Route Planning and Operator Allocation in Robot Fleets

by

Abhinav Dahiya

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Doctor of Philosophy
in
Electrical and Computer Engineering

Waterloo, Ontario, Canada, 2023

Examining Committee Membership

The following served on the Examining Committee for this thesis. The decision of the Examining Committee is by majority vote.

External Examiner:	Katherine Driggs-Campbell Assistant Professor, Dept. of Electrical and Computer Engineering, University of Illinois at Urbana-Champaign
Supervisor:	Stephen L. Smith Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Internal Member:	Kerstin Dautenhahn Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Internal Member:	Mahesh Tripunitara Professor, Dept. of Electrical and Computer Engineering, University of Waterloo
Internal-External Member:	Mark Hancock Professor, Dept. of Management Sciences, University of Waterloo

Author's Declaration

Statement of Contributions

This thesis comprises four published manuscripts, each of which was made possible through collaborative efforts with multiple researchers. Throughout these publications, Professor Stephen Smith served as the principal investigator, overseeing all research activities. The specific contributions to each manuscript are as follows:

Chapter 2: The work presented in this chapter is part of the following publication.

- Abhinav Dahiya, Alexander M. Aroyo, Kerstin Dautenhahn, and Stephen L. Smith. "A survey of multi-agent Human–Robot Interaction systems." Robotics and Autonomous Systems 161 (2023): 104335.

For this publication, I was the main author of the work, and Dr. Alexander Aroyo and Prof. Kerstin Dautenhahn from the Social and Intelligent Robotics Research Laboratory at the University of Waterloo provided their valuable feedback through the writing process.

Chapter 4: The work presented in this chapter is based on the following publication.

- Abhinav Dahiya and Stephen L. Smith, "Optimal Robot Path Planning In a Collaborative Human-Robot Team with Intermittent Human Availability," in IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), Busan, South Korea, 2023.

Chapter 5: The work presented in this chapter is based on the following publication.

- Abhinav Dahiya, Nima Akbarzadeh, Aditya Mahajan and Stephen L. Smith, "Scalable Operator Allocation for Multirobot Assistance: A Restless Bandit Approach," in IEEE Transactions on Control of Network Systems, vol. 9, no. 3, pp. 1397-1408, Sept. 2022.

This publication is the result of a collaboration with Nima Akbarzadeh and Prof. Aditya Mahajan from McGill University, Montreal. I took the lead in developing the mathematical background and solution to the problem at hand, and the simulation setup used for evaluation. We collaborated in establishing the problem setup and refining the proposed solution approach.

Chapter 6: The work presented in this chapter is based on the following publication.

- Abhinav Dahiya, Yifan Cai, Oliver Schneider and Stephen Smith, "On the Impact of Interruptions During Multi-Robot Supervision Tasks," in IEEE International Conference on Robotics and Automation (ICRA), London, United Kingdom, 2023.

This publication is the result of a collaboration with my labmate Yifan Cai and Prof. Oliver Schneider from the Department of Management Sciences at the University of Waterloo. I took the lead in defining the research questions and developing the user study. Yifan assisted in carrying out the study and data analysis. Prof. Schneider collaborated on refining the details of the user study and provided feedback throughout the project.

Other Publications: I have also worked on the following publications that are not presented in this thesis:

- Yifan Cai, Abhinav Dahiya, Nils Wilde and Stephen L. Smith, "Scheduling Operator Assistance for Shared Autonomy in Multi-Robot Teams," IEEE 61st Conference on Decision and Control (CDC), Cancun, Mexico, 2022, pp. 3997-4003.
- Ali Noormohammadi, Abhinav Dahiya, Alexander Mois Aroyo, Stephen L. Smith, and Kerstin Dautenhahn. "The effect of robot decision making on human perception of a robot in a collaborative task-a remote study," International Conference on Human-Agent Interaction, pp. 423-427. 2021.
- Pamela Carreno-Medrano, Abhinav Dahiya, Stephen L. Smith and Dana Kulić, "Incremental Estimation of Users' Expertise Level," IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), New Delhi, India, 2019, pp. 1-8.

Abstract

In this thesis, we address various challenges related to optimal planning and task allocation in a robot fleet supervised by remote human operators. The overarching goal is to enhance the performance and efficiency of the robot fleets by planning routes and scheduling operator assistance while accounting for limited human availability. The thesis consists of three main problems, each of which focuses on a specific aspect of the system.

The first problem pertains to optimal planning for a robot in a collaborative human-robot team, where the human supervisor is intermittently available to assist the robot to complete its tasks faster. Specifically, we address the challenge of computing the fastest route between two configurations in an environment with time constraints on how long the robot can wait for assistance at intermediate configurations. We consider the application of robot navigation in a city environment, where different routes can have distinct speed limits and different time constraints on how long a robot is allowed to wait. Our proposed approach utilizes the concepts of budget and critical departure times, enabling optimal solution and enhanced scalability compared to existing methods. Extensive comparisons with baseline algorithms on a city road network demonstrate its effectiveness and ability to achieve high-quality solutions. Furthermore, we extend the problem to the multi-robot case, where the challenge lies in prioritizing robots when multiple service requests arrive simultaneously. To address this challenge, we present a greedy algorithm that efficiently prioritizes service requests in a batch and has a remarkably good performance compared to the optimal solution.

The next problem focuses on allocating human operators to robots in a fleet, considering each robot's specified route and the potential for failures and getting stuck. Conventional techniques used to solve such problems face scalability issues due to exponential growth of state and action spaces with the number of robots and operators. To overcome these, we derive conditions for a technical requirement called indexability, thereby enabling the use of the Whittle index heuristic. Our key insight is to leverage the structure of the value function of individual robots, resulting in conditions that can be easily verified separately for each state of each robot. We apply these conditions to two types of transitions commonly seen in supervised robot fleets. Through numerical simulations, we demonstrate the efficacy of Whittle index policy as a near-optimal scalable approach that outperforms existing scalable methods.

Finally, we investigate the impact of interruptions on human supervisors overseeing a fleet of robots. Human supervisors in such systems are primarily responsible for monitoring robots, but can also be assigned with secondary tasks. These tasks can act as interruptions and can be categorized as either intrinsic, i.e., being directly related to the monitoring task, or extrinsic, i.e., being unrelated. Through a user study involving 39 participants, the findings reveal that task performance remains relatively unaffected by interruptions, and is primarily dependent on the number of robots being monitored. However, extrinsic interruptions led to a significant increase in perceived workload, creating challenges in switching between tasks. These results highlight the importance of managing user workload by limiting extrinsic interruptions in such supervision systems.

Overall, this thesis contributes to the field of robot planning and operator allocation in collaborative human-robot teams. By incorporating human assistance, addressing scalability challenges, and understanding the impact of interruptions, we aim to enhance the performance and usability of robot fleets. Our work introduces optimal planning methods and efficient allocation strategies, empowering the seamless operation of robot fleets in real-world scenarios. Additionally, we provide valuable insights into user workload, shedding light on the interactions between humans and robots in such systems. We hope that our research promotes the widespread adoption of robot fleets and facilitates their integration into various domains, ultimately driving advancements in the field.

Acknowledgements

I extend my sincere gratitude to Professor Stephen Smith for his invaluable support, motivation, and guidance throughout my Ph.D. program. His expertise and mentorship have been instrumental in shaping the direction of my research and fostering my academic growth. I would also like to express my appreciation to the members of the examination committee for their time and participation in evaluating my work. Lastly, I am deeply grateful for the friendship of those around me, and the unwavering love and support of my family. Their presence has been a constant source of encouragement and inspiration during this journey.

Dedication

I dedicate this dissertation to my family. Thank you for your love and support.

Table of Contents

List of Figures

True wisdom lies not only in finding the shortest path but also in embracing the most rewarding journey.

Unknown

Chapter 1

Introduction

With the progression of technology, the concept of robot fleets has gained substantial traction, finding practical applications across a spectrum of industries. From robotic delivery systems [1], data collection and surveillance [2, 3], to transportation solutions [4–6], these fleets have the potential to greatly improve system efficiency as autonomy continues to advance. However, despite their autonomous capabilities, human supervision of robot fleets remains essential to ensure safety, compliance with regulations and overall robustness. This paradigm of human-robot interaction holds great promise for improving productivity, efficiency, and safety in sectors such as manufacturing, logistics, agriculture, and search and rescue operations [7–10]. However, it also presents unique challenges and opportunities that require careful consideration and resolution [11, 12].

This thesis is dedicated to addressing several of the relevant challenges associated with supervised robot fleets. Specifically, we delve into the planning of routes/missions for robots within a fleet, where they operate autonomously while being overseen by a limited number of human operators in case they need assistance. A critical aspect of this system is the possible intermittent availability of human operators, who may have other responsibilities or be occupied with supervising other robots in the system. This intermittency poses a challenge as operators cannot be continuously present to supervise or assist individual robots. Consequently, it becomes crucial to develop planning strategies that can accommodate the availability of operators and ensure effective supervision of the fleet. Furthermore, we recognize the scenario where service requests for multiple robots arrive simultaneously, and it becomes important to decide the optimal order in which their routes should be planned. This decision is significant as planning for one robot changes operator availability for the later robots. Instead of arbitrarily choosing robots from the batch, it may be helpful to optimally prioritize the robots to ensure efficient utilization of human assistance and improve overall fleet performance.

Once the robots' routes are planned and the robots begin their designated routes, it is crucial to proactively anticipate potential trouble scenarios. Despite thorough planning, unforeseen circumstances such as robot failures or encountered obstacles may require human intervention. In such cases, it becomes crucial to establish a system for prioritizing which robots should receive assistance and when. This prioritization can be based on factors such as the severity of the situation, the criticality of the robot's task, or the overall impact on system performance. Additionally, taking proactive measures, such as allocating operators in advance to specific robots during potential trouble spots, can further enhance the system's responsiveness. By strategically allocating human assistance to individual robots in the fleet, we can effectively mitigate risks, and ensure prompt and effective human intervention when necessary.

In addition to addressing the challenges of operator allocation and planning, it is crucial to recognize and understand the human factors associated with the system of supervised robot fleets. While optimizing robot behavior and coordination remains essential, a comprehensive understanding of factors related to the supervision of multiple robots is equally vital. A key consideration in this context is the occurrence of different types of interruptions, which can have a significant impact on operator performance and workload, thereby influencing the overall effectiveness of the robot fleet. Examining the impact of interruptions within these systems allows us to acquire valuable insights into the dynamics between human operators and the robots under their supervision. By studying and addressing these human factors, we can enhance the design and implementation of supervised robot fleets, thereby improving their overall efficiency and usability.

For the majority of this thesis, we will primarily refer to and focus on the application of a fleet of robots navigating through a city environment as an illustrative example. In this scenario, each robot is tasked with traversing a set of waypoints to navigate from a given start location to a designated goal location. This example serves as a concrete and practical demonstration of the principles and methodologies developed throughout our research, allowing for a comprehensive exploration of the challenges and solutions in this context.

However, it is important to emphasize that the problem formulation we consider in this thesis extends beyond the example of a robot fleet in a city environment. Our framework has the flexibility to encompass a wide range of tasks that involve the completion of multiple sub-tasks with defined precedence and temporal constraints. For instance, our framework can be effectively applied to assembly tasks, where robots independently work on different components of an assembly process given the order of tasks and their corresponding time requirements. In such scenarios, the robots can utilize human assistance for specific procedures that require a higher level of precision or complex manipulation. Similarly, the proposed methods can be employed in surveillance missions, where robots are tasked with surveying a series of waypoints in an environment and may require human assistance for tasks such as target verification. Additionally, our approach can be extended to scenarios where different agents contend for other shared resources, such as access to communication bandwidth or limited workspace [13], while also being applicable to situations where we are required to optimize the fidelity level (resources allocated) of servicing the given tasks [14].

This versatility and adaptability of our approach make it applicable to a broad spectrum of real-world applications, highlighting the significance and impact of the research outcomes presented in this thesis. Through our research, we strive to enhance the efficiency, safety, and overall performance of supervised robot fleets, ultimately advancing the field of human-robot interaction and contributing to the realization of effective collaboration between humans and robots in real-world applications.

1.1 Contributions and Organization

The organization and contributions of this thesis are as follows.

Chapter 2: In this chapter, we present a brief overview of the multi-agent Human-Robot Interaction systems. We also provide a framework for understanding and characterizing such systems, and discuss how a supervised robot fleet fits within this framework.

The work presented in this chapter is part of the following publication.

- Abhinav Dahiya, Alexander M. Aroyo, Kerstin Dautenhahn, and Stephen L. Smith. "A survey of multi-agent Human–Robot Interaction systems." Robotics and Autonomous Systems 161 (2023): 104335 [11].

Chapter 3: In this chapter, we provide mathematical preliminaries and definitions relevant to the thesis. This includes discussion on graphs, shortest path problems, submodularity and restless multi-armed bandits.

Chapter 4: In this chapter, we present the problem of optimal planning for a robot in a collaborative human-robot team, where the human supervisor is intermittently available to assist the robot in completing tasks more quickly. We address the challenge of computing the fastest route between two configurations in an environment with time constraints on how long the robot can wait for assistance. To solve this problem, we propose a novel approach that utilizes the concepts of *budget* and *critical departure times*, which enables us to obtain optimal solutions while efficiently scaling to larger problem instances compared to existing methods. We also propose a greedy algorithm that efficiently prioritizes service requests for multiple robots, while having close to optimal performance.

The work presented in this chapter is based on the following publication.

- Abhinav Dahiya and Stephen L. Smith, "Optimal Robot Path Planning In a Collaborative Human-Robot Team with Intermittent Human Availability," in IEEE International Conference on Robot and Human Interactive Communication (RO-MAN), Busan, South Korea, 2023 [15].

Chapter 5: In this chapter, we discuss the problem of allocating operators among robots in a fleet, where each robot is required to perform an independent sequence of tasks, subject to the possibility of failing and getting stuck in a fault state at every task. If and when required, a human operator can assist or teleoperate a robot. Conventional dynamic programming-based techniques used to solve such problems face scalability issues due to exponential growth of state and action spaces with the number of robots and operators. We derive conditions under which the operator allocation problem satisfies a technical condition called indexability, thereby enabling the use of the Whittle index heuristic. The conditions are easy to check, and we show that they hold for a wide range of problems of interest. Our key insight is to leverage the structure of the value function of individual robots, resulting in conditions that can be verified separately for each state of each robot.

The work presented in this chapter is based on the following publication.

- Abhinav Dahiya, Nima Akbarzadeh, Aditya Mahajan and Stephen L. Smith, "Scalable Operator Allocation for Multirobot Assistance: A Restless Bandit Approach," in IEEE Transactions on Control of Network Systems, vol. 9, no. 3, pp. 1397-1408, Sept. 2022 [16].

Chapter 6: In this chapter, we present the design and outcomes of a user study conducted to investigate the impact of interruptions in systems where human supervisors monitor multiple robots. There can be several types of interruptions in such systems and we categorize the interruptions as either intrinsic, i.e., being directly related to the robot monitoring task, or extrinsic, i.e., being unrelated. We designed a web platform which allows users to monitor multiple robots, and report/fix faults in their navigation. During the experiments, we interrupt the users with secondary tasks and record parameters of performance and workload.

The work presented in this chapter is based on the following publication.

- Abhinav Dahiya, Yifan Cai, Oliver Schneider and Stephen Smith, "On the Impact of Interruptions During Multi-Robot Supervision Tasks," in IEEE International Conference on Robotics and Automation (ICRA), London, United Kingdom, 2023 [17].

Chapter 7: This chapter consists of a summary of the work presented in this thesis along with potential directions for future work.

Chapter 2

Multi-Agent Human-Robot Interaction

TL;DR: In this chapter, we provide a brief overview of the literature on Human-Robot Interaction (HRI), with a particular emphasis on multi-agent systems. This survey gives insights into the broader HRI landscape and helps contextualize supervised robot fleets within the field.

In order to establish a solid foundation, we begin by providing a framework for understanding and characterizing multi-agent HRI systems, drawing insights from the existing literature. Within this framework, we introduce the concept of an *interaction graph*, which serves as a visual representation of the interactions among different agents within an HRI system. Additionally, we take a moment to discuss how the system of a supervised robot fleet fits into this characterization, highlighting its role and relevance within the broader context of multi-agent HRI. This discussion sets the stage for the subsequent chapters, where we dive into specific research problems related to supervised robot fleets. Detailed reviews of the literature related to the specific problems are provided in the corresponding chapters.

Research in Human-Robot Interaction (HRI) has facilitated the introduction of robots in human spaces, and has gained significant momentum in the past two decades. Existing literature indicates that the vast majority of past research pertains to *dyadic* systems where a single human interacts with a single robot. However, this trend is changing now. With the improvement in physical and computational capabilities of robotic systems, we see robots working in teams of more than two, moving beyond the typical dyadic HRI systems. Even though multi-agent HRI systems are built on similar infrastructure as dyadic systems, there are several differences. Multi-agent HRI systems require more complex control strategies to coordinate several agents that may be dissimilar to one another (in roles, communication capabilities etc.), can involve interactions that connect more than two agents at once, and need special attention to address any conflicts that may arise from their interactions. These differences mean that control strategies developed for dyadic systems are either not applicable to multi-agent systems or they might not work as intended. As a result, a considerable body of research has emerged, addressing various aspects of multi-agent HRI systems specifically.

A human-robot system can have a varied number of agents, both humans and robots, interacting with each other. Even though any human-robot system is technically a multi-agent system (comprised of at least one human and one robot), in the HRI literature, the terminology is used differently. In HRI literature, *dyadic* systems refer to the ones built around interactions between exactly one robot and one human. *Triadic* systems have three agents, with two humans and a robot, or two robots and a human [18,19]. For systems with three or more agents, terms such as *teams* or *groups* are used [20]. In this thesis, we use the term *multi-agent* to refer to any human-robot system containing more than one human (multi-human) and/or more than one robot (multi-robot). Robots assisting humans in an industrial task, a group of humans and robots doing a social activity, or human operators controlling multiple robots are some examples of multi-agent HRI systems.

