a worker to join the other team.
_exchange	$Pub\mathcal{U}$	w	Robot received a message from the lead agent related to team switching.
chosen, notChosen	$\mathcal{U}$	w	Process the message to determine whether it has been chosen to switch to the other team.
nearLF, notNearLF	$\mathcal{U}$	w	Robot determines whether a team member was detected or not.

Force component $\mathbf{u}_i^{\text{rn}}$ causes robot i to get repelled from all neighbouring robots. It is defined as

$$\mathbf{u}_i^{\text{rn}} = -\frac{1}{|\mathcal{N}_i|} \sum_{j \in \mathcal{N}_i} \frac{\tau \sigma^4}{||\mathbf{p}_{ij}||^5} \frac{\mathbf{p}_{ij}}{||\mathbf{p}_{ij}||} \tag{3.4}$$

where $\mathcal{N}_i$ is the set of neighbours of robot i, σ is the target distance to maintain between the neighbours, and τ is the gain. The force is calculated using the generalised form of the Lennard-Jones (LJ) potential (also known as the Mie potential).

In addition to $\mathbf{u}_i^{\text{rn}}$, the force component $\mathbf{u}_i^{\text{ro}}$ uses proximity sensors to prevent collisions if the distance to an obstacle gets too close. It is defined as

$$\mathbf{u}_i^{\text{ro}} = -\frac{1}{8 d_{\max}} \sum_{j=1}^{8} (d_{\max} - d_j) \hat{\mathbf{v}}_j \tag{3.5}$$

where $d_{\max} = 10\,\text{cm}$ is the assumed range of the proximity sensors, d_j is the distance extracted from the j^{th} sensor, and $\hat{\mathbf{v}}_j$ is the unit vector pointing from the robot's centre to the j^{th} sensor. Where sensor j detects no object, we set $d_j = d_{\max}$.

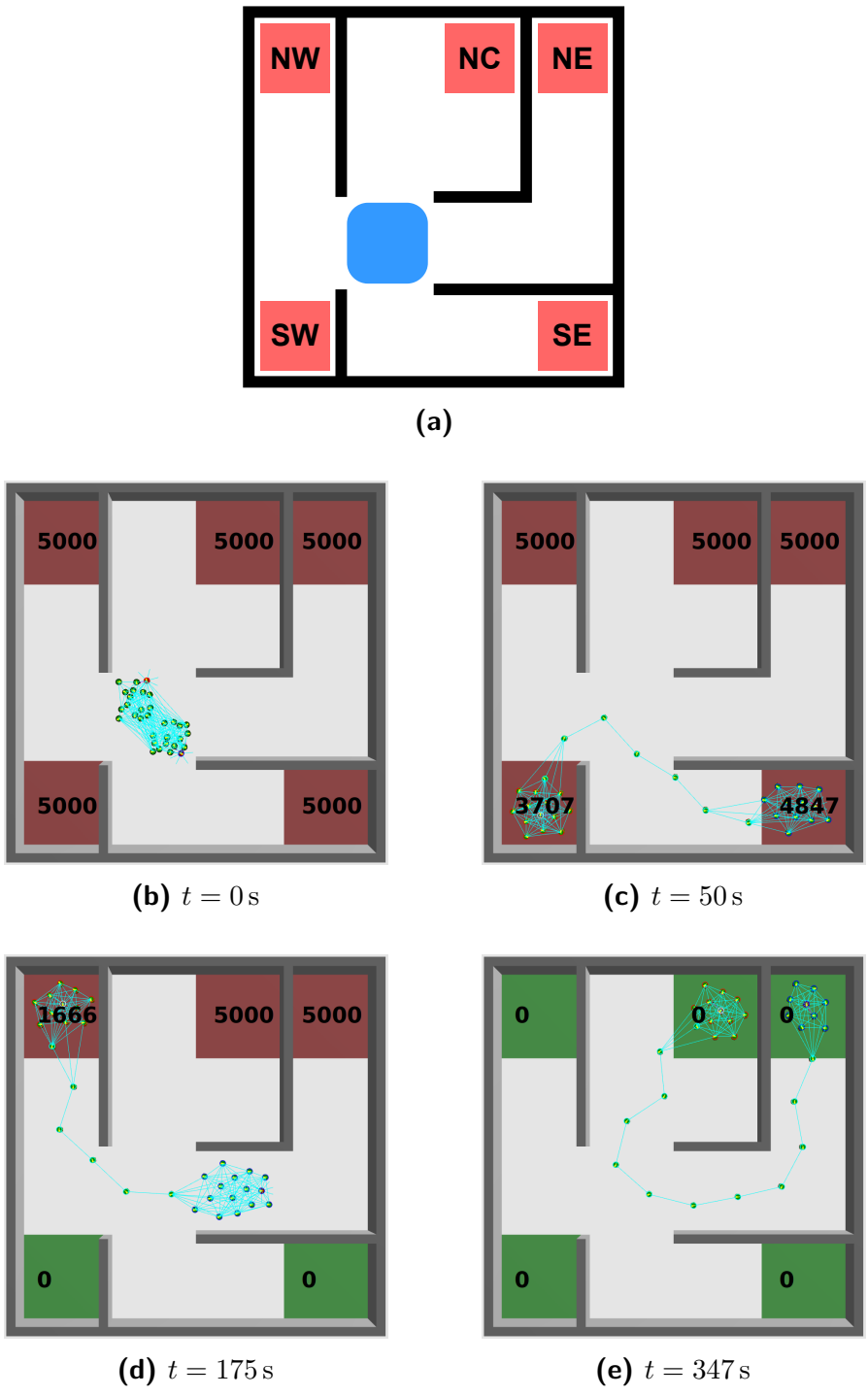
3.3 Simulation Studies

This section presents the simulation studies conducted to evaluate the connectivity-preserving swarm using simulated human operators.

3.3.1 Simulation Setup

We evaluate the performance of our method in simulation using the ARGoS simulator (Pinciroli et al., 2012). Trials were conducted in a $4\,\text{m} \times 4\,\text{m}$ arena containing walls (Figure 3.13a). Five tasks were distributed in the arena. The blue region represents the area where the lead agents and workers were initially deployed, separated as two team clusters.

We use the e-puck (Mondada et al., 2009) for our study. The initial position of the robots within each cluster was randomised in each trial. We use a communication range of $r_{\text{com}} = 0.8\,\text{m}$ for all robots, safety limit of $r_{\text{safe}} = 0.5\,\text{m}$, flocking weights and parameters of $\alpha = 1$, $\beta = 2$, $\gamma = 15$, $\tau = 1000$, and $\sigma = 0.15$ in all trials. A video recording of an example trial is available from the following link `https://youtu.be/Am9uaiGgg78`.



(a)

(b) $t = 0\,\mathrm{s}$

(c) $t = 50\,\mathrm{s}$

(d) $t = 175\,\mathrm{s}$

(e) $t = 347\,\mathrm{s}$

Figure 3.13. **(a)** The bounded arena with walls. Red areas represent task locations. The two lead agents and all workers are deployed in the blue area. **(b–e)** shows a sequence of snapshots taken from one of the simulations where 30 workers maintain a chain between two lead agents while performing the tasks. The numbers represent the task's remaining task demand. **(b)** The robots in their initial positions. **(c–e)** The simulation after 50, 175, and 347 seconds.

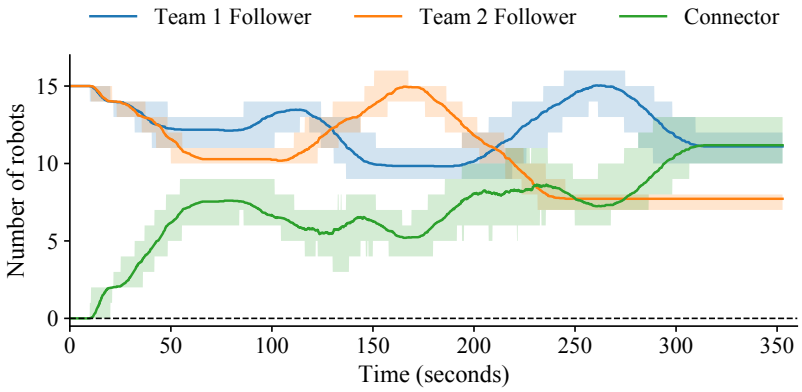


Figure 3.14. Average number of 30 workers in each role. Followers are shown separately according to their team. The solid lines represent the mean and the transparent regions represent the minimum and maximum values.

3.3.2 Performance of Chain Management

We first analyse the workers' ability to maintain a chain between the lead agents. Each task is given an initial demand of $w = 5000$ and requires $n^{\min} = 1$ robot to perform. Simulated operators are used to drive the lead agents to task locations and signal their workers to start or stop performing tasks. Each operator sends one heartbeat message per second to each other, which then should be relayed via the chain. We record whether these messages are successfully received. Fifty trials are performed with 20, 30, and 40 workers, respectively, that are split equally among the teams at the start. Trials are terminated once all tasks are completed or once 700 seconds have elapsed, whichever happens first.

Figures 3.13b – 3.13e shows a sequence of snapshots from a typical trial using 30 workers. The two operators first ask their teams to complete tasks SW and SE, respectively, then move to focus on tasks NW, NE and NC. Figure 3.14 shows the average number of workers in each role when using 30 workers. After initial deployment, the number of followers in each team reaches back to 15 workers for team 1 at around 260 seconds and for team 2 at around 165 seconds. This demonstrates the worker's ability to both build a chain to explore new areas but then subsequently shrink it, recovering the original state.

The full results are reported in Table 3.3. While task completion time decreased as more workers were used, the number of connectors at the end of the mission remained similar across all trials. This highlights how the number of connectors used to maintain a connection between the two teams is similar regardless of the number of workers in

Table 3.3. Simulation results for chain management.

No. of robots	Completion time (s)	Final no. of connectors	Successful message (%)
20	517.55	10.68	97.7
30	340.17	11.18	98.4
40	298.98	11.91	97.3

Table 3.4. Simulation results for robot exchange.

	Completion time (s)	Wait time (s)	Distance (m)
Team Migration	482.02	164.30	100.46
Robot Exchange	461.69	87.43	21.03

each team. Over 97% of messages were successfully exchanged between the lead agents in all three team sizes. This shows that the robot chain remained connected and the workers were able to relay the message as intended by the models.

3.3.3 Performance of Robot Exchange

To validate the exchange of robots over the chain, the scenario is altered by changing the minimum number of robots needed for task NC to be $n^{\min} = 15$ and the task demand of task NW to be $w = 7500$. The leader would request as many robots that were needed at the time of discovering the task, plus two. As a control study, we compare against the case where the whole team, receiving the request for additional robots, migrates upon completing their task, rather than sending the requested robots. The trials are repeated 50 times for each method using 30 workers.

The results are reported in Table 3.4. The strategy of exchanging robots along the chain completes the whole mission significantly faster than the strategy of migrating the whole team along the chain (two-sided, Mann-Whitney test, $p<0.001$). We also analysed the duration an operator has to wait from the moment of placing the request until the workers perform the task. For the robot exchange strategy, the wait time for task NC was almost half of that for team migration. This is because, for robot exchange, an operator can immediately send robots upon receiving a request even if it is currently working on a task. The total distance travelled by the lead agent and its workers since receiving the request is also largely reduced as only the requested

number of robots needed to travel to the other team. These results illustrate how robot exchange can be used by operators to efficiently share available workers together.

3.4 Summary

This chapter presented a method for two human operators to exert shared control over a swarm of robots to cooperatively perform spatially distributed tasks. SCT was used to formally design the control logic, which required only a formal description of the system and specifications, plus the implementation of all event-specific callback functions. The robots autonomously established a connectivity-preserving communication network that enabled the two lead agents (or human operators) to interact while independently moving through the environment. The communication network supported the exchange of robots between operators. The presented method was validated using embodied simulations.

The connectivity-preserving robot swarm presented in this chapter has so far only been tested with simulated human operators. The next chapter investigates the ability of real human users that interact with such robot swarms; how the ability to dynamically share robots and different communication types affects task performance and human factors.

Chapter 4

Sharing the Control of Robot Swarms: A User Study

The previous chapter considered the problem of forming a dynamic chain of robots connecting two operators exploring an environment. SCT was used to formally model the behaviour of both the lead agent and worker robots. The lead agents guided a subset of the swarm to work on tasks that each required different numbers of robots to complete. However, the behaviour of the lead agents was hard-coded: no human operators were involved. This chapter examines the ability of human operators in sharing robots using the aforementioned swarm.

One factor that affects the operators' performance is communication. Poor communication quality can hinder cooperation between operators as it becomes difficult to maintain a shared mental model of the situation. In addition, excessive communication can become an overhead for the operators, resulting in increased cognitive workload (MacMillan et al., 2004). Implicit or indirect communication (e.g. via a computer interface) can be effective when undertaking highly interdependent tasks, provided that the members have a sufficient understanding of the situation (Rico et al., 2008).

Another factor is the user interface. Achieving a high level of transparency is important, especially when operators are using the swarm as a shared resource (Roundtree et al., 2019). Information such as the robots' position and internal states can be helpful in understanding the state of the swarm. However, it may not be possible to obtain such information from all the robots of the swarm, and in practical scenarios, the operators may have to work with the information that is available locally on a single

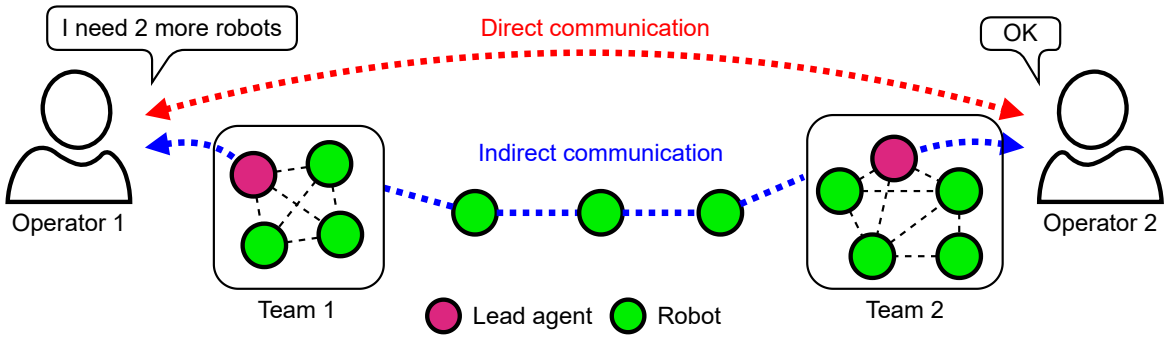


Figure 4.1. Illustration of two human operators each controlling a subset of the robot swarm (e.g. a team) via a lead agent. Other robots maintain connectivity between the two teams. An operator can request robots from the other operator either directly (i.e. verbally communicating) or indirectly (i.e. sending a message through the swarm).

robot. Having such restricted situational awareness can have a detrimental effect on the operators' ability to interact with a swarm (Kapellmann-Zafra et al., 2016), making it challenging for them to collaborate.

This chapter investigates the ability of multiple operators to dynamically share the control of robot swarms and the effects of different communication types on performance and human factors. To understand if the sharing of robots can be used effectively, the performance of a pair of operators in completing a set of spatially distributed tasks is compared with and without the ability to share robots. As sharing robots requires some level of coordination between the operators, we also investigate how either direct or indirect communication affects performance (Figure 4.1). To conduct the user study, the framework presented in the previous chapter is extended with a novel user interface that allows each operator to drive the respective lead agent of their team while being provided with its local camera feed, and request from or send robots to the other team. This is in contrast to previous work on multi-operator control of robot swarms as it (i) considers the case that the operators have dedicated teams and can exchange robots between them and (ii) restricts each operator's situational awareness by providing only a local view of the environment.

This chapter is organised as follows. Section 4.1 describes the scenario considered in this work, the system architecture that enables multiple users to simultaneously interact with a robot swarm, the graphical user interface, and the experimental design. Section 4.2 presents the experiment and questionnaire results obtained from the user

study. Section 4.3 analyses the results and discusses the findings. Finally, Section 4.4 summarises the chapter.

The work in this chapter is based on the author's publication (Miyauchi et al., 2023) and has been adapted for this thesis.

4.1 Methodology

In this section, the experimental design and the system used to conduct the user study are presented. The scenario considered in this study is described in Section 4.1.1. Next, the robot and simulation platforms are presented in Section 4.1.2, followed by the graphical user interface used by the operators presented in Section 4.1.3. Section 4.1.4 describes the performance measures, subjective questionnaires, a summary of the participants, and the procedure of the experiment conducted.

4.1.1 Scenario

Our study investigates the ability of two human operators to control a swarm of robots to complete a set of tasks scattered across a bounded environment with no obstacles. We use the connectivity-preserving robot swarm presented in the previous chapter. Each operator controls a *lead agent*, allowing them to move through the environment and provide instructions to the robots.

The *workers* are robots that are capable of completing tasks. At any moment in time, each worker is assigned to at most one lead agent. Throughout the mission, the workers autonomously maintain connectivity between the two teams by forming a robot chain. The workers can either assume the roles of *follower*, *connector*, or *traveller* as seen in Figure 4.2a and summarised in Table 3.1. Here, we describe the workers' behaviour in each role and any modifications made from the previous chapter.

Follower behaviour. A worker assigned to a lead agent is a follower. A follower performs a flocking motion based on virtual forces. It is attracted towards the lead agent and other followers within the same team and repulsed from any other lead agents or workers. When the lead agent and itself are both inside the same task area, it is considered to be working on that task. When the distance to the other team increases, the followers will leave the team, one at a time, to become a connector to maintain connectivity between the two teams by forming a robot chain.

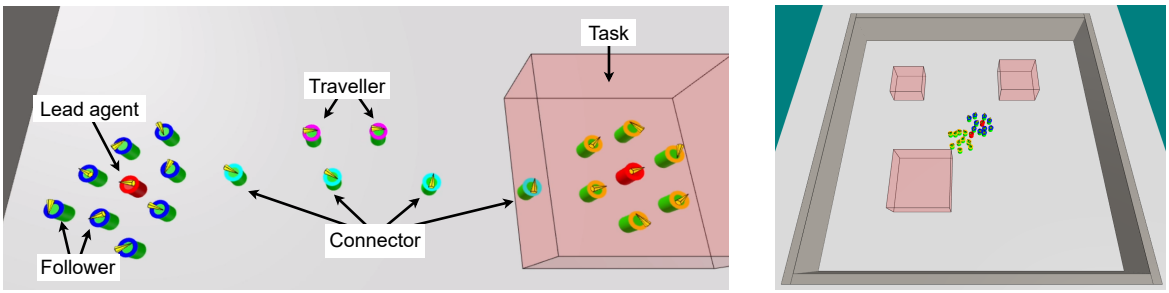


Figure 4.2. Overview of simulation scenario. **(a)** *Lead agents* and *workers* are represented as red and green cylinders respectively; yellow arrows represent their headings. The workers can assume three roles (indicated by the colour of their LED ring): *followers* (blue or orange) are part of a team, *connectors* (cyan) maintain connectivity among teams, and *travellers* (magenta) switch from one team to another. The task to complete is shown as a red transparent box. **(b)** Overhead view of an example arena containing three tasks. Tasks with a larger area require more followers to complete. Once a task is completed, it is removed from the arena and a new task appears at a random position.

Connector behaviour. A worker maintaining the robot chain is a connector. They allow the two teams to exchange information with each other. Unlike in the previous chapter, the connectors minimise the length of the chain by maintaining a straight line between the two teams. This is done by each connector finding the distance and angle to its two adjacent neighbours in the chain and moving to the middle point between these two neighbours. If the two teams move toward each other, the robot chain becomes shorter; the connector closest to the team leaves the chain to join the team.

Traveller behaviour. The lead agents are able to send a specified number of their followers to the other lead agent. When a follower receives a signal from its lead agent to join the other team, it becomes a traveller. A traveller moves along the robot chain until it reaches the other team to become a follower of that team. This allows the operators to rebalance the number of followers in each team.

Each task requires a specific number of followers to complete, which is visible to the operator when inside the task area. When outside, the operator can make a guess, as the task areas have lengths and widths of 0.4 m, 0.5 m, 0.6 m, 0.8 m, or 1.0 m, for tasks requiring 1, 3, 6, 9, or 12 followers, respectively. This resembles a search-and-rescue scenario in the real world; a rescuer might be able to predict the size of a task, but may only find out the exact number of robots or resources needed to accomplish it after

arriving at the mission site. To complete a task, the lead agent and its followers must remain inside the task area for a certain duration. The time required to remain inside the task area increases linearly to the number of followers needed to complete it, and having an excess number of followers does not make the task complete faster. Unlike in the previous chapter, the operators are no longer required to specifically instruct the robots to work on the task. These changes regarding how tasks are executed have been simplified to make the experiments suitable for the user study. In the extreme cases considered—single-robot and 12-robot tasks—the followers need to remain for 5 s and 30 s, respectively. Completing a task awards the pair of operators a score equal to the number of followers that were required to complete it. Once a task is completed, it is removed from the environment and a new task appears at a uniformly random location without overlapping with existing tasks.

The operators' joint objective is to score as many points as possible during the trial. The operators must guide their followers to the task areas, while also monitoring the number of followers in their team to ensure they have enough to complete the tasks. The full robot controller models can be found in Section 3.2.

4.1.2 Robot and Simulation Platform

The operators interact with the simulated robots running in the ARGoS simulator (Pinciroli et al., 2012). The simulation consists of 2 lead agents and 20 workers in a 4 m×4 m arena (Figure 4.2b). We use the e-puck (Mondada et al., 2009) for our study, which is a mobile differential-wheeled robot with a diameter of 7 cm. We use a maximum speed of 8 cm/s. The e-puck has eight proximity sensors with a range of 0.1 m distributed around its body. We assume a range-and-bearing system with a range of 0.8 m to communicate with neighbouring robots.

The code used for the study in this chapter is available from the following link: `https://gitlab.com/genki_miyauchi/multi-human-swarm-control`.

4.1.3 User Interface

The operators interact with the simulated robot swarm through a custom graphical user interface based on Webviz (Patel et al., 2022), a web interface plugin for ARGoS (Pinciroli et al., 2012). Webviz uses a client-server architecture, allowing multiple users to simultaneously interact with the same simulation from different devices (Figure 4.3).

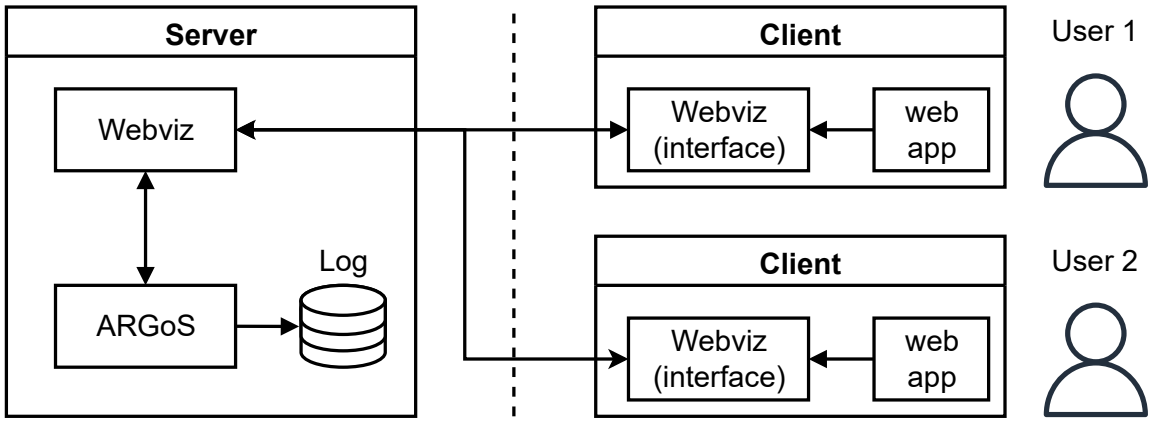


Figure 4.3. System overview. A user accesses the shared swarm simulation through a simple web-based app. When the user interacts with the interface, the inputs are sent to the server and reflected in the ARGoS simulation. The client-server architecture is realised using Webviz.

Figure 4.4 shows the interface presented to a user. It shows a first-person view of the environment from the lead agent's perspective. The interface consists of four panels that support the user in completing the tasks; *team information panel*, *task information panel*, *request-and-send panel*, and *log panel*. No overhead view of the environment is provided. The interface reflects the assumption in this work that lead agents are situated in the environment and are tele-operated by the user. The users have access to their surrounding information as well as information about the other operator via their respective lead agents. The users can drive the lead agent inside the simulation using a keyboard. Figure 4.5 illustrates a user controlling a lead agent to visually locate the task.

Team information panel. This panel provides information for the user to determine whether it or its partner has enough followers to complete their respective tasks. It is placed top-left in the interface and shows information about the two teams. The panel consists of two parts, which display the number of followers in the user and partner's team, respectively. In addition, if the partner is inside a task area, it also displays the size of that task (i.e. the number of followers needed) next to the partner's follower count.

Task information panel. This panel provides information about the current task and is placed at the top-centre of the interface. When a user is inside a task area, it displays the size of the task, the number of followers currently inside the task area, and a progress bar that fills up when the followers are performing the task. The team

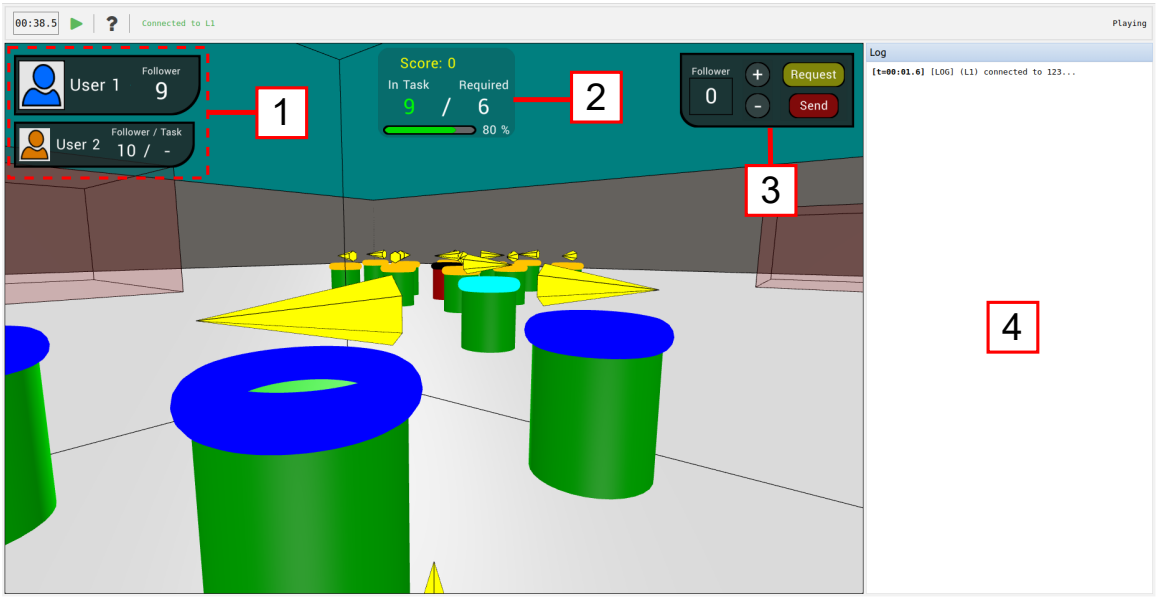


Figure 4.4. The interface used for the experiment. The operator is shown a first-person perspective of the lead agent situated in the simulated arena. The interface displays (1) the team information panel, (2) the task information panel, (3) the request-and-send panel, and (4) the log panel. No overhead view or mini-map of the arena is provided to the user.

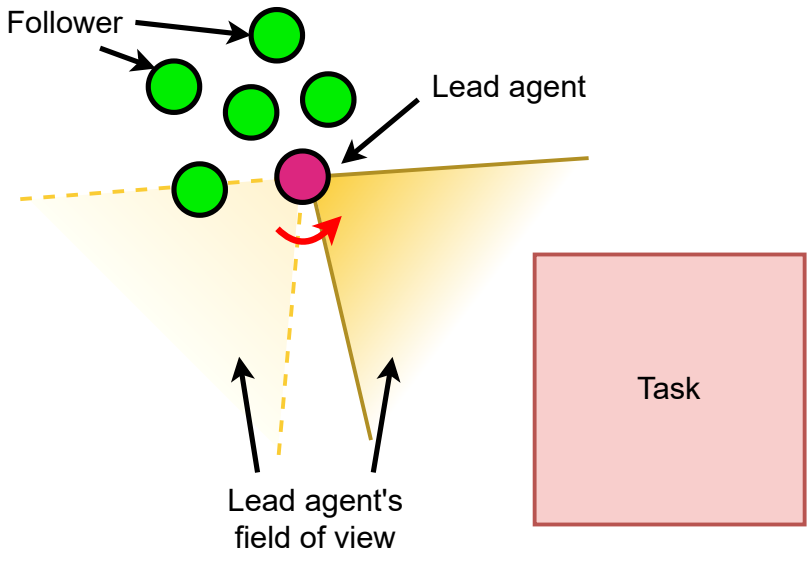


Figure 4.5. Illustration of the lead agent's field of view accessible to the user. The dotted and solid yellow regions show the previous and current field of view of the lead agent after rotating on the spot (shown by the red arrow), respectively. The user does not initially see the task until they rotate the lead agent to face the task.

score is also displayed at the top, which updates when either of the users completes a task. If the user moves outside of the task area, information about that task will no longer be displayed until the user re-enters the task area. Any progress made towards a task will remain even if a user exits the task area before it is completed.

Request-and-send panel. This panel is used to request or send followers to the partner and is placed top-right in the interface. A user can specify the number of followers using the plus and minus buttons and then press either the request or send buttons to confirm the action. If the `Request` button is selected, a message containing the specified number of followers will be sent to the partner via the robot chain. If the `Send` button is selected, the specified number of followers from the user's team will begin to travel to the partner's team. This allows the users to explicitly share their followers without needing to communicate with each other verbally.

Log panel. This panel displays the request and send messages received from the partner as well as any messages sent by the user, in reverse chronological order. It is placed on the right side of the interface. By monitoring the log panel, a user can keep track of their partner's request or confirm that their partner has sent followers to their team.

4.1.4 Experiment Design

Our experiment followed a 2×2 mixed factorial design (Wickens et al., 2004), where the robot-sharing condition (robot-sharing [RS] vs. no-robot-sharing [NRS]) was the within-subjects factor and the communication type (direct [DIR] vs. indirect [IND]) was the between-subjects factor. Within-subject evaluations assessed whether the dynamic sharing of robots affected the performance and human factors. Between-subject evaluations compared whether differences in communication style affected the performance and human factors. This means each participant experienced both robot-sharing conditions (RS and NRS) in one of the communication types (DIR or IND). Figure 4.6 summarises the experiment design.