In the literature, we find several studies and articles reviewing research in the field of HRI and Collaboration, e.g. [12, 21–23]. There are also studies that present taxonomies and classification criteria for HRI systems in general, and reviewing the work done in different social and industrial applications of HRI, e.g. [24, 25]. The literature review on human interaction with multiple remotely situated robots, conducted by M. Lewis [26], examines such systems under the notion of *command complexity* and analyzes its implications for the users. A comprehensive study by Thomaz et al. [23] provides a detailed review of the computational aspects of HRI, covering topics such as algorithms, modelling and computational framework design. Human-Swarm Interaction (HSI) is another subset of multi-agent systems where a small number of humans (commonly one) interact with a group of robots that coordinate among themselves and often act as a unified entity. Kolling et al. present a comprehensive survey of Human-Swarm Interaction systems and discuss core concepts for their design [27]. In the multi-agent HRI literature, Sebo et al. present a review of studies involving robots' interactions with a group or team of people, and explore characteristics of such systems [28]. The article discusses the role a robot plays in a group of humans and focuses on human-robot systems consisting of multiple humans irrespective of the number of robots involved. From a social psychology perspective, Oliveira et al. discuss methodological and transversal issues in HRI research conducted in small groups [29].

Expanding upon the existing surveys and reviews, our discussion in this chapter encompasses a broader range of systems. We include systems where multiple robots interact with either a single human or a group of humans, both within and outside of social settings. Moreover, we consider systems built around interactions among multiple humans and multiple robots simultaneously. Additionally, we incorporate studies on theoretical and computational research, which may not involve actual robots or humans but still encompass crucial aspects of an HRI system. Examples of such research include scheduling collaboration [30, 31] and optimizing collaborative manipulation [32]. Such studies are invaluable contributions to the multi-agent HRI literature and offer useful insight into the implementation of real-world systems.

Research in Human-Robot Interaction encompasses a broad scope, addressing various aspects of robot decision-making, human behavioral modeling, and system analysis. This research area includes systems where robots and humans coexist, whether in team settings or as independent entities pursuing shared or distinct objectives. This coexistence can manifest in physically co-located interactions or in systems where remotely located agents connect through virtual interfaces. Furthermore, the development of capable and efficient algorithms to enhance interaction between humans and robots has an important place in HRI research.

The research literature in this domain comprises three types of studies, which can complement and intersect with one another: 1) System Design: ones that propose a system useful in the context of HRI (direct application; e.g., [33]), 2) Observational: ones that detail a study designed to elicit different features of HRI systems (to understand system behavior; e.g., [34]), and 3) Algorithmic Design: ones that present computational research, including algorithms, models and frameworks to solve the decision-making problems in HRI systems (e.g., [16]). In the subsequent discussion, we consider all three kinds of studies from the multi-agent HRI literature.

2.1 Multi-Agent Human-Robot Systems

In order to formally define multi-agent Human-Robot Interaction systems, we first need to define the concept of 'interaction'. The American Psychological Association [35] defines a social interaction as a process that involves reciprocal stimulation or response between two or more individuals. In the context of HRI, however, reciprocity is not a necessary requirement. Therefore, we define an interaction in this context as follows:

> "information flow between two or more agents occurring as a result of communication, action, or presence of any of those agents".

This information flow leads to changes in the actions, behaviors, or mental states of the receiving agents. Using this definition, we can express a multi-agent HRI system using an interaction graph (Fig. 2.1). The figure illustrates a multi-agent HRI system and its corresponding interaction graph, where each agent is represented as a node in a directed graph. The edges in the graph depict interactions between pairs of agents and indicate the direction of information flow. We use this interaction graph structure to define multi-agent HRI systems as follows:

> Multi-agent Human-Robot Interaction systems are the ones for which the interaction graph contains three or more nodes (agents), with at least three nodes connected via interaction edges, including at least one interaction edge between a human node and a robot node.

Information exchange within these systems can occur through verbal or non-verbal communication channels, or through interaction interfaces. The agents in these systems can be homogeneous, sharing similar roles and identities, or heterogeneous, contributing differently to the overall system. Moreover, these agents can be located in separate environments interacting over virtual channels or they can share the same physical space. In the following discussion, we consider the literature on multi-agent HRI systems and try to characterize them based on various properties.

2.2 Characterization of Multi-agent HRI Systems

Human-Robot Interactions among multiple agents introduce a multitude of changes and complexities compared to dyadic interactions. With multiple agents present, multi-agent interactions and indirect interactions can emerge in the system. Notably, it becomes possible to have humans interacting with other humans and robots with other robots. These Human-Human Interactions (HHI) and Robot-Robot Interactions (RRI), in turn, have an impact on or are influenced by interactions between humans and robots. Moreover, the

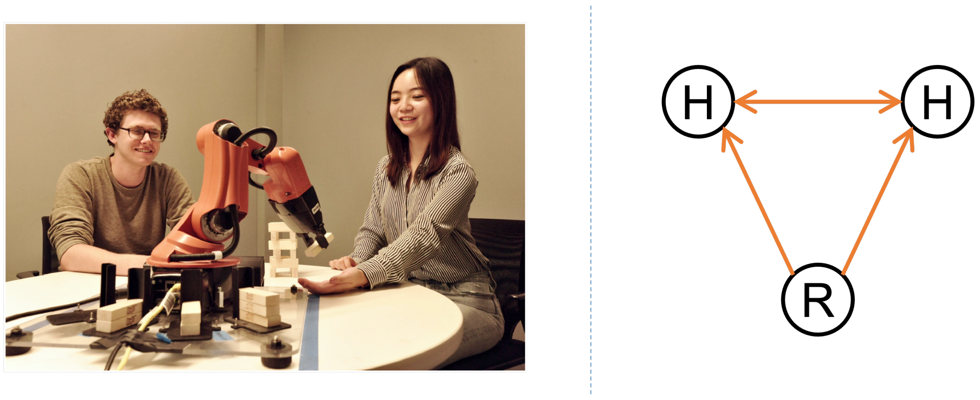


Figure 2.1: Left: HRI system for robot-assisted tower construction task where the robot allocates tower pieces between two human teammates [36]. Right: The corresponding interaction graph representation of the system. In the graph, each human and robot is denoted by a node. Each arrow represents an interaction and its direction signifies the direction of information flow. When information flow between two nodes is present in both directions, a bi-directional arrow is used. In this system, human behavior is dependent on robot's actions while the robot's actions are independent of its human teammates (thus the uni-directional arrows). Both humans can potentially discuss their next actions with the other human teammate (denoted by the bi-directional arrow).

involvement of multiple agents necessitates advanced communication modalities and interfaces to facilitate effective interaction among all the agents. As the number of agents in the system increases, the task of modeling their behavior and controlling their actions becomes increasingly challenging.

To provide a comprehensive understanding of multi-agent HRI systems, we propose a characterization framework based on three core aspects: 1) Team structure, 2) Interaction style and 3) Computational characteristics. These aspects represent the way a system is set up, and the way interactions take place and methods by which agents are controlled in that system.

Team Structure: The first aspect captures how the agents are organized within the system. It encompasses various parameters such as the composition of agents, including the number and types of agents, which can consist of both humans and robots. Additionally, team structure examines the degree of homogeneity among the agents, considering factors such as their roles, capabilities, embodiment, and assigned authorities.

Interaction Style: The second aspect of multi-agent HRI systems pertains to the manner in which the agents engage and interact with one another. This aspect encompasses various factors, including the modality of communication utilized among the agents and the implemented interaction models. This tells us the way in which interactions take place in the system (through a screen or speech etc.) as well as the agents involved in these interactions, whether it be a dyadic interaction between two agents or a collective interaction among multiple agents.

Computational Characteristics: The third aspect examines the computational aspects of HRI systems and focuses on how the behavior of various agents is controlled or influenced. This aspect encompasses elements such as robot task planning algorithms, model-based or model-free controllers, and other mechanisms that determine the actions of robots in the system. Additionally, it explores how these computational techniques are utilized to influence human behavior. By studying the computational characteristics, we gain insight into the underlying software components and decision-making processes that shape the interactions between humans and robots in the multi-agent HRI system.

These three aspects provide us with a framework to establish distinctions and make comparisons among multi-agent HRI systems. For instance, consider a system where a human operator is supervising a team of remote mobile robots (e.g., [37, 38]). We can characterize this system as one with a single human interacting with multiple robots (team size), and one where all the robots exhibit similar characteristics (homogeneity). The human might be able to give high-level commands to the robot group or control them individually (interaction model), using a screen-based interface (communication modality). The robot control can be made adaptive to actions of the operator or designed to optimize some utility function (computational characteristics). This example demonstrates how the three aspects enable a comprehensive characterization of multi-agent HRI systems.

Under these core aspects, we can characterize HRI systems based on five different attributes as shown in Fig. 2.2. It is important to note that these core aspects are interconnected, and attributes within one aspect can influence other system attributes. The nature of these interactions is further explored in Section 2.2.4. Additionally, it is worth mentioning that these core aspects do not aim to provide an exhaustive or exclusive means of distinguishing multi-agent HRI systems. Rather, they are selected to offer a taxonomy that is both broad enough to encompass diverse areas of research within multi-agent HRI and detailed enough to meaningfully characterize and compare these systems. In the following sections, we discuss the above three aspects and the associated attributes in more detail, and briefly present the categorizations that arise under these attributes. Under each category, we include examples from recent literature to understand its application, and analyze strengths and limitations of different types of systems.

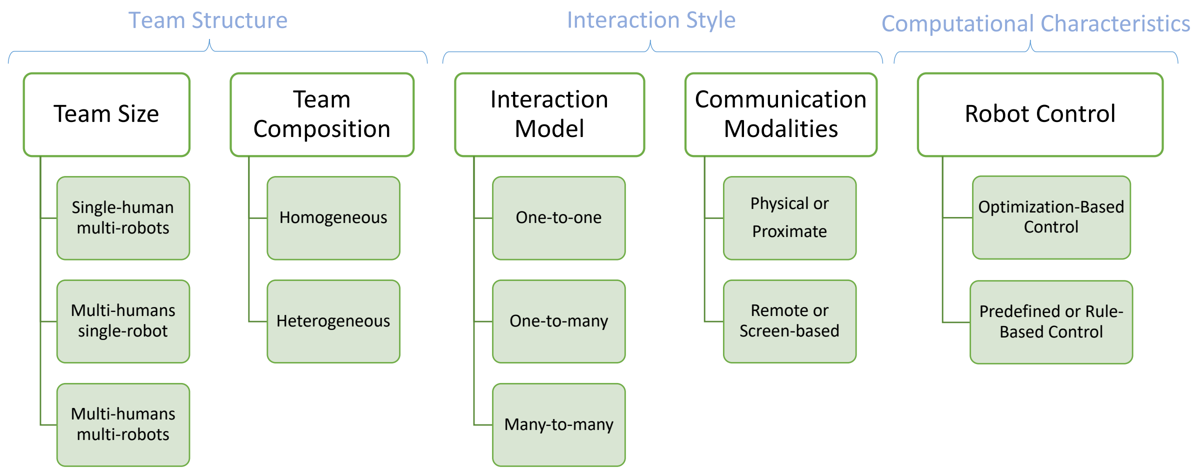


Figure 2.2: The different categorizations for multi-agent Human-Robot Interactions originating from the three core aspects: the team structure, the interaction style and the computational characteristics. These aspects bring forth several attributes of systems, each with its own features and applications.

2.2.1 Team Structure

The most perceptible feature of a multi-agent HRI system is the size and composition of the human-robot team. Depending on the application, a human-robot system can take advantage of more than one human and/or more than one robot in the team. Additionally, the task requirements may necessitate the involvement of agents with diverse capabilities and roles. Both the number and type of robots in a group can have significant effects on human perception and emotions towards the robots [39]. Since humans and robots usually have different ways of acting in a collaborative setting and interacting with their partners, the team structure decides various other aspects of the system including the way in which different agents interact, how they can share information and how their actions are planned [40, 41].

Based on team structure, multi-agent systems can differ from dyadic systems in two notable ways:

1. There can be several possibilities of having different numbers of agents (both humans and robots) in the team,

2. There exists a notion of homogeneity/heterogeneity among the agents, as agents may

share similar roles and capabilities or exhibit variations in their attributes.

Therefore, we discuss in the following two factors under team structure: Team size, and Team composition.

2.2.1.1 Team Size

Team size in an HRI system refers to the number of humans and robots present in the system. Based on team size, HRI systems can be categorized into four distinct groups.

Single-human – single-robot systems represent the most common type of HRI systems studied extensively in the field [22, 42–44]. These systems involve a single human interacting with a single robot, forming a dyadic relationship.

Single-human – multi-robot systems involve a single human interacting or collaborating with multiple robots. These systems are utilized in various applications, such as fault intervention or performance enhancement in multi-robot teams [37, 45], search-and-rescue operations [46], and Human-Swarm Interaction [27]. In some cases, the human user may not be an operator or supervisor but rather receive navigation instructions, co-operative navigation, or guidance from the robot team [47–51]. Additionally, there are instances where a team of robots collaboratively executes tasks like storytelling or drama for enhanced effectiveness [52, 53].

Multi-human – single-robot systems involve multiple human teammates interacting with a single robot. These systems have been used in applications such as search and exploration, unmanned aerial vehicle (UAV) operations, and resource distribution [36, 54, 55]. In these settings, humans assume different roles and manage various components of the robot's operation [56, 57]. Additionally, there are systems where a robot interacts with multiple humans for assistance [58–60], to moderate group interactions [61, 62], or to manage resource distribution [36, 58]. Furthermore, social robotics research has explored robots collaborating with multiple users in contexts like autism therapy [63–65], education [66–68], and interactions in public spaces [69]. For a more comprehensive review of systems with robots interacting with groups of humans, refer to [28].

Multi-human – multi-robot systems involve teams with multiple humans and multiple robots. These systems present significant challenges due to increased uncertainty and the exponential growth of possible states [16]. They have been applied in military scenarios where human teams coordinate with (semi)autonomous robot teams [70]. Additionally, such team compositions are found in search-and-rescue tasks [71–73] and supervisory control of heterogeneous human-robot teams [74–76]. Such teams have also been discussed in several theoretical/computational studies addressing the task allocation and operator scheduling problems [16, 31, 77, 78]. In social interaction and classroom settings, multi-human – multi-robot teams are explored for understanding socio-emotional aspects [79, 80] and enabling group learning [81, 82].

2.2.1.2 Team Composition

Besides the number of humans and robots, another important characteristic of a human-robot team is the team composition. This pertains to the aspect of homogeneity (or lack thereof) among the agents, be it humans or robots. In the case of robots, homogeneity may simply indicate whether there are different types of robots present in the team, with different hardware design [45], manipulation capabilities [33] or interaction interfaces [71]. In the case of humans, presence of different roles, capabilities or authority determines homogeneity among team members [19, 67].

Homogeneous team composition is commonly seen in applications where agents are primarily identified as a part of a group, such as robots in a swarm [27] or humans in a crowd [69,83]. In systems where agents work on similar tasks independently, a homogeneous team structure is common [11, 16]. In social HRI applications, humans are often considered equal members of the group, resulting in a homogeneous composition [84, 85].

Heterogeneous team composition is commonly observed when different types of robots are required, such as robots with different manipulators or mobility [33, 50]. In military applications, teams of heterogeneous humans with distinct roles, responsibilities, and authority are used to control complex robots [86]. Differences or similarities among human teammates can also introduce heterogeneity in system dynamics [87]. Assigning different roles to humans in HRI systems has been explored, similar to roles assigned to robots [71, 88, 89]. Industrial-oriented tasks often involve teams of heterogeneous humans in roles such as supervisors and assistants [19, 56, 57]. Heterogeneous team composition also finds applications in social/educational robotics, such as systems engaging with users from different generations [90, 91] or when humans have distinct roles or jobs within the group [92].

Homogeneity in a human-robot team greatly influences the type, level and efficiency of the interactions [93]. For instance, when managing a team of robots, the human operator needs to put in more interaction effort when the team is homogeneous as compared to the one with heterogeneous robots [26, 94], and this may lead to an increase in perceived workload and decrease in situational awareness [95, 96]. Having homogeneous robots may also lead to a simpler interaction interface [25]. However, heterogeneous robot teams may allow to use specific robot capabilities to carry out a variety of operations [45, 97]. It is generally agreed that control and operation of a team of heterogeneous agents is more demanding compared to a homogeneous team, therefore, the literature offers a variety of studies presenting efficient control strategies for the control of heterogeneous multi-robot teams [98–100].

2.2.2 Interaction Style

In the context of human-robot systems, interaction style is a broad term that can be used to refer to different aspects of interaction such as the modes of communication among agents [101], interaction models [69] and the interaction interface [102, 103]. It also includes communication methods (verbal/non-verbal) [104, 105], expression of affect [106] and spatial relationships [107, 108].

Interactions in multi-agent human-robot systems differ from those in dyadic systems in three ways:

1. First, with multiple agents present, it is possible to have interactions within a group of agents, viz. Human-Human Interactions [71] and Robot-Robot Interactions [109].
2. Second, in addition to the one-to-one interactions seen in dyadic HRI systems, one-to-many interactions are also realizable in multi-agent HRI systems [69].
3. Third, in multi-agent systems, there are additional types of interactions possible among the agents.

Patel et al. [110] discussed the differences between 'direct' and 'indirect' interactions between two humans in a system in which they can communicate either via verbal communication or through the interface. Che et al. [111] investigated the role of 'explicit' and 'implicit' communication in social navigation. In [20], authors used the terms 'the group' and 'the observer' to distinguish the two perspectives of measuring group cohesion in a multi-agent HRI setting.

Such distinctions of interaction types are useful to better understand HRI systems and can be applied to improve the interaction outcome. In the context of multi-agent HRI systems, we find it useful to distinguish 'direct' and 'indirect' interactions. Direct interaction between a sender and recipient occurs when the sender actively (i.e., intentionally and explicitly) communicates to the recipient using any mode, verbal or non-verbal. A third party may also receive information from this direct communication, and we refer to this "eavesdropping" as indirect interaction. Taking the example of a study presented by [50], a robot trying to transfer task information to another robot (e.g., through speech) is an example of direct interaction, whereas a human observing the robots interacting with each other is an example of indirect interaction. Making this distinction can help us understand if and how much the agents in the system are actively trying to communicate with each other, or if they are primarily co-existing in the same environment while observing each other. Modelling the indirect interactions may be useful to understand how direct interactions between any two agents can affect the behavior of others [50, 61, 112].