In the RS condition, the operators were able to request or send robots to each other using the interface. If an operator found that it did not have enough followers to execute the task, they could request followers from their partner. In the NRS condition, the operators could not share robots. This meant some tasks required more robots than what was originally available in a single team. In such cases, the task could still

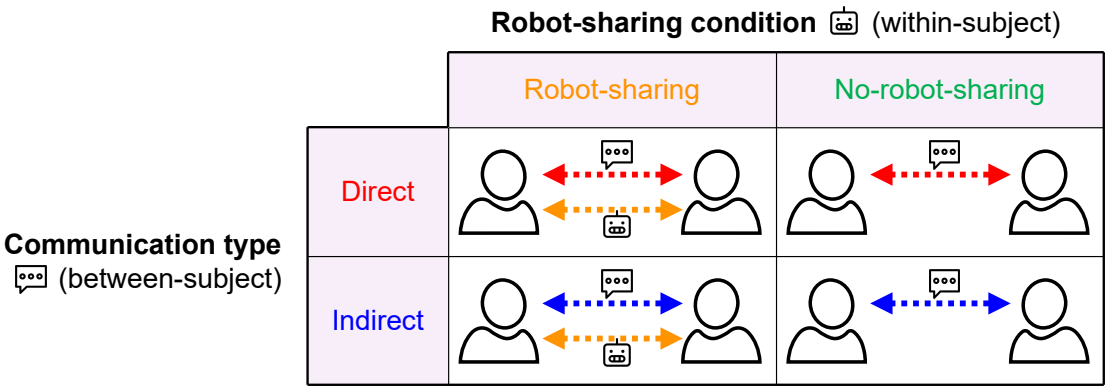


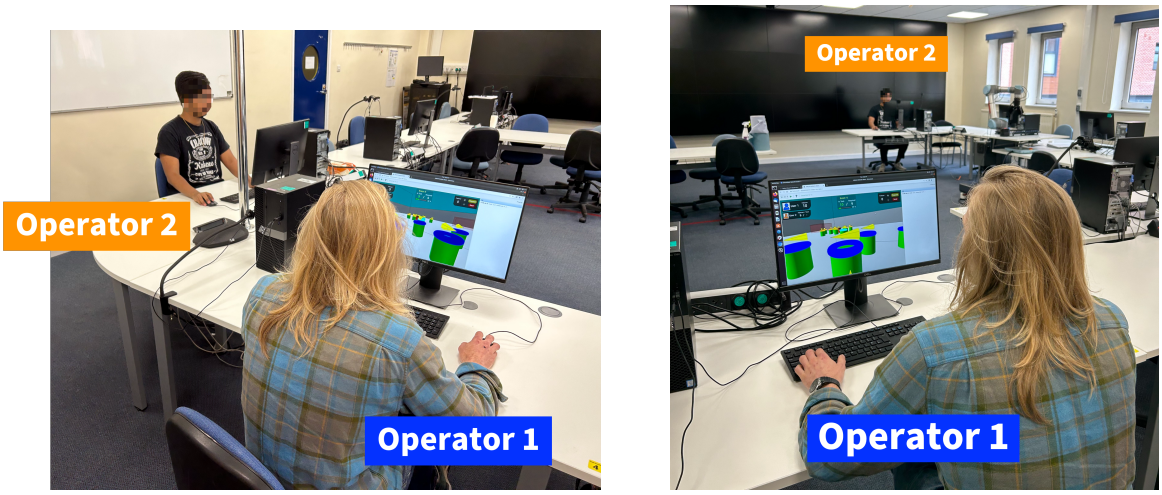
Figure 4.6. Illustration of the 2×2 mixed factorial design conducted in this study. Red and blue dashed arrows between participants indicate the communication type. Orange dashed arrows are shown if the participants were able to send robots to each other.

be completed if both teams entered the task area and the sum of followers reached the required number of followers for the task.

In DIR communication, the operators verbally communicated with their partners throughout the trial (Figure 4.7a). They were seated nearby but were unable to see each other's computer screens. To share robots, an operator verbally requested for more robots from their partner. Only the `Send` button was displayed as the operators directly asked their partners for more robots. In IND communication, the operators were not allowed to verbally communicate: they were only able to request a specific number of robots from their partner via the interface (Figure 4.7b). To ensure the operators do not communicate using non-verbal means, such as eye contact and gestures, up to six operators were invited to each experiment session and were randomly paired. The operators were not informed of who their partners were, making it difficult to identify who they were working with. The operators were not able to see their partner's computer screen. The operators used the `Request` button to request robots from their partners. The operator that received a request, either verbally or through the interface, then decided how many of its followers to send.

Performance Measures

Performance measures include *task score*, *distance travelled*, *team separation*, and *robots shared*. These data were obtained from the simulation logs, which recorded the position and state of every robot and task, and all inputs received from both operators. Task



(a) (b)

Figure 4.7. Illustration of how the communication types were controlled. **(a)** Direct communication, where a pair of participants were seated close to each other and were able to communicate verbally. **(b)** Indirect communication, where each participant was paired with another unknown participant seated far from them and was not able to communicate verbally. A pair of participants experienced only one of the two communication types.

score is the points obtained during the trial. If a task was being performed when the trial ended, we awarded partial points for the proportion of the task that was completed. Distance travelled is the total distance moved by the lead agents and workers during a trial. It is used here as a proxy for the total energy consumed by the swarm. Team separation is the average distance between the two teams throughout the trial. The separation was calculated by finding the geometric centre of both teams and taking the distance between the two points. This tells us how spread out the two teams were during the trial. For the RS condition, we recorded the number of followers shared between the teams.

Subjective Questionnaires

Subjective data were obtained individually through a paper-based questionnaire after each trial. The participant's subjective workload was obtained using the NASA-TLX (Hart and Staveland, 1988) on a 7-point Likert scale. Subjective situational awareness was obtained using the Situational Awareness Rating Technique (SART) (Taylor, 1990) on a 7-point Likert scale. We also asked participants about how well they understood their partner or the robots' actions, the usability of the interface, and

whether they found the ability to share robots useful on a 7-point Likert scale. The questionnaire concluded with open-ended questions asking about what their strategy was in completing the tasks and any additional comments they had about the experiment. The questionnaires used in the experiment can be found in Appendix B.

Participants

The study received ethical approval from The University of Sheffield. All participants were staff and students within the university. Participants were recruited within the university through a volunteering mailing list. Potential participants who showed interest completed an interest form, read the information sheet, and booked an available session. Participants were encouraged to choose a session which had an odd number of participants signed up at that time since the experiment had to be completed in pairs.

A total of 52 participants (34 males and 18 females) completed the experiment. All participants were adults over 18 and belonged to one of the age groups below (*Under 20*: 1, *20-29*: 28, *30-39*: 14, *40-49*: 5, *50-59*: 3, *60 & Over*: 1). Among the 52 participants, 28 participants experienced DIR communication, while 24 participants experienced IND communication.

Procedure

First, the participants completed a preliminary questionnaire that asked about their previous gaming experience. Next, the experimenter gave an introduction on how to use the interface and explained the participant's goal in the trials. The slides presented to the participants during this induction can be found in Appendix A. This was followed by a short training session for the participants to familiarise themselves with the robots' behaviours and the functionality of the interface.

For the main trials, participants were randomly paired to work together. The pair of participants each interacted with the robot swarm through the interface to score as many points as possible until the trials ended. The robot-sharing condition was controlled by showing or hiding the request-and-send panel in the interface. The communication type was controlled by assigning the pair to either DIR or IND communication. Participants completed two trials to experience both robot-sharing conditions (RS and NRS) within their assigned communication type (DIR or IND). The order of the robot-sharing conditions was counterbalanced across participants to minimise any learning effect. Each trial lasted for 10 minutes. After each trial, the

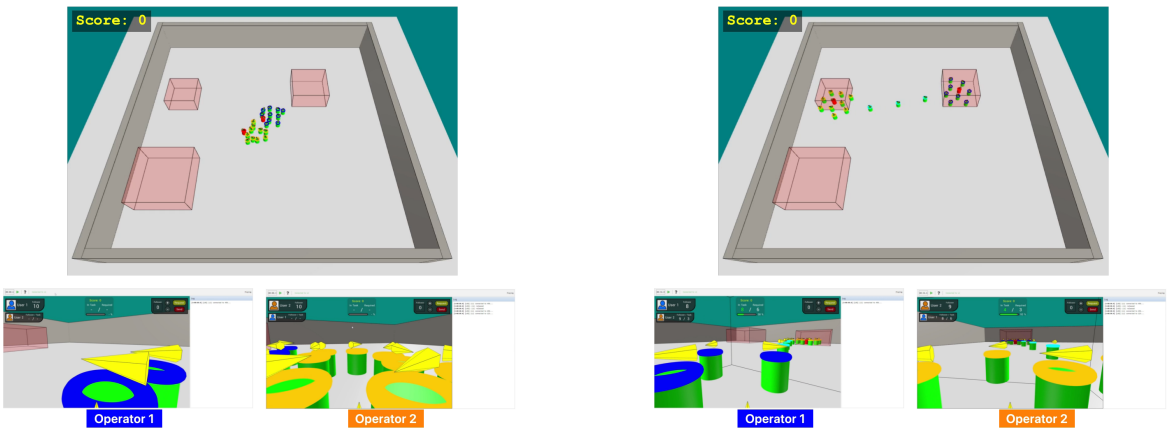
participants individually completed a post-trial questionnaire. Participants took a short break before moving on to the second trial. The overall experiment took around 75 minutes.

4.2 Results

The participants were randomly paired to form 26 teams. Among the 26 teams, 14 teams experienced DIR communication, while the remaining 12 teams experienced IND communication. The experiments were conducted on different dates over six weeks.

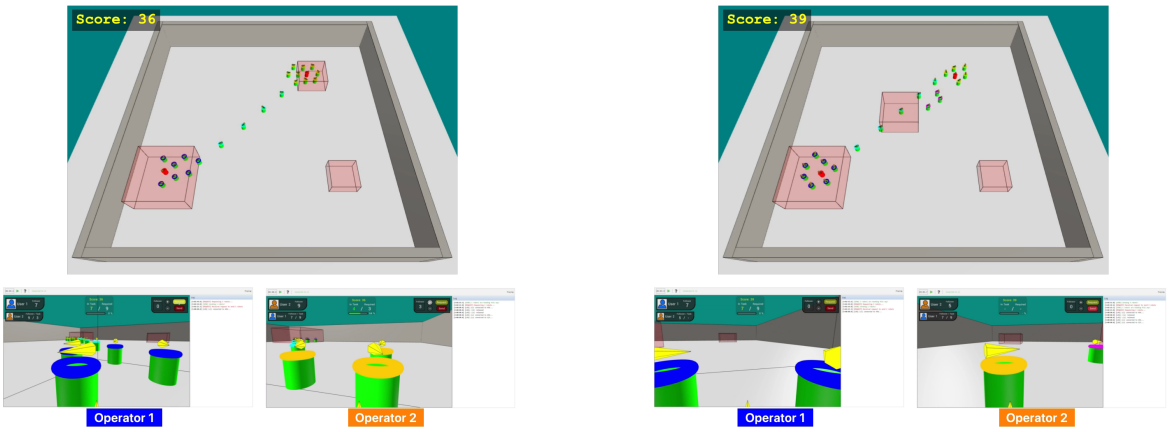
Figure 4.8 shows a sequence of snapshots from a trial conducted to test the functionalities of the simulation and user interface. At the beginning of the trial, the lead agents and workers were clustered near the centre of the arena that consisted of three tasks (Figure 4.8a). Once the trial started, the operators looked around their surrounding environment by controlling the lead agent and moved towards a task (Figure 4.8b). Figures 4.8c – 4.8e show the operators sharing robots to satisfy task requirements. In Figure 4.8c, Operator 1 sends a request for three workers as it does not have enough followers to complete the task; the task requires nine workers, while the operator only has seven. Here, the operator requested an additional worker than needed to ensure it will have enough workers to complete the task, as some followers may become connectors while it waits for the workers to arrive. This strategy was observed in some participants during the trials. Figure 4.8d shows three workers moving towards Operator 1's team and are seen to have arrived in Figure 4.8e. Figure 4.8f shows the end of the trial. A video recording of this trial is available from the following link: `https://youtu.be/DRYU4v8kkuo`.

We performed quantitative and qualitative analyses to examine the effect of the robot-sharing capability and communication type between two operators. Quantitative analyses were based on the simulation logs, which recorded the states and positions of robots and tasks, the points scored, and the user inputs to the interface at each time step. Qualitative analyses were based on questionnaire responses, which asked the participants to rate their experience on a set of scales. Table 4.1 summarises the results. A two-way mixed analysis of variance (ANOVA) was conducted to examine the effect of the two factors. Results are reported as significant when $p < 0.05$ and marginally significant when $p < 0.08$.



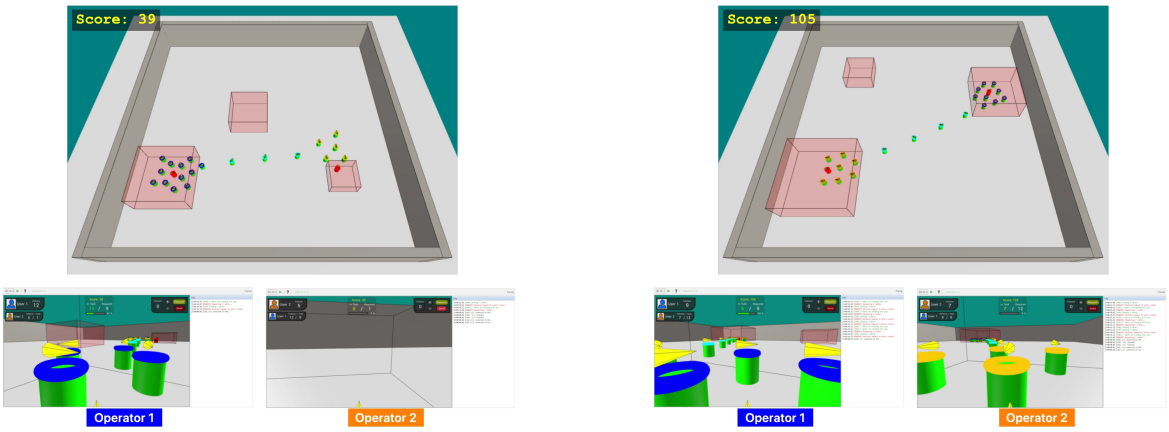
(a) $t = 0\,\mathrm{s}$

(b) $t = 37\,\mathrm{s}$



(c) $t = 169\,\mathrm{s}$

(d) $t = 184\,\mathrm{s}$



(e) $t = 216\,\mathrm{s}$

(f) $t = 600\,\mathrm{s}$

Figure 4.8. Sequence of snapshots showing an overhead view of the environment and the interface controlled by each operator in an example trial with IND communication and RS condition. The score indicates the performance of the team in completing tasks. The overhead view was not provided to the operators.

Table 4.1. Summary of performance measure and questionnaire results. N is the number of samples and SD is the standard deviation.

Measure	Condition	Communication	N	Mean	SD
Task Score	RS	DIR	14	100.39	16.84
		IND	12	92.25	16.71
	NRS	DIR	14	106.31	13.97
		IND	12	99.20	16.19
Distance Travelled (m)	RS	DIR	14	10.28	1.44
		IND	12	10.51	1.22
	NRS	DIR	14	10.79	1.59
		IND	12	11.54	1.29
Team Separation (m)	RS	DIR	14	1.42	0.21
		IND	12	1.48	0.14
	NRS	DIR	14	0.96	0.17
		IND	12	1.00	0.18
Robots Shared	RS	DIR	14	22.79	11.19
		IND	12	25.33	7.23
	NRS	DIR	—	—	—
		IND	—	—	—
Workload	RS	DIR	27	3.62	1.10
		IND	24	3.56	1.04
	NRS	DIR	28	3.42	1.00
		IND	24	3.18	1.09
Situational Awareness	RS	DIR	28	4.44	0.78
		IND	23	4.49	0.76
	NRS	DIR	28	4.36	0.65
		IND	24	4.14	0.72

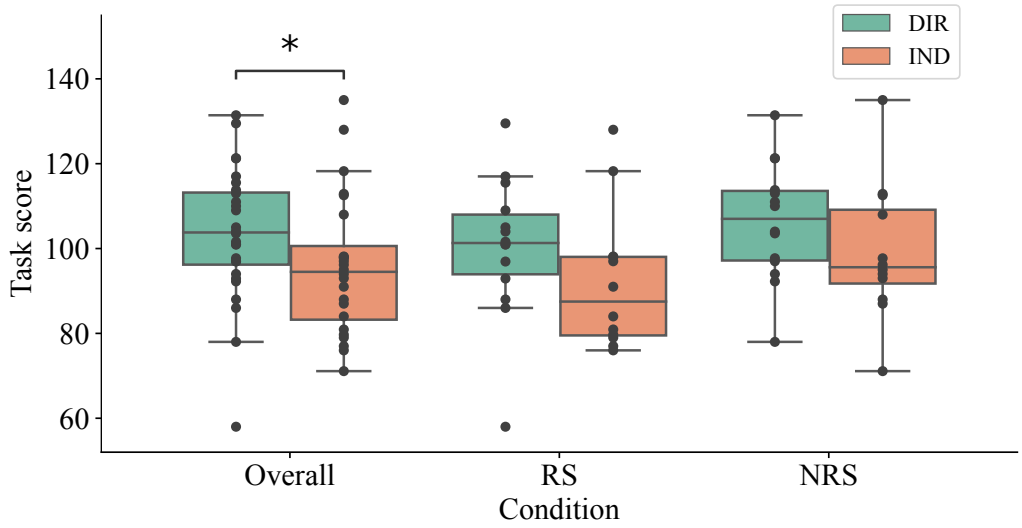


Figure 4.9. Comparisons of task score by robot-sharing condition and communication type. Statistically significant results ($p < 0.05$) are indicated with an asterisk ($*$).

A comparison between the communication types indicates that there was a significant difference in task score ($F(1, 23) = 5.36, p = 0.030$). Figure 4.9 shows that participants in the DIR communication scored more points ($M = 103.35, SD = 15.48$) than those in the IND communication ($M = 95.73, SD = 16.48$). No significant difference was found in the points scored between the robot-sharing conditions ($F(1, 23) = 2.72, p = 0.112$).

The ability to share robots had a significant effect on the total distance travelled by the robots ($F(1, 23) = 4.53, p = 0.044$). Figure 4.10a shows the total distance travelled in the RS and NRS conditions. Since each pair of participants experienced both conditions, a line is drawn to show the paired data. On average, participants in the RS condition ($M = 10.39, SD = 1.32$) travelled less than in the NRS condition ($M = 11.14, SD = 1.48$), and therefore consumed less energy. No significant effect was found on the distance travelled between communication types ($F(1, 23) = 0.62, p = 0.441$).

The ability to share robots also had a significant effect on team separation ($F(1, 23) = 86.2, p < 0.0001$). Figure 4.11 shows that participants in the RS condition ($M = 1.45, SD = 0.18$) were more likely to stay apart from the other team than in the NRS condition ($M = 0.98, SD = 0.18$) throughout the trial. There was no reportable difference in team separation between communication types ($F(1, 23) = 0.98, p = 0.333$).

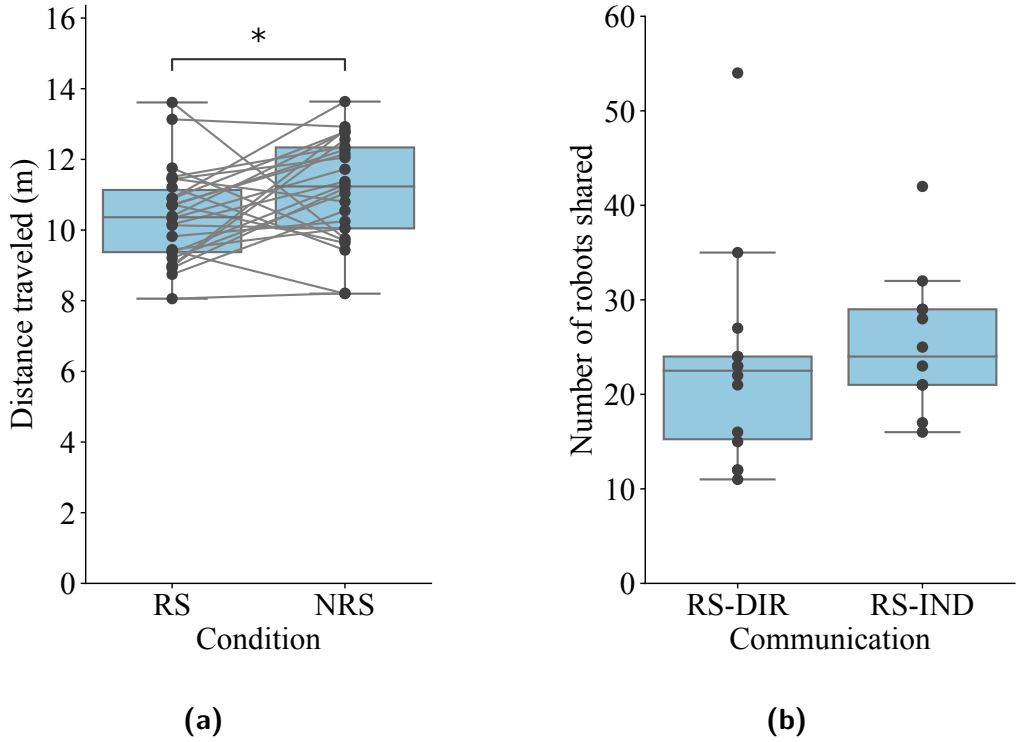


Figure 4.10. Comparisons of **(a)** the total distance travelled by the robots against the robot-sharing condition and **(b)** the total number of robots shared between the two teams in the RS condition against the communication type. Paired samples are joined by a line. Statistically significant results ($p < 0.05$) are indicated with an asterisk ($*$).

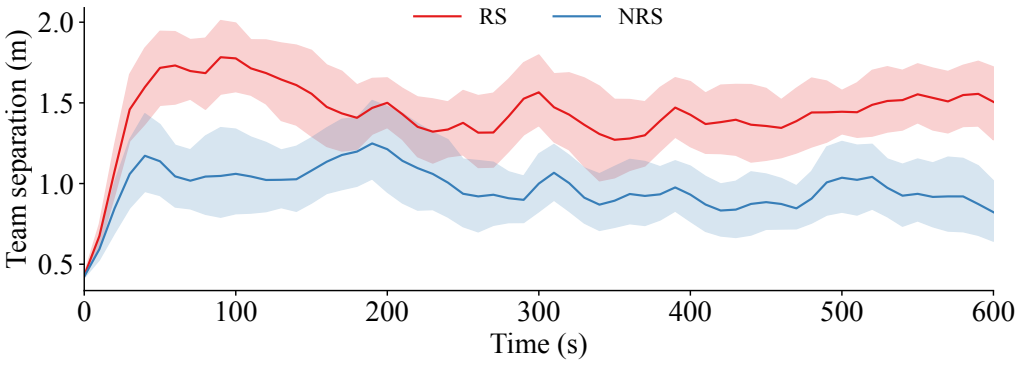


Figure 4.11. Average separation between the two teams under different robot-sharing conditions. The solid lines each represent the mean across 26 trials and the transparent regions represent the 95% confidence intervals.

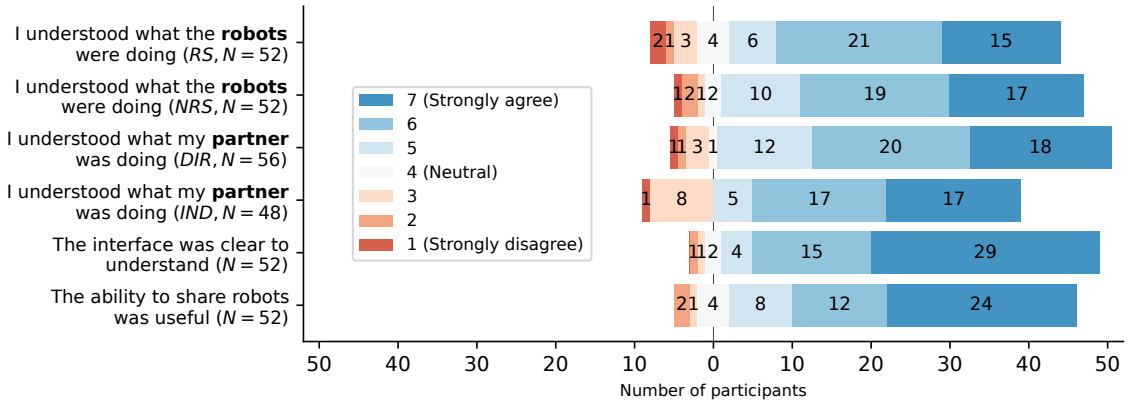


Figure 4.12. Post-trial questionnaire responses. Participants were asked to rate how well they understood their partner and the robots' actions, the interface, and whether they found the ability to share robots useful after the trial.

Within the RS condition, we examined if the communication type affected the number of robots shared between the participants. Figure 4.10b shows that there was no significant difference in the number of robots shared ($F(1, 24) = 0.46, p = 0.505$).

To compare the overall workload, we calculated the mean of the raw scores from the six categories in the NASA-TLX questionnaire. The ability to share robots had a marginally significant effect on the global workload ($F(1, 49) = 3.82, p = 0.056$). Participants in the RS condition ($M = 3.59, SD = 1.06$) reported a slightly higher global workload than in the NRS condition ($M = 3.31, SD = 1.04$). The overall situational awareness was also calculated using the mean of the raw scores from the nine categories in SART. The participants' global situational awareness was not affected by either communication type ($F(1, 49) = 0.18, p = 0.673$) or the ability to share robots ($F(1, 49) = 3.09, p = 0.085$).

4.3 Discussion

In all trials, every pair of participants was able to complete tasks and dynamically share robots, despite undergoing only a short training period (i.e. up to 10 minutes). Figure 4.12 shows that the lack of direct communication caused the percentage of participants understanding their partner's actions to drop from 89.3% to 81.3%. Despite the lack of direct communication and understanding of who their partners were, on average, participants in the IND communication were able to score 92.6% of the points

achieved by those using DIR communication. Overall, these results are in line with previous work, where the operators benefit from direct communication for coordinating their actions.

The ability to share robots (i.e. RS condition) decreased the total distance travelled by the robots, and hence the energy consumed by the swarm. This could be attributed to participants more likely focusing on different tasks at the same time. If the partner requested some followers, the participant could simply send the requested amount of followers, or even more if they choose to do so. By contrast, when participants were unable to share robots (i.e. NRS condition), they had to bring their teams into the same area if the task required many followers. This resulted in increasing the average distance travelled by each robot in the NRS condition.

Participants in the NRS condition were also hesitant to tackle larger tasks compared to those in the RS condition. Some participants reported that when they did not have the required number of followers to complete a task, they preferred to move on to another task when the other team was far away. This meant the teams in the RS condition were often exploring new task areas earlier on, as their ability to share robots provided added flexibility to meet the task requirements. This could be crucial during search-and-rescue missions.

Although the teams were faster at reaching the tasks in the RS condition, the task scores were slightly better in the NRS condition. This could be explained by the larger team separation seen in the RS condition. The larger team separation meant there were fewer followers in each team. This resulted in some participants spending more time waiting for the requested followers to arrive instead of working on the tasks. Furthermore, participants had to spend time on requesting robots and acknowledging the requests, which hindered some participants from completing many tasks. The best performing participants were those who prioritised tasks that required the same or slightly fewer followers than their current number of followers in the team. These participants were able to minimise the waiting time and obtain higher scores. The limiting factor in this study was the number of workers available to the participants compared to the size of the tasks. Increasing the number of workers may have increased the performance in the RS condition.

It was expected that participants in the RS condition would experience a higher workload than those in the NRS condition due to the addition of the robot-sharing capability. While this trend was observed, results indicated only a marginal significance, where the RS condition had a slightly higher workload. This suggests that the dynamic

sharing of robots between operators could be incorporated into human-swarm systems without having a significant effect on operator workload. According to the questionnaire responses in Figure 4.12, most participants found the ability to share robots useful, even among those that scored fewer points in the RS condition. Several participants expressed that they found the ability to share robots useful not only because they could adjust the number of followers in the two teams, but because it allowed them to work more independently on the tasks and potentially increase their performance by executing tasks in parallel. This behaviour to work independently matches with the increased team separation observed in the RS condition.

4.4 Summary

In this chapter, a user study was conducted to investigate the effects of dynamically sharing robots between two human operators on task-related performance measures and human factors. The operators were each provided only a local, first-person view of the simulated environment, and were accomplishing tasks, starting with preassigned teams of robots. They were able to request and share robots with each other either by communicating directly (i.e. verbally) or indirectly via simple messages that passed through the robot swarm. Our findings show that, despite the short training received, the operators were able to dynamically share robots under limited situational awareness. Although sharing robots did not necessarily increase task scores, it provided the operators with the flexibility to work independently or collaboratively, reduced the energy consumed by the swarm (i.e. the total distance travelled), and was considered useful by the operators. Regarding the communication type, a decrease in task scores was noted when verbal communication was prohibited. Further training could help operators understand when are effective moments to request or send robots to further improve task scores.

The study so far has focused on the case where two human operators share the control of a swarm of robots. The next chapter extends the connectivity-preserving robot swarm to facilitate the cooperation between an arbitrary number of operators.

Chapter 5

N-Operator Control of Robot Swarms

In Chapter 3, we proposed a framework that enables human operators to share the control of a robot swarm. The usability of the framework was examined through a user study in Chapter 4, which showed that the operators were able to successfully share the robots to complete tasks with different requirements. However, one shortcoming of the framework is its inability to facilitate collaboration between three or more operators, which may limit its application in the real world.

In this chapter, we extend the framework so that an arbitrary number of operators can interact with a robot swarm. Similarly to the previous chapters, each operator interacts with a dedicated lead agent to guide a subset of the swarm. We seek to minimise the number of robots that maintain connectivity between the operators as it allows most robots to work on tasks that the operators want to complete. Moreover, we minimise the length of the network to reduce the time robots would spend travelling between the teams. In the extended framework, the robots attempt to form a *Euclidean Steiner tree*, which is a minimum spanning tree connecting all terminal points (i.e. the teams) with the addition of *Steiner points* (Section 5.2.1). In order to adapt to the lead agents moving freely in the environment, the robots dynamically update the network by (i) creating and repositioning branches and (ii) adjusting the positions of robots maintaining it. We also propose a strategy an operator can employ when requesting robots from all other operators to receive the number of robots needed.

This chapter is organised as follows. Section 5.1 defines the problem formulation, particularly highlighting the changes from Chapter 3. Section 5.2 describes the process

of creating and repositioning branches in the network, the new SCT models, the motion of the workers in each role, and the strategy for sharing robots between multiple operators. Section 5.3 evaluates the extended framework with respect to a set of performance measures through multiple simulation studies. Section 5.4 concludes the chapter.

5.1 Problem Formulation

The problem considered in this chapter is based on the one described in Section 3.1. Two conditions are relaxed to focus on the scenario that involves $N \geq 2$ human operators that share control over a group of robots in an unbounded obstacle-free environment represented as $\mathcal{W} = \mathbb{R}^2$. This section describes the changes made to the original problem formulation.

The first change concerns how tasks are performed by the robots. Previously, each task was defined by the tuple $(A, w, n^{\min})$ and had an associated task demand w, which denoted the quantity of work to be performed. The rate at which a task was completed increased with the number of workers performing it. Furthermore, upon entering the task area, workers had to activate their working capability in order to reduce the remaining task demand. For this study, each task is defined by the tuple $(A, k, n^{\min})$, where $A \subset \mathcal{W}$ denotes the task area, k and $n^{\min}$ represent the time and the number of robots required to complete the task, respectively. The task is performed when the lead agent and a sufficient number of workers reside in the task area. This means tasks are performed at a fixed rate irrespective of the number of workers, provided the relevant threshold ($n^{\min}$) is met.

The second change concerns how communication between the robots are modelled. Previously, robots assumed line-of-sight communication to exchange information with another robot. For this study, we assume that any two robots (i.e. lead agents and workers) can communicate if the distance between them does not exceed the communication range r_{com}.

5.2 Methodology

We present an extension to the framework presented in Chapter 3, which helps preserve connectivity between an arbitrary number of operators while continuing to support

the sharing of information and workers with others. In Section 5.2.1, we describe our approach for workers to dynamically modify the network to maintain connectivity between multiple operators. In Section 5.2.2, we present the SCT models designed for the lead agents and workers. The motion of the workers in each role is described in Sections 5.2.3, 5.2.4, and 5.2.5. Section 5.2.6 presents a strategy for an operator to request more workers from all other connected operators.

5.2.1 Network Modification

To support an arbitrary number of operators, we discuss how the connectivity network dynamically evolves during the mission. First, we revisit the conditions to grow or shrink the branches in the network based on the original framework. Next, we define new conditions to allow workers to reposition the branches in the network.

Growing and Shrinking Branches

As the original framework considered two lead agents, each connector had to maintain connectivity with only two other connectors or teams to form a chain. To maintain connectivity between more than two lead agents, the connectors may need to create or remove branches to adapt to the moving operators. To do so, the conditions presented in Section 3.2 to dynamically grow and shrink the branches are slightly modified. A summary of the actors and roles involved in the scenario can be found in Table 3.1.

In the original framework, a follower switched to a connector and attached itself to the *tail connector*—the connector that is directly connected to the team—and became the new tail connector for the team it has just left. As before, we assume that a team has exactly one tail connector and members of the team (i.e. the lead agent and workers) are aware of this connector. In contrast, a connector can be a tail connector for multiple teams. Unlike in Chapter 3, which always had a tail connector only at the end of the chain, it is now possible for any connector in the network to be a tail connector.

For this study, we initialise a single worker as a connector (called the *initial connector*) and deploy it in a way so that it can detect at least one member from each team within its communication range. It essentially acts as a tail connector for all teams at the beginning and subsequent connectors would build the network from this connector. This allows us to assume that the network is a spanning tree in the

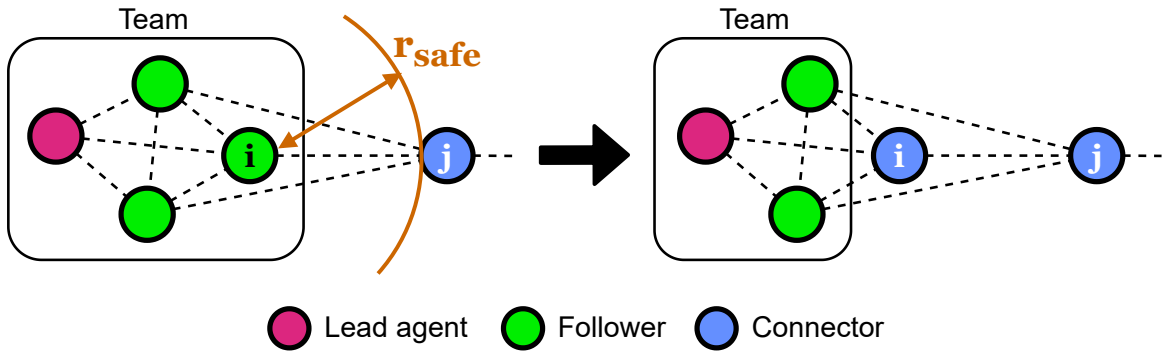


Figure 5.1. A situation prior (left) and after (right) a robot joins the network, thereby extending it. The box represents a team. Dashed lines represent connections between robots. Follower i is sufficiently far away from the tail connector j (condition $C1$), observes that it is closer to tail connector j than its peers (condition $C2$), and the lead agent is far from tail connector j (condition $C3$). It requests leaving its team to become a connector, which is approved by the tail connector.

beginning and to show that this property is retained through subsequent operations thereafter. Once the mission starts, the initial connector acts as a normal connector that may switch to become a follower.

Figure 5.1 illustrates the situation in which a new connector extends the network. The conditions for follower i that is part of team u to become a connector are as follows.

- $C1$: The distance between follower i and the tail connector of team u is greater than the safety limit $r_{\text{safe}} + \varepsilon$.
- $C2$: The distance between follower i and the tail connector of team u is the shortest among all distances between other followers in team u and the tail connector of team u.
- $C3$: The distance between the lead agent of team u and the tail connector of team u is greater than the safety limit $r_{\text{safe}} + \varepsilon$.

These conditions must simultaneously be satisfied for a follower to become a connector.

When two connectors have a direct connection between each other, they are said to be *adjacent*. Any connector (including the initial connector) that does not have any adjacent connectors is prevented from switching to a follower.

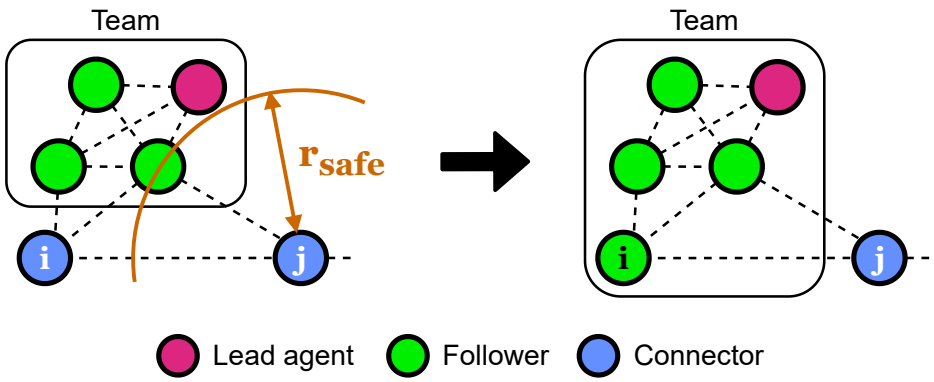


Figure 5.2. A situation prior (left) and after (right) a robot leaves the network, thereby shrinking it. Connector i is a tail connector of a neighbouring team (condition $F1$) and observes that the team is sufficiently close to an adjacent connector j (condition $F2$), rendering itself redundant. It leaves the network to join the nearby team.

Figure 5.2 illustrates a situation in which a tail connector switches to become a follower. Consider any connector i that has at least one adjacent connector j. The conditions for connector i to become a follower are as follows.

- $F1$: Connector i is a tail connector.
- $F2$: There is an adjacent connector j that has, for all teams u, a smaller hop count than connector i, or otherwise, a distance less than the safety limit $r_{\text{safe}} - \varepsilon$ between itself and the lead agent or follower of team u.

These conditions must simultaneously be satisfied for a connector to become a follower.

As connectivity must now be maintained between three or more teams, prior to switching to the follower role, connectors must ensure they are no longer needed as a connector for each team they are a tail connector for. To do so, connectors first list all the teams where condition $F1$ is true and then check whether condition $F2$ is true for all of these teams before it can switch to a follower.

Condition $F2$ has been modified to also check the distance of the lead agent. This has the benefit of allowing the tail connector to join the team more quickly since the lead agent is usually at the front of the team. However, the new follower is likely to switch back to a connector as it satisfies conditions $C1$ and $C2$. This potentially causes the worker to switch states rapidly between the two roles, which is not ideal. To

prevent this issue from occurring, condition $C3$ has been added to allow followers to become a connector only when the lead agent is far from the tail connector. One may expect that condition $C2$ could be modified to also consider the distance of the lead agent to the tail connector along with the other followers in the team instead of adding condition $C3$. However, the purpose of condition $C2$ is to allow the closest follower to become a connector; a lead agent cannot become a connector, which may cause the team to get disconnected from the rest of the network despite having followers in the team. Note that if the lead agent of team u is not within the communication range of follower i, we assume the lead agent is very far, and therefore condition $C3$ is evaluated as true.

Repositioning Branches

Modifying the network structure to minimise the total network length is important as it helps reduce the number of connectors maintaining the network, which in turn increases the number of followers in each team that are available to work on the tasks. In addition, it may also reduce the distance a robot must move along the network to travel between teams. As the teams move towards different task locations, the topology of the network should adapt to the movement of the teams.

A closely related problem is the *Euclidean Steiner tree problem*, which searches for the tree that interconnects N points in a plane with the minimum total length[1] (Prömel and Steger, 2002). To connect the points, additional points called *Steiner points* may be added when forming the Euclidean Steiner minimum tree (ESMT). Steiner points act as a hub for connecting three other points, essentially establishing a Y-junction where the edges are 120° apart. In our work, the network can be considered as a tree that interconnects all teams and the connectors can be considered as Steiner points that are part of the network. One difference from the ESMT is that we assume the teams are leaf nodes in the network; the path between any two teams must be a sequence of connectors. Although the Euclidean Steiner tree problem is known to be NP-hard, exact algorithms such as GeoSteiner[2] have been able to find optimal solutions for up to 100,000 terminals (Juhl et al., 2018). In the following, we describe constraints for connectors to ensure the network remains connected and to dynamically reconfigure the network to make it have a similar total length as the corresponding ESMT.

[1]The minimum spanning tree problem is a closely related problem, which is a special case of the Steiner problem

[2]`http://www.geosteiner.com/`

Figures 5.3 and 5.4 illustrate the situation in which a branch is being repositioned. The former considers connectors switching their connection with other neighbouring connectors. The latter considers a connector becoming a new tail connector for a neighbouring team. The conditions for connector i to replace its existing connection with an adjacent connector j to a non-adjacent connector or team k is as follows.

- $RB1_{\text{connector}}$: The distance between connector i and non-adjacent connector k is less than the distance between connector i and adjacent connector j.
- $RB1_{\text{team}}$: The distance between connector i and team k is less than the distance between adjacent connector j and team k.
- $RB2$: Adjacent connector j and a non-adjacent connector or team k are directly connected with each other.
- $RB3$: Adjacent connector j is connected to at least three other connectors or teams.

In order for a connector to modify its connections, the following conditions must be satisfied simultaneously; one of condition $RB1$ (i.e. $RB1_{\text{connector}}$ or $RB1_{\text{team}}$), condition $RB2$, and condition $RB3$.

Conditions $RB1_{\text{connector}}$ and $RB1_{\text{team}}$ check for new connections that can reduce the total network length. Depending on the type of the non-adjacent neighbour k, a different variant of condition $RB1$ is applied. If the non-adjacent neighbour is a connector, condition $RB1_{\text{connector}}$ is used; otherwise, condition $RB1_{\text{team}}$ is used.

Figure 5.3 illustrates the situation where the non-adjacent neighbour k is a connector. It applies $RB1_{\text{connector}}$ to check whether changing its current connection with an adjacent connector j to k would lead to a shorter edge length. When connector i detects a situation where all three conditions are satisfied, it notifies the other two connectors to apply the relevant modifications to their connections. The resulting network is shorter than the original, as the connection (i, j) is replaced with (i, k) only when the distance is shorter.

Figure 5.4 illustrates the situation where the non-adjacent neighbour k is a team. It applies $RB1_{\text{team}}$ to check whether it is closer to the team than an adjacent connector j, which is the current tail connector of team k. Unlike the previous case, connector i now checks whether or not the network length would become shorter if it became the new tail connector for team k. If all three conditions are satisfied, connector i notifies

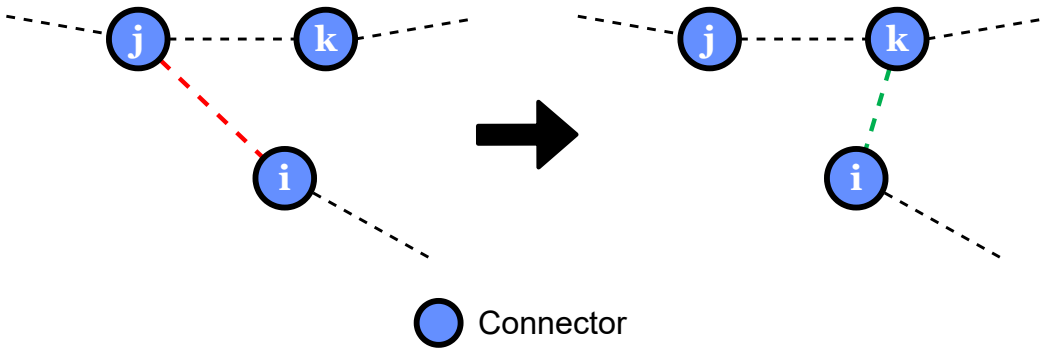


Figure 5.3. Repositioning the branch in the network. Situation prior (left) and after (right) the connections between robots are modified. Dashed lines represent connections between connectors. Connector i is closer to a non-adjacent connector k than its adjacent connector j (condition $RB1_{\text{connector}}$), j and k are directly connected with each other (condition $RB2$), and j has at least three connections (condition $RB3$). Connector i notifies j and k of the changes made to its connections. This reduces the total length of the network while preserving connectivity (network changes are indicated by the red and green dashed lines).

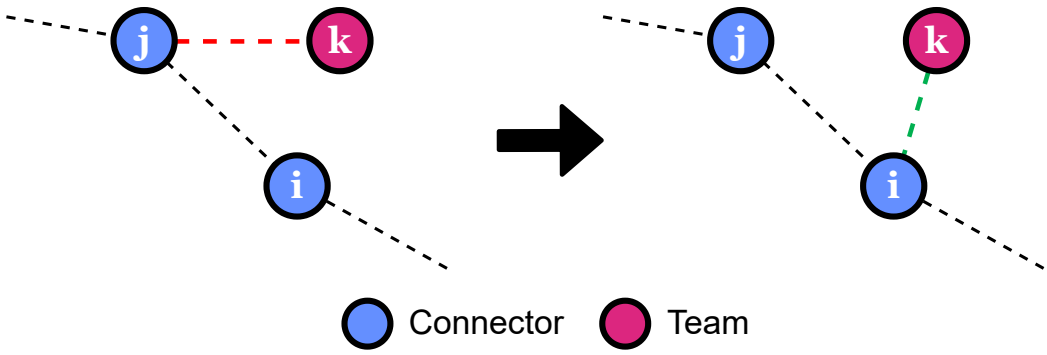


Figure 5.4. Switching the role as a tail connector. Situation prior (left) and after (right) the connections between robots are modified. Dashed lines represent connections between teams and connectors. Connector i is closer to a non-adjacent team k than its adjacent connector j (condition $RB1_{\text{team}}$), j and k are directly connected with each other (condition $RB2$), and j has at least three connections (condition $RB3$). Connector i notifies j to connect with itself, resulting in the removal of the connection between (j, k). This reduces the total length of the network while preserving connectivity (network changes are indicated by the red and green dashed lines).

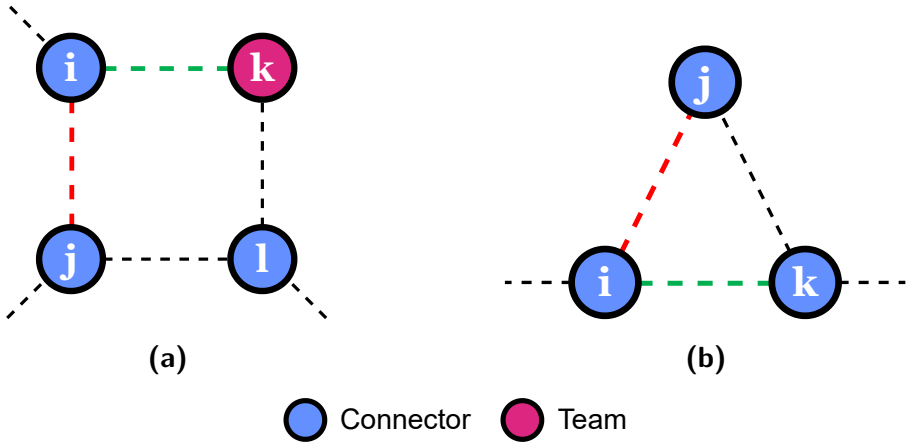
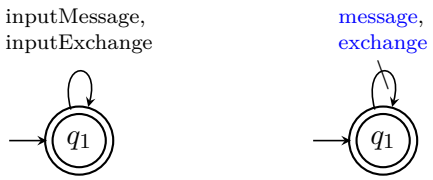


Figure 5.5. Example situations that conditions $RB2$ and $RB3$ prevent from occurring. The red and green dashed lines each indicate a potential connection removal or addition that would cause an issue in the network. **(a)** Condition $RB2$ ensures an adjacent connector j and team k are connected with each other to prevent connector i from becoming a second tail connector for team k. **(b)** Condition $RB3$ ensures an adjacent connector j has at least three connections to prevent it from becoming redundant.

adjacent connector j to stop being the tail connector for team k as it will replace j as the new tail connector. Similarly to before, the resulting network is shorter than the original.

Condition $RB2$ restricts the connection change to be considered only with those that are two hops away (i.e. the neighbour's neighbours). This is to ensure the network modification is applied incrementally. Figure 5.5a shows an example situation where condition $RB2$ is not enforced. If the highlighted connections changes were applied, team k would have two tail connectors (i.e. connectors i and l) instead of just having one. Furthermore, team k would no longer be a leaf node in the network, which contradicts our assumption.

Condition $RB3$ ensures that an adjacent connector j would have at least two connections with other connectors or teams after the connection with connector i is removed. This is to avoid connectors ending up with a single connection as it would not be useful since it is not maintaining the connectivity between any two teams. Figure 5.5b shows an example situation where a connector would become redundant if the highlighted connection was established.



(a) G_1^{nl} **(b)** G_2^{nl}

Figure 5.6. Free behaviour models for the lead agents representing their ability **(a)** to receive inputs from the operator in the form of messages intended to be sent to other operators and **(b)** to send a message that needs to be relayed to the other lead agents or inform a worker to join another team.

The connectors continuously move to align themselves with respect to their adjacent connectors (see Section 5.2.4 for more detail), which triggers these network modifications to optimise the structure of the network and allow the teams to travel further.

5.2.2 SCT Formulation

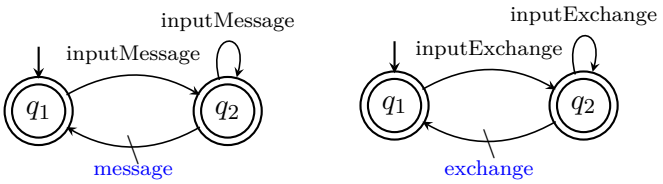
We present the SCT models designed for this study, which have been extended from the models described in Section 3.2.1. We describe the modifications made to the existing models, followed by new models that represent the conditions and behaviours of a worker to optimise the network. In this this study, we use $\theta \in \{nl, nw\}$ to refer to the extended models. nl refers to the models designed for the lead agnets, while nw refers to the models for the workers. G_i^θ denotes the ith free behaviour model and E_j^θ denotes the jth control specification.

Lead Agent Models

In this study, we assume tasks are executed as soon as the lead agent and its followers enter the task area. Therefore, events related to task execution (i.e. *pressStart*, *pressStop*, *start*, and *stop*) were removed from the lead agent models. Figures 5.6 and 5.7 show the new lead agent's free behaviour models and control specifications, respectively.

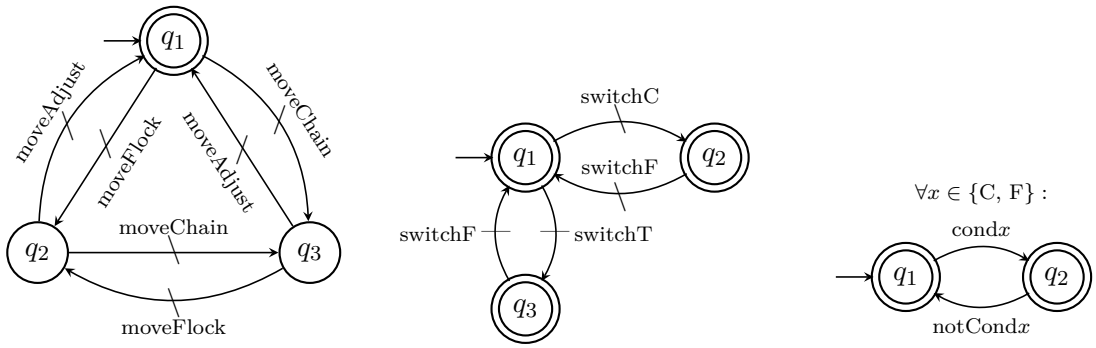
Worker Models

We present the free behaviour models and specifications for the workers. Figures 5.8 to 5.13 show the SCT models from Section 3.2.1 with changes made to certain models



(a) E_1^{nl} **(b)** E_2^{nl}

Figure 5.7. Control specification for the lead agents allowing them to transmit a signal upon receiving the corresponding operator input.



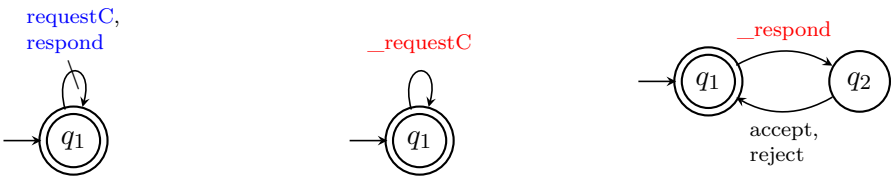
(a) G_1^{nw} **(b)** G_2^{nw} **(c)** $G_{3,x}^{nw}$

Figure 5.8. Free behaviour models regarding the worker's movement and ability to switch roles. **(a)** Motion capabilities; **(b)** role switching; **(c)** conditions for becoming a connector and a follower.

for this study. Figures 5.14 and 5.15 show the new SCT models that realise the network modification discussed in the previous section.

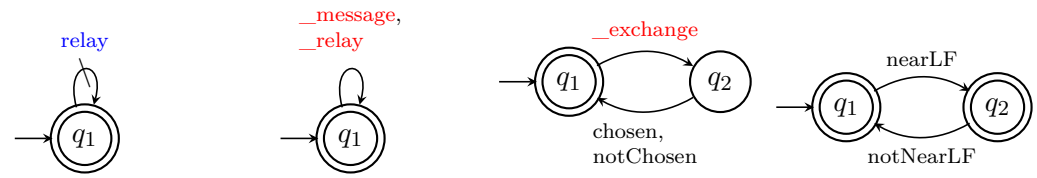
Figure 5.8 shows the free behaviour models related to the worker's movement and role-switching capabilities. G_1^{nw} and G_2^{nw} are the same as in Section 3.2.1. $G_{3,x}^{nw}$ has been modified to abstract individual conditions for becoming a connector or a follower, which are now evaluated at the operational procedures layer of the SCT architecture. This reduces the total number of states and transitions in the final supervisors while exhibiting the same behaviour.

Figure 5.9 shows the free behaviour models related to the worker's ability to extend the network. For G_4^{nw}, the ability to send a request for becoming the initial connector (event *requestL*) has been removed. Instead, for this study, we assume that there is always at least one connector between the teams by deploying the *initial connector*.



(a) G_4^{nw} **(b)** G_5^{nw} **(c)** G_6^{nw}

Figure 5.9. Free behaviour models regarding the worker's ability to extend the network. **(a)** Request to a neighbouring tail connector or reply to a request from a follower; **(b)** receive the request; **(c)** receive the response.



(a) G_7^{nw} **(b)** G_8^{nw} **(c)** G_9^{nw} **(d)** G_{10}^{nw}

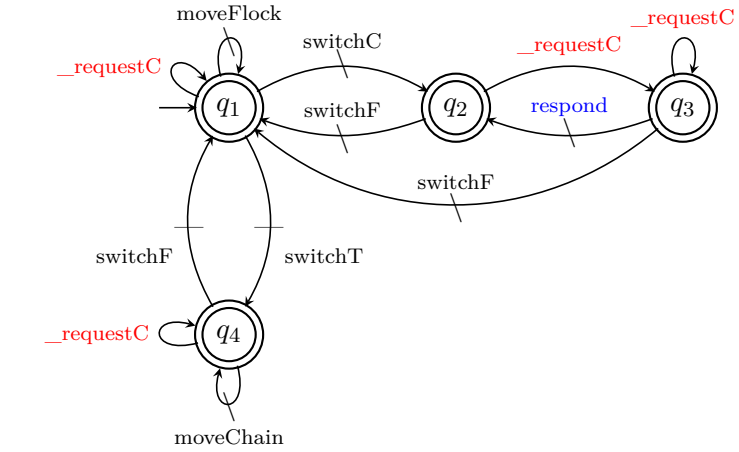
Figure 5.10. Free behaviour models regarding the worker's ability to relay messages and exchange workers between teams. **(a)** Send a message to be relayed; **(b)** receive a message to be relayed from the lead agent or a worker; **(c)** receive a message from the lead agent to join another team; **(d)** determine whether it is near the team that it needs to join.

For G_5^{nw}, the task-related events *_start* and *_stop* have been removed. G_6^{nw} remains the same as in Section 3.2.1.

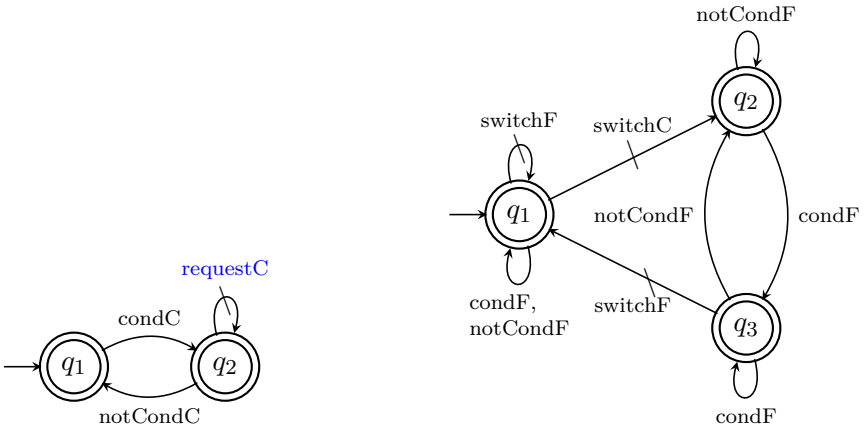
Figure 5.10 shows the free behaviour models related to the worker's ability to relay messages and exchange workers between teams. All the models G_7^{nw}, G_8^{nw}, G_9^{nw} and G_{10}^{nw} are identical to the models in Section 3.2.1.

Figure 5.11 shows the control specifications regarding the worker's ability to switch its role. Specification E_1^{nw} limits certain role-dependent controllable events to be enabled only in specific roles. Task-related event *taskStart* has been removed from state q_1. Specifications E_2^{nw} and E_3^{nw} each simplify the events involved to switch roles (events *condC* and *notCondC*, events *condF* and *notCondF*).

Figure 5.12 shows the control specifications regarding the procedures to follow when switching roles. Specification E_4^{nw} defines the steps for a follower to become a connector. Changes have been made in E_6^w from Figure 3.6, where event *requestL* and

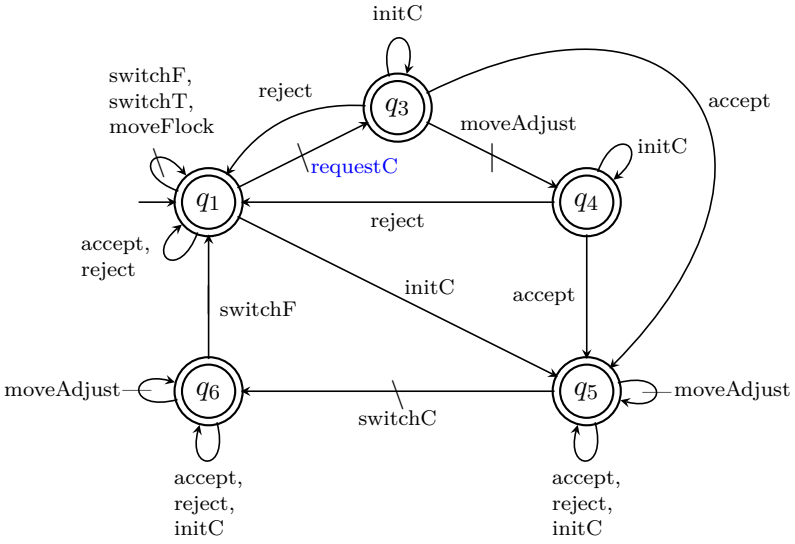


(a) E_1^{nw}

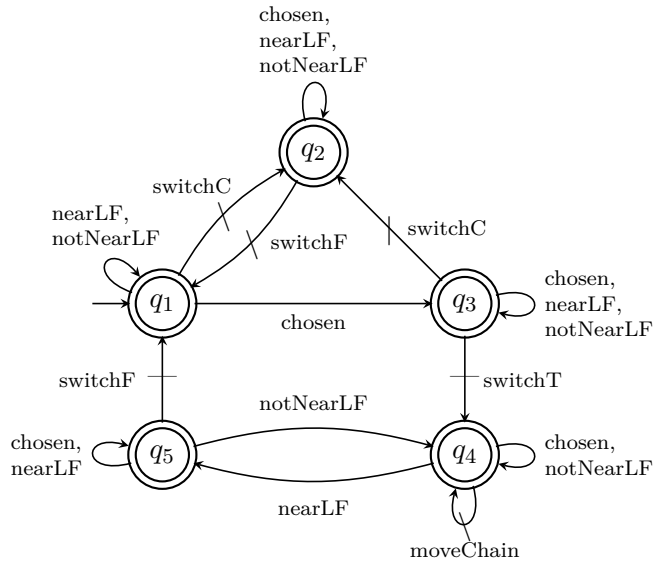


(b) E_2^{nw} **(c)** E_3^{nw}

Figure 5.11. Specifications regarding **(a)** the worker's allowed actions in each role and **(b–c)** conditions to switch roles.

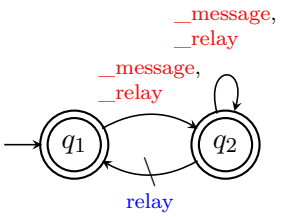


(a) E_4^{nw}



(b) E_5^{nw}

Figure 5.12. Specifications regarding the steps for a follower to **(a)** switch to a connector and **(b)** switch to a traveller.



(a) E_6^{nw}

Figure 5.13. Specification to relay messages received.

state q_2 in the original model have been removed[3]. Furthermore, a new event *initC* is added to state q_1 to initialise a single robot as a connector (i.e. the *initial connector*) at the beginning of a mission.

Figure 5.13 shows the specification E_6^{nw} for a worker to relay messages it has received. It is identical to the specification defined in Section 3.2.2.

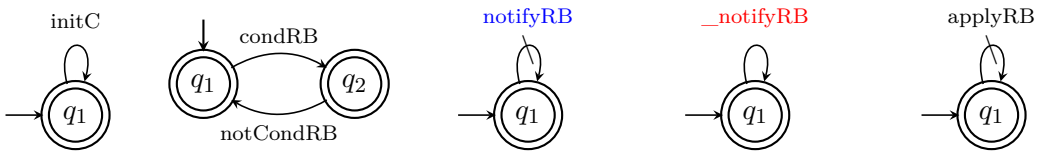
Figure 5.14 shows the free behaviour models for a connector to optimise the network. Free behaviour model G_{11}^{nw} represents an event that triggers a worker to initialise as a connector. One worker from a team is manually assigned to initialise as a connector. We assume this *initial connector* is positioned in a way so that it can detect at least one follower from each team at the beginning of the mission.

Free behaviour model G_{12}^{nw} represents the conditions to apply network optimisation with other neighbouring connectors. Uncontrollable event *condRB* is triggered when all three conditions $RB1$ (i.e. $RB1_{\text{connector}}$ or $RB1_{\text{team}}$), $RB2$, and $RB3$ are satisfied by a connector. Event *condRB* is only evaluated when it is a connector; if a worker is in any other role, event *notCondRB* will be triggered.

Free behaviour model G_{13}^{nw} is used to broadcast a signal to inform relevant connectors in its communication range to modify their connections. In free behaviour model G_{14}^{nw}, this signal is received by the relevant connectors. Finally, free behaviour model G_{15}^{nw} defines the action to apply the network modification. The event definitions can be found in Table 5.1.

Figure 5.15 shows the control specification regarding the network modification behaviour. Specification E_7^{nw} defines that when a connector satisfies the conditions to modify the connection with neighbouring connectors, it can broadcast a signal to other

[3]State q_2 was removed to minimise the worker's total number of states. E_4^{nw} achieves the same behaviour as E_6^{cw} with fewer states and transitions as it is not essential for event *moveFlock* to occur before making a request to connect (event *requestC*).



(a) G_{11}^{nw} **(b)** G_{12}^{nw} **(c)** G_{13}^{nw} **(d)** G_{14}^{nw} **(e)** G_{15}^{nw}

Figure 5.14. Additional free behaviour models for the workers. **(a)** Input to initialise a worker as a connector. **(b–e)** Models for the workers to modify their connections. They represent the ability **(b)** to determine whether the condition to switch its connections has been satisfied, **(c–d)** to transmit and receive messages to modify the network between adjacent connectors, and **(e)** to apply the network modification by updating its knowledge of active connections.

(a) E_7^{nw} **(b)** E_8^{nw}

Figure 5.15. Additional control specifications for the workers to modify their connections. The models allow them **(a)** to broadcast a message to neighbouring connectors when the condition to modify the network is satisfied, and **(b)** to update its own knowledge of the network when either it has sent or received the broadcast message to modify the network.

connectors. Specification E_8^{nw} ensures that when a connector sends or receives a signal to modify its connections, it will eventually apply those modifications.

The SCT models are synthesised using the local modular approach with public events.

5.2.3 Follower Behaviour

The robot uses virtual forces to determine its direction of movement. We use the flocking behaviour described in Section 3.2.4. $\mathbf{p}_{ij}$ denotes robot j's position in the local coordinate system of robot i. The virtual force of robot i in the follower role is given by

$$\mathbf{u}_i = \alpha_f \mathbf{u}_i^a + \beta_f \mathbf{u}_i^{rn} + \gamma_f \mathbf{u}_i^{ro} \tag{5.1}$$

where α_f, β_f and γ_f are positive scalars to weigh the influence of the force components.

Table 5.1. Summary of events' description used in this study. Private controllable, private uncontrollable, public controllable and public uncontrollable events are labelled $\mathcal{C}$, $\mathcal{U}$, $Pub\mathcal{C}$ and $Pub\mathcal{U}$, respectively.