Examples of systems with different types of interactions are given in Table 2.1. In the remainder of this section, we discuss two key aspects of interaction style: 1) Interaction model present, and 2) Communication modalities used in the system.

2.2.2.1 Interaction Models

While a multi-agent HRI system involves multiple agents, not every interaction in the system necessarily includes all agents simultaneously. Interactions can be implemented using different-sized communication channels, enabling one-to-one or one-to-many interaction models [69]. Examples of several multi-agent HRI systems with different interaction models are shown in Table 2.1 using the interaction graph structure defined in Section 2.1. These interaction models are an extension of the interaction types presented by [25] and capture a variety of interaction possibilities in multi-agent HRI systems, particularly in social interactions. They also allow us to specify the direction of information flow between agents, distinguishing between human-to-robot and robot-to-human interactions.

Note: These interaction models, in their basic form, do not capture all aspects of the complete system, such as embodiment, roles, heterogeneity among agents, and modes of communication. However, additional information can be incorporated to represent these aspects with slight modifications. Table 2.1 shows an example of such additional information, indicating whether an interaction is direct or indirect using solid and dashed arrows, respectively. Liu et al. [113] use edges in their interaction graphs to denote spatial interactions and distinguish between human-human and human-robot interactions.

As depicted in Table 2.1, interactions between different pairs of agents can be implemented using different models, such as one-to-one or one-to-many. The interaction model may also vary depending on the types of agents involved, such as human-to-robot, robot-to-human, robot-to-robot, and human-to-human interactions. For example, a human may give commands to a single robot at a time (one-to-one), but observe the behavior and receive messages/information from multiple robots simultaneously (one-to-many) [37, 46].

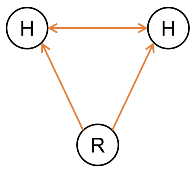
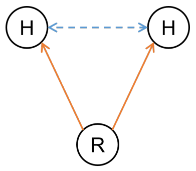
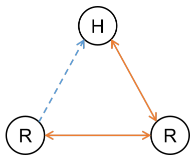
	Study	Interaction Graph	Remarks
(a)	[36, 61]		Robot actions are independent of human teammates. Humans decide their actions based on robot's actions and potential discussion with the other human teammate.
(b)	[58]		Robot actions are independent of human teammates. Humans decide their actions based on robot's actions, and observing actions of the other human teammate (without an explicit communication).
(c)	[50]		A human user directly interacts with one of the two robots, which then conveys the required information to the second robot while the user observes the two robots.

Table 2.1: Examples of multi-agent human-robot systems with respective interaction graphs. The orange solid arrows signify direct interactions between agents (e.g., explicit communication), while dashed blue arrows denote the indirect interactions (observational). A maximum of two humans and robots are shown in the figures but interaction graphs can be drawn for larger number of agents in a similar way. Note that this table only shows examples of possible interaction graphs and is not an exhaustive list.

These interaction models are further categorized based on the number of agents connected at each end of the communication channel.

One-to-one interaction model resembles the dyadic interactions commonly observed in human-robot systems, where one robot interacts with one human teammate at a time, despite the presence of other agents in the environment. This form of interaction is commonly seen in systems where one can afford the commands/instructions for an agent to be independent from others (e.g., in systems where coordination among agents is not required). A common application is seen in systems where a human operator manages a team of multiple robots by giving each of them separate commands. While a fleet of autonomous robots acts in separate environments, a human operator can monitor their states remotely, and can intervene to help a robot via teleoperation when the robot encounters a

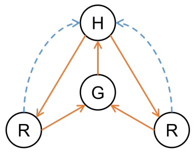
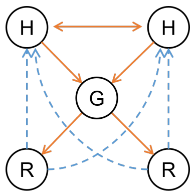
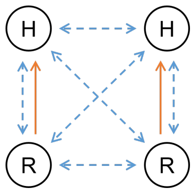
	Study	Interaction Graph	Remarks
(d)	[37,38]		A human operator controls each robot individually. Human's selection of robot is decided either by observation of robots' states or based on the suggestion shown by the interface (G).
(e)	[74]		Commands of multiple human operators are converted to actions required by individual robots. Operators observe each robot directly and can also interact with each other to resolve conflicts.
(f)	[80]		Two teams, each consisting of one human and one robot, play a competitive card game. Each agent decides on their actions based on observations of other agents' actions. The only direct communication in the system occurs when a robot speaks to its partner to convey its emotions.

Table 2.2: Examples of multi-agent human-robot systems with respective interaction graphs (cont'd).

challenging state [37,38]. The one-to-one interaction model can also be applied in human-robot swarm systems, e.g., using a leader-follower approach [114]. One-to-one interactions are also seen in office settings, where a single human interacts with a single robot, even in the presence of multiple humans or robots [47,60]. In settings where agents are located in the same environment, the system also has indirect interactions, occurring when an agent observes other agents and interactions are happening between those agents. These indirect interactions still follow a one-to-one model, as the flow of information from each agent is directed towards the observer.

One-to-many interaction model can be seen in situations where one robot is directly interacting with multiple humans at once [80,81,115], and where one human is simultaneously interacting with multiple robots [116–118]. One way to implement this interaction model is through verbal communication or gestures, enabling group commands to be issued to multiple agents simultaneously [119]. Consequently, this model is commonly observed in social settings where humans and robots coexist in the same environment [80,115]. Another similar application of this model is found in classroom settings where a robot serves as a tutor for a group of students to promote group-based learning [52, 81, 120]. Additionally, when a robot interacts with multiple humans in a public space, the one-to-many model naturally applies [69]. In industrial or military-oriented applications where humans and robots may be located in separate environments, the one-to-many model is often implemented using a group command node (see interaction graphs in Table 2.2; shown as node (G)). This group command node is an interpreter – an interface or an algorithm – that converts communication from one or more agents into information required for each individual agent before relaying that information to the intended recipient(s). For example, when controlling multiple ground or aerial robots, a human operator can issue commands to the entire fleet, and the group command node converts these commands into specific actions for each robot [116, 121]. Similarly, when humans interact with a swarm, a group command node is typically employed to convert human intentions or commands into control signals for all robots, ensuring efficient human control [27, 122].

Many-to-many interaction model can be observed in multi-human – multi-robot systems, multiple one-to-many interactions can occur among different agents simultaneously. Although less common in the literature, some studies [110, 123, 124] have presented interaction interfaces to facilitate many-to-many interactions. A common approach to enable interactions in such systems is through a proxy architecture designed to facilitate collaboration between humans and robots with varying levels of autonomy [70, 86, 125]. For instance, the system presented by Patel et al. [74] utilizes an AR interface on separate screens to enable multiple users to interact with multiple robots. In the absence of such proxies, the human users in the system must resolve any conflicts (in commands/decisions) by themselves and work out a common strategy [126]. Many-to-many interactions can be represented in the interaction graphs of Table 2.2 as a group node (G) with multiple incoming and outgoing edges. A many-to-many model provides a natural and efficient interaction setup in a multi-agent system as there are minimal constraints on when agents are allowed to communicate with each other, and information from multiple agents can be simultaneously relayed to their respective recipients.

2.2.2.2 Communication Modalities and Interfaces

Communication modalities refer to the modes through which different agents interact in the system (e.g., speech, haptics, screen etc.). Similar to conventional dyadic HRI systems, a multi-agent system can involve agents communicating through verbal or non-verbal modes, or a combination of multiple modes simultaneously. The choice of communication modes depends on factors such as the environment, system application, proximity of agents, and their interaction capabilities. Each modality has its own advantages, implementation requirements, and limitations. Proximity plays a significant role in determining feasible communication modalities for enabling efficient communication among humans and robots. Therefore, to discuss the different types of communication modalities used in multi-agent systems, it is helpful to group them based on the proximity of the agents.

Remote communication: is ubiquitous in systems designed to operate in potentially dangerous or inaccessible environments for enabling remote communication among humans and robots. Human control of a fleet of UAVs [57,127], supervision during search and rescue tasks [46,55,128] and teleoperation of underwater robots [129] are some of the applications that make use of remote communication techniques.

However, remote interaction with robots can lead to higher cognitive workload for human users [130], which further increases with an increasing number of robots [40,95]. Communicating plans to team supervisors and maintaining their situational awareness are also important challenges in collaborating with remote robots [131]. Thus, developing interaction interfaces that facilitate efficient and reliable human interaction with multiple remote robots is an important research area in remote HRI [75, 103, 132, 133]. In systems with a disproportionately high number of robots compared to human operators/supervisors, it becomes challenging for human users to interact with all robots efficiently due to perception, workload, and situational awareness challenges [40]. Intelligent interface designs are therefore required to mitigate these challenges [134]. The existing literature predominantly consists of screen-based interface designs, which are convenient to implement in such systems, as evidenced by the studies discussed earlier in this section. Common design features of multi-robot interfaces include camera feeds of multiple robots in small cards along the screen edge, an enlarged view of a selected robot, and a map or overview of all robots in the environment [38, 135]. Recently, immersive interfaces using Virtual Reality (VR) or Augmented Reality (AR) have gained popularity in multi-robot systems [133, 136] and have been shown to improve operator's situational awareness and decrease workload [137]. Some systems also incorporate multiple communication modes, such as haptics and audio, to enhance interaction outcomes [9, 138].

Proximate communication is found in systems where humans and robots are located in the same environment, as there is a wider range of options for system designers to choose from in terms of communication modes. These modes include auditory channels (speech and non-speech audio), visual channels (gestures, facial expressions, body postures and gaze), and physical channels (touch and force). For proximate communication, it is also common to use multiple modes simultaneously, resulting in a multi-modal communication architecture. For example, Pourmehr et al. [139] present a system that uses a combination of haptic and verbal inputs from a human user to control multiple UAVs. In another system by Gromov et al. [102], a human user can communicate with robots using speech and gestures, while robots provide visual and verbal feedback to the user.

Proximate communication is more commonly observed in social robotics, as it facilitates the personal interactions required in a social setup. Social cues, both verbal and non-verbal, play a significant role in conveying meaning during interactions [101]. Studies have explored the use of different cues, such as expressions and movements, to communicate a robot's intent, emotions, and information more clearly in a group setting [80, 140]. The spatial placement of agents relative to each other and the environment also influences human behavior in a group setting [141, 142].

When agents are co-located, the interaction between any two agents can have an effect on the behavior and future interactions of other agents, demonstrating indirect/implicit communication. Human-Human Interactions can be influenced by Human-Robot Interactions [63,90,91]. Similarly, Robot-Robot Interactions can impact future human interactions with the robots [50, 143] or affect a human's psychological state [144]. Moreover, the embodiment or mere presence of a robot can influence human behavior and their perception of the robots [145, 146]. Communication modalities can also vary depending on the agents involved and the direction of communication. For instance, Berg et al. [147] present an interaction system where the human to robot communication channel is realized through gestures and eye tracking while the information from the robot to the human is communicated using a projection. In the system presented by Rosenthal et al. [60], the robot communicates its queries using speech and receives human input using a visual interface on a laptop. Also, it is common to see different communication modalities between Human-Robot and Human-Human Interactions [71].

2.2.3 Computational Characteristics

In addition to the *perceptible* aspects of an HRI system, which define the presence and interactions of agents, computational aspects play a crucial role in influencing and controlling their behavior. Now, we look into computational aspects of the system, specifically how system designers can choose to influence/control the behavior of different agents in the system. In an HRI system, behavior and actions of robots are controlled directly, either governed by an optimization-based action-policy or via predefined/rule-based methods, or a combination of both. On the other hand, human behavior is influenced indirectly using robots—via robots' actions, their interactions with the humans, other robots or the environment, or by explicit communication. Some systems incorporate human behavioral models to determine robot actions, while others adopt a model-free approach.

For the purpose of this article, we prioritize the discussion of robot control in multi-agent HRI systems, while including relevant aspects of human behavior when necessary. Specifically, we examine the two main types of robot control: 1) Optimization-based control, and 2) Predefined and rule-based control. Furthermore, we also discuss the differences that the presence of multiple agents brings to the system.

Optimization-based control refers to a framework of computing robot actions to optimize one or more performance-defining parameters established for the system. The performance parameters often represent factors like time of task completion [30], cost incurred (resources spent) [16] and reward earned (value produced) [37, 148]. In the multi-agent HRI literature, to pose the mathematical optimization problem, we find examples of systems being modeled in the form of time-series [129], outcome probabilities [16, 45] or Dynamic Bayesian Network [149]. Machine-learning and other data-driven methods are also some of the tools used in optimization-based control, e.g., [37, 89, 150]. Such system models are often motivated by literature from different areas of behavioral study such as psychology, economics and social sciences. For example, Shannon et al. [151] present the Pew model from psychology, Swamy et al. [37] make use of the Luce Choice model from economics, and Bera et al. [152] use entitativity related psychology research in their system.

Optimization-based control is seen in studies where the system behaviour can be modelled reliably using existing theories, or where researchers are trying to validate a new approach for the same. These systems also require that there exist quantifiable and measurable parameters that can be used as optimizing metrics (such as task completing time, error rate, etc.). When implementing such robot control in multi-agent systems, there are a few considerations to handle. When multiple agents are present in the system, the required information access and the possibility of interactions may increase exponentially with the number of agents. This problem has motivated a whole segment of research on the development of computationally efficient decision-making and control techniques for multi-agent HRI systems. Among other applications, this research is seen in systems enabling a robot influence a team of multiple humans [89], enabling multiple robots to safely navigate among other robots and humans [153], predicting human behavior while supervising multiple heterogeneous unmanned vehicles [154], and finding task allocation and sequencing for multiple robots travelling to collaborate with humans [30]. When dealing with a large number of robots, human users' ability to maintain awareness of the system's state might be insufficient [155, 156] and thus the users may benefit from a decision support system (DSS). Applications of such DSSs are seen in systems enabling a human operator to assist multiple remote robots [37], and in allocating operators to multiple navigating robots [16,38,78]. There are also several studies on human supervision of fleets or swarms of remote robots. Lewis [26] presents a review of systems enabling such supervision under the construct of command complexity, while Kolling et al. [27] review research on human control of robot swarms.

Predefined and rule-based control methods provide a convenient way of implementing robot control in an HRI system without the use of computational models or data. This includes techniques like Wizard of Oz, expert knowledge-based *if...then* rules or pre-specified sequences of actions. These methods help simplifying the robot's decision-making and allow the researchers to focus on other aspects of Human-Robot Interaction that the system is designed to investigate, e.g., the outcome of a controlled interaction [50, 143]. In some systems, direct use of an optimization-based control is not possible (e.g., due to lack of a numerical model), and a subjective interpretation of system events is required based on expert knowledge. Such systems often have robot action policies implemented as *if...then* rules instead of numerically computed conditions. For example, Correia et al. [80] present an algorithm to generate robot emotions given system events, based on knowledge from psychology. Rule-based control is more commonly seen in studies that deal with subjective metrics of outcome (e.g., human perception of robots, workload, etc.). It is to be noted that the optimization-based and rule-based control methods are not mutually exclusive. It is possible to implement a combination of an optimizing policy with an expert knowledge-based model. For example, in the system presented by Alves et al. [81], the robot's behavior is decided by a hybrid controller that combines manually-encoded behavioral rules and a machine learning-based mapping function.

While the use of a rule-based control can enable a system to decide the robot's behavior based on some feedback from the environment, it is also possible to pre-specify robot behavior without any decision-making component. This robot control method facilitates studies to manipulate different test conditions where feedback from the environment is not required. Such implementation is commonly seen in studies which investigate the effects of specific robot's behaviors on human users, e.g., users' perception of robots [157], knowledge acquisition [66], group emotions [158] and perceived legibility [159]. Referring back to the interaction graphs defined in Section 2.1, this type of robot control results in unidirectional edges from robots to humans. Under predefined and rule-based control methods, the Wizard of Oz (WoZ) is a common technique of implementing human-assisted robot control in multi-agent (mainly multi-human) systems. In this technique, a hidden human 'Wizard' (teleoperator) controls some or all of the robots' actions, speech, gestures and behavior, unknown to other human users in the system [160]. Depending on the level of robot autonomy, the Wizard can be used to replace certain parts of the robots' perception or cognitive capabilities (e.g., [161,162]), thus overcoming the robots' limitations. It is also possible to have a mixed-initiative approach where either the robots or the human user can take control of the robots' actions (e.g., [46, 129, 163]).

Note on Group/Individual behavioral parameters: Regardless of the type, robot control in multi-agent systems can be implemented in two ways: either based on parameters of individual agents or based on the group as a whole. Taking the parameter of trust as an example in a system with a single human and multiple robots, one can either plan robots' actions by considering trust of the human in each individual robot [164], or one can consider the human's trust in the whole robot team [165]. Other examples of robot control based on individual parameters can be seen in studies with parameters like engagement [82] and attention [143]. Such individual modelling provides a simple method to expand dyadic HRI research to the multi-agent setting, and to test any differences between the two. Group-based robot control has its own benefits. Often, the humans in the system are not working independently of other agents. So, one may find it useful to decide robot control as a function of group-based parameters, something not possible in dyadic systems. Such control is particularly useful in systems where agents act as a coordinating team, or in applications where performance or behavior of the whole group is central. For example, a robot can express group-based emotions to increase its likability among human teammates [80], or use audio features to affect group entitativity [166].

2.2.4 Interactions between the Core Aspects and Attributes

When examining the multi-agent HRI literature across the three core aspects, we can observe that the system attributes chosen by researchers within one core aspect are influenced by attributes within other aspects. To understand the interaction between these core aspects, we consider a specific set of attributes as the "configuration" of the system under each core aspect. For example, the team structure is defined by its size and composition. The categorizations across the different core aspects are designed in such a way that each core aspect's configuration can be independently determined during system design. In other words, one can select any attribute from each column of the table in Figure 2.2 to build a multi-agent HRI system[1]. This implies that a viable system can have any of the three configurations for team size, featuring homogeneous or heterogeneous agents, adopting one-to-one or one-to-many interaction models, employing proximate or remote communication modes, and using optimization-based or rule-based control for the robot(s) in the system.

[1]With the exception that having a many-to-many interaction model requires a team size comprising multiple humans and multiple robots.

However, the configuration of one core aspect can influence how another core aspect of the system should ideally be configured. For instance, in the case of a large team size, a one-to-many communication model may be more efficient, particularly in co-located settings as demonstrated by Kim et al. [119]. Additionally, systems with a large number of robots often incorporate a group command node to facilitate coordination among the robots, as it can be challenging for the human(s) to communicate individually with each robot. This interaction model is therefore commonly utilized in Human-Swarm Interaction scenarios [27].

Similarly, the Wizard of Oz control technique is prevalent in systems with a limited number of robots because of its ease of implementation [160]. However, this technique may hinder scalability of such systems and restrict researchers to use homogeneous or a limited number of robots. This highlights the need of developing efficient tools for implementing robot decision-making to facilitate HRI research in large-scale systems. Moreover, a system requiring a human Wizard to closely monitor the agent(s) can be cognitively demanding and time consuming. The choice of configuration is also influenced by the target application and objective of the study. As seen in Section 2.2.2, systems designed for social settings often adopt a one-to-many interaction model. In cases where a human operator needs to supervise multiple robots, a simple screen-based interface is a popular choice among system designers.

However, it is important to emphasize that research in multi-agent HRI is still in its early stages, and it does not yet provide a definitive guide for achieving an "ideal" system configuration, or even establish whether such an ideal configuration exists. For example, although screen-based interfaces have been prevalent in systems with multiple robots, researchers are beginning to explore the potential of VR and AR-based interfaces for their immersive and enhanced capabilities. As the literature in this field continues to grow, future developments may enable us to determine whether these existing configurations can serve as guidelines for designing multi-agent HRI systems, or if they represent opportunities to explore novel configurations.

2.2.5 Supervised Robot Fleet as a Multi-Agent HRI System

In the upcoming chapters, our attention centers on a scenario involving a fleet of robots navigating an environment under the remote supervision of a limited number of human operators. Here we will provide an overview of how our system aligns with the classification framework outlined earlier, shedding light on its classification.

To begin, our system's team structure consists of multiple robots and multiple human operators. A distinguishing characteristic is that all robots share identical attributes, forming a team of homogeneous robots and similarly all humans form a homogeneous team as well.

In terms of interaction style, our system employs a central computational entity to facilitate interactions between the human operators and the robots. This entity gathers data from all robots and then allocates each human operator to the robots requiring assistance, resulting in a many-to-many information exchange. After allocation, a human operator may engage in direct interaction with a specific robot, resulting in a one-to-one interaction mode. The interaction graph of our system is illustrated in Figure 2.3.

Given that the operators are located remotely, our system necessitates the use of a remote interface to facilitate communication between the robots and operators. In Chapter 6, for example, we employ a web-based screen interface within the context of the study presented.

Lastly, regarding robot control strategies, our system adopts different approaches across the studies featured in various chapters. In Chapter 4, the systems utilize an optimization-based control approach. Conversely, the studies explored in Chapters 5 and 6 employ predefined behaviors for the robots.

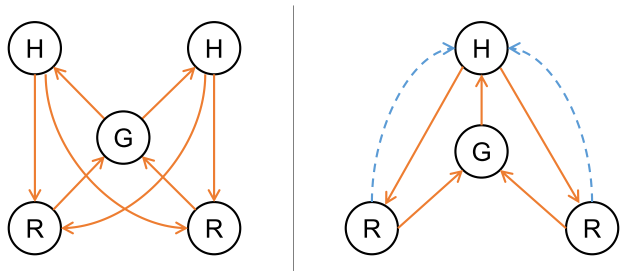


Figure 2.3: Interaction Graphs of the supervised robot fleets as HRI systems discussed in this thesis. Left: In Chapters 4 and 5, the systems in consideration have a central computational entity (G) – a decision support system – that collects data from all robots and then allocates human operator(s) to the robots that need assistance or schedules assistance for future instances. Operators can directly assist each robot individually, and are allocated by the central computational entity. Right: In Chapter 6, we look into a scenario where a single user monitors multiple robots, and interacts with them when required. Moreover, the interface (G) can direct the user's focus to specific robots as prompted by interruptions.

Chapter 3

Preliminaries

In this chapter, we provide mathematical preliminaries and definitions relevant to the thesis.

3.1 Graphs

A graph G is defined as a pair of sets $G = (V, E)$, where the set V represents the vertices in the graph and $E \subseteq V \times V$ is the set of edges between the vertices. An edge in E is an ordered tuple (u, v) connecting $u \in V$ to $v \in V$.

Definition 3.1 (Neighbours of a Vertex)**.** *In a graph $G = (V, E)$, the neighbours of a vertex $u \in V$ is the set of vertices to which there exists a direct edge connecting them to u, i.e.,*

$$\mathcal{N}(u) = v \in V \mid (u, v) \in E.$$

For brevity, an edge (u, v) can be denoted as e_{uv}.

Definition 3.2 (Weighted Graph)**.** *A weighted graph is defined as triplet $G = (V, E, c)$, where $c : E \to \mathbb{R}$ is a function assigning weights (costs) to each edge in the graph.*

When the set of edges E and the associated edge weights c remain constant over time, the graph is referred to as a static graph. In this thesis, we also look into time-dependent graphs, where certain properties of the graph are function of time, and therefore, the time spent traversing an edge or waiting at a vertex are also considered.

Definition 3.3 (Time-Dependent Graph)**.** *A time-dependent graph is defined as* $G = (V, E, c, c_w, \tau)$*, where the function* $c : E \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ *assigns costs to edges,* $c_w : V \times \mathbb{R}_{\geq 0} \times \mathbb{R}_{\geq 0} \to \mathbb{R}$ *assigns costs to waiting at vertices, and* $\tau : E \times \mathbb{R}_{\geq 0} \to \mathbb{R}_{\geq 0}$ *assigns durations (traversal time) to each edge in the graph.*

We represent the cost of an edge $e = (u, v)$ as $c(e, t)$ or $c(u, v, t)$ for a departure time $t \in \mathbb{R}_{\geq 0}$. Similarly, duration of and edge can be represented as $\tau(e, t)$ or $\tau(u, v, t)$. The cost of waiting for a duration of $w_u \in \mathbb{R}_{\geq 0}$ can be represented as $c_w(u, w_u, t)$. The time variable t can either be continuous ($\in \mathbb{R}_{\geq 0}$) or discrete ($\in \mathbb{Z}_{\geq 0}$).

Definition 3.4 (Topological Path)**.** *A topological path* $P^{\top}$ *is defined as a sequence of vertices* $\langle v_1, v_2, \ldots v_n \rangle$*, where*

- $v_i \in V$ *for* $i \in [1, n]$*, and*
- $(v_i, v_{i+1}) \in E$ *for* $1 \leq i \leq n - 1$.

A topological path represents a sequence of edges taken to travel from one vertex to another in a graph. In time-dependent graphs, for a path to be valid, additional conditions need to be met, which can be expressed using an *execution path.*

Definition 3.5 (Execution Path)**.** *An execution path* P^X *is defined as a sequence of triplets* $\langle (v_1, t_1, w_1), (v_2, t_2, w_2), \ldots, (v_n, t_n, w_n) \rangle$*, where*

- $v_i \in V$ *for* $i \in [1, n]$*,*
- $(v_i, v_{i+1}) \in E$ *for* $1 \leq i \leq n - 1$*,*
- $w_i \in \mathbb{R}_{\geq 0}$ *for* $i \in [1, n]$*, and*
- $t_{i+1} = t_i + w_i + \tau(v_i, v_{i+1}, t_i + w_i)$ *for* $1 \leq i \leq n - 1$.

Here, w_i denotes the waiting time at vertex v_i and t_i denotes the arrival time at vertex v_i. It is possible to have additional constraints over the waiting limits, or edge traversability depending on the problem requirements. In later chapters, we denote the execution paths P^X simply as P, dropping the superscript X and only provide clarifications where required.

3.1.1 Shortest Path Problems

In a general form, the shortest path problem can be defined as follows for the static and time-dependent graphs.

3.1.1.1 Static Graphs

On a static weighted graph, the shortest path problem aims to find a topological path with the minimum total weight or cost between two specified vertices, a source vertex and a goal vertex.

Problem 3.1. *Given a weighted graph $G = (V, E, c)$, and the set $\mathcal{P}^\top$ of all possible topological paths $P^\top$ of arbitrary length $n(\geq 0)$, such that $P^\top := \langle v_1, v_2, \dots v_n \rangle$, find the path with the minimum total weight or cost between vertices s and g, i.e.,*

$$\begin{aligned} \min_{P^\top \in \mathcal{P}^\top} \quad & \sum_{i=1}^{n-1} c(v_i, v_{i+1}) \\ s.t. \quad & v_1 = s, \\ & v_n = g. \end{aligned}$$

3.1.1.2 Time-Dependent Graphs

Shortest path problem in time-dependent graphs is commonly implemented as a time-constrained problem. Given a time-dependent graph, the shortest path problem aims to find an execution path with the minimum total weight or cost between two specified vertices under the constraint that the total duration of the path does not exceed some given limit.

Problem 3.2. *Given a time-dependent graph $G = (V, E, c, c_w, \tau)$, and the set $\mathcal{P}^X$ of all valid execution paths P^X of arbitrary length n, such that $P^X := \langle (v_1, t_1, w_1), (v_2, t_2, w_2), \dots, (v_n, t_n, w_n) \rangle$, find the path with the minimum total cost between vertices s and g such that the path duration does not exceed $T \in \mathbb{R}_{\geq 0}$, i.e.:*

$$\begin{aligned} \min_{P^X \in \mathcal{P}^X} \quad & \sum_{i=1}^{n-1} c_w(v_i, w_i, t_i) + c(v_i, v_{i+1}, t_i) \\ s.t. \quad & v_1 = s, \\ & v_n = g, \\ & t_n \leq T. \end{aligned}$$

3.1.2 Graph Search for Shortest Path Problems

Performing an informed search through the given graph is a common class of methods to solve the shortest path problems on static graphs (Problem 3.1). Various algorithms have been developed in this solution domain, each with its own strengths and limitations. One widely used approach is the A^* algorithm, which combines elements of the Dijkstra's algorithm [167] with an admissible heuristics [168] to guide the search towards the goal vertex more efficiently.

Let $h : V \to \mathbb{R}_{\geq 0}$ be a function denoting the heuristic value assigned to vertex v, and let $c^*(v)$ denote the optimal (minimum) cost from vertex v to the goal.

Definition 3.6 (Admissible Heuristic)**.** *A heuristic function $h(v)$ is admissible if for every vertex v in the graph, it satisfies the condition:*

$$h(v) \leq c^*(v).$$

In other words, the heuristic function provides a lower bound estimate of the cost to reach the goal. By considering both the actual cost from the source vertex and an estimated cost to the goal vertex, A^* can efficiently find the shortest path in static graphs. Algorithm 3.1 gives a pseudo-code for finding the shortest (minimum cost) path in a static graph.

Algorithm 3.1 A^*

```
1: **Input:** $G = (V, E, c)$, source vertex $s$, goal vertex $g$, heuristic function $h$
2: **Output:** Minimum cost path $P$ from $s$ to $g$
3: $d(v) \leftarrow \infty$ for all $v \in V$
4: $d(s) \leftarrow 0$
5: $Q \leftarrow$ initialize priority queue
6: $Q$.insert$(s, d(s) + h(s))$
7: $pred \leftarrow$ initialize predecessor info
8: while $Q$ not empty do
9:    $u \leftarrow Q$.extract-min()    // Vertex with minimum $d(u) + h(u)$
10:    if $u = g$ then
11:       break
12:    for all $v \in$ neighbors$(u)$ do
13:       $dist \leftarrow d(u) + c(u, v)$
14:       if $dist < d(v)$ then
15:          $d(v) \leftarrow dist$
16:          $Q$.insert$(v, d(v) + h(v))$
17:          $pred(v) \leftarrow u$    // Store predecessor vertex
18: $P \leftarrow [\,]$    // Initialize empty list for path
19: $v \leftarrow g$    // Start from the goal vertex
20: while $v \neq s$ do
21:    $P$.append$(v)$    // Add current vertex to the path
22:    $v \leftarrow pred(v)$    // Move to the predecessor vertex
23: $P$.append$(s)$
24: $P$.reverse
25: return $P$
```

As we discuss in Chapter 4, this algorithm can also be adapted to solve a discrete time version of the time-dependent shortest path problem (Problem 3.2) by creating a copy of each vertex for each time step $t \in [0, T]$, and adding appropriate edges. For instance, if $e_{uv} \in E$, then new edges will connect u^t to $v^{t+\tau(u,v,t)}$ for all $t \in [0, T - \tau(u, v, t)]$. We provide more details and an admissible heuristic in Chapter 4.

3.1.3 Time Dependent Shortest Path Problems

The Time Dependent Shortest Path (TDSP) Problems can also be solved using an iterative approach as shown by Cai et al. [169]. In this thesis, we are concerned about a variant of TDSP problems which has constraints on the maximum wait time allowed at a vertex. Cai et al. [169] present a solution to the problem of finding time-varying constrained shortest path (TCSP) from a given start vertex to other vertices in a time-varying graph constrained by a maximum time T, and having constrained wait times (CWT). The algorithm works in a discrete-time setting. In Alg. 3.2, we provide the pseudo-code.

Algorithm 3.2 *TCSP-CWT*

```
1: Input: $G = (V, E, c, \tau)$, source vertex $s$, time constraint $T$, waiting constraints $\overline{w}_v$ for
   all $v \in V$
2: Output: Minimum cost $d_C^*(v)$ for all $v \in V$ under the given time constraint
3: $d_C(v,t) \leftarrow \infty$ for all $v \in V$, $t \in \{0,\dots,T\}$
4: $d_C(s,0) \leftarrow 0$
5: $Q_v \leftarrow \{d_C(v,0)\}$ for all $v \in V$ // Initialize priority heaps for each vertex
6: $d_C^m(v,0) \leftarrow d_C(v,0)$ for all $v \in V$
7: Sort all values $t + \tau(u,v,t)$ for all $t \in \{1,\dots,T\}$ and $(u,v) \in E$
8: for $t = \{1,\dots,T\}$ do
9:     for all $(u,v) \in E$ do
10:        $\Gamma_C(u,v,t) \leftarrow \infty$
11:    for all $\{(u,v) \in E$ and all $t_D$ s.t. $t_D + \tau(u,v,t_D) = t$ do
12:        $\Gamma_C(u,v,t) \leftarrow \min\{\Gamma_C(u,v,t),\ d_C^m(u,t_D) + c(u,v,t_D)\}$
13:    for all $v \in V$ do
14:        $d_C(v,t) \leftarrow \min_{u|\{(u,v)\in E\}} \Gamma(u,v,t)$
15:        $Q_v$.insert$(d_C(v,t))$
16:        if $t > \overline{w}_v$ then
17:            $Q_y$.delete$(d_C(v,t-\overline{w}_v-1))$
18:        $t_A \leftarrow \arg\min_t d_C(v,t)$     // Can be extracted from $Q_v$
19:        $d_C^m(v,t) \leftarrow d_C(v,t_A)$
20:        $d_C^*(v) \leftarrow \min_{0\le t\le T} d_C(v,t)$
21: return $d_C^*$     // Return the final minimum cost
```

Note that this algorithm does not consider cost of waiting and does not return the required path, which will require some additional steps similar to Alg. 3.1.

3.2 Submodularity

In this section we review some essential concepts related to submodular set functions and submodular sequence functions [170–172]. The concept of submodularity provides a foundational framework for various optimization challenges characterized by diminishing returns. This property has significant implications across many disciplines, serving as a representation of scenarios where resource allocation yields decreasing improvements. This mathematical principle holds particular relevance in real-world applications, such as resource distribution and information retrieval.

Let E be a finite set. A function f over E assigns a value to every subset of E, denoted as $f : 2^E \to \mathbb{R}$.

Definition 3.7 (Normalized and Monotone)**.** *The function f is said to be normalized if $f(\emptyset) = 0$. It is also considered monotone non-decreasing if, for any subsets A and B of E where $A \subseteq B \subseteq E$, we have $f(A) \leq f(B)$.*

Definition 3.8 (Submodularity)**.** *The function f is called submodular if, for all subsets A and B of E where $A \subseteq B \subseteq E$, and for all elements $x \in E \setminus B$, the inequality $f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$ holds.*

Submodular functions exhibit the property of *diminishing marginal returns.* This means that the contribution of any element x to the total value of a set function decreases as the set size increases. Formally, for all subsets A and B of E where $A \subseteq B \subseteq E$, it follows that $f(A \cup \{x\}) - f(A) \geq f(B \cup \{x\}) - f(B)$.

3.2.1 Sequence Submodularity

Instead of sets, let us consider a function f defined over sequences derived from a base set E. For our purposes, a sequence $\mathcal{A} = \langle a_1, \ldots, a_k \rangle$ of length $k \in \mathbb{Z}_{>0}$ is composed of k elements from the base set E, i.e., $a_i \in E$. Given two sequences $\mathcal{A} = \langle a_1, \ldots, a_k \rangle$ and $\mathcal{B} = \langle b_1, \ldots, b_\ell \rangle$ defined over the same base set, they can be combined into a larger sequence by concatenation denoted as $\mathcal{A} \parallel \mathcal{B} = \langle a_1, \ldots, a_k, b_1, \ldots, b_\ell \rangle$. A sequence $\mathcal{A}$ is called a *subsequence* of another sequence $\mathcal{B}$, represented as $\mathcal{A} \subseteq \mathcal{B}$, if there exists a sequence $\mathcal{C}$ such that $\mathcal{B} = \mathcal{A} \parallel \mathcal{C}$. A *sequence function* f maps sequences derived from a base set E to real numbers. The value of a sequence function depends on both the elements in the sequence and the order of those elements.

Let $\mathcal{A} = \emptyset$ denote an empty sequence, and let $E \setminus \mathcal{A}$ denote the set of elements of E not in the sequence $\mathcal{A}$.