Event	Type	Used In	Description
moveFlock	$\mathcal{C}$	nw	Robot flocks with the lead agent.
moveChain	$\mathcal{C}$	nw	Robot moves along the chain to the other team.
moveAdjust	$\mathcal{C}$	nw	Robot adjusts its position in the chain.
switchC, switchF, switchT	$\mathcal{C}$	nw	Robot switches to the connector, follower, or traveller role.
inputMessage, inputExchange	$\mathcal{U}$	nl	Robot detects an operator input.
requestC	$Pub\mathcal{C}$	nw	Robot sends a request message to a connector.
_requestC	$Pub\mathcal{U}$	nw	Robot received a request from a worker.
respond	$Pub\mathcal{C}$	nw	Robot sends a reply to the request received.
_respond	$Pub\mathcal{U}$	nw	Robot received a response to the request it made to switch to a connector.
accept, reject	$\mathcal{U}$	nw	Process the response to determine whether its request to switch to a connector was accepted or rejected.
cond$x$, notCond$x$	$\mathcal{U}$	nw	Robot determines whether condition $x \in \{C, F\}$ was satisfied or not.
message	$Pub\mathcal{C}$	nl	Robot sends a message to the other lead agent.
relay	$Pub\mathcal{C}$	nw	Robot relays a message to the target lead agent.
_message, _relay	$Pub\mathcal{U}$	nw	Robot received a message that needs to be relayed.
exchange	$Pub\mathcal{C}$	nl	Robot sends a message that specifies a worker to join the other team.
_exchange	$Pub\mathcal{U}$	nw	Robot received a message from the lead agent related to team switching.
chosen, notChosen	$\mathcal{U}$	nw	Process the message to determine whether it has been chosen to switch to the other team.
nearLF, notNearLF	$\mathcal{U}$	nw	Robot determines whether a team member was detected or not.
initC	$\mathcal{U}$	nw	Input to initialise as a connector.
condRB, notCondRB	$\mathcal{U}$	nw	Determine whether the condition to reposition the branch was satisfied or not.
notifyRB	$Pub\mathcal{C}$	nw	Broadcast to neighbouring robots that it will reposition the branch.
_notifyRB	$Pub\mathcal{U}$	nw	Broadcast received to reposition the branch.
applyRB	$\mathcal{C}$	nw	Apply repositioning of the branch.

Force component $\mathbf{u}_i^a$ represents the attraction towards robot i's leader. It is defined as

$$\mathbf{u}_i^a = \frac{1}{|\mathcal{N}_i^a|} \sum_{j \in \mathcal{N}_i^a} \mathbf{p}_{ij} \tag{5.2}$$

where $\mathcal{N}_i^a$ denotes the set of neighbours of robot i that are either the lead agent or members of the same team and are closer to the lead agent than robot i (based on hop count).

Force component $\mathbf{u}_i^{rn}$ causes robot i to get repelled from all neighbouring robots. It is defined as

$$\mathbf{u}_i^{rn} = -\frac{1}{|\mathcal{N}_i|} \sum_{j \in N_i} \frac{\tau \sigma^4}{||\mathbf{p}_{ij}||^5} \frac{\mathbf{p}_{ij}}{||\mathbf{p}_{ij}||} \tag{5.3}$$

where $\mathcal{N}_i$ is the set of neighbours of robot i, σ is the target distance to maintain between the neighbours, and τ is the gain.

In addition to $\mathbf{u}_i^{rn}$, the force component $\mathbf{u}_i^{ro}$ uses proximity sensors to prevent collisions if the distance to nearby obstacles (i.e. other robots) gets too close. It is defined as

$$\mathbf{u}_i^{ro} = -\frac{1}{8 d_{\max}} \sum_{j=1}^{8} (d_{\max} - d_j) \hat{\mathbf{v}}_j \tag{5.4}$$

where $d_{\max} = 10\,\text{cm}$ is the assumed range of the proximity sensors, d_j is the distance extracted from the j^{th} sensor and $\hat{\mathbf{v}}_j$ is the unit vector pointing from the robot's centre to the j^{th} sensor. Where sensor j detects no object, we set $d_j = d_{\max}$.

5.2.4 Connector Behaviour

In this section, we describe the connector motion that causes a connector to move towards the centroid of all its connected neighbours, while ensuring the connectivity with all its neighbours is maintained. Figure 5.16 illustrates the desired connector behaviour to move towards the centroid of its neighbours when there are two or three neighbours, although it is possible to have more. This behaviour allows the connectors to minimise the length of the network. The virtual force of robot i in the connector role is given by

$$\mathbf{u}_i = \alpha_{c1} \mathbf{u}_i^{an} + \alpha_{c2} \mathbf{u}_i^{at} + \beta_c \mathbf{u}_i^{rn} + \gamma_c \mathbf{u}_i^{ro} \tag{5.5}$$

where α_{c1}, α_{c2}, β_c and γ_c are positive scalars to weigh the influence of the force components.

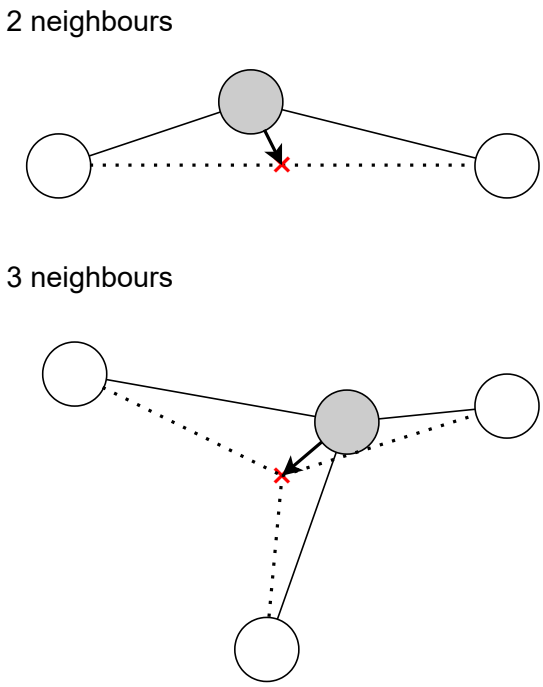


Figure 5.16. Illustration of a connector (grey) with two or three connectors moving to the centroid (red cross) of its neighbours (white).

Force component $\mathbf{u}_i^{an}$ represents the attraction towards neighbouring robots in the network. This ensures the connectors and teams remain connected at all times. It is defined as

$$\begin{aligned} \mathbf{u}_i^{an} &= \frac{1}{|\mathcal{N}_i^{an}|} \sum_{j \in \mathcal{N}_i^{an}} \mathbf{p}_{ij} \phi^{an} \\ \phi^{an} &= \begin{cases} \dfrac{r_{\text{com}} - r_{\text{safe}}}{r_{\text{com}} - ||\mathbf{p}_{ij}||}, & \text{if } ||\mathbf{p}_{ij}|| > r_{\text{safe}} \\ 1, & \text{otherwise} \end{cases} \end{aligned} \tag{5.6}$$

where $\mathcal{N}_i^{an}$ denotes the set of neighbours of robot i that are directly connected to it (i.e. connector or the closest member from a team). ϕ^{an} regulates the influence of the attraction to the neighbour, which increases as the distance $||\mathbf{p}_{ij}||$ grows larger than r_{safe} (note that $||\mathbf{p}_{ij}||$ is bounded by r_{com}).

For tail connectors that are directly connected to a team, we apply an additional force component $\mathbf{u}_i^{at}$ to make the robot move closer to the team. This can be thought of as each team pulling the network towards their team. This keeps the connectors in the network separated from each other, which helps minimise the number of followers

required to extend the network. The force component $\mathbf{u}_i^{at}$ is defined as

$$\begin{aligned} \mathbf{u}_i^{at} &= \frac{1}{|\mathcal{N}_i^{at}|} \sum_{j \in N_i^{at}} \mathbf{p}_{ij} \phi^{at} \\ \phi^{at} &= \max\left(0, \frac{r_{\text{safe}} + \Delta - d_i^{\max}}{r_{\text{safe}} + \Delta - r_{\text{safe}}}\right) \end{aligned} \tag{5.7}$$

where $\mathcal{N}_i^{at} \subset \mathcal{N}_i^{an}$ denotes the set of neighbours of tail connector i that are the closest members from the team. $d_i^{\max}$ denotes the distance of the furthest neighbour of robot i. The attraction tends to zero as $d_i^{\max}$ gets closer to r_{safe} at which there will be no attraction. To avoid the robot from being attracted beyond the communication range with its other neighbours, this force is reduced accordingly to the longest distance to a neighbour against a threshold value. The attraction to the team is only applied if there are no neighbouring connections that are critically close to being disconnected. To avoid the robot from being attracted beyond the communication range with its other neighbours, this force is reduced accordingly to the longest distance to a neighbour against a threshold value.

The connectors use the same repulsion force $\mathbf{u}_i^{rn}$ from equation 5.3. However, instead of getting repelled from all neighbours, it is repelled only by $\mathcal{N}_i^{an}$. This means connectors have a higher priority than other workers when moving as they are essential in maintaining the network. Connectors also use the force component $\mathbf{u}_i^{ro}$ to avoid obstacles that get too close to itself.

5.2.5 Traveller Behaviour

The virtual force of robot i in the traveller role is given by

$$\mathbf{u}_i = \alpha_t \mathbf{u}_i^t + \beta_t \mathbf{u}_i^{rn} + \gamma_t \mathbf{u}_i^{ro} \tag{5.8}$$

where α_t, β_t and γ_t are positive scalars to weigh the influence of the force components.

Force component $\mathbf{u}_i^t$ represents the attraction for the traveller to move along the network to reach the team that it needs to join. If a member of the team is within the communication range of robot i, then it will move directly towards the team. Otherwise, it will move towards the connector that has the smallest hop count for the team to join as it means the team can be found in that direction in the network. It is

defined as

$$\mathbf{u}_i^t = \begin{cases} \frac{\sum_{j \in \mathcal{N}_i^t} \mathbf{p}_{ij}}{|\mathcal{N}_i^t|}, & \text{if } |\mathcal{N}_i^t| > 0 \\ \mathbf{p}_{ix} + \frac{\mathbf{p}_{ix}}{||\mathbf{p}_{ix}||} \frac{\pi}{2} d_{\text{sep}}, & \text{otherwise} \end{cases} \tag{5.9}$$

where $\mathcal{N}_i^t$ denotes the set of followers in the communication range of robot i that belong to the team to join, $\mathbf{p}_{ix}$ denotes the position of the connector x which has the smallest hop count to the team to join and d_{sep} denotes a fixed separation from the connector. This makes the travellers move along the left side of the connectors instead of moving directly towards them.

5.2.6 Requesting Workers

Sending a request for more workers to multiple teams can potentially cause communication overhead and interrupt the individual tasks that each operator is carrying out. Furthermore, if an operator simply broadcasts a request to all other operators, those receiving the request may each send the requested number of workers, which could result in exceeding the number of workers that were actually needed. Sending excess workers temporarily reduces the number of workers available to perform tasks and wastes energy by travelling between teams.

We present a requesting strategy that operators can employ to send the exact number of workers to the operator who is making the request. Rather than broadcasting a single request to all operators, it sends targeted requests to each operator in sequential order. Figure 5.17 describes the steps an operator would follow when requesting workers. First, all other operators are sorted by their hop count to its team in ascending order. Next, the operator requests for the remaining number of workers it needs to the first operator in the list[4]. Once the other operator decides the number of workers they would like to send (possibly zero), the requesting operator receives an acknowledgement message indicating how many workers will be sent to its team. If the number of workers joining the team satisfies the requirement, no further requests are made. If the request was only partially satisfied, the operator sends another request to the next operator in the list for the remaining number of workers. This process is repeated until either the required number of workers has been satisfied or it has sent a request to every operator. If there are still not enough workers after sending the requests, the operator

[4]The request message has been extended to include a field that specifies which operator the request is targeted for.

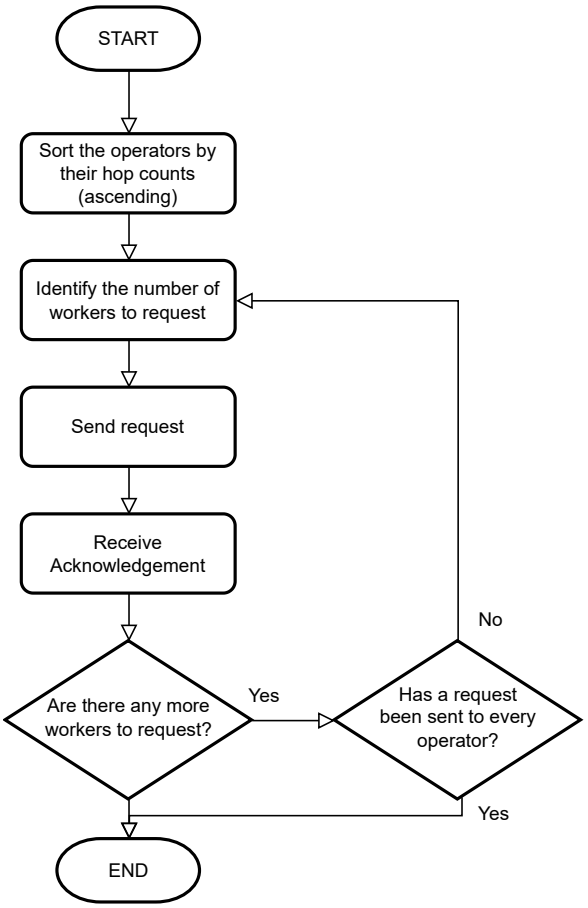


Figure 5.17. Flowchart showing the steps that an operator would follow when requesting workers.

may choose to go through the process again as some teams may have completed a task since they previously received the request.

5.3 Simulation Studies

This section presents the simulation studies conducted to evaluate the performance of the proposed extension.

5.3.1 Simulation Setup

We conduct simulation trials using the ARGoS simulator (Pinciroli et al., 2012). We use the e-puck (Mondada et al., 2009) for our study, which is a mobile differential-wheeled

Table 5.2. Design parameters.

Role	Symbol	Value
Follower	α_f	1.5
	β_f	2
	γ_f	15
Connector	α_{c1}	1
	α_{c2}	1
	β_c	1
	γ_c	15
Traveller	α_t	1
	β_t	0.3
	γ_t	5
	d_{sep}	0.2 (m)
Common	σ	0.15 (m)
	τ	1000

robot with a diameter of 7 cm. It has eight proximity sensors with a range of 0.1 m distributed around its body, We assume a range-and-bearing system with a range $r_{\text{com}} = 0.8\,\text{m}$ to communicate with neighbouring robots and a safety limit of $r_{\text{safe}} = 0.5\,\text{m}$. Table 5.2 reports the design parameters used for the simulation studies. We consider simulated human operators, who each control a lead agent and move it towards an assigned task in the environment. The simulation physics and the robot control cycle are updated every 0.1 s.

5.3.2 Performance Measures

For this study, we examine the following three performance measures:

- *Network length*: The sum of the edge lengths between all connectors and the team centroids.
- *Number of connectors*: The number of connectors used to maintain connectivity between the teams. Having a smaller number of connectors is preferred as this means more workers are available to perform tasks as a follower for the lead agent.

- *Average path length*: The length of the average path between any pair of teams in the network. As travellers move between teams by following the connectors in the network, a shorter path between teams reduces the time a worker needs to travel to join another team.

The performance measures are related to each other to a certain extent, as a longer network length may likely result in more connectors needed to maintain global connectivity and a longer path between any two teams.

To analyse the performance of the network formed by the workers, we compare it against a Euclidean Steiner tree found using the GeoSteiner algorithm (Juhl et al., 2018). We provide the set of team centroids as input to the algorithm, which returns an optimal tree with the newly added points as output. To make the comparison fair between the optimal solution and the network formed in our trials, we make the following modifications to the solution found by the GeoSteiner algorithm. First, if the solution contains edges that are longer than the communication range r_{com}, we "steinerise" the edge into multiple equal-length segments, each shorter than r_{com} and introduce additional points for connecting them. Second, if the solution contains a point representing a team centroid that is not a terminal, we add a new point incident at the team centroid, replace the edges of the team centroid with this new point, and make an edge between the team centroid and the new point. The resulting tree has the same network length, but an extra point added to the original solution. The above modifications allow us to compare the number of connectors that were necessary to maintain the network in the trials against the number of points needed to construct the optimal tree (i.e. ESMT) found by the GeoSteiner algorithm.

5.3.3 Behavioural Analysis

The simulation is conducted in an unbounded circular arena. To conduct simulations with various numbers of teams and workers per team, the arena radius R_{arena} is defined relative to n_t, which is the initial number of workers per team.

$$R_{\text{arena}} = \frac{n_t}{2} r_{\text{com}} \tag{5.10}$$

It is defined so that approximately half of the workers are required for a team located at the centre to reach the edge of the arena.

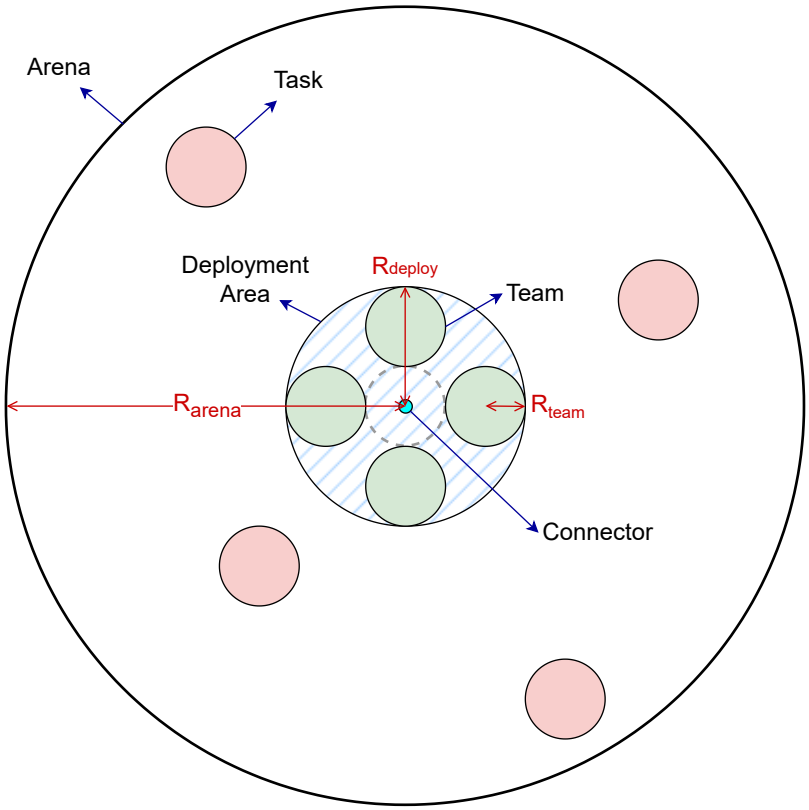


Figure 5.18. Illustration of the simulation environment. The arena is represented by the outer circle. Each team (green circle) is initialised inside the deployment area (circle with a blue hatch). Tasks (red circle) are scattered within the arena. A single connector (cyan circle) is initially placed at the centre of the deployment area.

Initially, the lead agents and their followers are deployed within the deployment area. Within the deployment area, the team (i.e. a lead agent and its followers) are placed close together. The radius of the deployment area R_{deploy} is defined as

$$\begin{aligned} R_{\text{deploy}} &= 3R_{\text{team}} \\ R_{\text{team}} &= \frac{A(n_t+1)}{\rho\pi} \end{aligned} \tag{5.11}$$

where R_{team} is the radius of a circle defining the area a single team is deployed, A is the footprint of an e-puck, and ρ is the density at which the lead agent and workers are distributed within the area defined by R_{team}. We use $\rho = 0.2$ in all our simulations. Figure 5.18 summarises the simulation setup.

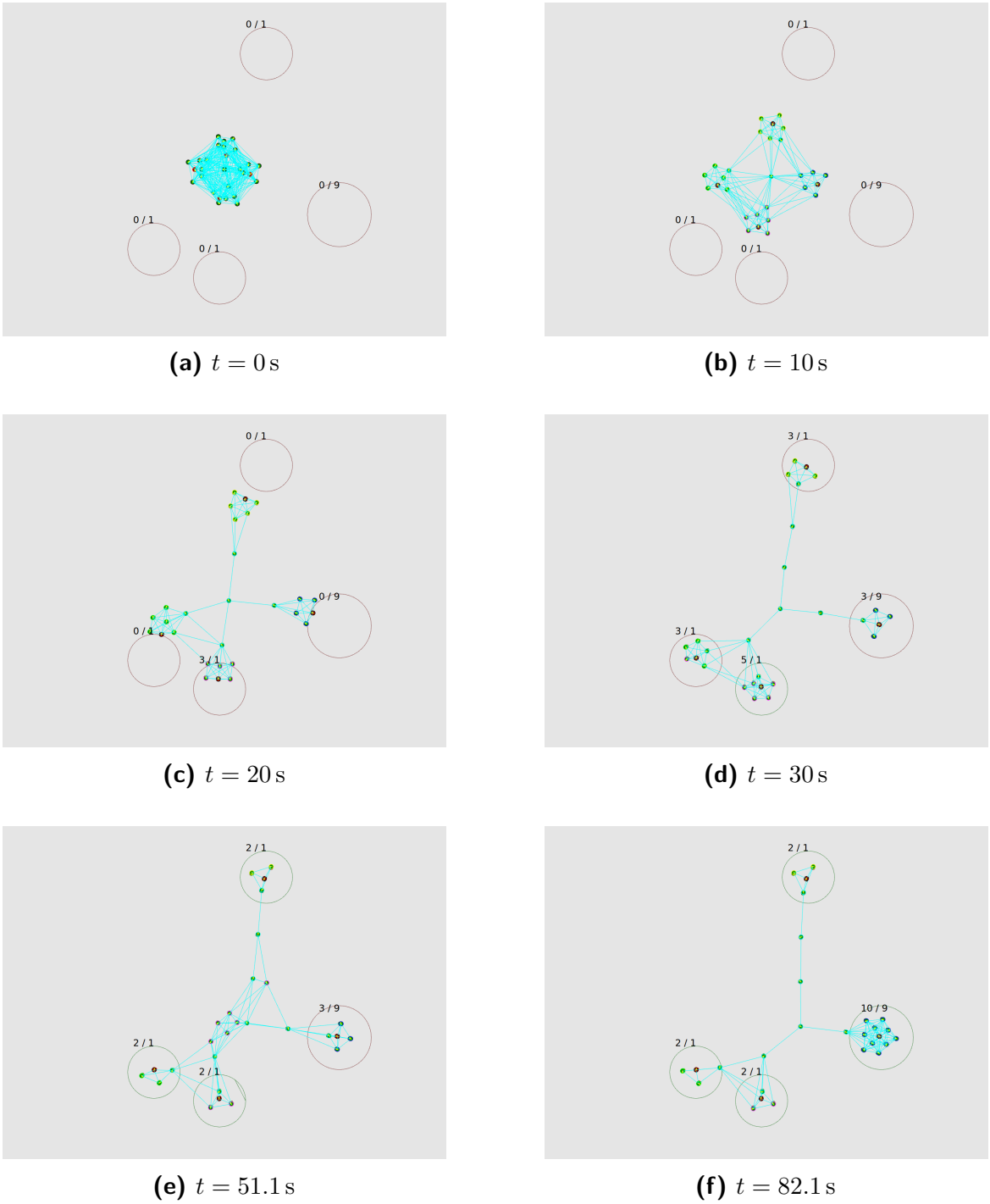
We consider 4 lead agents and 24 workers. Each lead agent is initially assigned 6 workers. Four tasks are uniformly randomly distributed within the arena without

overlapping with another task or the deployment area. Three of the tasks require $n^{\min} = 1$, while one task requires $n^{\min} = 9$. Each lead agent is assigned a unique task to complete; the first lead agent is assigned to its closest task and the remaining lead agents and tasks are paired in sequence in the order they appear in the anti-clockwise direction. The presence of a large task means one of the lead agents will not be able to complete the task with its initially assigned workers and must request more from other teams.

A total of 100 trials were performed. Figure 5.19 shows a sequence of snapshots from one of the simulation trials. As the lead agents move towards their assigned tasks, the workers can be seen to maintain global connectivity by leaving the team to become connectors. For tasks that require one robot to complete, the team simply enters the task and waits until the task is completed. In Figure 5.19d, the team on the right require 9 workers to complete. As it does not have enough workers in its team to complete the task, the lead agent sends a request message to other lead agents, one at a time as discussed in Section 5.2.6, for the remaining number of workers needed. For this study, the simulated operators were programmed to have at least 2 workers in their team and send all remaining workers when requested by another operator. This can be seen in Figure 5.19e, where the lead agents in the bottom, left and top teams each send three, two and one workers, respectively. Upon receiving workers from the other teams, the team on the right successfully completes its assigned task (Figure 5.19f).

Figure 5.20 shows the final timestep of the simulation from Figure 5.19f and the optimal tree found by the GeoSteiner algorithm when the team centroids were provided as input. We can see that the simulation trial shows eight connectors, while the solution found by the algorithm shows six connectors. The total network length was 5.83 m for the simulation trial and 5.56 m for the optimal tree.

Figure 5.21 shows a comparison of the simulation trials and the optimal tree. The diagonal dashed line indicates identical results between the simulation trial and the optimal tree. If a point is above the dashed line, it means the trial result had more connectors than the optimal solution and vice versa. It can be seen from Figure 5.21a that the trial results had a slightly longer network length than the optimum achievable for a given set of team centroids with relatively little variation. The average network length was 5.34 m for the simulation trials and 5.06 m for the optimal tree. Figure 5.21b shows that the trial results often had the same or required up to three more connectors to form the network when compared to the optimal tree. The average number of connectors was 7.12 for the simulation trials and 6.24 for the optimal tree. The deviation



(a) $t = 0\,\mathrm{s}$ **(b)** $t = 10\,\mathrm{s}$

(c) $t = 20\,\mathrm{s}$ **(d)** $t = 30\,\mathrm{s}$

(e) $t = 51.1\,\mathrm{s}$ **(f)** $t = 82.1\,\mathrm{s}$

Figure 5.19. Sequence of snapshots showing how four teams complete four tasks. Cyan lines indicate two robots that are within each other's communication range. Tasks are represented as red or green circles, indicating an incomplete or complete task, respectively. The numbers above each task show the current number of workers inside the task area (left) and the minimum number of workers needed (right), respectively.

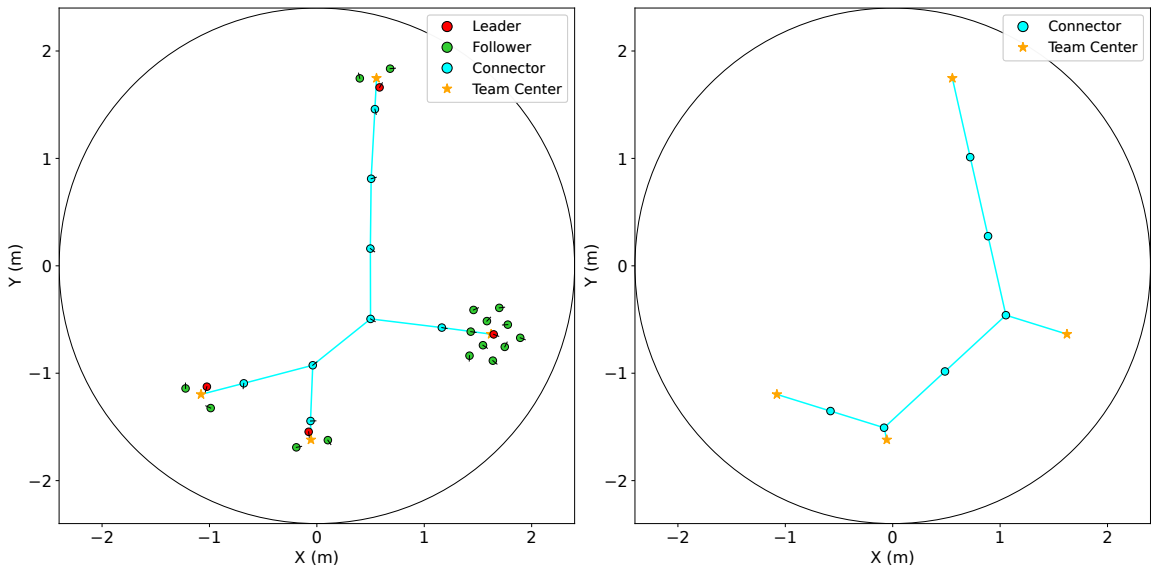


Figure 5.20. Robots at the final timestep of a simulation trial (left) and the solution found by the GeoSteiner algorithm (right). Red, green, and cyan circles represent the lead agents, followers, and connectors, respectively. Cyan lines represent the connection maintained by the connectors. Yellow stars indicate the centroid of each team.

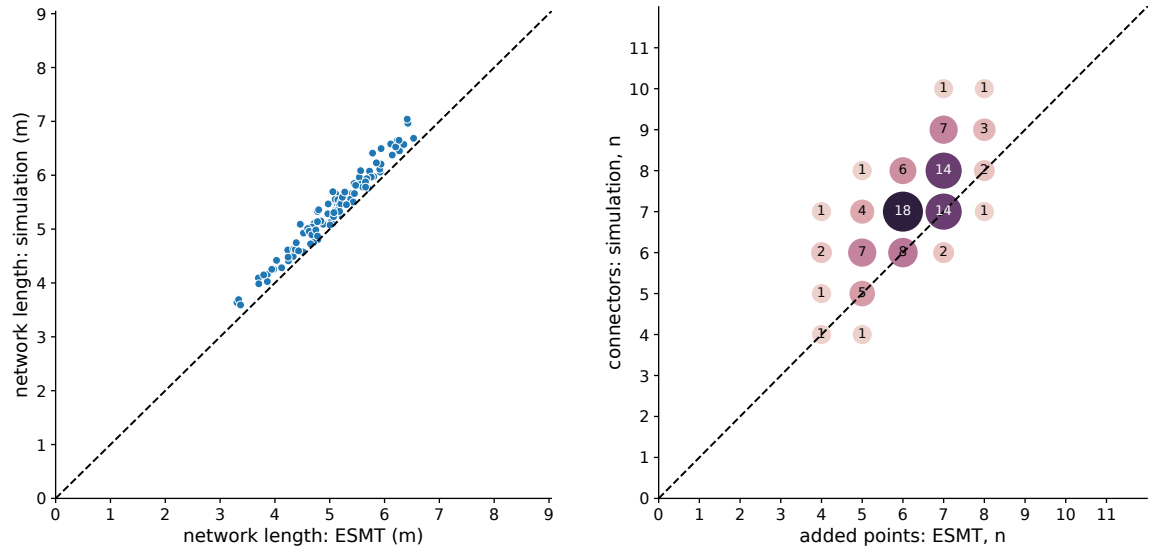


Figure 5.21. Comparison of **(a)** the network length and **(b)** the number of connectors between the simulation trials and the optimal solutions found by the GeoSteiner algorithm (ESMT).