Definition 3.9 (Sequence Normalized and Monotone)**.** *The sequence function f is said to be normalized if for an empty sequence $\mathcal{A} = \emptyset$, $f(\mathcal{A}) = 0$. The sequence function f is monotone non-decreasing if for all subsequences $\mathcal{A}$ of a sequence $\mathcal{B}$, i.e., $\mathcal{A} \subseteq \mathcal{B}$, $f(\mathcal{A}) \leq f(\mathcal{B})$.*

Definition 3.10 (Sequence Submodularity)**.** *The function f is sequence submodular if for all sequences $\mathcal{A}$ and $\mathcal{B}$ derived from the set E such that $\mathcal{A} \subseteq \mathcal{B}$, and for all sequences $\mathcal{C}$ derived from the set $E \setminus \mathcal{B}$, we have $f(\mathcal{A} \parallel \mathcal{C}) - f(\mathcal{A}) \geq f(\mathcal{B} \parallel \mathcal{C}) - f(\mathcal{B})$.*

The property of submodularity significantly influences optimization methods for a given problem, particularly in the context of greedy algorithms. Greedy algorithms, that sequentially add elements to maximize the marginal gain, are shown to have a remarkably good performance in maximizing submodular functions. The *set greedy algorithm* starts with an empty set S and repeatedly adds an element $x \in E \setminus S$ to S that maximizes the marginal gain $f(S \cup \{x\}) - f(S)$. Similarly, the *sequence greedy algorithm* begins with an empty sequence $\mathcal{S}$ and repeatedly appends an element $x \in E \setminus \mathcal{S}$ that maximizes the marginal gain $f(\mathcal{S} \parallel \{x\}) - f(\mathcal{S})$.

3.3 Restless Multi-Armed Bandits

In this section we provide an overview of Restless Multi-Armed Bandits (RMAB), indexability and the Whittle index policy.

3.3.1 Restless Bandit Process

A restless bandit (RB) process is a controlled Markov process $(\tilde{\mathcal{Z}}, \{0,1\}, \tilde{T}, \tilde{C}, \tilde{z}_0)$ where $\tilde{\mathcal{Z}}$ is the state space, $\{0,1\}$ is the action space, $\tilde{T}\colon \tilde{\mathcal{Z}} \times \tilde{\mathcal{Z}} \times \{0,1\} \to \mathbb{R}_{[0,1]}$ is the transition probability function, $\tilde{C} : \tilde{\mathcal{Z}} \times \{0,1\} \to \mathbb{R}$ is the per-step cost function, and $\tilde{z}_0 (\in \tilde{\mathcal{Z}})$ is the initial state. By convention, action 0 is called the *passive* action and action 1 is called the *active* action.

3.3.2 Restless Multi-armed Bandit Problem

A Restless Multi-armed Bandit (RMAB) is a collection of K independently evolving RBs $(\tilde{\mathcal{Z}}^k, \{0,1\}, \tilde{T}^k, \tilde{C}^k, \tilde{z}_0^k)$, $k \in \mathcal{K} := \{1, \ldots, K\}$. Each process is conventionally called an *arm*. A decision-maker selects at most M arms ($M < K$) at each time instance. Let $\tilde{Z}_t^k$ and $\tilde{A}_t^k$ denote the state of arm k and the action chosen for arm k at time t. Let $\{\tilde{\boldsymbol{Z}}_t\}_{t\geq 0}$ and $\{\tilde{\boldsymbol{A}}_t\}_{t\geq 0}$ where

$$\tilde{\boldsymbol{Z}}_t \coloneqq (\tilde{Z}_t^1, \dots, \tilde{Z}_t^K) \text{ and } \tilde{\boldsymbol{A}}_t \coloneqq (\tilde{A}_t^1, \dots, \tilde{A}_t^K),$$

denote the states and actions of all arms. As the dynamics of each arm are independent, we have

$$\tilde{T}(\tilde{\boldsymbol{Z}}_{t+1}|\tilde{\boldsymbol{Z}}_t, \tilde{\boldsymbol{A}}_t) = \prod_{k\in\mathcal{K}} \tilde{T}^k(\tilde{Z}_{t+1}^k|\tilde{Z}_t^k, \tilde{A}_t^k).$$

The instantaneous cost of the system is the sum of costs incurred by each arm. The performance of any time homogeneous Markov policy $\tilde{\boldsymbol{\pi}} : \prod_{k=1}^K \mathcal{Z}^k \to \{\boldsymbol{a} \in \{0,1\}^K : ||\boldsymbol{a}||_1 \leq M\}$ is measured by

$$\tilde{J}(\tilde{\boldsymbol{\pi}}) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \sum_{k=1}^{K} \tilde{C}^k(\tilde{Z}_t^k, \tilde{\boldsymbol{\pi}}(\tilde{Z}_t^k)) \middle| \tilde{z}_0^1, \dots, \tilde{z}_0^K\right], \tag{3.1}$$

where $\gamma \in (0,1)$ denotes the discount factor. Finally, the RMAB optimization problem is as follows:

Problem 3.3. *Given a discount factor $\gamma \in (0,1)$, a collection of arms $\{(\tilde{\mathcal{Z}}^k, \{0,1\}, \tilde{T}^k, \tilde{C}^k, \tilde{z}_0^k)\}_{k\in\mathcal{K}}$, and the number M of arms to be chosen at each time, choose a policy $\tilde{\boldsymbol{\pi}} : \prod_{k=1}^K \mathcal{Z}^k \to \{\boldsymbol{a} \in \{0,1\}^K : ||\boldsymbol{a}||_1 \leq M\}$ that minimizes $\tilde{J}(\tilde{\boldsymbol{\pi}})$.*

Even though the arms operate independently, the actions applied to them are not independent. They are coupled through the operator allocation constraints. Therefore, we cannot decompose the dynamic programming into multiple smaller MDPs. As discussed earlier, the Whittle index policy is one of the commonly used heuristics to solve an RMAB problem [173] and it addresses the scalability issues of dynamic programming-based solutions. This policy is computationally efficient and it readily generalizes to the setting where K or M changes over time. Next, we present the required definitions.

3.3.3 Indexability and the Whittle index policy

In this section, we restrict our discussion to a single arm and therefore omit the superscript k for the ease of notation. Consider an arm $(\tilde{\mathcal{Z}}, \{0,1\}, \tilde{T}, \tilde{C}_\lambda, \tilde{z}_0)$ where, for some penalty $\lambda \in \mathbb{R}$, modify the per-step cost as

$$\tilde{C}_\lambda(z,a) \coloneqq \tilde{C}(z,a) + \lambda a, \quad \forall\, z \in \tilde{\mathcal{Z}}, a \in \{0,1\}. \tag{3.2}$$

Then the performance of any given time-homogeneous Markov policy $\tilde{\pi} : \tilde{\mathcal{Z}} \to \{0,1\}$ is given by

$$\tilde{J}_\lambda(\tilde{\pi}) := \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \tilde{C}_\lambda(\tilde{Z}_t, \tilde{\pi}(\tilde{Z}_t)) \middle| \tilde{Z}_0 \sim \tilde{z}_0\right]. \tag{3.3}$$

Now consider the following auxiliary problem:

Problem 3.4. *Given an arm $(\tilde{\mathcal{Z}}, \{0,1\}, \tilde{T}, \tilde{C}, \tilde{z}_0)$, the discount factor $\gamma \in (0,1)$ and the penalty $\lambda \in \mathbb{R}$, choose a Markov policy $\tilde{\pi} : \tilde{\mathcal{Z}} \to \{0,1\}$ to minimize $\tilde{J}_\lambda^{(\tilde{\pi})}(\tilde{z}_0)$ given by (3.3).*

Problem 3.4 is a Markov decision process. Let us denote the optimal policy of Problem 3.4 by $\tilde{\pi}_\lambda$. It is assumed that the optimal policy picks passive action at any state where both the active and passive actions result in same expected cost. Next, define passive sets and indexability.

Definition 3.11 (Passive set)**.** *Given $\lambda \in \mathbb{R}$, the passive set $\tilde{\mathcal{P}}(\lambda)$ is the set of states where passive action is prescribed by $\tilde{\pi}_\lambda$, i.e.,*

$$\tilde{\mathcal{P}}(\lambda) := \{z \in \mathcal{Z} : \tilde{\pi}_\lambda(z) = 0\}.$$

Definition 3.12 (Indexability)**.** *An arm is indexable if $\tilde{\mathcal{P}}(\lambda)$ is non-decreasing in λ, i.e., for any $\lambda_1, \lambda_2 \in \mathbb{R}$,*

$$\lambda_1 \leq \lambda_2 \implies \tilde{\mathcal{P}}(\lambda_1) \subseteq \tilde{\mathcal{P}}(\lambda_2).$$

A RMAB problem is indexable if all n arms are indexable.

Definition 3.13 (Whittle index)**.** *For an indexable arm, the Whittle index of the state z of an arm is the smallest value of λ for which state z is part of $\tilde{\mathcal{P}}(\lambda)$, i.e.,*

$$\tilde{w}(z) = \inf\left\{\lambda \in \mathbb{R} : z \in \tilde{\mathcal{P}}(\lambda)\right\}. \tag{3.4}$$

Equivalently, the Whittle index $\tilde{w}(z)$ is the smallest value of λ for which $\tilde{\pi}_\lambda$ is indifferent between the active action and passive action when the arm is in state z.

The Whittle index policy is as follows: *At each time, compute the Whittle indices of the current state of all arms and select the arms in states with M highest Whittle indices (provided they are positive).*

Chapter 4

Route Planning in Collaborative Human-Robot Teams

TL;DR: We define the route planning problem for a single robot, introduce the notion of budgets and critical departure times, and propose an optimal solution method based on these concepts. We then extend this approach to address the problem of prioritizing route planning for multiple robots in the fleet.

Robots have come a long way in the past decades, with increasing levels of autonomy transforming the way they operate in different domains, from factories and warehouses to homes and public spaces [11, 174, 175]. However, navigating dynamic environments effectively continues to be a formidable challenge. Despite the significant strides made in robot autonomy, human oversight remains vital in enhancing safety, efficiency or to comply with regulatory requirements. For example, a robot navigating through an urban environment must abide by traffic regulations and may require human assistance in busy or construction areas to ensure safety or expedite operations. Similarly, in an exploration task, robots may require replanning due to changes in the environment, while the supervisor has already committed to a supervision schedule for other robots and is only intermittently available. By considering the operator's availability and environmental restrictions, robots can plan their routes more efficiently, avoid unnecessary waiting and decide when to use human assistance.

In this chapter, we consider the problem of robot planning with the objective of finding the fastest route between two configurations. We demonstrate our approach through an example of robot navigation in an urban environment with intermittent operator availability, varying travel speeds, and waiting limits. In this context, a route can be defined as an *execution path* that includes details about the locations to visit, necessary waiting periods, and specific segments where operator assistance should be scheduled. Figure 4.1 illustrates the problem overview[1]. Specifically, we consider a city road network where the robot can traverse through different locations either autonomously or with the assistance of a human supervisor, each taking different amounts of time. However, the supervisor is only available at certain times, and the robot has a limited amount of time to wait at a location before it must move on to its next destination. By formulating the problem in this way, we aim to address the challenge of collaborative robot planning in real-world environments where the availability of human supervisors may be limited and thus can affect the optimal route for the robot.

The problem of robot path planning with operator allocation in dynamic networks is inspired by real-world scenarios where the availability of human assistance and thus the robot's speed of travel and its ability to traverse certain paths can change over time, e.g., [176]. Traditional methods, such as time-dependent adaptations of the Dijkstra's algorithm, are not designed to handle situations in dynamic environments where waiting is limited, and the task durations may not follow the first-in-first-out (FIFO) property [177]. This means that a robot may arrive at its target location earlier by departing later from its previous location, for example, by using human assistance. To address these challenges, we draw on techniques from the time-dependent shortest path literature to solve the problem. Unfortunately, existing optimal solution techniques are severely limited by their computational runtime. In this chapter, we propose a novel algorithm that is guaranteed to find optimal solution and runs orders of magnitude faster than existing solution techniques.

Contributions: The main contributions of this work are as follows:

1) We propose a novel graph search algorithm for the collaborative planning problem with intermittent human availability. The algorithm operates by intelligently selecting the times of exploration and by combining ranges of arrival times into a single search node.

2) We provide the proof that the algorithm generates optimal solutions.

3) We demonstrate, using simulations, the effectiveness of our approach in a city road-network, and show that it outperforms existing approaches in terms of computational time

[1]The map shown in Fig. 4.1 shows the road network of the city of Waterloo, generated using QGIS, OpenStreetMap and OpenRouteService.

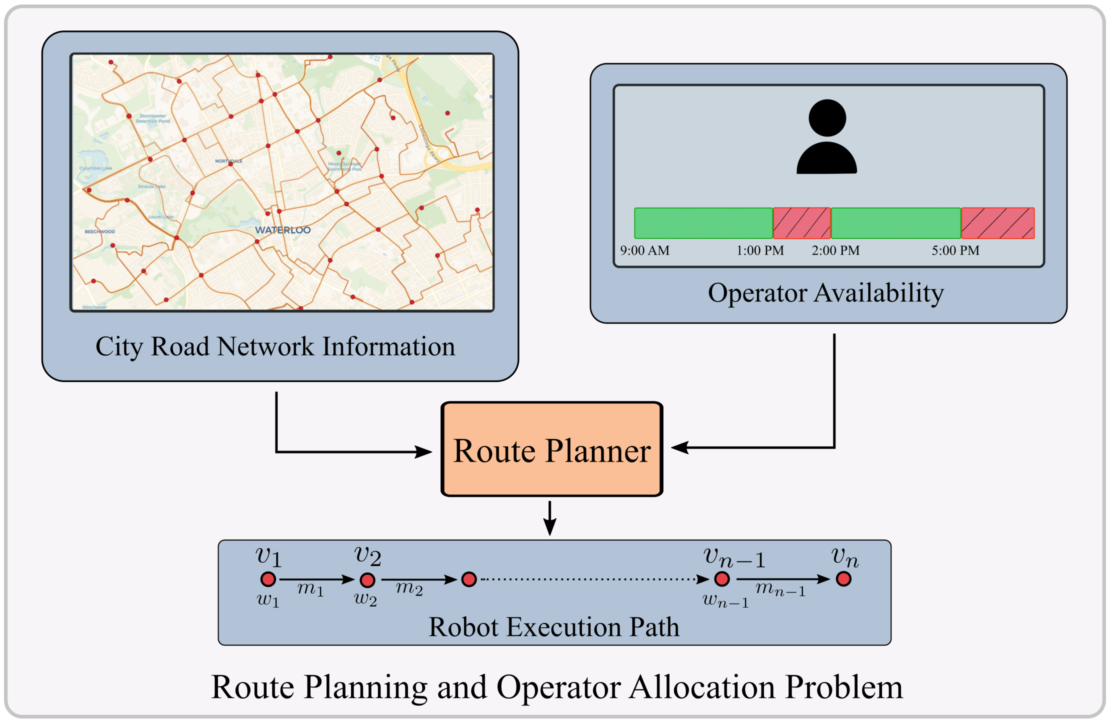


Figure 4.1: Problem Overview: Given the information of a city road network with speed and waiting restrictions, and the availability of human assistance, the objective is to determine a route for the robot in form of an *execution path* that results in the fastest arrival at a goal location. The execution path consists of three components: (1) the vertices or locations to navigate through, denoted as $(v_1, v_2, \ldots, v_n)$; (2) the amount of waiting required at intermediate locations, denoted as $(w_1, w_2, \ldots, w_{n-1})$; and (3) whether to use human assistance, denoted as $(m_1, m_2, \ldots, m_{n-1})$. In this work, we propose a novel and efficient planning algorithm to obtain the required execution path.

and/or solution quality.

4) We extend the problem to multiple robots and show how a simple greedy method can be used to efficiently prioritize a batch of route planning problems.

4.1 Background and Related Work

In this section, we discuss some relevant studies from the existing literature in the area of robot planning with human supervision/collaboration. We also look into how the presented problem can be solved using existing techniques from related fields.

Planning with Human Collaboration: The problem of task allocation and path planning for robots operating in collaboration with humans has been studied extensively in recent years. Researchers have proposed approaches such as a data-driven approach for human-robot interaction modelling that identifies the moments when human intervention is needed [37], and a probabilistic framework that develops a decision support system for the human supervisors, taking into account the uncertainty in the environment [16]. In the context of autonomous vehicles, studies have investigated cooperative merging of vehicles at highway ramps [178] and proposed a scheduling algorithm for multiple robots that jointly optimize task assignments and human supervision [179].

Task allocation is a common challenge in mixed human-robot teams across a variety of applications, including manufacturing [180], routing [30], surveying [181], and subterranean exploration [176]. In addition, the problem of computing the optimal path for a robot under time-varying human assistance bears similarity to queuing theory applications, such as optimal fidelity selection [14] and supervisory control of robots via a multi-server queue [182]. These studies provide insights into allocating assistance and path planning for robots in collaborative settings, but do not address our specific problem of computing the optimal path for a robot under bounded waiting and intermittent assistance availability. Additionally, our problem differs in that the robot can operate autonomously even when assistance is available, i.e., the collaboration is optional.

Time-Dependent Shortest Paths: The presented problem is also closely related to time-dependent shortest path (TDSP) problems, which aim to find the minimum cost or minimum length paths in a graph with time-varying edge durations [177, 183]. Existing solution approaches include planning in graphs with time-activated edges [177], implementing modified A^* [184], and finding shortest paths under different waiting restrictions [185, 186]. Other studies have explored related problems such as computing optimal temporal walks under waiting constraints [187], and minimizing path travel time with penalties or limits on waiting [188]. Many studies in TDSP literature have addressed the first-in-first-out (FIFO) graphs [177], while others have explored waiting times in either completely restricted or unrestricted settings. However, the complexities arising from non-FIFO properties, bounded waiting and the need to make decisions on the mode of operation, i.e., autonomous or assisted, have not been fully addressed in the existing literature [186, 188, 189].

The most relevant solution technique that can be used to solve our problem is presented by Cai et al. [169], which solves a TDSP problem where the objective is to minimize the path cost constrained by the maximum arrival time at the goal vertex. This method iteratively computes the minimum cost for all vertices for increasing time constraint value. A time-expanded graph search method [190] is another way of solving the presented problem by creating separate edges for autonomous and assisted modes. We discuss these two methods in more detail in Sec. 4.5. As we will see, the applicability of these solution techniques to our problem is limited due to their poor scalability for large time horizons and increasing graph size.

4.2 Problem Definition

The problem can be defined as follows. We are given a directed graph $G = (V, E)$, modelling the robot environment, where each edge $e \in E$ has two finite travel times corresponding to the two modes of operation: an autonomous time $\tau(e, 0)$ and an assisted time $\tau(e, 1)$, with the assumption that $\tau(e, 0) \geq \tau(e, 1)$. When starting to traverse an edge, the robot must select the mode of operation for the traversal that is used for the entire duration of the edge. While the autonomous mode is always available, the assisted mode can only be selected if the supervisor is available for the entire duration of the edge (under assisted mode). The supervisor's availability is represented by a binary function μ, with $\mu([t_1, t_2]) = 1$ indicating availability during the time window $[t_1, t_2]$, and 0 otherwise. Additionally, at each vertex $v \in V$, the robot can wait for a maximum duration of $\overline{w}_v \geq 0$ before starting to traverse an outgoing edge.

The robot's objective is to determine how to travel from a start vertex to a goal vertex. This can be represented as an execution path P, specified as a list of edges to traverse, the amount of waiting required at intermediate vertices and the mode of operation selected for each edge. The objective of this problem is to find an execution path (simply referred to as a path) from a start vertex $s \in V$ to a goal vertex $g \in V \setminus \{s\}$, such that the arrival time at g is minimized.

Problem 4.1. *Given a set $\mathcal{P}$ of all possible paths P of arbitrary length n, such that $P :=$*

$\langle(v_1, t_1, w_1, m_1),\ (v_2, t_2, w_2, m_2), \ldots, (v_n, t_n, w_n, m_n)\rangle$, *we can write the problem objective as follows:*

$$\begin{aligned}
\min_{P \in \mathcal{P}} \quad & t_n \\
s.t. \quad & v_1 = s,\ v_n = g \\
& e_{v_i v_{i+1}} \in E && \forall\ i \in [1, n-1] \\
& t_{i+1} = t_i + w_i + \tau(e_{v_i v_{i+1}}, m_i) && \forall\ i \in [1, n-1] \\
& w_i \leq \overline{w}_{v_i} && \forall\ i \in [1, n-1] \\
& m_i = 1 \Rightarrow \mu([t_i + w_i, t_{i+1}]) = 1 && \forall\ i \in [1, n-1].
\end{aligned}$$

The first constraint ensures that the path starts at s and ends at g. The second constraint ensures that the topological path is valid in the graph. The third constraint ensures that the path does not violate travel duration requirements at any edge. Fourth constraint ensures that the waiting restrictions are met at each vertex. Finally, the fifth condition ensures that an edge can only be assisted if the operator is available throughout the edge traversal.

To efficiently solve this problem, we must make three crucial decisions: selecting edges to travel, choosing the mode of operation, and determining the waiting time at each vertex. Our proposed method offers a novel approach to computing the optimal solution. However, before delving into the details of our solution, it is necessary to grasp the concept of budget and how new nodes are generated during the search process.

4.3 Budget and Node Generation

Since the robot is allowed to wait (subject to the waiting limits), it is possible to delay the robot's arrival at a vertex by waiting at one or more of the preceding vertices. Moreover, the maximum amount of time by which the arrival can be delayed at a particular vertex depends on the path taken from the start to that vertex. Our key insight is that this information about the maximum delay can be used to efficiently solve the given problem by removing the need to examine the vertices at every possible arrival time. We achieve this by augmenting the search space into a higher dimension, using additional parameters with the vertices of the given graph. A node in our search is defined as a triplet (x, a_x, b_x), corresponding to a vertex $x \in V$, arrival time $a_x \in \mathbb{R}_{\geq 0}$ and a budget $b_x \in \mathbb{R}_{\geq 0}$. The *budget* here defines the maximum amount of time by which the arrival at the given vertex can be delayed. Thus, the notion of budget allows a single node (x, a_x, b_x) to represent a range of arrival times from $[a_x, a_x + b_x]$ at vertex x. Therefore, the allowed departure time from this vertex lies in the interval $[a_x, a_x + b_x + \overline{w}_x]$. Thus budget of a node can also be regarded as the additional wait that one can afford before moving to the next vertex.

4.3.1 Node Generation

The proposed algorithm is similar to standard graph search algorithms, where we maintain a priority search queue, with nodes prioritized based on the earliest arrival time (plus any admissible heuristic). Nodes are then extracted from the queue, their *neighbouring* nodes are generated and are added to the queue based on their priority. Since in our search a node is defined by the vertex, arrival time and budget, we must determine these parameters for the newly generated nodes when exploring a given node. To characterize the set of nodes to be generated during the graph search in our proposed algorithm, we define the notion of direct reachability as follows.

Definition 4.1 (Direct reachability)**.** *A node (y, a_y, b_y) is said to be* directly reachable *from a node (x, a_x, b_x) if x and y are connected by an edge, i.e., $e_{xy} \in E$, and it is possible to achieve all arrivals times in $[a_y, a_y + b_y]$ at y through edge e_{xy} for some departure time $t_D \in [a_x, a_x + b_x + \overline{w}_x]$ from x and some mode of travel.*

As an example, consider a node $(x, 10, 2)$ with $\tau(e_{xy}, 0) = 5$ and $\overline{w}_x = 3$. Then the nodes $(y, 15, 5), (y, 16, 4)$ and $(y, 17, 3)$ are a few directly reachable nodes from $(x, 10, 2)$ (corresponding to departure times 10, 11 and 12, respectively).

Like standard graph search methods, our algorithm aims to generate all nodes directly reachable from the current node during the exploration process. One approach is to generate all directly reachable nodes from the given node (x, a_x, b_x) for all possible departure times in $[a_x, a_x + b_x + \overline{w}_x]$. However, this results in redundancy when multiple nodes can be collectively represented using a single node with a suitable budget.

As the operator availability changes, the possible arrival times at the next vertex may present themselves as separate blocks of time. A block of arrival times can be represented using a single node, and thus we only need to generate a new node for each arrival time block. To understand this, we consider the example given in Fig. 4.2, where a node (x, a_x, b_x) is being extracted from the queue, and we want to generate the nodes corresponding to a neighbouring vertex y. The arrival time range at x, $[a_x, a_x + b_x]$ is shown

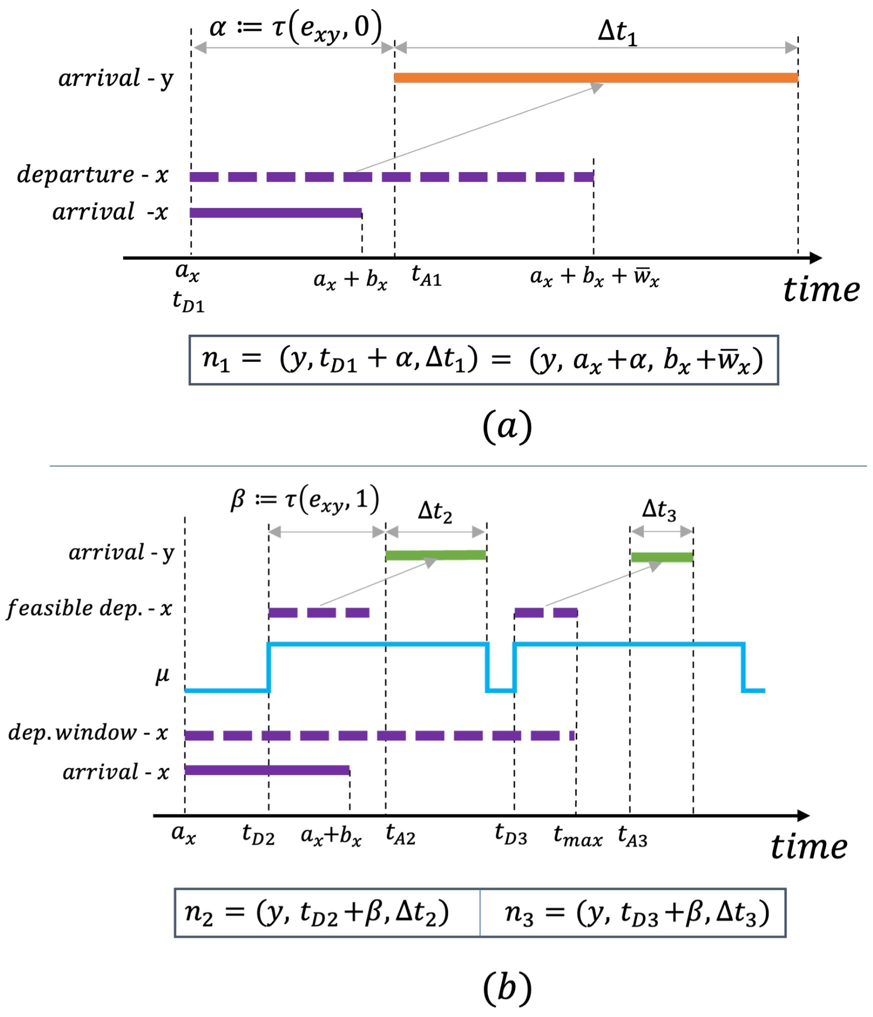

$\alpha \coloneqq \tau(e_{xy}, 0)$
Δt_1
arrival - y
departure - x
arrival - x
a_x
t_{D1}
$a_x + b_x$
t_{A1}
$a_x + b_x + \overline{w}_x$
time
$n_1 = (y, t_{D1} + \alpha, \Delta t_1) = (y, a_x + \alpha, b_x + \overline{w}_x)$
(a)
$\beta \coloneqq \tau(e_{xy}, 1)$
Δt_2
Δt_3
arrival - y
feasible dep. - x
μ
dep. window - x
arrival - x
a_x
t_{D2}
$a_x + b_x$
t_{A2}
t_{D3}
t_{max}
t_{A3}
time
$n_2 = (y, t_{D2} + \beta, \Delta t_2)$
$n_3 = (y, t_{D3} + \beta, \Delta t_3)$
(b)

Figure 4.2: An example explaining the notion of budget and determining the resulting node when a node (x, a_x, b_x) is explored. (a) Under autonomous mode, the earliest departure from x is a_x (corresponding to no waiting). Since the autonomous mode is always available to the robot, the corresponding arrival time at y forms a single block, shown as a solid orange line. This can be thus represented using a single node n_1. (b) Under the assisted mode, feasible departure times from x are governed by the operator availability function μ, and a time $t_{max} = \min(\alpha - \beta, b_x, \overline{w}_x)$. In this example, the resulting arrival time at y forms two separate blocks, shown as solid green lines on the time axis. Thus, exploring the node under assisted operation generates two nodes, n_2 and n_3.

as solid purple line. The possible departure window $[a_x, a_x + b_x + \overline{w}_x]$ is shown as purple dashed line.

Let us denote the autonomous travel time $\tau(e_{xy}, 0)$ as α and the assisted travel time $\tau(e_{xy}, 1)$ as β. Under autonomous operation, the edge can be traversed by departing at any time in the departure window, resulting in possible arrival time at vertex y in the interval $[a_x + \alpha, a_x + b_x + \overline{w}_x + \alpha]$, shown as solid orange line. Therefore, we can represent these possible arrival times using the node $n_1 = (y, a_x + \alpha, b_x + \overline{w}_x)$, where $a_x + \alpha$ is the earliest arrival time at y and $b_x + \overline{w}_x$ is the new budget. Note that the new budget is increased from the previous value by an amount of $\overline{w}_x$.

Under assisted operation, only a subset of departure window is feasible, as shown in Fig. 4.2(b). This results in separate blocks of arrival times at y, shown as solid green lines. The range of arrival times corresponding to these blocks become the budget values for the new nodes. In the given example, two nodes are generated: $n_2 = (y, t_{D2} + \beta, \Delta t_2)$ and $n_3 = (y, t_{D3} + \beta, \Delta t_3)$.

Note that the feasible departure times are limited by operator availability and t_{max}. The value of t_{max} is the minimum of $b_x + \overline{w}_x$ and $\alpha - \beta$. The former quantity limits the departure from x to $a_x + b_x + \overline{w}_x$, while the latter comes from the observation that any departure time $t_D > a_x + (\alpha - \beta)$ will result in arrival at y at a time $a_y > a_x + \alpha$, and a budget $b_y < b_x + \overline{w}_x$. However, this arrival time range is already covered by the node generated under autonomous operation.

Critical departure times: Note that the earliest arrival times for each of the three new nodes correspond to unique departure times from x (t_{D1}, t_{D2}, t_{D3} in Fig. 4.2). We refer to these times as critical departure times, as exploring a node only at these times is sufficient to generate nodes that cover all possible arrival times at the next vertex. Since in the presented problem, the edge duration depends on the mode of operation selected, the set of critical departure times for a node is a subset of times when the operator availability changes, and thus can be efficiently determined.

Next, we present how these concepts are used by our proposed Budget-A^* algorithm to solve the given problem.

4.4 Budget A^* Algorithm

This section details the proposed Budget A^* algorithm and its three constituent functions: EXPLORE, REFINE and GET-PATH. To recall, a node in our search is defined as a tuple

(x, a_x, b_x). A pseudo-code for the Budget-A^* algorithm is given in Alg. 4.1, and more details on the constituent functions follow. The algorithm initializes an empty priority queue Q,

Algorithm 4.1 Budget A^*

```
1: Input: $G = (V, E), \tau, \mu, \overline{w}, s, g$
2: $Q \leftarrow$ initialize priority queue
3: $S \leftarrow \emptyset$
4: $\psi(x, a) \leftarrow nil$ for all $x \in V, a \in \mathbb{Z}_{\geq 0}$    // Initialize predecessor function
5: $Q$.push($(s, 0, 0)$)
6: while $Q$ not empty do
7:   currNode $:= (x, a_x, b_x) \leftarrow Q$.extract-min()
8:   $S \leftarrow S \cup$ currNode
9:   if $x = g$ then
10:     break
11:   for all $y \in$ neighbors($x$) do
12:     $\mathcal{N} \leftarrow$ EXPLORE($a_x, b_x, \overline{w}_x, e_{xy}, \tau, \mu$)
13:     for all $(y, a_i, b_i, m_i) \in \mathcal{N}$ do
14:       newNode $\leftarrow (y, a_i, b_i)$
15:       $Q, \psi \leftarrow$ REFINE($Q$, newNode, currNode, $\psi, m_i$)
16: $path \leftarrow$ GET-PATH(currNode, $\overline{w}, \psi, \tau$)
```

a processed set S and a predecessor function ψ. It then adds a node $(s, 0, 0)$ to Q denoting an arrival time of exactly 0 at s. The algorithm iteratively extracts the node with the earliest arrival time (plus an admissible heuristic) from Q, adds it to S, and generates new candidate nodes for each of its neighbors using the EXPLORE function. The REFINE function then checks if these nodes can be added to the queue, removes redundant nodes from Q, and updates predecessor information. The algorithm continues until Q is empty or the goal vertex is reached. The GET-PATH function generates the required path using the predecessor data and waiting limits.

4.4.1 Exploration

In this step, we generate a list of potential nodes to add to the search queue. The EXPLORE function takes in several input parameters: arrival time a_x, budget b_x, waiting limit $\overline{w}_x$, edge e_{xy}, travel durations τ and operator availability μ. The function returns a set $\mathcal{N}$ of candidate nodes of the form (y, a_i, b_i, m_i), where a_i, b_i, m_i are the arrival time, budget and the mode of operation respectively, corresponding to all critical departure times from the node (x, a_x, b_x) to vertex y. A pseudo-code is shown in Alg. 4.2.

As discussed in Sec. 4.3, the autonomous mode generates one new node, while assisted mode can generate multiple nodes depending on operator availability, node budget and task duration. The function first adds a node $(y, a_x + \alpha, b_x + \overline{w}_x)$ corresponding to the autonomous mode to $\mathcal{N}$. The node is $(y, a_x + \alpha, b_x + \overline{w}_x)$, where $\alpha = \tau(e_{xy}, 0)$, the autonomous duration of edge e_{xy}.

For the assisted mode, it first computes the maximum useful delay in departure t_{max}. Next, it generates an ordered set $\mathcal{F}$ of feasible departure times from the current node as the times in departure window when it's possible to depart under assisted mode, computed using μ and β (line 5). Lines 7-8 generate a new node (y, a_y, b_y) for each critical departure time t_d, with a budget $b_y = 0$ and arrival time $a_y = t_d + \beta$. The budget is then incremented for each consecutive departure time in $\mathcal{F}$. A gap in $\mathcal{F}$ means a gap in arrival time at y indicating that we have considered the complete arrival time range for that critical departure time. This condition is checked in line 11, and the node $(y, a_y, b_y, 1)$ is added to $\mathcal{N}$.

Algorithm 4.2 EXPLORE$(a_x, b_x, \overline{w}_x, e_{xy}, \tau, \mu)$

```
1:  $\alpha \leftarrow \tau(e_{xy}, 0)$, $\beta \leftarrow \tau(e_{xy}, 1)$
2:  $\mathcal{N} \leftarrow \{(y, a_x + \alpha, b_x + \overline{w}_x, 0)\}$
3:  $t_{max} \leftarrow \min(\alpha - \beta, b_x + \overline{w}_x)$
4:  $\mathcal{T}_D \leftarrow [a_x, a_x + t_{max}]$    // Possible departure window
5:  $\mathcal{F} \coloneqq [t \in \mathcal{T}_D \;\; s.t. \;\; \mu([t, t+\beta]) = 1]$    // Feasible set
6:  for all $t_d \in \mathcal{F}$ do
7:      if $t_d - 1 \notin \mathcal{F}$ then
8:          $(y, a_y, b_y) \leftarrow (y, t_d + \beta, 0)$
9:      else
10:         $b_y \leftarrow b_y + 1$
11:     if $t_d + 1 \notin \mathcal{F}$ then
12:         $\mathcal{N} \leftarrow \mathcal{N} \cup (y, a_y, b_y, 1)$
13: return $\mathcal{N}$
```

Once all departure times in $\mathcal{F}$ are accounted for, the set $\mathcal{N}$ contains all required arrival time and budget pairs (along with the mode of operation) for the given node (x, a_x, b_x) and neighbour y while accounting for the operator availability, the travel time and the waiting limit.

4.4.2 Node Refinement

The REFINE function plays a crucial role in determining the inclusion and removal of nodes in the priority queue Q. Its primary objective is to evaluate the newly generated nodes and assess their relevance among the existing nodes in the queue. To achieve this, the function compares the arrival time range of the new node with those already present in Q (with the same vertex).