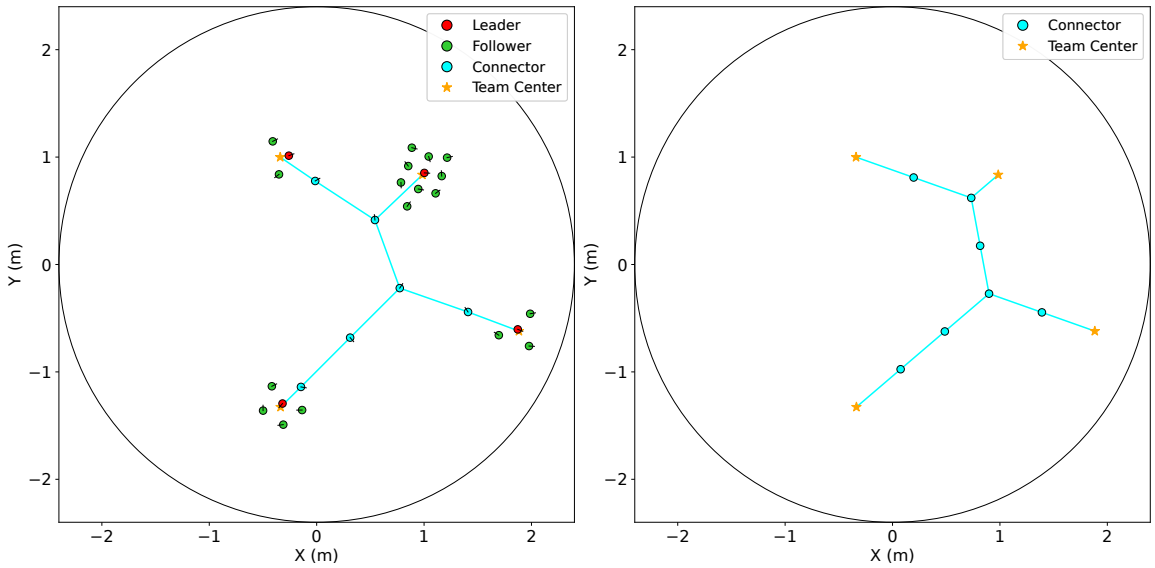


Figure 5.22. Final timestep of a simulation trial (left) and the solution found by the GeoSteiner algorithm (right), where the simulation trial has fewer connectors. Red, green, and cyan circles represent the lead agents, followers, and connectors, respectively. Cyan lines represent the connection maintained by the connectors. Yellow stars indicate the centroid of each team.

of the number of connectors from the optimal tree was worse than the network length. One reason for this could be due to r_{safe} being smaller than r_{com}; the workers extend the chain before the separation reaches the maximum communication range, which may have caused the network to consist of more connectors than the optimal tree. In four trials, the network consisted of fewer connectors than the optimal tree. This is possible as the GeoSteiner algorithm minimises the network length instead of the number of points it adds and our modification to "stenerise" the edges into equal lengths may also affect the output. This can be seen in Figure 5.22, where the trial result had six connectors, while the optimal tree added seven points.

Figure 5.23 shows the minimum, maximum and average path lengths between the teams. Although the average path lengths were similar between the trials and the optimal trees, they showed a different distribution. We found that the maximum path lengths in the trials were significantly smaller and more concentrated than the optimal trees, which had a concentrated minimum path length but a much more scattered maximum path length. A consistently shorter and relatively small variation in the average path length between any two teams can be advantageous as it means the time for travellers to move between them is more consistent and hence predictable to

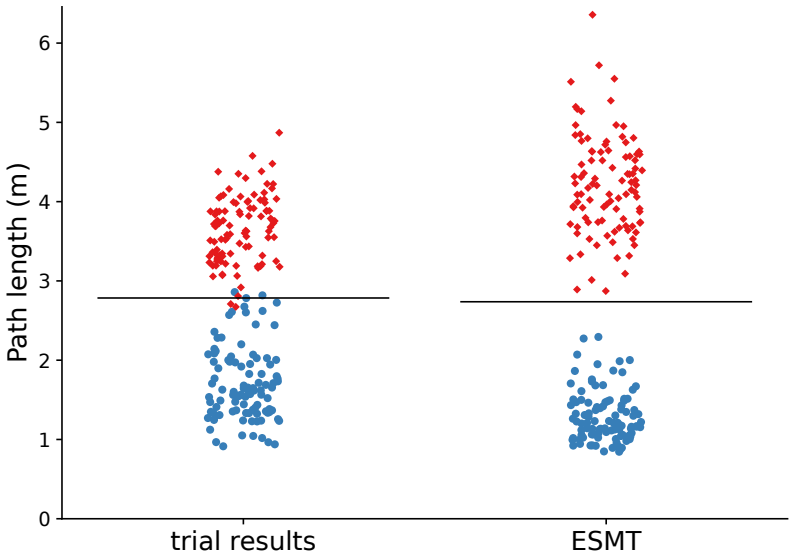


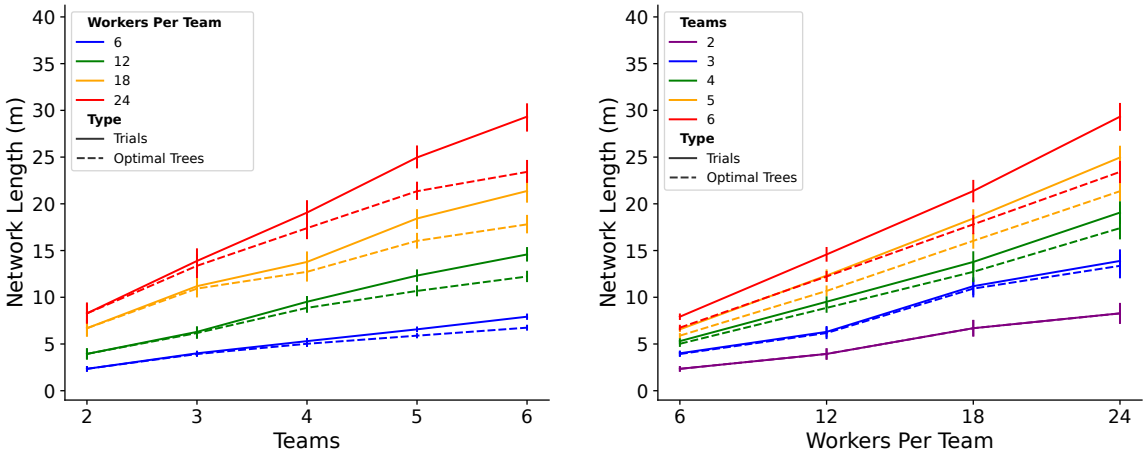
Figure 5.23. Minimum and maximum path lengths between a pair of teams for the simulation trials and the optimal solutions found by the GeoSteiner algorithm (ESMT). Red and blue points represent the maximum and minimum path lengths of a given trial, respectively. The black line shows the mean of all path lengths in every trial.

the operators. The shorter maximum path length observed when using our method indicates that travellers are likely to travel a shorter distance in the worst case.

5.3.4 Scalability Analysis

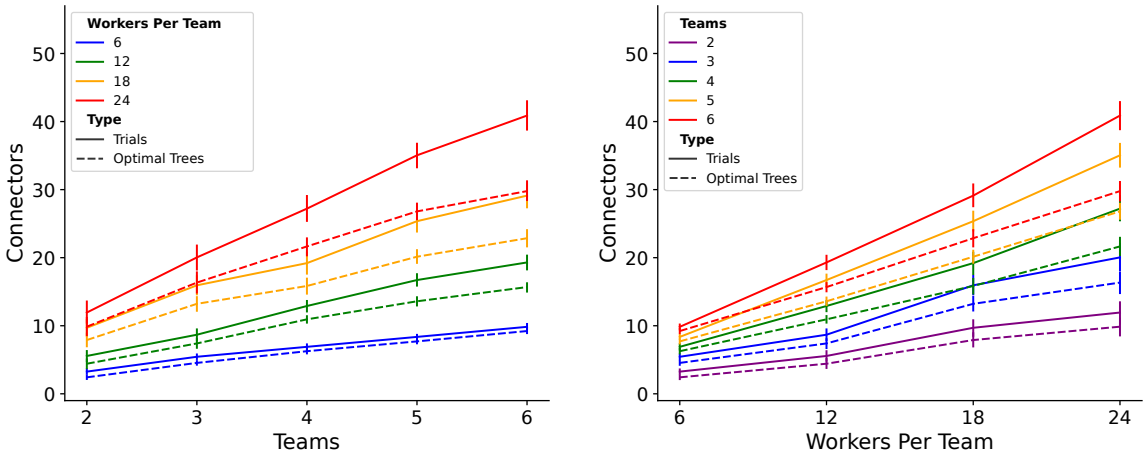
We investigate the performance of the swarms of $n \in \{2, 3, 4, 5, 6\}$ teams and $m \in \{6, 12, 18, 24\}$ workers per team in an environment of variable size with respect to m. In total, there were 20 configurations. We ran 50 trials for each configuration, totalling 1000 runs. n tasks were uniformly randomly distributed in the arena. Similarly to the previous study, each leader was assigned a unique task. The simulation was terminated when all tasks were completed or the timeout $T = 300\,\mathrm{s}$ was reached, whichever occurred first.

Figures 5.24 and 5.25 show the average network length and the number of connectors at the final timestep of the simulation trials and the number of points added in the optimal solutions, respectively. We see that increasing the number of teams and workers per team both cause an increase in the total network length and the number of connectors. However, the average path length between the teams remains relatively



(a) **(b)**

Figure 5.24. Comparison of the total network length when modifying **(a)** the number of teams and **(b)** the number of workers per team. The solid and dashed lines represent the number of connectors from the simulation trials and the solutions found by the GeoSteiner algorithm, respectively.



(a) **(b)**

Figure 5.25. Comparison of the number of connectors when modifying **(a)** the number of teams and **(b)** the number of workers per team. The solid and dashed lines represent the number of connectors from the simulation trials and the solutions found by the GeoSteiner algorithm, respectively.

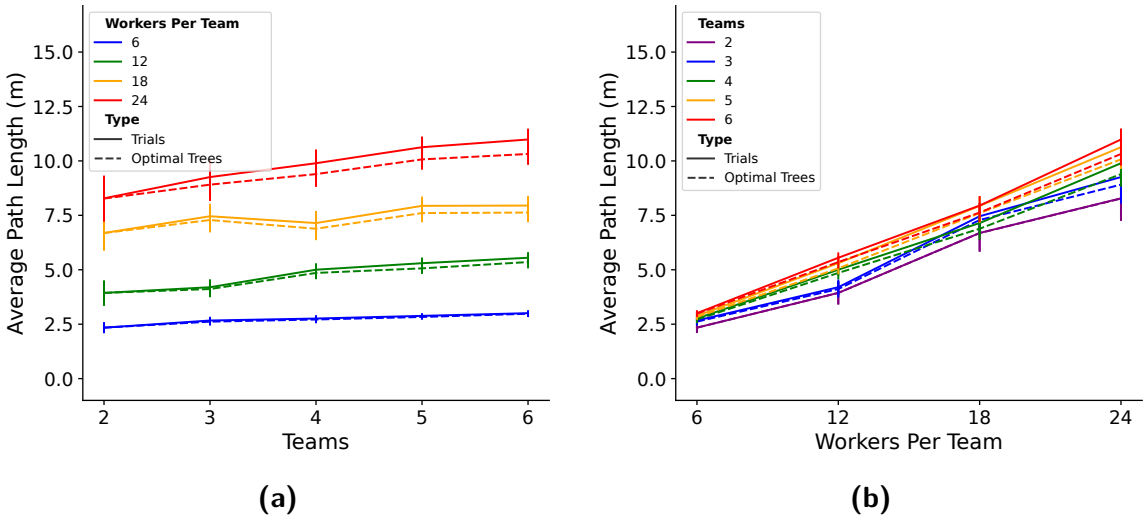


Figure 5.26. Comparison of the average path length between pairs of teams when modifying **(a)** the number of teams and **(b)** the number of workers per team. The solid and dashed lines represent the number of connectors from the simulation trials and the solutions found by the GeoSteiner algorithm, respectively.

similar to the optimal solution (Figure 5.26). This is an interesting result, as we expected this to increase as well, following the increase of the other two performance measures.

To quantify how much the tree produced from the simulation trials diverged from the optimal solutions, we examined the relative performance of our results against the optimal trees, which is shown as a heatmap in Figure 5.27. The value in each cell shows the results of the trials divided by the results of the optimal tree. For network length, a gradual increase from left to right (up to 25%) can be observed indicating that an increase in the number of teams has a greater effect on the total network length, while an increase in the number of robots per team has a smaller effect. One factor of this trend could be due to longer networks accumulating the deviation from the optimal tree as it grows in size. The deviation may continue to increase for every connector forming a Y-junction that does not have edges separated by 120° from each other due to the difference in the network structures.

A mixed result can be observed for the number of connectors in the network. Increasing the number of workers per team reduces the ratio when the number of teams is small (i.e. 2 teams) and increases it when there are more teams (i.e. 4, 5, and 6 teams). Similarly increasing the number of teams reduces the ratio when the number of workers per team is small (i.e. 6 and 12 workers per team) and increases it for larger

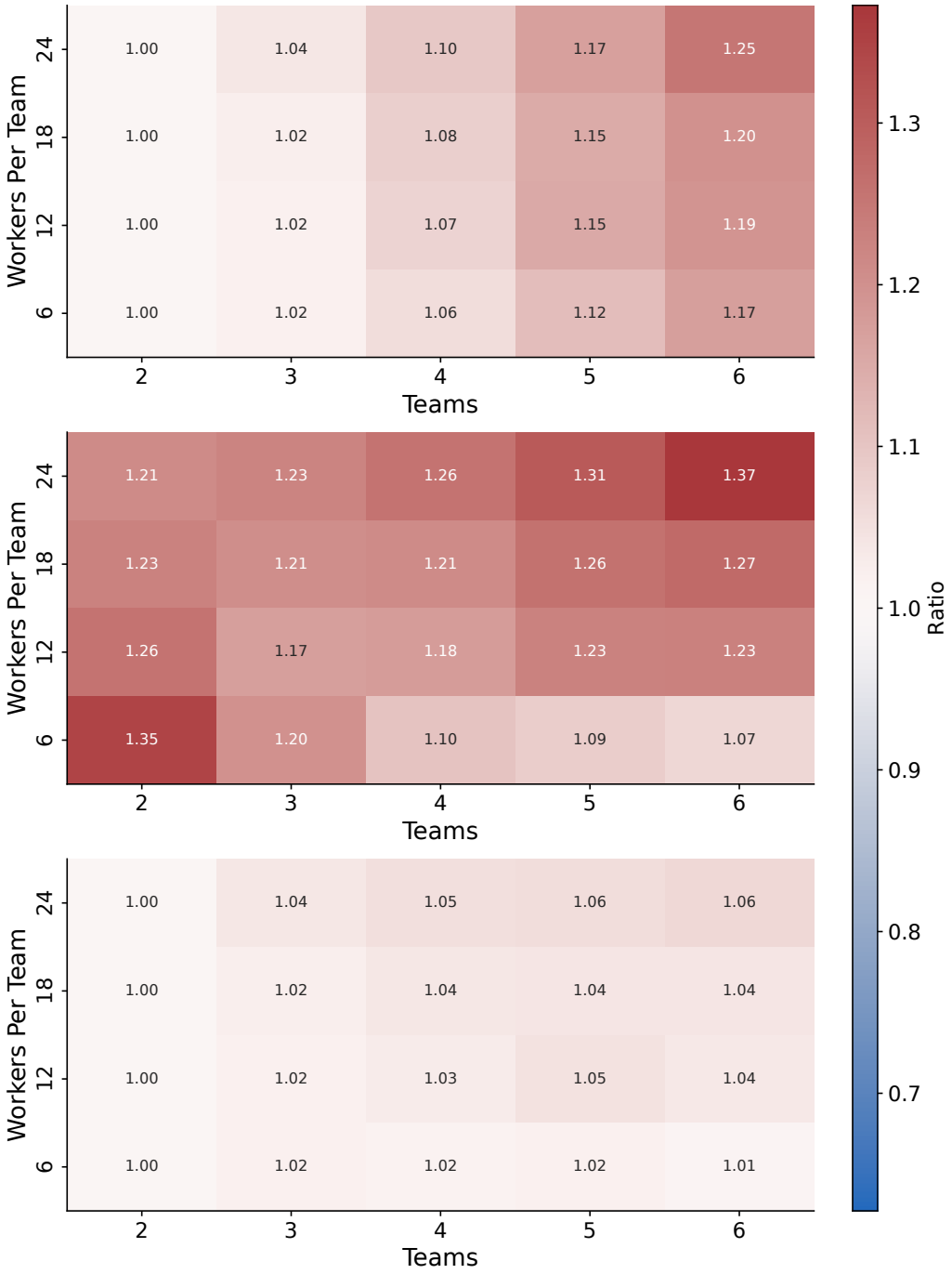


Figure 5.27. Performance of the tree produced by the proposed method relative to the tree that is optimal with respect to the total length (smaller is better). (Top) The network length; (middle) number of connectors; (bottom) average path length. The number in the cell represents how the obtained result compares to the optimal tree.

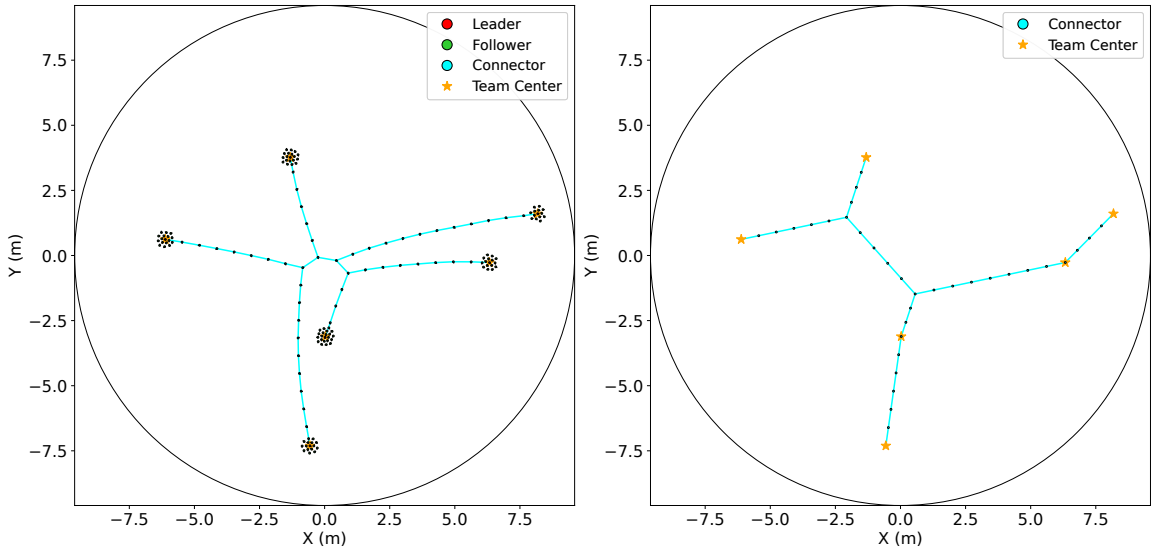


Figure 5.28. Final timestep of a simulation trial (left) and the solution found by the GeoSteiner algorithm (right). The simulation trial had 46 connectors, while the algorithm produced a tree with 33 relay nodes.

teams (i.e. 18 and 24 workers per team). These effects appear to flip when there are 3 teams and 12 or 18 workers per team. The high ratio observed for a smaller number of teams and workers can be explained by the small number of connectors needed to form the network; adding an extra connector to the network affects the ratio more significantly than if the network already consisted of many connectors.

The ratio was also high for larger teams, where 6 teams and 24 workers per team showed a 37% increase compared to the optimal tree. Figure 5.28 shows a situation where many connectors were required to form a network, highlighting the limitation of the proposed method. The trial resulted in 46 connectors, while the optimal tree had 33 added points. If we look at the two teams on the right from the trial result, we see that there are two long branches connecting each team with the rest of the network. Ideally, we would like a single branch connecting both of these teams to reduce the number of connectors. The reason for the formation of these two branches stems from the conditions $RB1$–$RB3$ as connectors can only consider switching their current connector to a different one if it is already adjacent to the current connector. Further improvements to the conditions that relax some of these constraints without losing connectivity could allow the formation of networks requiring fewer connectors.

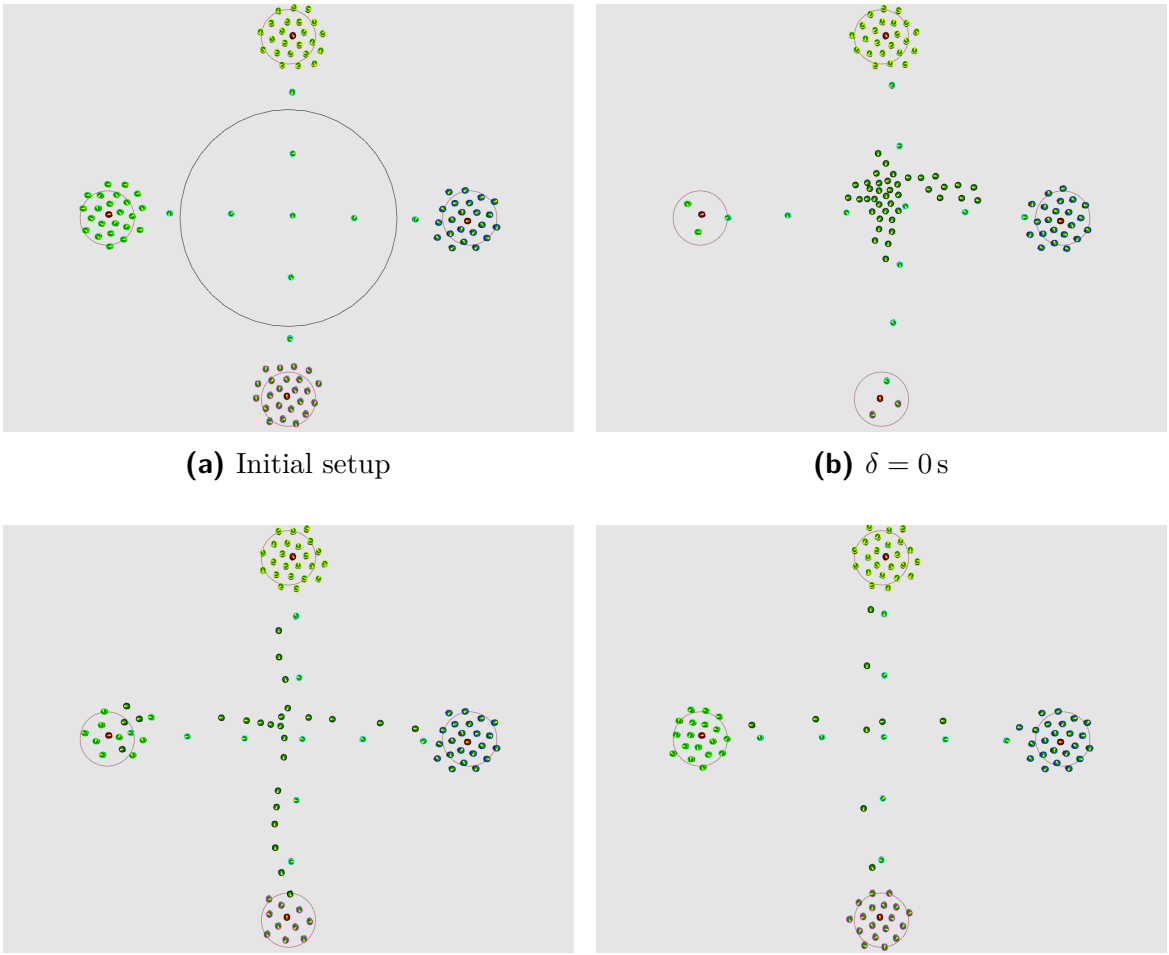
The average path length showed a relatively small deviation from the optimal tree for various numbers of teams and workers per team. When workers are shared

between two teams by moving them along the network, the path length between them determines how long it takes for the workers to reach the other team. The result highlights the robustness of the network formed by the connectors in keeping the distance between the teams consistent.

Among the simulations conducted, two trials were unsuccessful in maintaining global connectivity between the teams. Upon further inspection, we found that followers in one team were being split and left behind from their lead agent due to another team attempting to move through the team. If the lead agent is split from its followers who are connected to the rest of the network, it loses connectivity with other teams. Connectivity can be recovered if the lead agent moves within the communication range of the group of followers that were split from its team.

5.3.5 Traveller Congestion Analysis

In this section, we investigate the robustness of the network when a large number of connectors move along the network causing high congestion, and how much delay should be added between sending each traveller. For this study, we use 4 teams and 25 workers per team. Two types of congestion are considered; *CROSS* and *LINE*. Both setups involve 4 tasks. Figure 5.29a shows the *CROSS* setup. Four tasks were placed 2 m away from the arena centre and separated at 90° from each other. At $T = 50$ s, the teams on the left and bottom each sent 20 workers to the teams on the right and top, respectively. This setup causes two groups of travellers to intersect at the centre of the arena. Figure 5.30a shows the *LINE* setup. Four tasks were evenly placed 2 m away from each other in a line. At $T = 50$ s, the team on the far left sent 20 workers to the team on the far right. This setup caused the travellers to move through or near the teams that were located along the network. We intentionally set a large value for $n^{\min}$ (i.e. $n^{\min} = 100$) so that the task cannot be completed. In all our previous simulation studies, we have been sending travellers every 3 s. We modify the time to wait (i.e. $\delta = 0, 1, 2, 3, 6, 9$ and 12 s) between sending travellers to see whether this has an effect on global connectivity and safety (i.e. collisions). We ran 50 trials for each setup and delay, totalling 700 runs. The simulation was terminated when there were no more travellers in the simulation or the timeout $T = 1000$ s was reached, whichever occurred first. Trials were considered unsuccessful if global connectivity was lost or the travellers did not reach the target team before the timeout was reached. Figures 5.29b – 5.29d show typical trials from the *CROSS* setup, while Figures 5.30b – 5.30d show typical trials from the *LINE* setup.



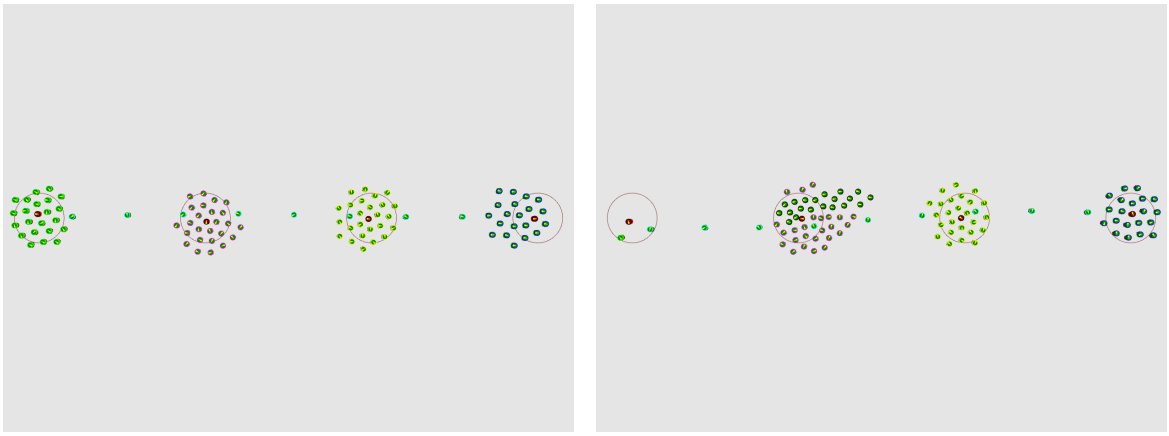
(a) Initial setup

(b) $\delta = 0\,\mathrm{s}$

(c) $\delta = 3\,\mathrm{s}$

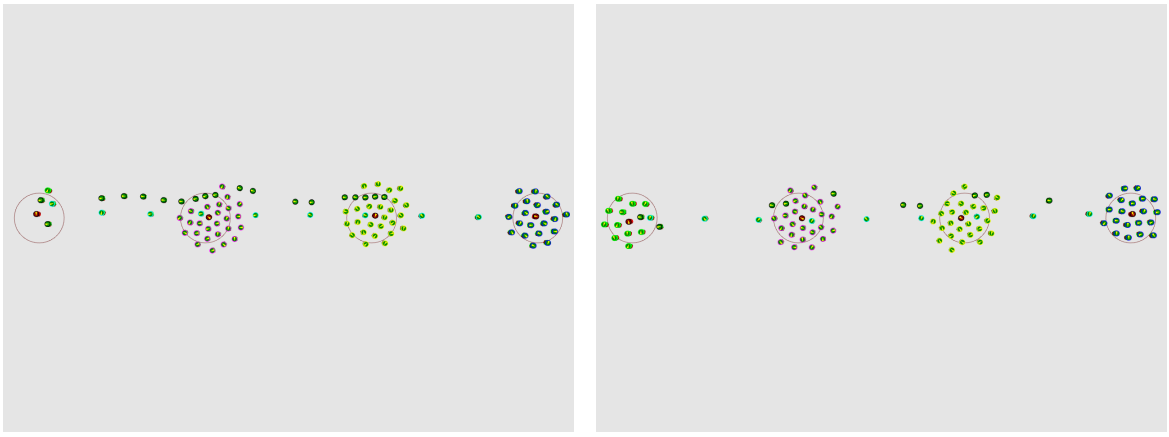
(d) $\delta = 9\,\mathrm{s}$

Figure 5.29. Snapshots of the *CROSS* congestion trials. **(a)** *CROSS* setup before sending travellers. **(b–d)** A typical trial where the travellers are moving along the network when dispatched every $\delta = 0, 3$ and $9\,\mathrm{s}$, respectively.



(a) Initial setup

(b) $\delta = 0\,\mathrm{s}$



(c) $\delta = 3\,\mathrm{s}$

(d) $\delta = 9\,\mathrm{s}$

Figure 5.30. Snapshots of the *LINE* congestion trials. **(a)** Initial *LINE* setup before sending travellers. **(b–d)** A typical trial where travellers are moving along the network when dispatched every $\delta = 0, 3$ and $9\,\mathrm{s}$, respectively.

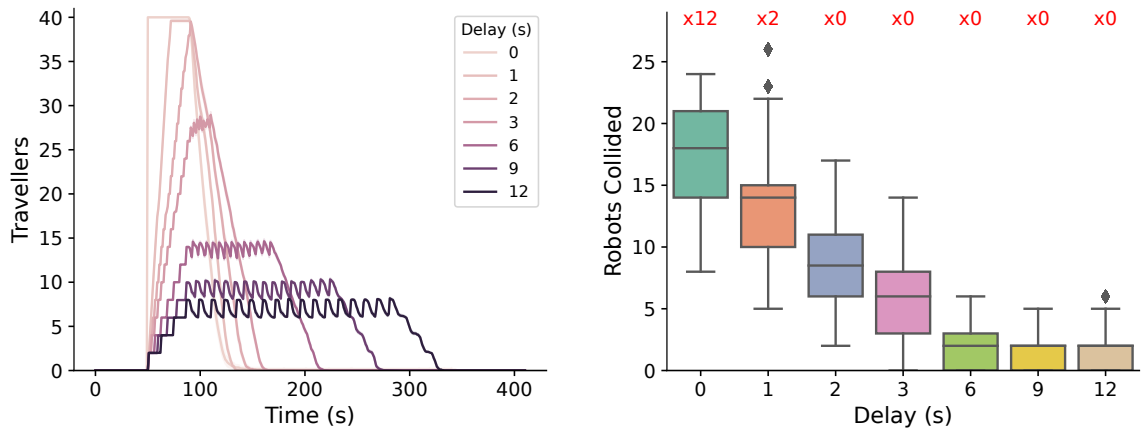


Figure 5.31. Performance of traveller congestion analysis where a new traveller is dispatched every $\delta = 0, 1, 2, 3, 6, 9$ and 12 s by the lead agents (*CROSS* setup). **(a)** The number of travellers during the mission. **(b)** The number of robots that collided. The text in red at the top reports the number of unsuccessful trials.

Figure 5.31a shows the average number of travellers in the *CROSS* setup. Starting at $T = 50$ s, we can see that the number of travellers increases at different rates. The travellers move along the network and join their target team faster when the delay in sending travellers is shorter. However, as shown in Figure 5.31b, the number of robots that collided during the trials was higher when the delay was short and plateaued at around 6 s. This was a result of travellers being highly congested at the point where they intersected each others' paths which was observed when δ was small (Figure 5.29b). The large surge of travellers from behind caused those at the intersection to collide with each other more frequently. Moreover, a number of trials with short delays (i.e. $\delta = 0$ and 1 s) lost global connectivity between the teams. This was due to the travellers trapping one or more followers from the team they had just left and moving them away from the lead agent. When the lead agent was not directly connected to the network and all of its followers connected to the network were pushed away from the team, the lead agent lost connectivity with the other teams.

Figure 5.32 shows the average number of travellers and the number of robots that collided in the *LINE* setup. Similarly to the *CROSS* setup, travellers reached their target team faster when the delay was short, but resulted in more collisions. The collisions mostly occurred between the travellers and the followers of teams that were situated along the network. Loss of global connectivity was also more likely for shorter delays, similar to that of the *CROSS* setup.