Line 5 of Algorithm 4.3 implements a check to identify if any existing node in Q already encompasses the arrival time range of the new node. If such an overlap exists, the new node does not provide additional information and can be disregarded. Conversely, if the condition is not met, the new node is deemed relevant and is added to Q. Subsequently, line 7 of the algorithm conducts a further check to identify any redundant nodes in Q whose arrival time window falls entirely within the window of the new node. These redundant nodes are then removed from Q, ensuring the queue remains efficient. Finally, the function returns the modified queue along with the predecessor function.

In summary, the REFINE function manages the inclusion and removal of nodes in the priority queue, optimizing the utilization of budget in making the search process more efficient.

4.4.3 Path Generation

When the goal vertex is extracted from the priority queue for the first time, its corresponding arrival time represents the earliest arrival time at that vertex (as explained in Section 4.4.4). To construct the execution path from the start to the goal, we utilize the predecessor data stored in the function ψ.

Since the problem does not adhere to the optimal substructure property, merely storing the predecessor vertices as nodes are created is insufficient. It is crucial to capture the precise path leading to the goal vertex. This necessitates storing both the predecessor vertex and its associated arrival time, as a vertex and arrival time pair uniquely identify a search configuration. For a given vertex-time pair (y, a), invoking the function $\psi(y, a)$ retrieves the predecessor node vertex and its earliest arrival time (x, a_x) along with the corresponding mode of travel m_x required to traverse the edge e_{xy}. This information,

Algorithm 4.3 REFINE(Q, `newNode`, `currNode`, ψ, m)

```
1:  toRemove ← $\emptyset$
2:  $(y, a_y, b_y)$ ← newNode
3:  $(u, a_u, b_u)$ ← currNode
4:  for all $(x, a, b) \in Q$ do
5:      if $x = y$ and $a \leq a_y$ and $a + b \geq a_y + b_y$ then
6:          return $Q, \psi$
7:      else if $x = y$ and $a \geq a_y$ and $a + b \leq a_y + b_y$ then
8:          toRemove ← toRemove $\cup (x, a, b)$
9:  $Q \leftarrow Q \setminus$ toRemove
10: $Q \leftarrow Q \cup$ newNode
11: $\psi(y, a_y) = (u, a_u, m)$
12: return $Q, \psi$
```

along with the wait limits, enables us to construct the exact path taken to reach the goal vertex, including the sequence of edges, waiting times at each intermediate vertex, and the mode of operation selected for each edge.

To accomplish this, we employ the GET-PATH function depicted in Alg. 4.4. This function backtracks from the goal vertex to the start, calculating the precise departure time from the predecessor vertex based on the earliest arrival time at the current vertex and the mode of operation (line 6).

Subsequently, the exact arrival time is determined by incorporating the departure times and the maximum allowed waiting $\overline{w}$ (lines 6-9). The resulting path is stored as a list of tuples, with each tuple representing a vertex, its arrival time, waiting time, and the mode of operation utilized. This comprehensive path construction procedure ensures the validity of the generated paths.

4.4.4 Correctness Proof

Let a vertex-time pair (y, a_y) be called directly reachable from a node (x, a_x, b_x), if vertex y can be reached by departing vertex x between time a_x and $a_x + b_x + \overline{w}_x$ through edge $e_{xy} \in E$.

Lemma 1. *Consider a node (x, a_x, b_x) extracted from Q (line 7), and a $y \in neighbors(x)$ inspected in lines 11-15. If (y, a_y) is directly reachable from (x, a_x, b_x), then there is a node*

Algorithm 4.4 GET-PATH($(y, a, b), \overline{w}, \psi, \tau$)

```
1: Initialize an empty path list $P$
2: Append $(y, a, 0, 0)$ to $P$
3: rem $\leftarrow 0$
4: while $\psi(y, a) \neq nil$ do
5:     $(x, a', m) \leftarrow \psi(y, a)$
6:     $t_D \leftarrow a - m\ \tau(e_{xy}, 1) - (1 - m)\tau(e_{xy}, 0) + \texttt{rem}$
7:     $w_x \leftarrow min(t_D - a', \overline{w}_x)$
8:     rem $\leftarrow t_D - a' - w_x$
9:     $a_x \leftarrow a' + \texttt{rem}$    // Required arrival at $x$
10:    Append $(x, a_x, w_x, m)$ to $P$
11:    $(y, a) \leftarrow (x, a')$
12: return $P$
```

in Q with vertex y whose arrival time range contains a_y.

Proof. For a given node, the critical departure times represent the number of separate arrival time blocks. Also, as discussed earlier, a single block of arrival times can be represented by a node having the earliest arrival time in that block and budget equal to the width of the block. The EXPLORE function gets called for each neighbour of x (Alg. 4.1 line 11) and generates new nodes corresponding to each critical departure (Alg. 4.2 lines 6-12). Therefore the resulting nodes cover all possible arrival times at every neighbouring vertex of x when departing at a time in the range $[a_x, a_x + b_x + \overline{w}_x]$.

During the refinement step, only those nodes are removed for which the arrival time range is already covered by another node (Alg. 4.3 line 7). Therefore, after execution of the EXPLORE and REFINE functions, there exist nodes for all achievable arrival times at the neighboring vertices corresponding to the node (x, a_x, b_x). □

Lemma 2. *When a node (x, a_x, b_x) is extracted from Q, for every achievable arrival time $a' < a_x$ at x (through any path from the start vertex), there exists at least one node with vertex x in the explored set S for which the arrival time range includes a'.*

Proof. We will use proof by induction.

Base case: Consider the starting node $(s, 0, 0)$, the first node extracted from Q. Since it has an arrival time of 0, and arrival times are non-negative there is no earlier achievable arrival time at vertex s, so the statement is true.

Induction step: Assume the statement is true for the first k nodes extracted and added to S. We want to show that it is also true for the next node (x, a_x, b_x) extracted from Q. We will prove this by contradiction.

Suppose there exists an achievable arrival time $a' < a_x$ at x such that no node of vertex x in S has an arrival time range that includes a'. Let (x, a') is achieved via some path[2] $(s, 0) \rightsquigarrow (u, a_u) \to (v, a_v) \rightsquigarrow (x, a')$, where (u, a_u) and (v, a_v) are two consecutive entries in the path. Let (v, a_v) be the first pair in the path for which a node enclosing arrival time a_v is not present in S. This can also be (x, a') itself. Since (v, a_v) is directly reachable from (u, a_u), when exploring the node corresponding to (u, a_u), a node corresponding to arrival time a_v at v must have been inserted (or already present) in Q (Lemma 1). Let this node be (v, a'_v, b'_v).

We have $a_v \in [a'_v, a'_v + b'_v]$. Since $b'_v \geq 0$, we get $a'_v \leq a_v$. Also, $a_v \leq a'$ because (v, a_v) and (x, a') lie on a valid path. Since we assumed $a' < a_x$, we get $a'_v < a_x$. However, since (x, a_x, b_x) is extracted from Q first, we must have $a_x \leq a'_v$. Therefore, the initial assumption must be incorrect, and the statement holds for any node extracted from Q. □

Theorem 1. *Consider a vertex x, and let (x, a_x, b_x) be the first node with vertex x that is extracted from Q. Then a is the earliest achievable arrival time at x.*

Proof. By Lemma 2, if there exists an arrival time $a' < a_x$ at x which is achievable through any path from the start, a corresponding node must be in the explored set. Since (x, a_x, b_x) is the first node with vertex x that is extracted from Q, there is no node with vertex x in the explored set S. Therefore, a is the earliest achievable arrival time at x. □

4.5 Simulations and Results

In this section, we present the simulation setup and discuss the performance of different solution methods.

4.5.1 Baseline Algorithms

In this section, we present some solution approaches that we use to compare against the proposed Budget-A^* algorithm.

[2]Here, only the vertex-time pairs are used to denote a path. Wait times and mode of travel are omitted for simplicity of expressions.

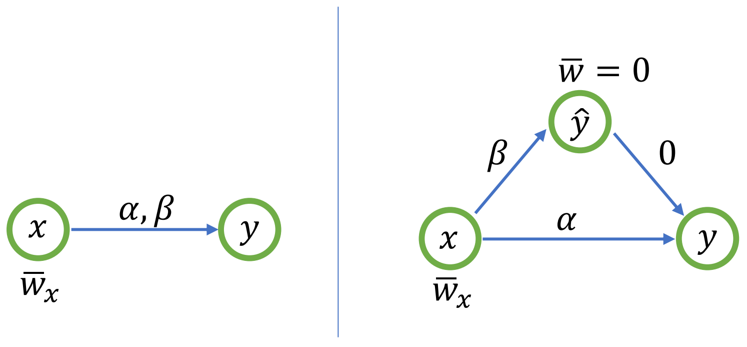


Figure 4.3: Method for modifying the given graph for TCSP-CWT algorithm. Left: Two vertices and connecting edge in original graph with autonomous edge duration α, assisted edge duration β and a wait limit $\overline{w}_x$ at vertex x. Right: Modified vertices and edges in the new graph. This allows for two different paths from x to y based on mode of operation selected.

4.5.1.1 TCSP-CWT

The TCSP-CWT algorithm (Time-varying Constrained Shortest Path with Constrained Waiting Times), presented by Cai et al. [169], solves the shortest path problem under the constraint of a bounded total travel time. Details on the algorithm are provided in Section 3.3.3. The original algorithm can optimize the travel cost but does not take into account the cost of waiting, and requires an upper bound T on travel time as an input. To solve the given problem, we modify the original algorithm as follows.

First, we determine the upper bound on travel time T to search by solving the static version of the problem, where each edge duration is set to its corresponding autonomous travel time value. Next, we obtain a new graph by creating two copies of each vertex in the given graph: one corresponding to autonomous operation (v) and another to the assisted operation ($\hat{v}$). For each edge $e_{uv} \in E$, two edges are created: e_{uv} - from u to v - representing autonomous travel, and $\hat{e}_{uv}$ - from u to $\hat{v}$ - representing assisted travel. We also add an edge from $\hat{v}$ to v with a duration of 0. Waiting limit at any v is the original given limit $\overline{w}_v$, while no waiting is allowed at $\hat{v}$. These modifications are shown in Fig. 4.3.

The autonomous edges can be traversed at any time, while the assisted edges are dependent on the operator availability. The cost of each edge is set to its travel duration given the departure time and operator availability. Finally, instead of storing cost of reaching a vertex, we store the time it is reached. Since we are interested in finding the minimum arrival time only at the goal vertex, when looping through the time steps $t \in [1, T]$, we stop the search at the first t for which the arrival time at the goal vertex

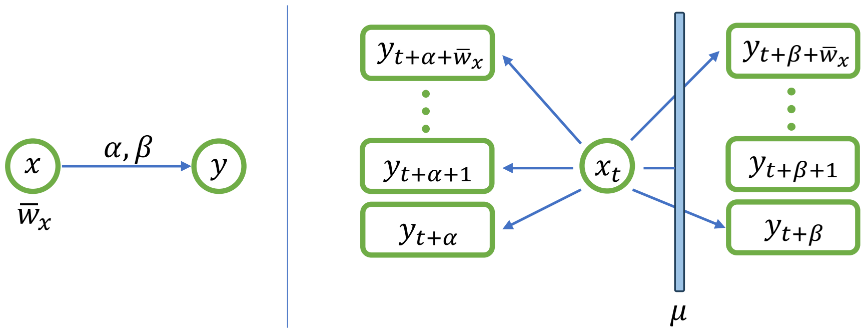


Figure 4.4: Method for modifying the graph for the Time-Expanded A^* algorithm. Left: Two vertices and connecting edge in original graph with autonomous edge duration α, assisted edge duration β and a wait limit $\overline{w}_x$ at vertex x. Right: Modified vertices and edges in the new graph for the vertex x_t. There are $\alpha+\overline{w}_x+1$ edges added for autonomous operation, while the edges for assisted operation gets filtered based on the operator availability. In the above example, assisted operation is not available for departure time $t+1$, so the corresponding edge is not added. This process is repeated for each copy of vertex x for $t \in \{0, 1, \dots, T\}$.

has a finite value. The optimal path is then generated by backtracking the arrival times at each intermediate vertex.

4.5.1.2 Time-expanded A^*

The Time-expanded A^* algorithm [190] is a modified version of the A^* algorithm that can be used to solve the given problem. It is called "time-expanded" because it creates a separate node for each vertex at each time step, thus expanding the search space. New edges are then added to the graph, with edge durations corresponding to the departure time at a vertex in the expanded graph. This approach is equivalent to our proposed algorithm without the notion of budget and it allows us to utilize the standard A^* algorithm used for static graphs (Algorithm 3.1) to solve our time-dependent problem. Similar to the TCSP-CWT, we duplicate the edges and determine the upper bound of time steps T by solving the problem with autonomous travel durations.

4.5.1.3 Greedy (Fastest Mode) Method

One efficient method for obtaining a solution is to combine a time-dependent greedy selection with a static graph search method, as described in Algorithm 4.5. This approach is similar to an A^* search on a static graph, but takes into account the arrival time at each vertex while exploration. To determine the edge duration to the neighboring vertices, we consider the faster of the two alternatives: traversing the edge immediately under autonomous mode or waiting for the operator to become available (subject to waiting limits). This way, the algorithm computes the fastest way to reach the neighboring vertices from each vertex, given the current time and edge durations. Once the goal vertex is extracted from the priority queue, we can stop the search and use the predecessor data to obtain the path.

Algorithm 4.5 Greedy Search

```
1: **Input:** $G = (V, E), \tau, \mu, \overline{w}, s, g$
2: $Q \leftarrow$ initialize priority queue
3: $S \leftarrow \emptyset$
4: $u.pred \leftarrow nil$ for all $u \in V$
5: $Q$.insert($(s, 0)$)
6: **while** $Q$ not empty **do**
7:     $(u, t) \leftarrow Q$.extract-min() // Extract node with minimum $t$
8:     $S \leftarrow S \cup u$
9:     **if** $u = g$ **then**
10:        **break**
11:    **for all** $v \in$ neighbors($u$) **do**
12:        **if** $v \notin S$ **then**
13:            $\Delta t \leftarrow timeUntilOprAvail(t, mu, \overline{w}_u)$
14:            $timeAuto \leftarrow t + \tau(e_{uv}, 0)$
15:            $timeAssist \leftarrow t + \Delta t + \tau(e_{uv}, 1)$
16:            $Q.push((v, \min\{timeAuto, timeAssist\}))$
17:            $v.pred \leftarrow u$
```

If it is not possible to traverse the edge under assisted mode under the given waiting limit, the Δt compute in Line 13 takes the value of infinity (or a large positive number).

4.5.2 Problem instance generation

For generating the problem instances, we use the map of the city of Waterloo, Ontario, Canada (a $10km \times 10km$ area around the city centre). Using the open source tools QGIS and OpenStreetMap, we place a given number of points at different intersections and landmarks. These points serve as vertices in our graph. Next, we use Delaunay triangulation to

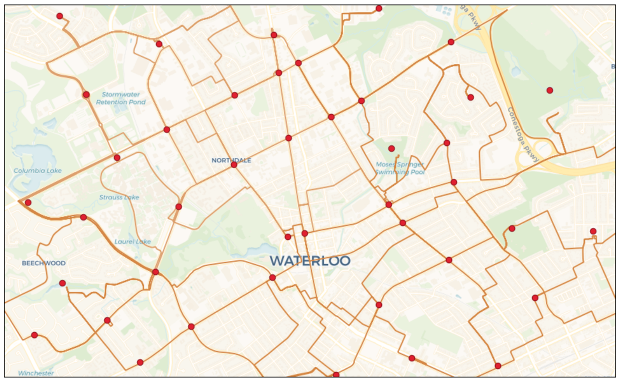


Figure 4.5: Example of the street network graph of the city of Waterloo used in the simulations. The figure shows a screenshot from the QGIS software. The shortest paths (orange lines) between vertices (red dots) are generated using the OpenRouteService.

connect these vertices and use OpenRouteService (ORS) to compute the shortest driving distance between these vertices. An example graph of the city is shown in Figure 4.5. To obtain the travel durations at each edge, we first sample robot speeds from a uniform random distribution. The travel durations under the two modes are then computed by dividing the edge length (computed using ORS) by the speed values and rounding off to the nearest integer. The travel speeds are sampled as follows: autonomous speed $u^0_{xy} \sim U[0, 40]$; assisted speed $u^1_{xy} \sim U[10 + u^0_{xy}, 30 + u^0_{xy}]$. The maximum waiting duration at each vertex x is sampled from a uniform random distribution as $\overline{w}_x \sim U[0, 15]$. The operator availability function is generated by randomly sampling periods of availability and unavailability, with durations of each period sampled from the range of $[10, 200]$. The distance values used in our simulations are in meters, times are in minutes and speeds are in meters/minute.

We test the algorithms using the Waterloo city map with varying vertex density, by selecting 64, 100 or 225 vertices to be placed in the map. We generate 20 problem instances for each density level (varying speeds, waiting limits and operator availability), and for each instance, we solve the problem for 100 randomly selected pairs of start and goal vertices. The algorithms are compared based on solution time and the number of explored nodes. We also examine some of the solutions provided by the greedy method.

Note on implementation: All three graph search algorithms (Budget-A^*, Greedy and Time-expanded A^*) use the same heuristic, obtained by solving a problem instance under the assumption that operator is always available. This heuristic is admissible in a time-dependent graph [184] and can be computed efficiently. The priority queues used in all methods are implemented as binary heaps, allowing for efficient insertion, extraction and search operations. Additionally, all the methods require computation of the feasibility set (Alg. 4.2 line 5). This is pre-computed for all departure times and is given as input to each algorithm.

4.5.3 Results

Figure 4.6 compares the performance of the Budget-A^* algorithm with that of the Greedy algorithm in terms of durations of the generated paths. From the figure, we observe that the Greedy algorithm is able to generate optimal or close-to-optimal solutions for a large proportion of the tested problem instances. However, for many instances, the path generated by the greedy approach is much longer than that produced by the Budget-A^* algorithm, reaching up to twice the duration.