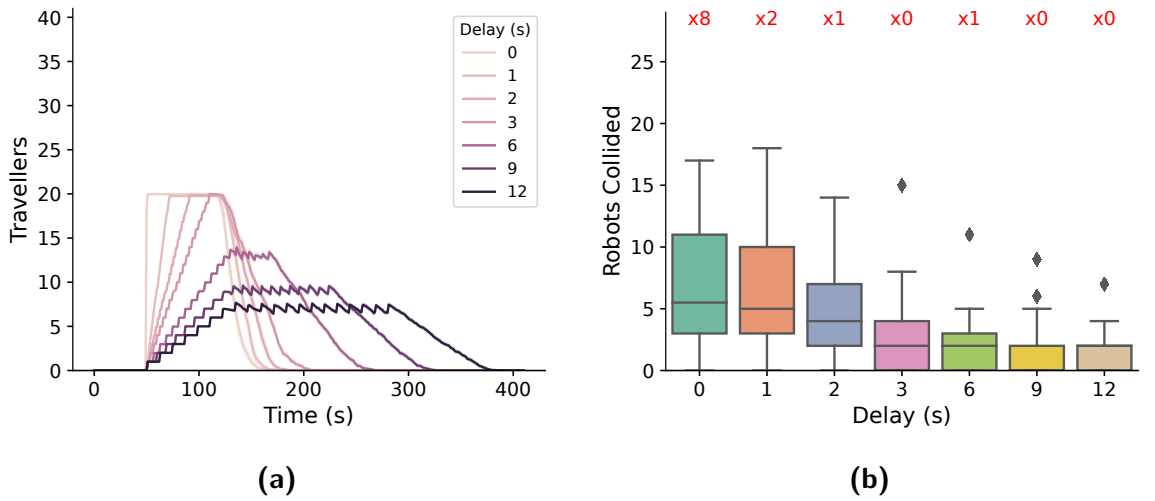


Figure 5.32. Performance of traveller congestion analysis where a new traveller is dispatched every $\delta = 0, 1, 2, 3, 6, 9$ and 12 s by the lead agents (*LINE* setup). **(a)** The number of travellers during the mission. **(b)** The number of robots that collided. The text in red at the top reports the number of unsuccessful trials.

The results of the two congestion studies suggest that workers should start moving along the network one at a time to reduce the chance of colliding with other robots and avoid the risk of causing the team to get disconnected from the network by dragging a follower away from its team.

5.4 Summary

In this chapter, we showed how the control of a robot swarm can be shared between an arbitrary number of human operators. The robot swarm autonomously maintained connectivity between all lead agents controlled by the operators by forming new branches in the network. To minimise the number of robots in the network, a set of constraints were designed to allow the robots to dynamically modify the network topology as the lead agents moved through the environment. These constraints were incorporated into the robot controller developed using SCT. In addition, a sequential robot-requesting strategy for operators was presented. This provided the operator the ability to negotiate with one operator at a time, instead of having to broadcast the request to all operators simultaneously.

Using computer simulations involving up to 6 lead agents and 144 robots, we demonstrated that the robot swarm successfully maintains a network between the lead

agents and allows the operators to share robots along the network. When compared against an optimal tree with minimum network length, the network formed by the robots was shown to be near the optimal length when the number of teams was small, but gradually deviated further as more teams were added. However, the average path length between a pair of teams remained close to that of the optimal tree regardless of the number of teams. In fact, the maximum path length was often shown to be shorter than the ones in the optimal tree, which helps in reducing the time a traveller spends moving between teams. The network was also tested under high congestion of robots moving along it. Although the network did manage to maintain connectivity in most trials, we showed that a small delay should be added between every robot being sent to avoid the likelihood of collisions between robots.

Chapter 6

Real Robot Validation

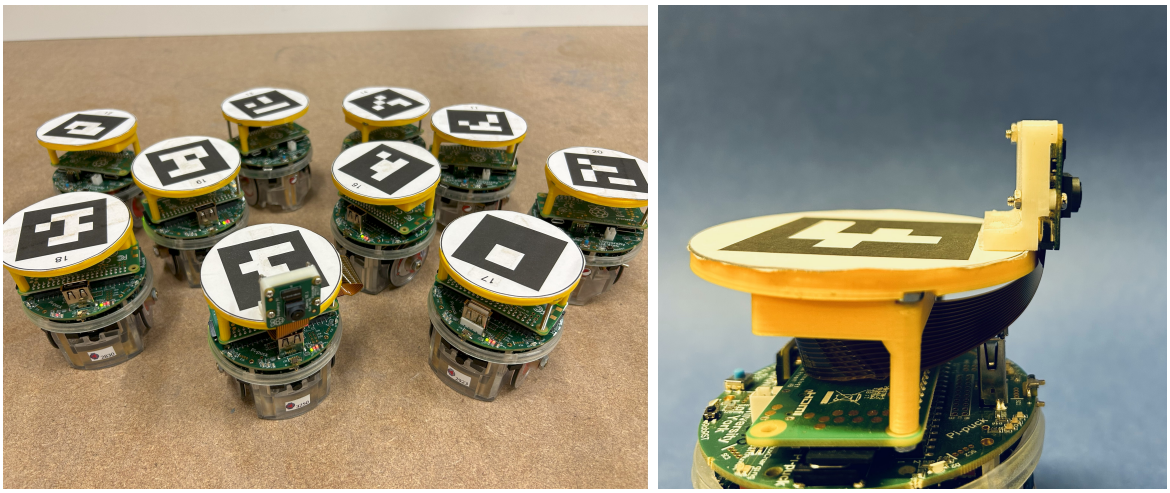
In the previous chapter, we proposed an extension for our framework to allow an arbitrary number of operators to interact with the robot swarm. This was achieved through the dynamic branching and optimisation of the connectivity network structure in a distributed manner. The performance of the extension was evaluated in simulation.

In this chapter, we validate our framework using a physical robotic platform, demonstrate the usability of the graphical user interface when multiple operators interact with real robots, and discuss the problems and challenges that have been identified when deploying the framework on real robots.

This chapter is organised as follows. Section 6.1 describes the robotic platform and system architecture for conducting the real robot experiments as well as how the framework and interface were connected with the rest of the system. Section 6.2 presents the experiments conducted to validate the framework involving both simulated and real human operators. Section 6.3 concludes the chapter.

6.1 Methodology

This section presents the robotic platform and system setup prepared for this study and how the user interface developed in Chapter 4 is connected with the rest of the system.



(a) **(b)**

Figure 6.1. (a) Pi-puck robots used for the experiments. **(b)** Close-up view of the Raspberry Pi camera module attached to the Pi-puck via the camera mount.

6.1.1 Robot Platform

We use the Pi-puck robots (Millard et al., 2017) to validate the framework on a physical platform (Figure 6.1a). The Pi-puck is an extended version of the e-puck that is equipped with a Raspberry Pi via the Pi-puck extension board. The Raspberry Pi provides Linux support and improved computational capabilities for the e-puck. The Pi-pucks used in this chapter are mounted with a Raspberry Pi Zero W with Raspbian Buster installed.

The Pi-puck is equipped with a yellow hat that displays a unique ArUco marker. The marker is used to detect the position and orientation of the robot using an overhead camera (further details are discussed in Section 6.1.2). The height of the Pi-puck equipped with the hat is 8.4 cm.

The Pi-pucks can optionally be equipped with a Raspberry Pi camera module[1] (Figure 6.1b). For this proof-of-concept study, we attached a camera module to two Pi-pucks that acted as lead agents. In particular, we attached the 8-mega pixel Raspberry Pi Camera Module v2 or v2.1 to the yellow hat using a custom designed camera mount. The camera mount has a height of 2.4 cm and a width of 2.5 cm. The combined thickness of the camera module and the mount is 2.0 cm. The camera module has a horizontal field of view of 62.2° and a vertical field of view of 48.8°. This provides a local view of the surrounding environment from the Pi-puck's perspective, which

[1] `https://www.raspberrypi.com/documentation/accessories/camera.html`

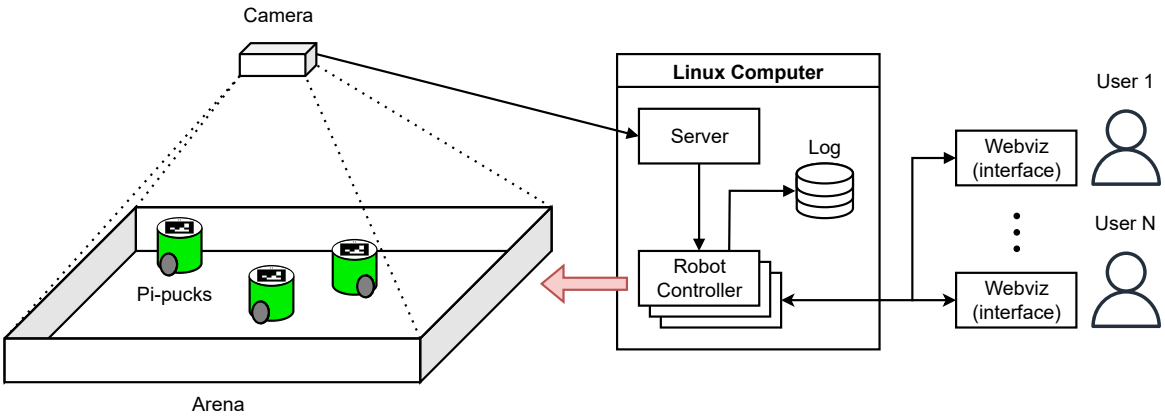


Figure 6.2. SwarmHack system overview. The overhead camera captures the global positions and orientations of Pi-pucks and other objects with an ArUco marker. This data is passed to the server, which provides localised information to each robot controller. Each robot controller determines the behaviour of each Pi-puck. A user can connect to a lead agent via the interface to interact with the swarm.

closely resembles to the local view examined in Chapter 4. The maximum resolution to record a video using the camera module is 1920 x 1080 at 30 frames per second. In order to provide a consistent video stream to the user interfaces in real-time, we used a lower resolution of 640 x 480 at 5 frames per second.

6.1.2 SwarmHack System

Figure 6.2 illustrates the SwarmHack setup that has been used at the University of Sheffield for this proof-of-concept study. The robots operate within a bounded 2 m × 1 m arena (Figure 6.3), part of a setup developed for the SwarmHack 22' event by the group led by Dr Alan Millard at the University of York. The arena floor is made of a sheet of medium-density fibreboard. An overhead camera is mounted to the arena frame 1.5 m above the centre of the arena floor. The overhead camera is connected to a Linux computer and detects the positions and orientations of ArUco markers, such as those attached to the robots and to the arena corners.

The robots' controllers to operate the Pi-pucks in the arena are located within the Linux computer, but are managed independently from one another. The server processes the ArUco markers captured by the overhead camera. The server then provides each robot with information about other neighbouring robots within their communication range and any task area that they occupy, relative to their current

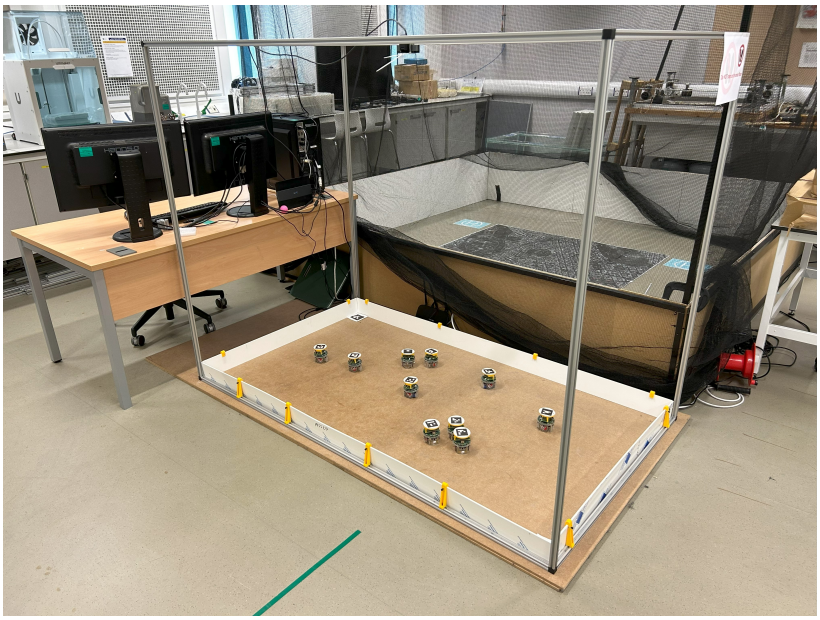


Figure 6.3. The SwarmHack arena setup.

position and orientation in the arena. It is possible to obtain the proximity sensor values from each Pi-puck, though this feature was not used for this study as it occasionally caused high latency depending on the Pi-puck's signal strength.

The SwarmHack setup is written in Python, whereas the SCT supervisors discussed throughout this thesis are executed by a library written in `C++`. Therefore, the framework and the script to run the SCT supervisors—also known as the *generator player*—have been implemented in Python for compatibility.

Figure 6.4 shows the graphical user interface that was adapted for the SwarmHack setup. The interface is based on Webviz (Patel et al., 2022), which uses websockets to communicate the simulation state and user commands with a simulation running in ARGoS. As the SwarmHack system also uses websockets to communicate between the computer and the robots, we modified it so that it uses the same message formats to send and receive information as those that were used by the interface. This allowed information such as the team, task, and other logs to be updated in the interface in real-time. The interface preserved the same keyboard and mouse inputs from the original one used for the user study in Chapter 4 to teleoperate a lead agent and request or send robots to another operator from the request-and-send panel. The video stream displayed in each interface was provided by the computer which rebroadcasted the view from the Pi-puck cameras.

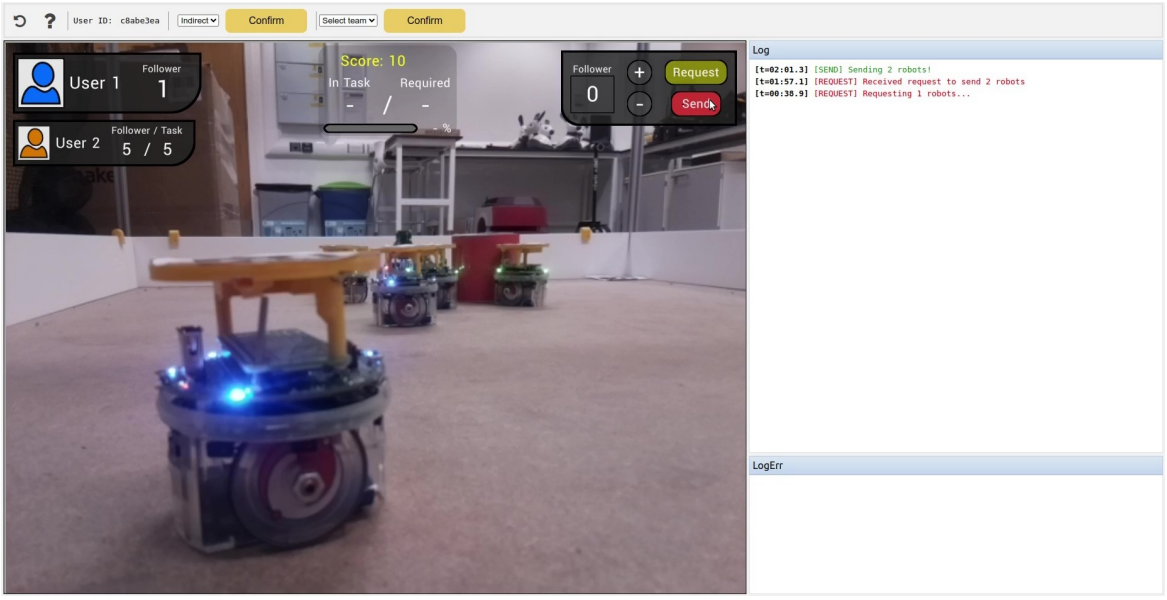


Figure 6.4. The user interface adapted for the SwarmHack system.

6.2 Experiments

This section describes the experiments conducted with real robots. The first experiment considers lead agents controlled by simulated operators to validate the connectivity-preserving framework in real life. The second experiment involves lead agents controlled by real human operators to examine the usability of the graphical user interface developed in Chapter 4.

6.2.1 Real Robots and Simulated Operators

To validate the feasibility of the connectivity-preserving robot swarm with multiple operators, we conducted experiments using ten physical Pi-pucks; three of these were lead agents and seven of these were workers. One lead agent was initially assigned three workers and the other two were each assigned two workers. The arena consisted of three tasks, which were performed when a lead agent and a sufficient number of workers entered the task area. The tasks had a circular area. Three tasks were uniformly randomly distributed in the arena in a way so that the task centres were within the boundaries of the arena and no task areas were overlapping with one another. Two of the tasks required $n^{\min} = 1$ workers to complete, while one task required $n^{\min} = 2$ workers. The task radius was set to $0.2\sqrt{n^{\min}}$ m. We used a communication range of 0.6 m for the Pi-pucks. The lead agents and workers were initially clustered at the

centre (Figure 6.5a). The simulated operators were each preassigned a unique task to move their respective lead agent towards. If the lead agent was inside the task area and there were not enough followers to carry out the task, the simulated operator requested for workers to the other two operators as described in Section 5.2.6.

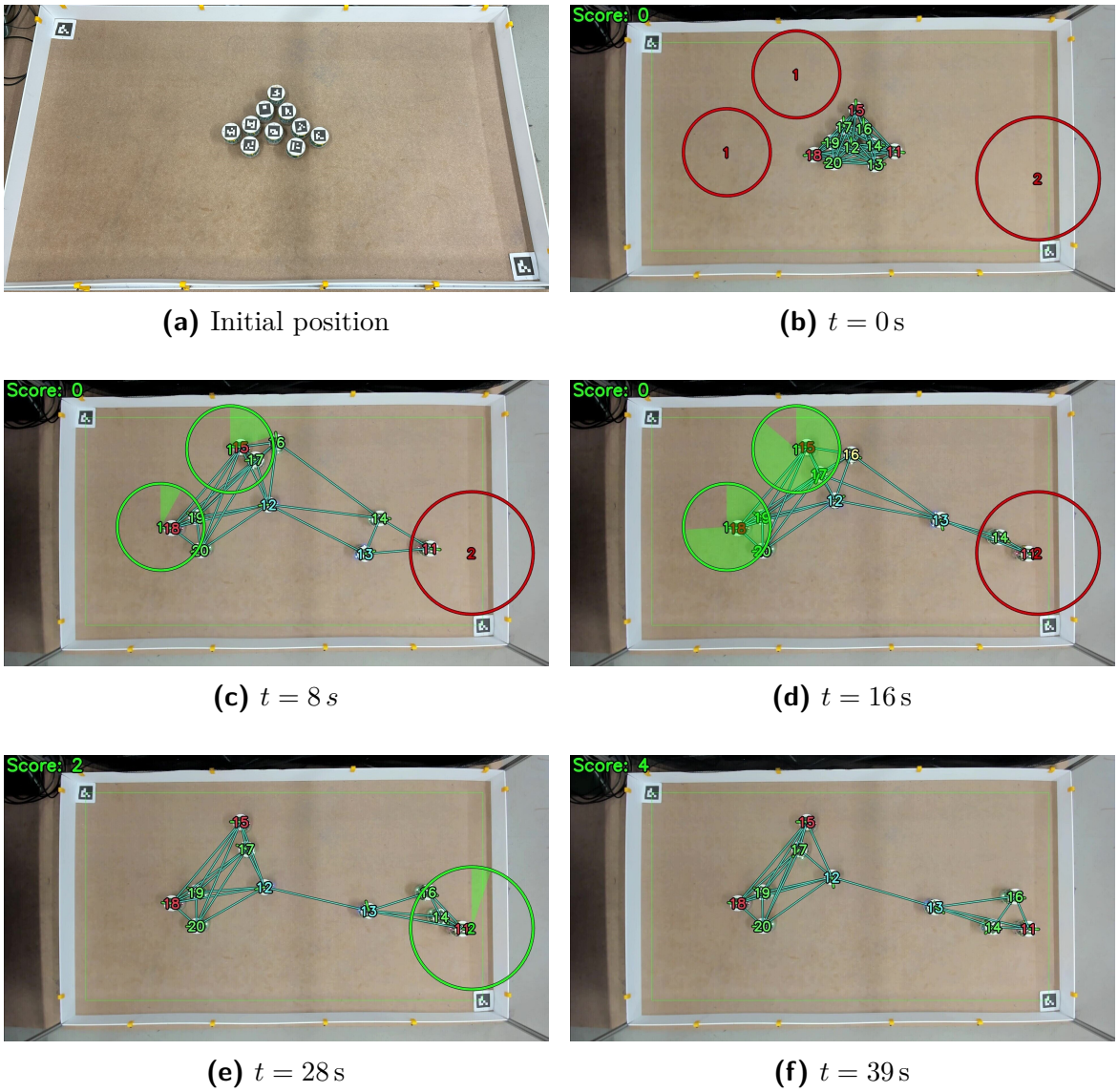
A total of 10 experimental trials were performed. All trials were recorded by the overhead camera. Figures 6.5b – 6.5f shows a sequence of snapshots from one of the experiment trials. As the lead agents move towards their assigned tasks, the workers can be seen to maintain connectivity between the lead agents. In Figure 6.5c, lead agent 11 arrives at the task requiring two workers but only has one (worker 14), so it requests an additional worker to lead agent 15. Lead agent 15 accepts the request and sends worker 16 to lead agent 11 as its task only requires one worker to complete. Worker 16 travels towards lead agent 11 (Figure 6.5d) and completes the task together with the existing worker (Figures 6.5e and 6.5f). A video recording of this trial is available from the following link: `https://youtu.be/25NVMd-p3kU`.

Figure 6.6 shows the robots at the final timestep of the trial from Figure 6.5 and the Euclidean Steiner minimum tree (ESMT) with steinerised edges. The team centroids were provided as input to the GeoSteiner algorithm to find the ESMT, which is optimal with respect to its total length. Both the experiment result and the optimal tree resulted in two connectors. The total network length was 1.467 m for the trial and 1.419 m for the optimal tree.

Figure 6.7 shows a comparison of the performance measures from Section 5.3.2 between the experimental trials and the optimal tree for all ten trials. The total network length of the experiment trials was slightly longer than the optimal tree and had either the same number or one extra connector maintaining the connectivity between the lead agents. Figure 6.8 shows the average path lengths between any two teams in the network. The experiment trials had longer minimum path lengths but shorter maximum path lengths, causing them to be more clustered than the optimal tree. These results are in line with the results observed in the simulation trials in Chapter 5.

6.2.2 Real Robots and Human Operators

This section demonstrates how the user interface can be used by human operators to complete the tasks. The tasks in the previous experiment were virtual so they were not visibly present in the arena. As the human operators need to be able to recognise the tasks through the Raspberry Pi camera module attached to the lead agents, red



(a) Initial position

(b) $t = 0\,\mathrm{s}$

(c) $t = 8\,s$

(d) $t = 16\,\mathrm{s}$

(e) $t = 28\,\mathrm{s}$

(f) $t = 39\,\mathrm{s}$

Figure 6.5. Sequence of snapshots from an experimental trial. **(a)** Initial positions of the Pi-pucks. **(b–f)** Overhead camera showing a real-robot experiment with three teams completing three tasks. Red, green, cyan, and yellow numbers are overlayed on the lead agent, follower, connector, and traveller robots respectively. Cyan lines indicate pairs of robots that are within each other's communication range. Red and green circles represent a task that has not yet been performed or partially performed, respectively. The number inside the circle shows the minimum number of workers needed.

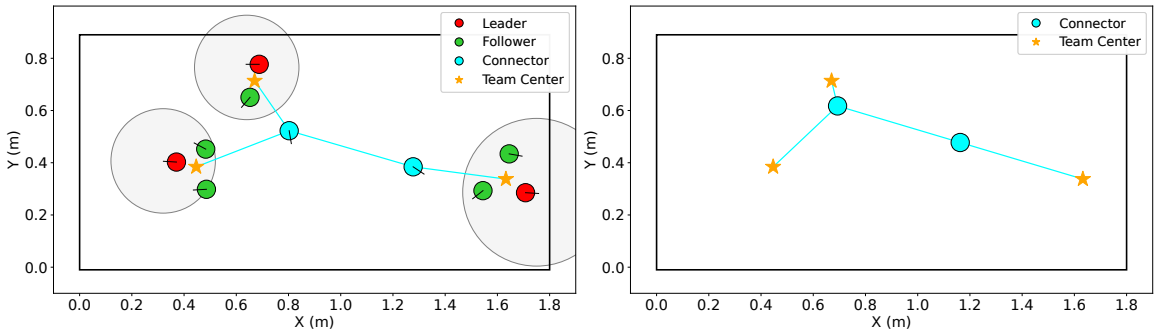


Figure 6.6. Robots at the final timestep of an experiment trial (left) and the optimal solution found by the GeoSteiner algorithm (right). Red, green, and cyan circles represent the lead agents, followers, and connectors, respectively. Cyan lines represent the connection maintained by the connectors. Yellow stars indicate the centroid of each team.

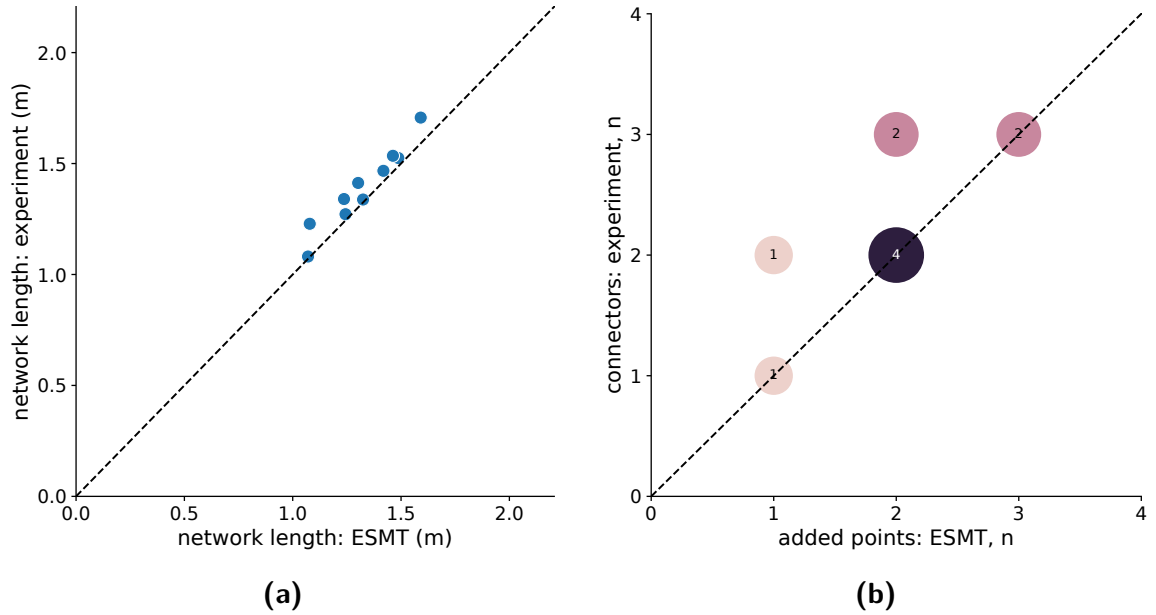


Figure 6.7. Comparison of **(a)** the total network length and **(b)** the number of connectors between the experiments and the optimal solution found by the GeoSteiner algorithm (ESMT).

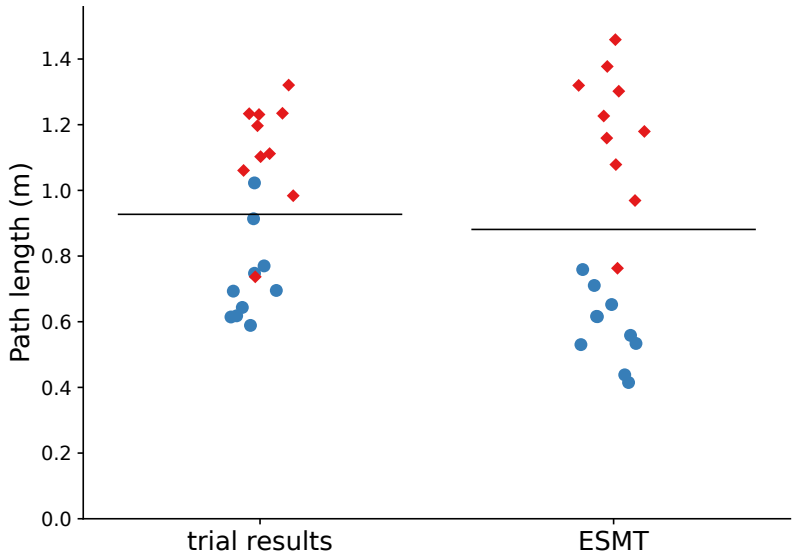


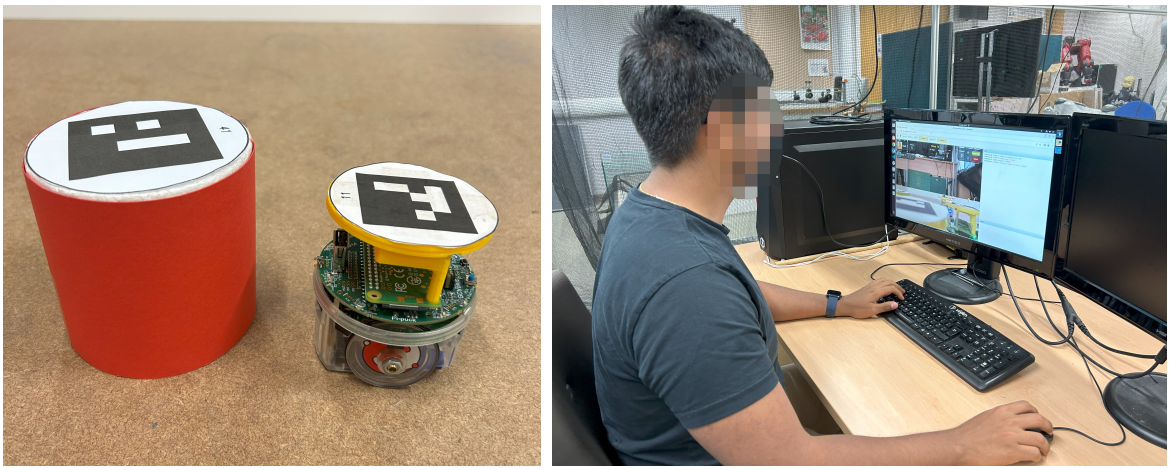
Figure 6.8. Minimum and maximum path lengths between a pair of teams for the experiments and the optimal solution found by the GeoSteiner algorithm (ESMT). Red and blue points represent the maximum and minimum path lengths of a given trial, respectively. The black line shows the mean of all path lengths in every trial.

cylinder objects made of expanded polystyrene were used to represent the task centres (Figure 6.9a). Task objects were made taller than the Pi-pucks so that an operator could find a task even when the lead agent was surrounded by other robots. Each task object had a unique ArUco marker that could be detected by the overhead camera. To prevent task objects from being pushed around by the Pi-pucks, double-sided tape was applied to the bottom of each object to make them stick to the arena floor.

The operators were tasked to complete five tasks. The minimum number of robots required to complete the tasks was predefined, ranging between one to five robots. The task radius was set to $0.2\sqrt[3]{n^{\min}}$ m, which is slightly larger than the previous experiment since the centre of the task area was now occupied by the task object. Two task objects were present in the arena at any time. The tasks were replaced with new ones as they were completed.

The trial was conducted by two operators. One operator was a student from the research group who received an hour of training to use the interface, while the other operator was the author himself. Figure 6.9b shows one of the operators using the interface.

Figure 6.10a shows the position of the robots and a view of the user interfaces controlled by the two human operators at the beginning of the trial. Figures 6.10b



(a) **(b)**

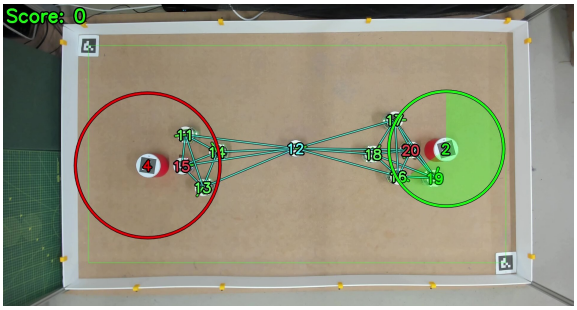
Figure 6.9. (a) A red cylinder object used to represent a task in the arena. Each task had an ArUco marker that could be detected by the overhead camera. **(b)** An operator interacting with the user interface.

— 6.10e show a sequence of snapshots where one operator sent a robot to the other operator to complete a large task. Upon entering the task area, Operator 1 (who is controlling lead agent 15) sends a request via the interface for one robot to Operator 2 (who is controlling lead agent 20). Operator 2 responds by sending one robot, which travels (Figure 6.10c) and joins Operator 1's team to work on the task (Figure 6.10d). The task is successfully completed, while a new task has been placed in the arena (Figure 6.10e).

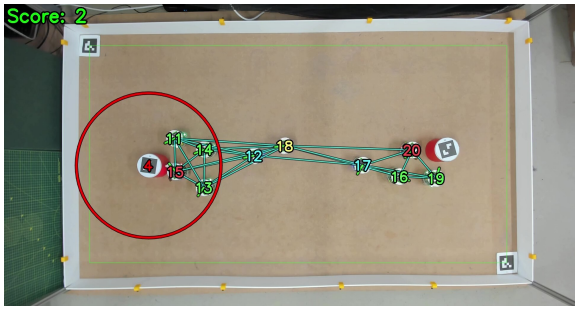
The operators continued to work on the tasks and successfully completed all five tasks in 2 min 29 s, scoring 15 points (i.e. the tasks awarded 1, 2, 3, 4, and 5 points). Similarly to the user study with simulated robots from Chapter 4, the operators can be seen to navigate their followers towards the tasks and share robots with each other through the user interface. The robots were able to maintain connectivity between the lead agents throughout the trial. This demonstrates the applicability of the proposed connectivity-preserving method and the usability of the user interface on real robotic platforms. Note that the level of complexity of the tasks used in this experiment was simplified from the one used during the user study due to limitations of the arena size and the pi-puck's battery capacity.

The trial also highlighted several issues, some of which have emerged while transitioning from the simulation to the real world. Despite efforts to reduce the delay in the video stream by reducing the resolution and frame rate, there was approximately

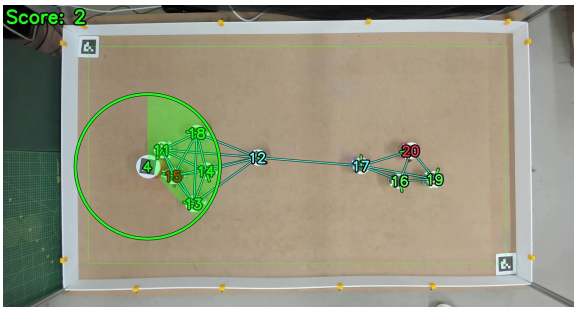
(a)



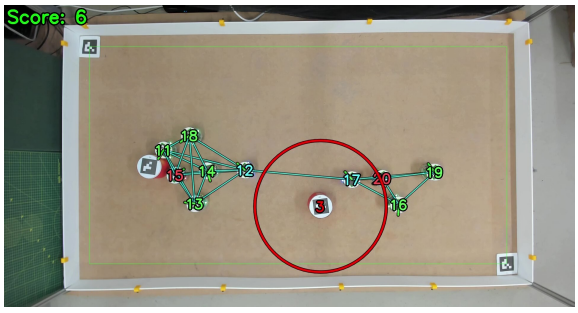
(b)



(c)



(d)



(e)

Figure 6.10. **(a)** The initial position of the robots, the tasks, and the user interfaces used by the operators. **(b–e)** Sequence of snapshots from the overhead camera showing operator 2 (team on the right) sending one of its workers to operator 1 (team on the left).

a one-second delay in displaying the stream to the user interface. This caused some difficulty when teleoperating the lead agent, since the robot would move almost instantaneously, while the stream would lag behind. Furthermore, motion parameters had been manually tuned before the trials and would benefit from further optimisation. A limitation of the user interface was the lack of support for three or more operators. This was due to reusing the same interface developed for the user study in Chapter 4, which was assumed to be used by two human operators. Future improvements in the user interface to support an arbitrary number of operators would be beneficial.

6.3 Summary

In this chapter, we verified the connectivity-preserving robot swarm and the user interface developed in the previous chapters in the real world. To make the transition from simulation to the real world, the existing SwarmHack setup using Pi-pucks was extended to support the necessary capabilities. To execute the SCT supervisors, the generator player was implemented in Python. No changes were applied to the supervisors themselves, which meant there was no need to reimplement the control logic; it was sufficient to implement the callback functions associated with each event. A camera was attached to the Pi-pucks that were used as lead agents to provide a local view of the environment from the robot's perspective. The SwarmHack system was extended to be compatible with the user interface to support its functionalities to update the information displayed on the interface, present a real-time video stream received from the lead agent, teleoperate the lead agent and communicate with another operator to share robots.

Trials were performed where real robots interacted with either simulated or real human operators to demonstrate how the swarm can maintain global connectivity and allow the operators to share robots while working on spatially distributed tasks. It was found that the network formed by the robots to maintain connectivity between three simulated operators produced similar results across all performance metrics compared to those seen in the simulation. We also showed a proof-of-concept for a pair of human operators and a group of robots completing tasks in the real world. The trial demonstrated that human operators can successfully interact with the swarm and request robots from the other operator using the user interface. Future work could extend the interface to support three or more operators, reduce the latency of the video stream and perform extensive user studies.

Chapter 7

Conclusions

Human-swarm interaction has attracted a lot of interest in the last decade. This has led to a large body of work that has focused on how a single human operator can influence the behaviour of a swarm of robots. However, it becomes increasingly difficult for a single operator to manage multiple subgroups within the swarm that are working on different tasks at the same time. This could be addressed by involving multiple operators who jointly manage the swarm, but also highlights new problems that need to be considered, such as inter-human communication and sharing of robots among the operators.

This thesis considered a robot swarm as a resource that can be shared among the operators and focused on how they can dynamically exchange robots with one another. The objective of the human-swarm team was to complete spatially distributed tasks within an environment that lacked the infrastructure to support global communication. Each operator was initially assigned a subswarm to carry out the tasks. We developed a swarm that maintains connectivity between the operators by forming an ad-hoc network with the robots. This network allowed the operators to share mission-related information as well as exchange robots with each other depending on the current task requirements at the time. The robots' control logic was formally modelled using supervisory control theory (SCT). The use of this formal approach facilitated the development of swarm behaviours and reduced the amount of implementation required through automatic code generation. The robots operated in a distributed manner, so no centralised controller was needed.

In Chapter 3, we presented a robot swarm that maintains connectivity between two human operators. A set of conditions was defined to allow the robots to dynamically

switch their roles to perform tasks for the human operators and to maintain connectivity between them as they navigated through the environment. These conditions and the robot behaviours were formally modelled using SCT. Furthermore, we used public events to model communication between robots that affected the behaviour of the robots. Simulation studies showed that the robots were able to successfully maintain connectivity between the operators. We also highlighted a situation where the ability to send robots between operators reduced the time to complete all tasks as well as the distance travelled by the robots.

In Chapter 4, we conducted a user study to investigate the ability of a pair of human operators to dynamically share robots in a swarm. Each operator interacted with the swarm and their partner through a user interface, which provided information about the state of the swarm, the task and a restricted local view of the environment; no global view was provided. By using the interface, the operators were able to request and send robots to each other. Through a mixed factorial study, we examined the usefulness of the ability to share robots and the effects of different communication types on performance and human factors. It was found that sharing robots did not necessarily increase mission-related performance measures, but it enabled the operators to explore further, provided the flexibility to work independently and collaboratively, and reduced the total energy consumed by the robots.

While previous chapters considered a pair of human operators sharing the control of a robot swarm, Chapter 5 presented an extension that allows an arbitrary number of operators to share the swarm. To adapt to the operators exploring the environment, the swarm dynamically updated the communication network by modifying its topology and adjusting the positions of the robots. We showed that the swarm was able to evolve a network that was comparable to the optimal tree with respect to the total network length and the number of robots that maintained it. Furthermore, the distances between the operators along the network were found to be similar but more consistent than the optimal tree, which means the maximum distance a robot must travel in the worst case is kept short. It was also found that the network was robust to high volumes of robots travelling along the network.

In Chapter 6, the framework was verified in real life using the Pi-pucks (Millard et al., 2017) within the arena based on the design of the SwarmHack system. The system was modified to incorporate the user interface and other functions necessary to run experiments. No changes were made to the robots' control logic; the same SCT models were used to operate the Pi-pucks. The behaviour of the swarm was found to

translate well to the real world, demonstrating similar trends in performance that were observed in simulation. A proof-of-concept trial demonstrated how a pair of human operators were able to interact with the swarm via the user interface and share robots among themselves.

This thesis has presented a framework for multiple operators to share the control of a robot swarm. Such human-swarm teams would be beneficial in the real world where the tasks are scattered across the environment and global communication cannot be guaranteed. Application scenarios may include search and rescue, firefighting, infrastructure inspection, forest monitoring, and agriculture. In any of the above scenarios, the robot swarm could work alongside humans to augment their abilities to carry out a wide range of tasks. This may also involve operators dynamically exchanging some of their robots with each other depending on the current needs. However, the proposed method to share robots among multiple operators is only one way to design the relationship between humans and swarms; there is still a lot of room for improvement to realise human-swarm teams that operate in real-life applications. We hope that this thesis inspires future researchers to investigate how the collaboration between a team of humans and robot swarms can solve a wide range of complex problems in the real world.

7.1 Future work

While the proposed framework provides a method for multiple human operators to share the control of a connectivity-preserving robot swarm, there are still many directions in which this work could be extended.

Throughout this thesis, robots that were shared between operators had to travel along the chain of robots to reach and join the other team. One way to improve this could be to shift the chain of robots between the teams. In other words, an operator that requires more robots could essentially "pull" the chain towards itself, adding the closest connectors as new followers to its team until the required number of robots has joined. As the robots considered in this thesis were homogeneous in terms of capability, it did not matter which robot joined the team. Such shifting strategy would greatly reduce the time spent on sharing robots as the time required for a single robot to travel between the teams would be distributed among the connectors in the network. Nevertheless, further consideration is needed for shifting the chain when there are three or more teams present as it may affect other parts of the network.

An inherent limitation of the SCT framework is the use of complex models. If we were to model the behaviour of the entire swarm using a single supervisor, it would suffer from state space explosion even for a few robots due to the composition of all individual robot states. Rather than modelling the entire swarm, this thesis used local modular (de Queiroz and Cury, 2002) supervisors to formally define the behaviour of each individual robot, limiting the growth of the state space. Next, public events (Lopes et al., 2020) were used to specifically model the communication between individual robots to act in a coordinated manner. The use of other supervisory control methods such as hierarchical supervisory control (Zhong and Wonham, 1990) could improve how we model the behaviour of the swarm and individual robots while further minimising the state space. This may also make it possible to model the behaviours and specifications at the global swarm level, rather than at the local robot level which requires careful design of individual robot-level behaviours to anticipate the global behaviour.

Robots in the real world are prone to noise, delay, energy consumption and sudden failures. The method presented in this thesis has not been tested under the presence of such conditions. Communication delays and robot failures may cause a pair of connectors to lose connectivity with each other, resulting in the loss of communication between the operators. Therefore, mechanisms to prevent the network from disconnecting or repairing the network are very much needed. For example, the framework could be extended to form networks that guarantee k-connectivity. To fix a broken network, the connectors adjacent to a failed connector could either move towards it to establish a new connection with other connectors that were also adjacent or inform and recruit one or more followers from the teams to replace any failed connectors, which may travel along the network to reach the broken part. This would improve the swarm's robustness and adaptability to potential hazards in the real world.

This work considered homogeneous robots, but heterogeneous robots that each possess a unique capability could be considered. For example, some robots may be equipped with special tools to perform tasks that can only be performed by them. In such cases, operators will be required to specify the type of robot whenever they make a request. Another example might be assigning dedicated roles to certain robots. For example, UGVs may be required to perform tasks for the operators, while UAVs may be responsible for maintaining global connectivity. Introducing a variety of robot types in the swarm is likely to increase an operator's workload so assistive tools that support them could be beneficial.

The idea of mixing robots with different roles could be extended to humans as well. While this work has considered a flat team structure, where all operators assume the same role, as the number of operators continues to increase, it may be useful to consider teams that have a hierarchical structure. One example would be to have one operator take on a supervisory role to oversee and support other operators who are working alongside the swarm. This special operator could provide advice on areas to explore or manage the robot requests from other operators and reallocate robots among the teams. Such teams would significantly change the nature of inter-human communication, so a user study investigating their effectiveness would be extremely valuable.

Finally, an experiment using real robots that can work alongside human operators is needed. One of the motivating scenarios in this thesis involved operators who are working alongside the swarm to identify tasks and instruct the robots to carry them out. The use of UAVs or UGVs that can traverse different terrains would allow experiments to be carried out in larger outdoor environments. Furthermore, the development of a suitable user interface for outdoor use that enables the operators to intuitively send signals to the robots and communicate with other operators to share robots would truly demonstrate the human-swarm team's potential in the real world.

Bibliography

Abdi, S. S. and Paley, D. A. (2023). Safe Operations of an Aerial Swarm via a Cobot Human Swarm Interface. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1701–1707.

Abu-Aisheh, R., Bronzino, F., Salaün, L., and Watteyne, T. (2022). CARA: Connectivity-Aware Relay Algorithm for Multi-Robot Expeditions. *Sensors*, 22(23):9042.

Alhafnawi, M., Hunt, E. R., Lemaignan, S., O'Dowd, P., and Hauert, S. (2022). MOSAIX: A Swarm of Robot Tiles for Social Human-Swarm Interaction. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 6882–6888.

Alonso-Mora, J., Haegeli Lohaus, S., Leemann, P., Siegwart, R., and Beardsley, P. (2015). Gesture based human - Multi-robot swarm interaction and its application to an interactive display. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5948–5953.

Bashyal, S. and Venayagamoorthy, G. K. (2008). Human swarm interaction for radiation source search and localization. In *2008 IEEE Swarm Intelligence Symposium*, pages 1–8.

Birattari, M., Ligot, A., Bozhinoski, D., Brambilla, M., Francesca, G., Garattoni, L., Garzón Ramos, D., Hasselmann, K., Kegeleirs, M., Kuckling, J., Pagnozzi, F., Roli, A., Salman, M., and Stützle, T. (2019). Automatic Off-Line Design of Robot Swarms: A Manifesto. *Frontiers in Robotics and AI*, 6:59.

Bonabeau, E. (1999). *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press.

Bozhinoski, D. and Birattari, M. (2023). Enhancing the technological maturity of robot swarms. In *2023 IEEE/ACM 5th International Workshop on Robotics Software Engineering (RoSE)*, pages 5–8.

Brambilla, M., Brutschy, A., Dorigo, M., and Birattari, M. (2014). Property-Driven Design for Robot Swarms: A Design Method Based on Prescriptive Modeling and Model Checking. *ACM Transactions on Autonomous and Adaptive Systems*, 9(4):17:1–17:28.

Brambilla, M., Ferrante, E., Birattari, M., and Dorigo, M. (2013). Swarm robotics: A review from the swarm engineering perspective. *Swarm Intelligence*, 7(1):1–41.

Carrillo-Zapata, D., Milner, E., Hird, J., Tzoumas, G., Vardanega, P. J., Sooriyabandara, M., Giuliani, M., Winfield, A. F. T., and Hauert, S. (2020). Mutual Shaping in Swarm Robotics: User Studies in Fire and Rescue, Storage Organization, and Bridge Inspection. *Frontiers in Robotics and AI*, 7:53.

Chandarana, M., Luo, W., Lewis, M., Sycara, K., and Scherer, S. (2018). Decentralized Method for Sub-Swarm Deployment and Rejoining. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 1209–1214. IEEE.

Chandarana, M., Meszaros, E. L., Trujillo, A., and Danette Allen, B. (2017). Natural Language Based Multimodal Interface for UAV Mission Planning. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 61(1):68–72.

Chen, J., Wang, H., Rubenstein, M., and Kress-Gazit, H. (2020). Automatic Control Synthesis for Swarm Robots from Formation and Location-Based High-Level Specifications. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8027–8034.

Christensen, A. L., Grøntved, K. A. R., Oanh Hoang, M.-T., van Berkel, N., Skov, M., Scovill, A., Edwards, G., Geipel, K. R., Dalgaard, L., Lundquist, U. P. S., Constantiou, I., Lehrer, C., and Merritt, T. (2022). The HERD Project: Human-Multi-Robot Interaction in Search & Rescue and in Farming. In *Adjunct Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1–4.

Clarke, E. M. (1999). *Model Checking*. MIT Press.

Couzin, I. D., Krause, J., James, R., Ruxton, G. D., and Franks, N. R. (2002). Collective Memory and Spatial Sorting in Animal Groups. *Journal of Theoretical Biology*, 218(1):1–11.

Crandall, J. W., Anderson, N., Ashcraft, C., Grosh, J., Henderson, J., McClellan, J., Neupane, A., and Goodrich, M. A. (2017). Human-Swarm Interaction as Shared Control: Achieving Flexible Fault-Tolerant Systems. In *Engineering Psychology and Cognitive Ergonomics: Performance, Emotion and Situation Awareness*, Lecture Notes in Computer Science, pages 266–284. Springer International Publishing.

de Queiroz, M. H. and Cury, J. E. R. (2002). Synthesis and implementation of local modular supervisory control for a manufacturing cell. In *Sixth International Workshop on Discrete Event Systems, 2002. Proceedings.*, pages 377–382.

Delmerico, J., Mintchev, S., Giusti, A., Gromov, B., Melo, K., Horvat, T., Cadena, C., Hutter, M., Ijspeert, A., Floreano, D., Gambardella, L. M., Siegwart, R., and Scaramuzza, D. (2019). The current state and future outlook of rescue robotics. *Journal of Field Robotics*, 36(7):1171–1191.

Deneubourg, J. L., Aron, S., Goss, S., and Pasteels, J. M. (1990). The self-organizing exploratory pattern of the Argentine ant. *Journal of Insect Behavior*, 3(2):159–168.

Divband Soorati, M., Clark, J., Ghofrani, J., Tarapore, D., and Ramchurn, S. D. (2021). Designing a User-Centered Interaction Interface for Human–Swarm Teaming. *Drones*, 5(4):131.

Dorigo, M., Floreano, D., Gambardella, L. M., Mondada, F., Nolfi, S., Baaboura, T., Birattari, M., Bonani, M., Brambilla, M., Brutschy, A., Burnier, D., Campo, A., Christensen, A. L., Decugniere, A., Di Caro, G., Ducatelle, F., Ferrante, E., Forster, A., Gonzales, J. M., Guzzi, J., Longchamp, V., Magnenat, S., Mathews, N., Montes de Oca, M., O'Grady, R., Pinciroli, C., Pini, G., Retornaz, P., Roberts, J., Sperati, V., Stirling, T., Stranieri, A., Stutzle, T., Trianni, V., Tuci, E., Turgut, A. E., and Vaussard, F. (2013). Swarmanoid: A Novel Concept for the Study of Heterogeneous Robotic Swarms. *IEEE Robotics Automation Magazine*, 20(4):60–71.

Dorigo, M., Theraulaz, G., and Trianni, V. (2020). Reflections on the future of swarm robotics. *Science Robotics*, 5(49).

Dorigo, M., Tuci, E., Groß, R., Trianni, V., Labella, T. H., Nouyan, S., Ampatzis, C., Deneubourg, J.-L., Baldassarre, G., Nolfi, S., Mondada, F., Floreano, D., and Gambardella, L. M. (2005). The SWARM-BOTS Project. In *Swarm Robotics*, Lecture Notes in Computer Science, pages 31–44. Springer.

Drury, J., Scholtz, J., and Yanco, H. (2003). Awareness in human-robot interactions. In *SMC'03 Conference Proceedings. 2003 IEEE International Conference on Systems, Man and Cybernetics*, volume 1, pages 912–918. IEEE.

Duarte, M., Costa, V., Gomes, J., Rodrigues, T., Silva, F., Oliveira, S. M., and Christensen, A. L. (2016). Evolution of Collective Behaviors for a Real Swarm of Aquatic Surface Robots. *PLOS ONE*, 11(3):e0151834.

Dulce-Galindo, J. A., Santos, M. A., Raffo, G. V., and Pena, P. N. (2019). Autonomous navigation of multiple robots using supervisory control theory. In *2019 18th European Control Conference (ECC)*, pages 3198–3203. IEEE.

Endsley, M. R. (1995). Toward a Theory of Situation Awareness in Dynamic Systems. *Human Factors*, 37(1):32–64.

Francesca, G., Brambilla, M., Brutschy, A., Garattoni, L., Miletitch, R., Podevijn, G., Reina, A., Soleymani, T., Salvaro, M., Pinciroli, C., Mascia, F., Trianni, V., and Birattari, M. (2015). AutoMoDe-Chocolate: Automatic design of control software for robot swarms. *Swarm Intelligence*, 9(2):125–152.

Francesca, G., Brambilla, M., Brutschy, A., Trianni, V., and Birattari, M. (2014). AutoMoDe: A novel approach to the automatic design of control software for robot swarms. *Swarm Intelligence*, 8(2):89–112.

G. Lagoudakis, M., Markakis, E., Kempe, D., Keskinocak, P., Kleywegt, A., Koenig, S., Tovey, C., Meyerson, A., and Jain, S. (2005). Auction-Based Multi-Robot Routing. In *Robotics: Science and Systems I*, volume 5, pages 343–350. Robotics: Science and Systems Foundation.

Gao, F., Cummings, M. L., and Solovey, E. T. (2014). Modeling teamwork in supervisory control of multiple robots. *IEEE Transactions on Human-Machine Systems*, 44(4):441–453.

Garattoni, L. and Birattari, M. (2018). Autonomous task sequencing in a robot swarm. *Science Robotics*, 3(20).

Garzón Ramos, D. and Birattari, M. (2020). Automatic Design of Collective Behaviors for Robots that Can Display and Perceive Colors. *Applied Sciences*, 10(13):4654.

Gerkey, B. P. and Matarić, M. J. (2004). A formal analysis and taxonomy of task allocation in multi-robot systems. *The International Journal of Robotics Research*, 23(9):939–954.

Ghosh, P., Gasparri, A., Jin, J., and Krishnamachari, B. (2019). Robotic Wireless Sensor Networks. In Ammari, H. M., editor, *Mission-Oriented Sensor Networks and Systems: Art and Science: Volume 2: Advances*, Studies in Systems, Decision and Control, pages 545–595. Springer International Publishing, Cham.

Goodrich, M. A., Pendleton, B., Kerman, S., and Sujit, P. (2013). What types of interactions do bio-inspired robot swarms and flocks afford a human? In *Proc. Robot.: Sci. Syst. VIII*, pages 105–112.

Gromov, B., Gambardella, L. M., and Di Caro, G. A. (2016). Wearable multi-modal interface for human multi-robot interaction. In *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 240–245.

Grosh, J. R. and Goodrich, M. A. (2020). Multi-human Management of Robotic Swarms. In *Human-Computer Interaction. Multimodal and Natural Interaction*, Lecture Notes in Computer Science, pages 603–619. Springer International Publishing.

Hamann, H. (2018). *Swarm Robotics: A Formal Approach*. Springer.

Hart, S. G. and Staveland, L. E. (1988). Development of NASA-TLX (task load index): Results of empirical and theoretical research. In *Advances in Psychology*, volume 52, pages 139–183. Elsevier.

Hasbach, J. D. and Bennewitz, M. (2022). The design of self-organizing human–swarm intelligence. *Adaptive Behavior*, 30(4):361–386.

Hasselmann, K. and Birattari, M. (2020). Modular automatic design of collective behaviors for robots endowed with local communication capabilities. *PeerJ Computer Science*, 6:e291.

Hexmoor, H., Mclaughlan, B., and Baker, M. (2005). Swarm Control in Unmanned Aerial Vehicles. In *Proceedings of the International Conference on Artificial Intelligence*, pages 911–917. Army Research Laboratory.

Hogg, E., Hauert, S., Harvey, D., and Richards, A. (2020). Evolving behaviour trees for supervisory control of robot swarms. *Artificial Life and Robotics*, 25(4):569–577.

Hsieh, M. A., Halász, Á., Berman, S., and Kumar, V. (2008). Biologically inspired redistribution of a swarm of robots among multiple sites. *Swarm Intelligence*, 2(2):121–141.

Hussein, A. and Abbass, H. (2018). Mixed Initiative Systems for Human-Swarm Interaction: Opportunities and Challenges. In *2018 2nd Annual Systems Modelling Conference (SMC)*, pages 1–8.

Jang, I., Hu, J., Arvin, F., Carrasco, J., and Lennox, B. (2021). Omnipotent virtual giant for remote human-swarm interaction. In *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, pages 488–494.

Jones, S., Studley, M., Hauert, S., and Winfield, A. (2018). Evolving Behaviour Trees for Swarm Robotics. In Groß, R., Kolling, A., Berman, S., Frazzoli, E., Martinoli, A., Matsuno, F., and Gauci, M., editors, *Distributed Autonomous Robotic Systems*, volume 6, pages 487–501. Springer International Publishing, Cham.

Juhl, D., Warme, D. M., Winter, P., and Zachariasen, M. (2018). The GeoSteiner software package for computing Steiner trees in the plane: An updated computational study. *Mathematical Programming Computation*, 10(4):487–532.

Kalra, N., Zlot, R., Dias, M. B., and Stentz, A. (2005). Market-Based Multirobot Coordination: A Comprehensive Survey and Analysis:. Technical report, Defense Technical Information Center.

Kanakia, A., Klingner, J., and Correll, N. (2016). A Response Threshold Sigmoid Function Model for Swarm Robot Collaboration. In *Distributed Autonomous Robotic Systems*, Springer Tracts in Advanced Robotics, pages 193–206. Springer Japan.

Kapellmann-Zafra, G., Salomons, N., Kolling, A., and Groß, R. (2016). Human-robot swarm interaction with limited situational awareness. In *Swarm Intelligence*, Lecture Notes in Computer Science, pages 125–136. Springer.

Karavas, G. K., Larsson, D. T., and Artemiadis, P. (2017). A hybrid BMI for control of robotic swarms: Preliminary results. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5065–5075.

Kim, L. H., Drew, D. S., Domova, V., and Follmer, S. (2020). User-defined Swarm Robot Control. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, CHI '20, pages 1–13. Association for Computing Machinery.

Kim, L. H. and Follmer, S. (2017). UbiSwarm: Ubiquitous Robotic Interfaces and Investigation of Abstract Motion as a Display. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 1(3):66:1–66:20.

Koenig, S., Tovey, C., Lagoudakis, M., Markakis, V., Kempe, D., Keskinocak, P., Kleywegt, A., Meyerson, A., and Jain, S. (2006). The power of sequential single-item auctions for agent coordination. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1625–1629. AAAI Press.

Kolling, A., Nunnally, S., and Lewis, M. (2012). Towards human control of robot swarms. In *Proceedings of the Seventh Annual ACM/IEEE International Conference on Human-Robot Interaction*, HRI '12, pages 89–96. Association for Computing Machinery.

Kolling, A., Sycara, K., Nunnally, S., and Lewis, M. (2013). Human-swarm interaction: An experimental study of two types of interaction with foraging swarms. *Journal of Human-Robot Interaction*, 2(2):103–129.

Kolling, A., Walker, P., Chakraborty, N., Sycara, K., and Lewis, M. (2016). Human Interaction With Robot Swarms: A Survey. *IEEE Transactions on Human-Machine Systems*, 46(1):9–26.

Kuckling, J., Ubeda Arriaza, K., and Birattari, M. (2020). AutoMoDe-IcePop: Automatic Modular Design of Control Software for Robot Swarms Using Simulated Annealing. In *Artificial Intelligence and Machine Learning*, Communications in Computer and Information Science, pages 3–17, Cham. Springer International Publishing.

Kuckling, J., van Pelt, V., and Birattari, M. (2021). Automatic Modular Design of Behavior Trees for Robot Swarms with Communication Capabilites. In Castillo, P. A. and Jiménez Laredo, J. L., editors, *Applications of Evolutionary Computation*, Lecture Notes in Computer Science, pages 130–145, Cham. Springer International Publishing.

Le Goc, M., Kim, L. H., Parsaei, A., Fekete, J.-D., Dragicevic, P., and Follmer, S. (2016). Zooids: Building Blocks for Swarm User Interfaces. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*, UIST '16, pages 97–109. ACM.

Lee, D., Lu, Q., and Au, T.-C. (2022). Dynamic Robot Chain Networks for Swarm Foraging. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 4965–4971.

Lewis, M., Wang, H., Chien, S.-Y., Scerri, P., Velagapudi, P., Sycara, K., and Kane, B. (2010). Teams organization and performance in multi-human/multi-robot teams. In *2010 IEEE International Conference on Systems, Man and Cybernetics*, pages 1617–1623. IEEE.

Lewis, M., Wang, J., and Scerri, P. (2006). Teamwork Coordination for Realistically Complex Multi Robot Systems. In *NATO Symposium on Human Factors of Uninhabited Military Vehicles as Force Multipliers*, pages 1–12.

Ligot, A., Hasselmann, K., and Birattari, M. (2020). AutoMoDe-Arlequin: Neural Networks as Behavioral Modules for the Automatic Design of Probabilistic Finite-State Machines. In *Swarm Intelligence*, Lecture Notes in Computer Science, pages 271–281, Cham. Springer International Publishing.

Lin, C., Luo, W., and Sycara, K. (2021). Online Connectivity-aware Dynamic Deployment for Heterogeneous Multi-Robot Systems. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8941–8947.

Lopes, Y. K. (2016). *Supervisory Control Theory for Controlling Swarm Robotics Systems*. PhD thesis, University of Sheffield.

Lopes, Y. K., Trenkwalder, S. M., Leal, A. B., Dodd, T. J., and Groß, R. (2016). Supervisory control theory applied to swarm robotics. *Swarm Intelligence*, 10(1):65–97.

Lopes, Y. K., Trenkwalder, S. M., Leal, A. B., Dodd, T. J., and Groß, R. (2017). Probabilistic Supervisory Control Theory (pSCT) Applied to Swarm Robotics. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, pages 1395–1403. International Foundation for Autonomous Agents and Multiagent Systems.

Lopes, Y. K., Trenkwalder, S. M., Leal, A. B., Dodd, T. J., and Groß, R. (2020). Supervisory Control of Robot Swarms Using Public Events. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7193–7199.

Luo, W., Chakraborty, N., and Sycara, K. (2020a). Minimally Disruptive Connectivity Enhancement for Resilient Multi-Robot Teams. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11809–11816.

Luo, W., Yi, S., and Sycara, K. (2020b). Behavior Mixing with Minimum Global and Subgroup Connectivity Maintenance for Large-Scale Multi-Robot Systems. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9845–9851.

MacMillan, J., Entin, E. E., and Serfaty, D. (2004). Communication overhead: The hidden cost of team cognition. In *Team Cognition: Understanding the Factors That Drive Process and Performance.*, pages 61–82. American Psychological Association.

Majcherczyk, N., Jayabalan, A., Beltrame, G., and Pinciroli, C. (2018). Decentralized Connectivity-Preserving Deployment of Large-Scale Robot Swarms. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4295–4302.

Marlow, S. L., Lacerenza, C. N., Paoletti, J., Burke, C. S., and Salas, E. (2018). Does team communication represent a one-size-fits-all approach?: A meta-analysis of team communication and performance. *Organizational Behavior and Human Decision Processes*, 144:145–170.

Mayet, R., Roberz, J., Schmickl, T., and Crailsheim, K. (2010). Antbots: A Feasible Visual Emulation of Pheromone Trails for Swarm Robots. In *Swarm Intelligence*, Lecture Notes in Computer Science, pages 84–94. Springer.

Millard, A. G., Joyce, R., Hilder, J. A., Fleşeriu, C., Newbrook, L., Li, W., McDaid, L. J., and Halliday, D. M. (2017). The Pi-puck extension board: A raspberry Pi interface for the e-puck robot platform. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 741–748.

Miller, C., Funk, H., Wu, P., Goldman, R., Meisner, J., and Chapman, M. (2005). The Playbook™ Approach to Adaptive Automation. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 49(1):15–19.

Min, B.-C., Parasuraman, R., Lee, S., Jung, J.-W., and Matson, E. T. (2018). A directional antenna based leader–follower relay system for end-to-end robot communications. *Robotics and Autonomous Systems*, 101:57–73.

Miyauchi, G., Lopes, Y. K., and Groß, R. (2022). Multi-operator control of connectivity-preserving robot swarms using supervisory control theory. In *2022 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6889–6895.

Miyauchi, G., Lopes, Y. K., and Groß, R. (2023). Sharing the control of robot swarms among multiple human operators: A user study. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 8847–8853.

Molins, P., Stillman, N., and Hauert, S. (2019). Trail Formation Using Large Swarms of Minimal Robots. *Cybernetics and Systems*, 50(8):693–710.

Mondada, F., Bonani, M., Raemy, X., Pugh, J., Cianci, C., Klaptocz, A., Magnenat, S., Zufferey, J.-C., Floreano, D., and Martinoli, A. (2009). The e-puck, a robot designed for education in engineering. *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions*, 1(1):59–65.

Nagavalli, S., Chandarana, M., Sycara, K., and Lewis, M. (2017). Multi-Operator Gesture Control of Robotic Swarms Using Wearable Devices. In *Proceedings of the Tenth International Conference on Advances in Computer-Human Interactions*. IARIA.

Nagavalli, S., Chien, S.-Y., Lewis, M., Chakraborty, N., and Sycara, K. (2015). Bounds of Neglect Benevolence in Input Timing for Human Interaction with Robotic Swarms. In *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 197–204.

Nagi, J., Giusti, A., Gambardella, L. M., and Di Caro, G. A. (2014). Human-swarm interaction using spatial gestures. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3834–3841.

Nehaniv, C., Dautenhahn, K., Kubacki, J., Haegele, M., Parlitz, C., and Alami, R. (2005). A methodological approach relating the classification of gesture to identification of human intent in the context of human-robot interaction. In *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005.*, pages 371–377.

Nouyan, S., Campo, A., and Dorigo, M. (2008). Path formation in a robot swarm. *Swarm Intelligence*, 2(1):1–23.

Nouyan, S., Groß, R., Bonani, M., Mondada, F., and Dorigo, M. (2009). Teamwork in Self-Organized Robot Colonies. *IEEE Transactions on Evolutionary Computation*, 13(4):695–711.

Nouyan, S., Groß, R., Dorigo, M., Bonani, M., and Mondada, F. (2006). Group transport along a robot chain in a self-organised robot colony. *Proc. of the 9th Int. Conf. on Intelligent Autonomous Systems*, pages 433–442.

Nunnally, S., Walker, P., Lewis, M., Chakraborty, N., and Sycara, K. (2013). Using Haptic Feedback in Human Robotic Swarms Interaction. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 57(1):1047–1051.

Olsen, D. R. and Goodrich, M. A. (2003). Metrics for Evaluating Human-Robot Interactions. *Proceedings of PERMIS*, 2003.

Olsen, D. R. and Wood, S. B. (2004). Fan-out: Measuring human control of multiple robots. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '04, pages 231–238. Association for Computing Machinery.

Patel, J. and Pinciroli, C. (2020). Improving human performance using mixed granularity of control in multi-human multi-robot interaction. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*, pages 1135–1142.

Patel, J., Sonar, P., and Pinciroli, C. (2022). On multi-human multi-robot remote interaction: A study of transparency, inter-human communication, and information loss in remote interaction. *Swarm Intelligence*, 16(2):107–142.

Patel, J., Xu, Y., and Pinciroli, C. (2019). Mixed-granularity human-swarm interaction. In *2019 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1059–1065.

Payton, D., Daily, M., Estowski, R., Howard, M., and Lee, C. (2001). Pheromone Robotics. *Autonomous Robots*, 11(3):319–324.

Pinciroli, C., O'Grady, R., Christensen, A. L., and Dorigo, M. (2009). Self-organised recruitment in a heteregeneous swarm. In *2009 International Conference on Advanced Robotics*, pages 1–8.

Pinciroli, C., O'Grady, R., Christensen, A. L., and Dorigo, M. (2010). Coordinating Heterogeneous Swarms through Minimal Communication among Homogeneous Subswarms. In *Swarm Intelligence*, Lecture Notes in Computer Science, pages 558–559. Springer.

Pinciroli, C., Trianni, V., O'Grady, R., Pini, G., Brutschy, A., Brambilla, M., Mathews, N., Ferrante, E., Di Caro, G., Ducatelle, F., Birattari, M., Gambardella, L. M., and Dorigo, M. (2012). ARGoS: A modular, parallel, multi-engine simulator for multi-robot systems. *Swarm Intelligence*, 6(4):271–295.

Pinheiro, L. P., Lopes, Y. K., Leal, A. B., and Rosso Junior, R. S. U. (2015). Nadzoru: A Software Tool for Supervisory Control of Discrete Event Systems. *IFAC-PapersOnLine*, 48(7):182–187.

Pini, G., Brutschy, A., Birattari, M., and Dorigo, M. (2009). Interference reduction through task partitioning in a robotic swarm. In *Sixth International Conference on Informatics in Control, Automation and Robotics–ICINCO*, pages 52–59.

Podevijn, G., O'Grady, R., Mathews, N., Gilles, A., Fantini-Hauwel, C., and Dorigo, M. (2016). Investigating the effect of increasing robot group sizes on the human psychophysiological state in the context of human–swarm interaction. *Swarm Intelligence*, 10(3):193–210.

Podevijn, G., O'Grady, R., Nashed, Y. S. G., and Dorigo, M. (2014). Gesturing at subswarms: Towards direct human control of robot swarms. In Natraj, A., Cameron, S., Melhuish, C., and Witkowski, M., editors, *Towards Autonomous Robotic Systems*, pages 390–403, Berlin, Heidelberg. Springer Berlin Heidelberg.

Pourmehr, S., Monajjemi, V. M., Vaughan, R., and Mori, G. (2013). "You two! Take off!": Creating, modifying and commanding groups of robots using face engagement and indirect speech in voice commands. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 137–142.

Prabhakar, A., Abraham, I., Taylor, A., Schlafly, M., Popovic, K., Diniz, G., Teich, B., Simidchieva, B., Clark, S., and Murphey, T. (2020). Ergodic Specifications for Flexible Swarm Control: From User Commands to Persistent Adaptation. In *Robotics: Science and Systems XVI*. Robotics: Science and Systems Foundation.

Prömel, H. J. and Steger, A. (2002). *The Steiner Tree Problem: A Tour through Graphs, Algorithms, and Complexity*. Vieweg+Teubner Verlag, Wiesbaden.

Ramadge, P. J. and Wonham, W. M. (1987). Supervisory Control of a Class of Discrete Event Processes. *SIAM Journal on Control and Optimization*, 25(1):206–230.

Rico, R., Sánchez-Manzanares, M., Gil, F., and Gibson, C. (2008). Team implicit coordination processes: A team knowledge-based approach. *Academy of management review*, 33(1):163–184.

Rockbach, J. D. and Bennewitz, M. (2022). Robot swarms as embodied extensions of humans. *IOP Conference Series: Materials Science and Engineering*, 1261(1):012015.

Roundtree, K. A., Cody, J. R., Leaf, J., Demirel, H. O., and Adams, J. A. (2021). Human-collective visualization transparency. *Swarm Intelligence*, 15(3):237–286.

Roundtree, K. A., Goodrich, M. A., and Adams, J. A. (2019). Transparency: Transitioning From Human–Machine Systems to Human-Swarm Systems. *Journal of Cognitive Engineering and Decision Making*, 13(3):171–195.

Roundtree, K. A., Manning, M. D., and Adams, J. A. (2018). Analysis of Human-Swarm Visualizations. *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, 62(1):287–291.

Saez-Pons, J., Alboul, L., Penders, J., and Nomdedeu, L. (2010). Multi-robot team formation control in the GUARDIANS project. *Industrial Robot: An International Journal*, 37(4):372–383.

Şahin, E. (2004). Swarm Robotics: From Sources of Inspiration to Domains of Application. In *Swarm Robotics*, Lecture Notes in Computer Science, pages 10–20. Springer.

Salman, M., Ligot, A., and Birattari, M. (2019). Concurrent design of control software and configuration of hardware for robot swarms under economic constraints. *PeerJ Computer Science*, 5:e221.

Schneider, E., Sklar, E. I., and Parsons, S. (2016). Evaluating Multi-Robot Teamwork in Parameterised Environments. In Alboul, L., Damian, D., and Aitken, J. M., editors, *Towards Autonomous Robotic Systems*, Lecture Notes in Computer Science, pages 301–313, Cham. Springer International Publishing.

Serpiva, V., Karmanova, E., Fedoseev, A., Perminov, S., and Tsetserukou, D. (2021). SwarmPaint: Human-Swarm Interaction for Trajectory Generation and Formation Control by DNN-based Gesture Interface. In *2021 International Conference on Unmanned Aircraft Systems (ICUAS)*, pages 1055–1062.

Sheridan, T. B. (2002). *Humans and Automation: System Design and Research Issues.* Wiley Series in System Engineering and Management: HFES Issues in Human Factors and Ergonomics Series. Human Factors and Ergonomics Society.

Smith (1980). The Contract Net Protocol: High-Level Communication and Control in a Distributed Problem Solver. *IEEE Transactions on Computers*, C-29(12):1104–1113.

Soma, K., Khateri, K., Pourgholi, M., Montazeri, M., Sabattini, L., and Beltrame, G. (2023). A Complete Set of Connectivity-aware Local Topology Manipulation Operations for Robot Swarms. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5522–5529.

Talamali, M. S., Bose, T., Haire, M., Xu, X., Marshall, J. A. R., and Reina, A. (2020). Sophisticated collective foraging with minimalist agents: A swarm robotics test. *Swarm Intelligence*, 14(1):25–56.

Tarapore, D., Groß, R., and Zauner, K.-P. (2020). Sparse Robot Swarms: Moving Swarms to Real-World Applications. *Frontiers in Robotics and AI*, 7.

Taylor, R. M. (1990). Situation awareness rating technique (SART): The development of a tool for aircrew systems design. In *Situational Awareness in Aerospace Operations (AGARD-CP-478)*, pages 3/1–3/17. NATO-AGARD.

Tsykunov, E., Labazanova, L., Tleugazy, A., and Tsetserukou, D. (2018). SwarmTouch: Tactile Interaction of Human with Impedance Controlled Swarm of Nano-Quadrotors. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4204–4209.

Turgut, A. E., Çelikkanat, H., Gökçe, F., and Şahin, E. (2008). Self-organized flocking in mobile robot swarms. *Swarm Intelligence*, 2(2):97–120.

Varadharajan, V. S., Adams, B., and Beltrame, G. (2019). The Unbroken Telephone Game: Keeping Swarms Connected. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '19, pages 2241–2243, Richland, SC. International Foundation for Autonomous Agents and Multiagent Systems.

Varadharajan, V. S., St-Onge, D., Adams, B., and Beltrame, G. (2020). Swarm relays: distributed self-healing ground-and-air connectivity chains. *IEEE Robotics and Automation Letters (RA-L)*, 5(4):5347–5354.

Vásárhelyi, G., Virágh, C., Somorjai, G., Nepusz, T., Eiben, A. E., and Vicsek, T. (2018). Optimized flocking of autonomous drones in confined environments. *Science Robotics*, 3(20).

Walker, P., Amraii, S. A., Chakraborty, N., Lewis, M., and Sycara, K. (2014a). Human control of robot swarms with dynamic leaders. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1108–1113.

Walker, P., Amraii, S. A., Lewis, M., Chakraborty, N., and Sycara, K. (2013). Human Control of Leader-Based Swarms. In *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pages 2712–2717.

Walker, P., Amraii, S. A., Lewis, M., Chakraborty, N., and Sycara, K. (2014b). Control of swarms with multiple leader agents. In *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3567–3572.

Walker, P., Lewis, M., and Sycara, K. (2016). The effect of display type on operator prediction of future swarm states. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 2521–2526.

Walker, P., Nunnally, S., Lewis, M., Kolling, A., Chakraborty, N., and Sycara, K. (2012). Neglect benevolence in human control of swarms in the presence of latency. In *2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 3009–3014.

Webster, M., Western, D., Araiza-Illan, D., Dixon, C., Eder, K., Fisher, M., and Pipe, A. G. (2020). A corroborative approach to verification and validation of human–robot teams. *The International Journal of Robotics Research*, 39(1):73–99.

Whetten, J. M., Goodrich, M. A., and Guo, Y. (2010). Beyond robot fan-out: Towards multi-operator supervisory control. In *2010 IEEE International Conference on Systems, Man and Cybernetics*, pages 2008–2015. IEEE.

Wickens, C. D., Lee, J. D., Liu, Y., and Gordon Becker, S. E. (2004). *An introduction to human factors engineering*. Pearson Education, Upper Saddle River, NJ: Prentice Hall., 2nd ed., international ed. edition.

Wilson, J., Chance, G., Winter, P., Lee, S., Milner, E., Abeywickrama, D., Windsor, S., Downer, J., Eder, K., Ives, J., and Hauert, S. (2023). Trustworthy Swarms. In *Proceedings of the First International Symposium on Trustworthy Autonomous Systems*, TAS '23, pages 1–11, New York, NY, USA. Association for Computing Machinery.

Winfield, A. F., Sa, J., Fernández-Gago, M.-C., Dixon, C., and Fisher, M. (2005). On Formal Specification of Emergent Behaviours in Swarm Robotic Systems. *International Journal of Advanced Robotic Systems*, 2(4):39.

Yanco, H. and Drury, J. (2004). Classifying human-robot interaction: An updated taxonomy. In *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No.04CH37583)*, volume 3, pages 2841–2846 vol.3.

Yi, S., Luo, W., and Sycara, K. (2021). Distributed topology correction for flexible connectivity maintenance in multi-robot systems. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 8874–8880. IEEE.

Zahadat, P. and Schmickl, T. (2016). Division of labor in a swarm of autonomous underwater robots by improved partitioning social inhibition. *Adaptive Behavior*, 24(2):87–101.

Zhang, L. and Vaughan, R. (2016). Optimal robot selection by gaze direction in multi-human multi-robot interaction. In *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5077–5083.

Zhong, H. and Wonham, W. (1990). On the consistency of hierarchical supervision in discrete-event systems. *IEEE Transactions on Automatic Control*, 35(10):1125–1134.

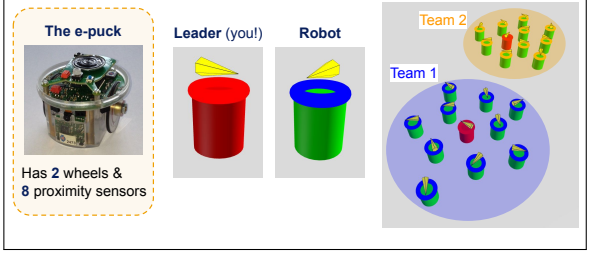
Appendix A

User Study Presentation Slides

This appendix contains the induction slides presented to the participants at the beginning of an experiment outlined in Chapter 4.

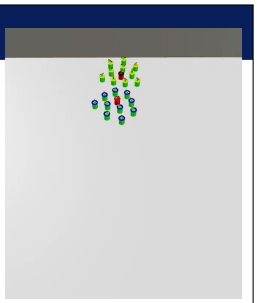
Participants were presented with slightly different presentation slides depending on the communication type they were allocated to and the order they experienced the robot-sharing condition. The particular slides listed in this appendix were presented to participants who were allocated to the *indirect (IND)* communication type and completed the experiments in the order: *no-robot-sharing (NRS)*, then *robot-sharing (RS)*.

The Robots



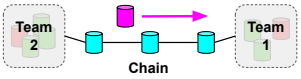
Robot Chain (1)

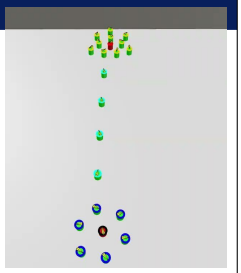
- Robots keep the two teams **connected**
- **Moving away** from the other team will make some of your **robots leave the team**
- **Moving closer** to the other team will make **robots join your team**



Robot Chain (2)

- You can **send** your robots to the other team
- Robots travel along the chain

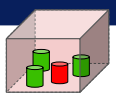


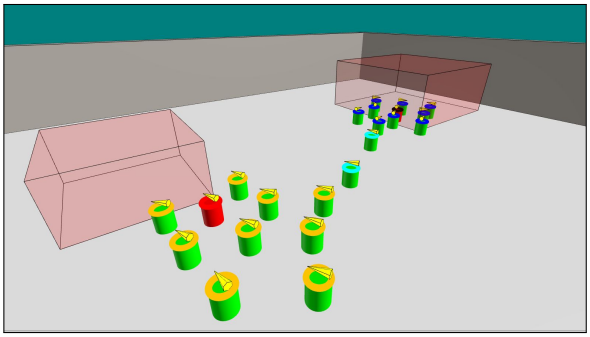


The Task

- Tasks represented by **red zones**
  - Need a specified number of robots to complete each one
  - Size of zone is proportional to the number of followers needed
- **Aim:** **Work together to score as many points as possible!**
  - Score increases by the size of the task you complete

Guide the robots *to the task zones*
Share robots *with each other if needed*



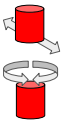


How to Move

Hold Arrow keys or WASD to move

Direction (forward/backward) : Up / Down key or W / S

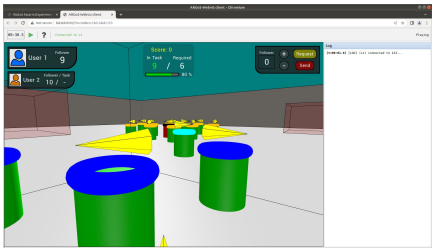
Rotation (left/right) : Left / Right key or A / D



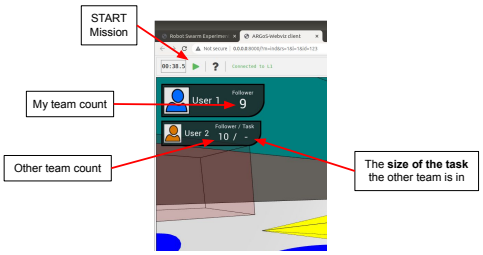
Combine **Direction** and **Rotation** to turn

E.g. and :

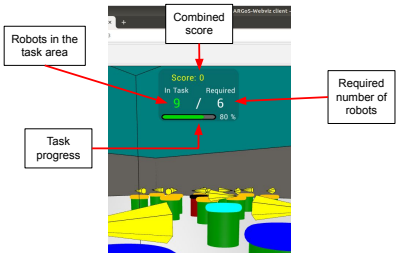
User Interface



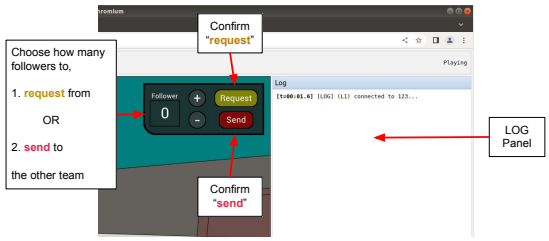
User Interface (Team Info)



User Interface (Task Info)

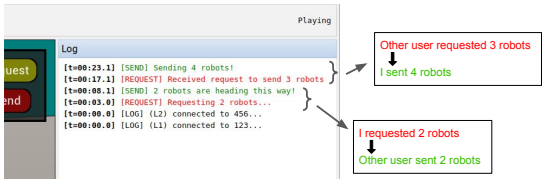


User Interface (Request / Send robots)



User Interface (Request / Send robots)

Example:



Training

Practice with a computer

1. **Move** to the closest task
   - Keep the robots inside the box to complete the task!
2. **Request** robots
   - If there is not enough followers
3. **Send** robots
   - Try sending your followers to the other team

ANY QUESTIONS?

Schedule:

1. Trial 1 (10 minutes)
2. Questionnaire + break
3. Trial 2 - (10 minutes)
4. Questionnaire

Trial 1 (10 min)

Goal: Obtain as many points as possible!

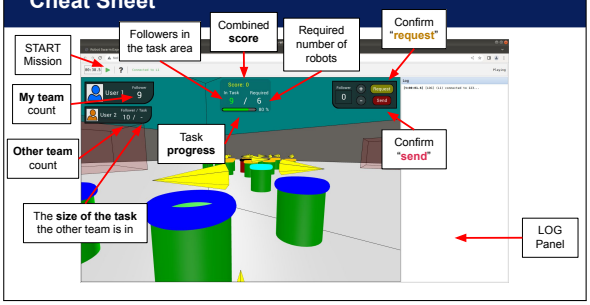
Restriction: Sending robots is disabled

Trial 2 (10 min)

Goal: Obtain as many points as possible!

Restriction: None

Cheat Sheet



Thank you for your help!

Appendix B

User Study Questionnaire

This appendix contains the questionnaires provided to the participants in the user study outlined in Chapter 4. The participants each completed three questionnaires during the experiment.

The *Preliminary Questionnaire* (1 page) asked about the participant's background and any previous gaming experiences.

The *Post-Trial Questionnaire (1)* (2 pages), and *Post-trial Questionnaire (2)* (3 pages) asked the participants to rate their experience upon completing each trial. After the second trial, the participants were also asked to answer additional questions related to the overall experience (found on the last page of Post-trial Questionnaire (2)).

Preliminary Questionnaire

Sharing the Control of Robot Swarms User Study

Participant ID: ☐

Instruction: Mark your answer.

1. What is your age?

   ☐ Under 20 ☐ 40-49 ☐ Prefer not to say

   ☐ 20-29 ☐ 50-59

   ☐ 30-39 ☐ 60 & over

2. To which gender identity do you most identify?

   ☐ Male ☐ Self-describe: _____

   ☐ Female ☐ Prefer not to say

3. How often do you play video games (including console, PC, mobile)?

   ☐ I play on a daily basis

   ☐ I play a few times a week

   ☐ I play a few times a month

   ☐ I don't play video games regularly but I used to play regularly in the past

   ☐ I don't play video games regularly and I have not played regularly in the past

4. If you play or used to play video games, what is your favourite genre? Please select all that apply.

   ☐ Action ☐ Role-playing game ☐ Puzzle

   ☐ Shooter ☐ Simulation ☐ Rhythm

   ☐ Fighting ☐ Strategy ☐ Other: _____

   ☐ Sports / Racing ☐ MOBA

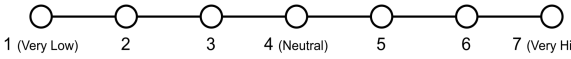
Post-Trial Questionnaire (1)

Sharing the Control of Robot Swarms User Study

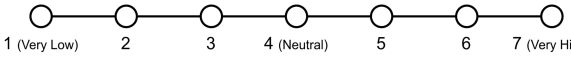
Participant ID: ____

Instruction: Mark one answer.

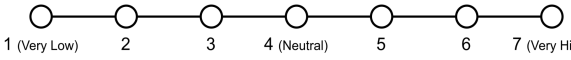
1. **Mental Demand:** How mentally demanding was the task?



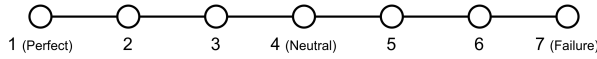
2. **Physical Demand:** How physically demanding was the task?



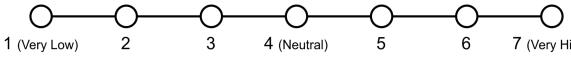
3. **Temporal Demand:** How hurried or rushed was the pace of the task?



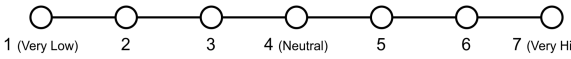
4. **Performance:** How successful were you in accomplishing what you were asked to do?



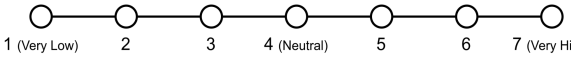
5. **Effort:** How hard did you have to work to accomplish your level of performance?



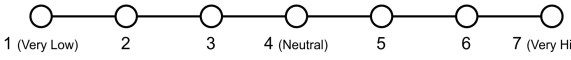
6. **Frustration:** How insecure, discouraged, irritated, stressed, and annoyed were you?



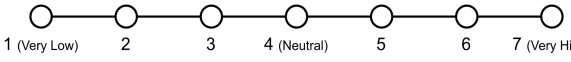
7. **Instability of Situation:** How changeable is the situation? Is the situation highly unstable and likely to change suddenly (High) or is it very stable and straightforward (Low)?



8. **Complexity of Situation:** How complicated is the situation? Is it complex with many interrelated components (High) or is it simple and straightforward (Low)?

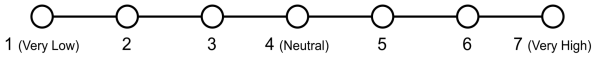


9. **Variability of Situation:** How many variables are changing within the situation? Are there a large number of factors varying (High) or are there very few variables changing (Low)?

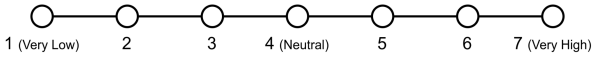


Flip over to next page

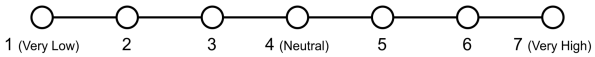
10. **Arousal:** How aroused are you in the situation? Are you alert and ready for activity (High) or do you have a low degree of alertness (Low)?



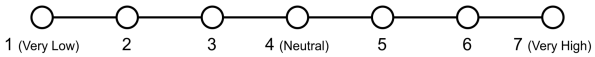
11. **Concentration of Attention:** How much are you concentrating on the situation? Are you concentrating on many aspects of the situation (High) or focused on only one (Low)?



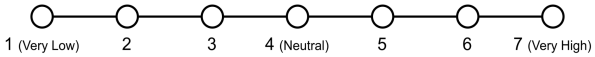
12. **Division of Attention:** How much is your attention divided in the situation? Are you concentrating on many aspects of the situation (High) or focused on only one (Low)?



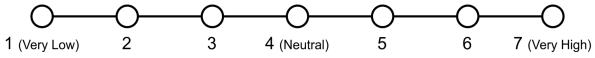
13. **Spare Mental Capacity:** How much mental capacity do you have to spare in the situation? Do you have sufficient to attend to many variables (High) or nothing to spare at all (Low)?



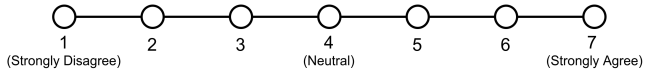
14. **Information Quantity:** How much information have you gained about the situation? Have you received and understood a great deal of knowledge (High) or very little (Low)?



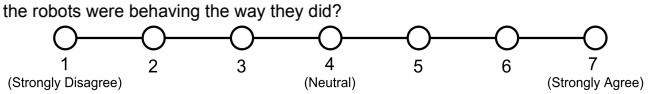
15. **Familiarity with Situation:** How familiar are you with the situation? Do you have a great deal of relevant experience (High) or is it a new situation (Low)?



16. Did you understand what your **teammate was doing** at any particular time? Were you able to understand why your teammate was taking a certain action?



17. Did you understand what the **robots were doing** at any particular time? Were you sure how and why the robots were behaving the way they did?



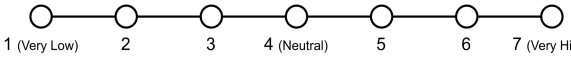
Post-Trial Questionnaire (2)

Sharing the Control of Robot Swarms User Study

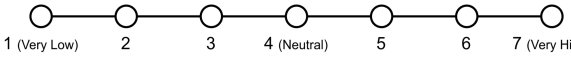
Participant ID: ____

Instruction: Mark one answer.

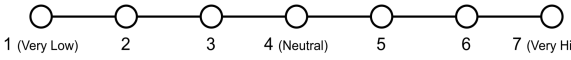
1. **Mental Demand:** How mentally demanding was the task?



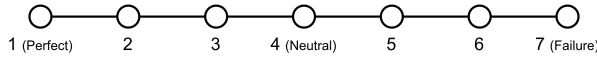
2. **Physical Demand:** How physically demanding was the task?



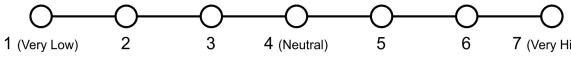
3. **Temporal Demand:** How hurried or rushed was the pace of the task?



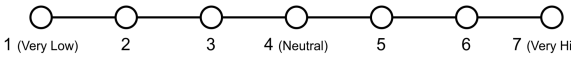
4. **Performance:** How successful were you in accomplishing what you were asked to do?



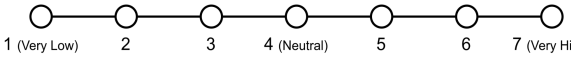
5. **Effort:** How hard did you have to work to accomplish your level of performance?



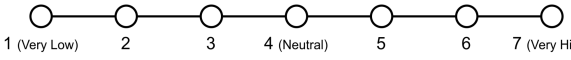
6. **Frustration:** How insecure, discouraged, irritated, stressed, and annoyed were you?



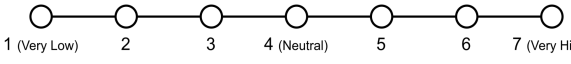
7. **Instability of Situation:** How changeable is the situation? Is the situation highly unstable and likely to change suddenly (High) or is it very stable and straightforward (Low)?



8. **Complexity of Situation:** How complicated is the situation? Is it complex with many interrelated components (High) or is it simple and straightforward (Low)?

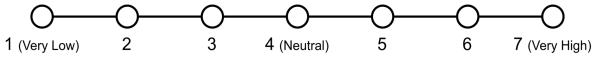


9. **Variability of Situation:** How many variables are changing within the situation? Are there a large number of factors varying (High) or are there very few variables changing (Low)?

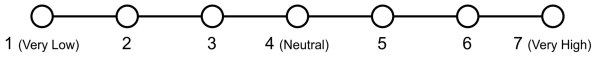


Flip over to next page

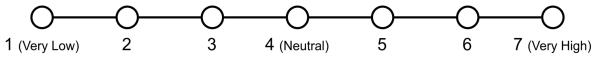
10. **Arousal:** How aroused are you in the situation? Are you alert and ready for activity (High) or do you have a low degree of alertness (Low)?



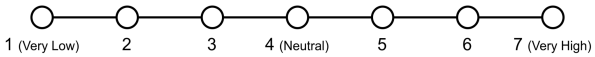
11. **Concentration of Attention:** How much are you concentrating on the situation? Are you concentrating on many aspects of the situation (High) or focused on only one (Low)?



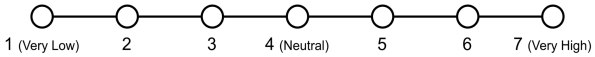
12. **Division of Attention:** How much is your attention divided in the situation? Are you concentrating on many aspects of the situation (High) or focused on only one (Low)?



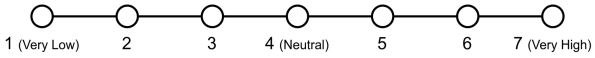
13. **Spare Mental Capacity:** How much mental capacity do you have to spare in the situation? Do you have sufficient to attend to many variables (High) or nothing to spare at all (Low)?



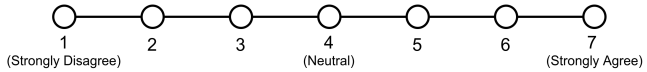
14. **Information Quantity:** How much information have you gained about the situation? Have you received and understood a great deal of knowledge (High) or very little (Low)?



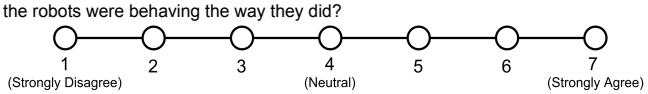
15. **Familiarity with Situation:** How familiar are you with the situation? Do you have a great deal of relevant experience (High) or is it a new situation (Low)?



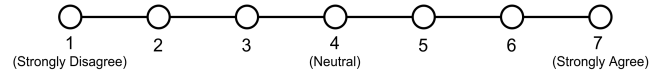
16. Did you understand what your **teammate was doing** at any particular time? Were you able to understand why your teammate was taking a certain action?



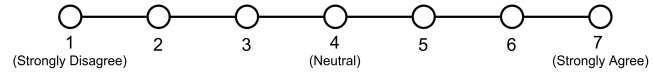
17. Did you understand what the **robots were doing** at any particular time? Were you sure how and why the robots were behaving the way they did?



18. Was the information provided by the interface **clear to understand**?



19. Did you find the **ability to share robots** with your teammate useful in completing the tasks?



20. What was your **strategy** for completing the tasks? Was there anything in particular that you paid attention to during the trial?

21. If you have any additional comments, please write them here.

Thank you for taking part in the study!