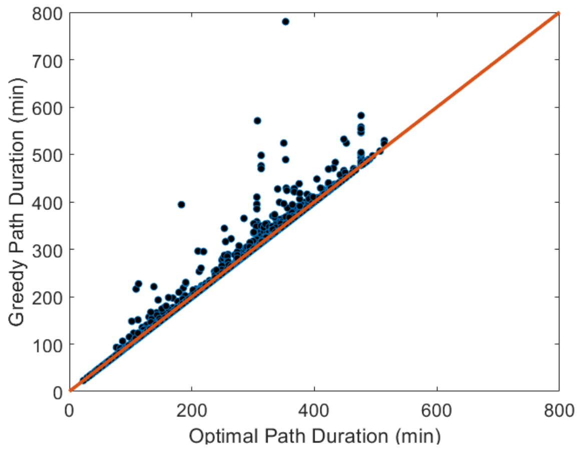


Figure 4.6: Performance comparison of the Greedy algorithm to the proposed Budget-A^* algorithm. Each point in the graph denotes a problem instance, where the x and y coordinates correspond to durations of the paths generated by the Budget-A^* and the Greedy algorithm respectively. The diagonal line represents equal path duration and the vertical distance from the line indicates the difference in duration between the two solutions.

To gain further insight into our results, we present Fig. 4.7, highlighting example instances where the greedy approach fails to generate an optimal solution. Through these examples, we demonstrate how our algorithm makes effective decisions regarding path selection, preemptive waiting, and not utilizing assistance to delay arrival at a later vertex. These decisions ultimately result in improved arrival time at the goal.

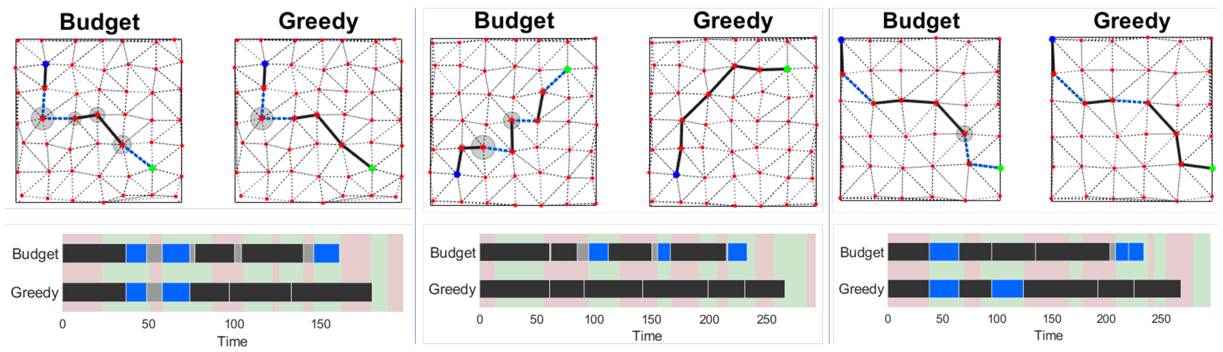


Figure 4.7: Example graphs comparing paths generated by the proposed Budget-A^* and the Greedy algorithm. Black solid lines represent autonomous mode and blue dotted lines represent assisted mode. Grey circles denote waiting, with circle size proportional to waiting duration. The red and greed shaded regions in the bottom plots represent operator availability. (a) The Budget-A^* algorithm preemptively waits at initial vertices to use autonomous mode for later edges, resulting in a faster path. The greedy algorithm moves towards the goal quickly, but cannot use operator availability later due to waiting limits. (b) The Budget-A^* algorithm uses operator availability more efficiently by selecting a longer path. (c) The Budget-A^* algorithm chooses not to use operator's assistance even when it is available, so that it can be used later when the assistance is more beneficial.

Figure 4.8 compares the computation time required by different solution methods for varying number of vertices and the duration of the optimal path between the start and goal vertices. The plots demonstrate that the proposed algorithm consistently outperforms the other optimal methods in terms of computation time, with the greedy method being the fastest but providing suboptimal solutions. The computation time for all methods increases with the number of vertices. The path duration has the greatest impact on the performance of the TCSP-CWT algorithm, followed by the Time-expanded A^*, the Budget-A^* algorithm, and finally the Greedy algorithm.

Figure 4.9 compares the number of nodes generated and explored by Time-expanded A^*, Budget A^*, and Greedy search algorithms. The number of nodes is a key metric to evaluate search efficiency as it reflects the number of insertions and extractions from the priority queue. The Time-expanded A^* generates nodes at a faster rate with increasing vertices, while the proposed algorithm generates an order of magnitude fewer nodes, indicating better efficiency and scalability. The Greedy search algorithm terminates after exploring the least number of nodes, indicating that it sacrifices optimality for speed. In contrast, both the Time-expanded A^* and Budget A^* algorithms guarantee optimality in their search results.

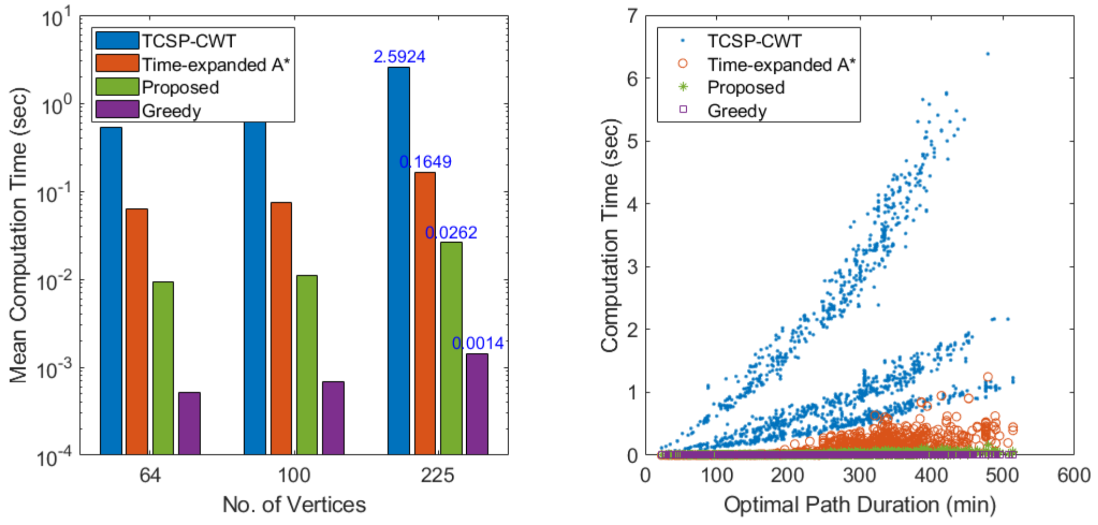


Figure 4.8: Computation time comparison for different methods. **Left:** Mean computation time as a function of the number of vertices in the graph, averaged over all test instances. Note that the time is plotted on a log scale. **Right:** Computation time as a function of actual duration of the optimal path, shown for all test instances.

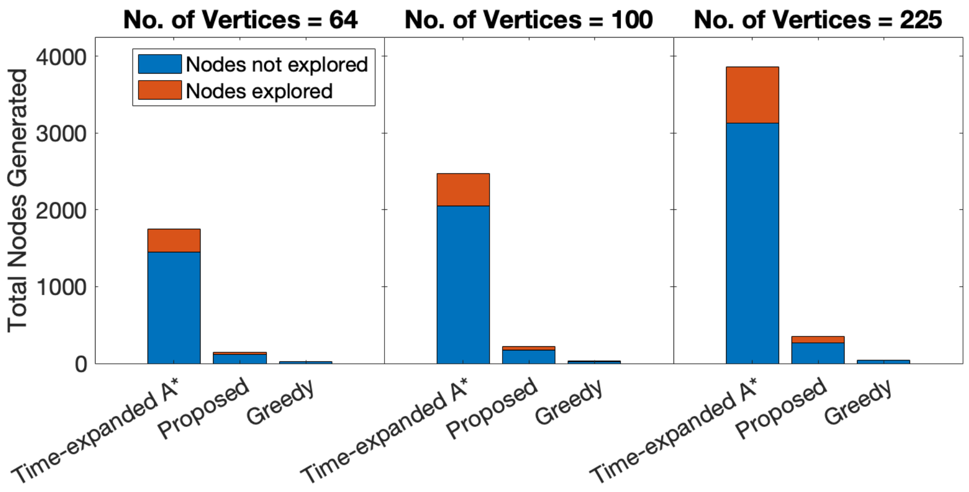


Figure 4.9: Mean number of nodes generated during graph search under the three methods for different number of vertices in the map. The bar height shows the total number of nodes generated and added to the queue during the search. The orange stack denotes the number of nodes explored before the goal is reached and the search is terminated.

4.6 Implementation and Efficiency

In this section, we discuss the time efficiency of the given algorithm in a discrete-time setting. Although the Budget-A^* algorithm can inherently operate in continuous time, we adapt it to a discrete time setting to maintain consistency with other baseline algorithms used for comparison (as discussed in Section 4.5.1). In this discrete-time implementation, we introduce a time constraint T that represents the upper bound on the arrival time at the goal location. One approach to determine the value of T is to solve the static version of the problem, assuming that the operator is always unavailable, which yields a suitable value for T. Alternatively, we can use a fast (greedy) algorithm to find a tighter bound on this time constraint, further optimizing the algorithm's performance. Importantly, the Budget-A^* algorithm's performance remains unaffected by the chosen unit of time steps in the implementation. This characteristic ensures the robustness and consistency of the results, regardless of the specific time granularity used.

Examining the time efficiency of the Budget-A^* algorithm (Alg. 4.1), observe that a node in the priority queue is determined by the vertex and its earliest arrival time. Therefore, the primary *while* loop (line 6) runs for $\mathcal{O}(T\,|V|)$ times. The priority queue Q is implemented as a binary heap, resulting in an extraction time of $\mathcal{O}(\log(T\,|V|))$. The REFINE function can be implemented in $\mathcal{O}(\log(T\,|V|))$ time by exploiting the fact that nodes in the queue are arranged by their arrival times. The GET-PATH function is a simple backtracking through the predecessor nodes and takes $\mathcal{O}(|V|)$ time. By pre-computing the feasible set outside the while loop, the EXPLORE function in our algorithm achieves a time complexity of $\mathcal{O}(|\mathcal{N}|)$, where $|\mathcal{N}|$ represents the number of new nodes generated. This optimization significantly improves the efficiency of the exploration process.

An essential feature of our problem is that the number of newly generated nodes is constrained by the budget at a vertex and the duration of the edge under autonomous and assisted operation. In Section 4.3, we discussed how feasible departure times from a node are constrained by t_{max}. For a node (x, a_x, b_x) and an edge e_{xy}, the value of t_{max} is given by

$$\min\{b_x + \overline{w}_x, \tau(e_{xy}, 0) - \tau(e_{xy}, 1)\}.$$

Furthermore, for an edge to be eligible for assistance, μ must be 1 for the whole duration of the edge. This means that even for an arbitrarily large value of budget, and even if the operator availability changes arbitrarily, the maximum number of critical departure times under assisted operation for a node are limited by

$$\left\lceil \frac{\tau(e_{xy},0) - \tau(e_{xy},1)}{\tau(e_{xy},1)} \right\rceil.$$

As a result, the EXPLORE function generates a maximum of $1 + \left\lceil \frac{\tau(e_{xy},0)-\tau(e_{xy},1)}{\tau(e_{xy},1)} \right\rceil$ new nodes. Defining $\gamma = \max_{e \in E} \frac{\tau(e,0)}{\tau(e,1)}$, we observe that the EXPLORE function runs in $\mathcal{O}(\gamma)$ time. Consequently, the Budget-A^* algorithm exhibits a time complexity of

$$\mathcal{O}(\gamma\, T\, |V|^2 \log(T\, |V|)).$$

In the worst case, $|\mathcal{N}| = T$. Therefore, the time complexity of the given algorithm will be $\mathcal{O}(T^2\, |V|^2 \log(T\, |V|))$, which is the same as that of the time-expanded A^* algorithm. However, in practice, the value of γ is expected to be much less than T, and thus the algorithm performs much better than the time-expanded A^* algorithm, as seen in Section 6.3.

Remark. It is also important to note that the runtime of our algorithm remains unaffected even if all edge durations are scaled by a constant factor. This is because there are only $\mathcal{O}(T/\lambda)$ possible times to arrive at a given vertex, where λ represents the highest common factor among all edge durations. As a result, the maximum length of the priority queue is bounded by $\mathcal{O}(\frac{1}{\lambda} T\, |V|)$. Thus, the time complexity of our algorithm can be more accurately expressed as

$$\mathcal{O}\left(\frac{\gamma}{\lambda} T\, |V|^2 \log\left(\frac{1}{\lambda} T\, |V|\right)\right).$$

4.7 Extension to multi-robot case

While the Budget-A^* algorithm presented in this chapter is well suited for a single robot scenario, we aim to investigate the feasibility and benefits of planning for multiple robots concurrently.

In the single-robot case, the objective was to minimize the arrival time at the goal location for the given robot, under operator availability constraints. In a multi-robot setting, several robots compete for the same shared resource, which is the operator assistance. This implies that trying to minimize the travel time for one robot using operator assistance can potentially increase the travel time for other robots in the system. Therefore, for the multi-robot case, we must choose a different, more appropriate reward.

Consider a set of K robots denoted as $\{1,\ldots,K\}$, each with its path planning problem represented as a *service request* defined by a start and goal vertex pair. These service requests form a batch, and the planned path for robot $k \in \{1,\ldots,K\}$ is represented as P^k. The length of a path, denoted by $l(P^k)$, signifies the arrival time at the goal vertex along the execution path P^k. Note that for each robot $k \in \{1,\ldots,K\}$, there exists at least one path that leads from its start to the corresponding goal vertex without using operator assistance. Among these paths, let the shortest one be denoted as $\tilde{P}^k$, with the path length $l(\tilde{P}^k)$. We consider the following two reward functions:

Reward-1: Net Reduction in Travel Time: Our first reward function quantifies the cumulative reduction in travel time (path length) for all robots within the batch:

$$r(\{P^1,\ldots,P^K\}) = \sum_{k=1}^{K} l(\tilde{P}^k) - l(P^k). \tag{4.1}$$

Reward-2: Reduction in Makespan: Similarly, we define an alternative reward function based on the decrease in makespan, which is the path length of the slowest robot among all the robots in the batch, and has been used to guide decision-making in similar applications [179, 191]:

$$r(\{P^1,\ldots,P^K\}) = \max_{k\in\{1,\ldots,K\}} l(\tilde{P}^k) - \max_{k\in\{1,\ldots,K\}} l(P^k). \tag{4.2}$$

Given the reward function, we can write the multi-robot path planning problem as follows:

Problem 4.2. *Given a graph $G = (V, E)$ and operator availability μ as described in Sec. 4.2, we consider a batch of service requests for K robots defined by pairs of start and goal vertices (s^k, g^k) for each robot $k \in \{1,\ldots,K\}$. Denoting the path for robot k as $P^k := \langle(v_1^k, t_1^k, w_1^k, m_1^k),\ (v_2^k, t_2^k, w_2^k, m_2^k), \ldots,\ (v_{n^k}^k, t_{n^k}^k, w_{n^k}^k, m_{n^k}^k)\rangle$, the problem objective can* *be written as follows:*

$$
\begin{aligned}
\max_{P^1,\dots,P^K \in \mathcal{P}} \quad & r\Big(\{P^1, \dots, P^K\}\Big) & \\
s.t. \quad & v_1^k = s^k,\ v_{n^k}^k = g^k & \forall\ k \in [1, K] \\
& e_{v_i^k v_{i+1}^k} \in E & \forall\ k \in [1, K], i \in [1, n^k - 1] \\
& t_{i+1}^k = t_i^k + w_i^k + \tau(e_{v_i^k v_{i+1}^k}, m_i^k) & \forall\ k \in [1, K], i \in [1, n^k - 1] \\
& w_i^k \leq \overline{w}_{v_i^k} & \forall\ k \in [1, K], i \in [1, n^k - 1] \\
& m_i^k = 1 \Rightarrow \mu([t_i^k + w_i^k, t_{i+1}^k]) = 1 & \forall\ k \in [1, K], i \in [1, n^k - 1] \\
& \textstyle\sum_{k=1}^{K}\ u(t, k) \leq 1 & \forall\ t \in [0, \max_k t_{n^k}^k] \\
& u(t, k) = 1 \iff \exists\ i \in [1, n^k - 1]\ s.t.\ t_i^k \leq t < t_{i+1}^k, m_i^k = 1, &
\end{aligned}
$$

where $u(t, k)$ is an indicator function representing whether robot k is scheduled to receive operator assistance at time t.

The first five constraints are direct extensions of the corresponding constraints of Problem 4.1. The last two constraints ensure that the operator cannot be assigned to more than one robot simultaneously[3].

To address the multi-robot planning problem, one approach is to extend the time-expanded A^* algorithm. In this extension, the system state encompasses the combination of individual robot states, forming a K-dimensional vector, while actions encompass the combinations of next edges selected for each robot. This leads to a centralized solution approach where the plans for all robots are collectively generated. Nevertheless, for a system with K robots, this centralized approach results in a graph of size $V^K \times T$, which can be computationally intensive to execute [16]. Similar extension of the Budget-A^* method can also lead to centralized solutions for multi-robot scenarios. Alternatively, it's possible to formulate this problem as a Mixed Integer Linear Program (MILP). Nevertheless, such formulations can also encounter scalability issues and be operationally infeasible [179].

However, in the context of autonomous delivery applications, we argue that a decentralized planning approach is more preferable. Here, each robot's plan is individually generated, independent of the other robots. This approach is supported by the nature of

[3]Although this discussion assumes a single operator for all robots, the presented formulation can be easily extended to a multi-operator scenario.

service requests in such applications, which are typically received individually. This enables the treatment of each robot as a separate entity during the planning process, addressing their missions based on the sequential order of service requests.

Moreover, the robots in such systems largely operate independently of each other, with the allocation of operator assistance being the only point of interdependence among them. Since the use of operator assistance is optional for each task, it is possible for the robots to independently complete their missions without impacting the plans of other robots. This autonomy implies that centralizing the planning for all robots might not yield significantly better results than a decentralized prioritization-based approach [16]. Furthermore, a decentralized planning approach offers advantages in terms of robustness over centralized approaches as it avoids interdependencies among robot plans [13]. Such interdependence could be detrimental in cases of failure or delays, potentially propagating adverse effects across the plans of multiple robots [192].

Given these practical considerations, adopting a decentralized approach to our problem appears to be more feasible and advantageous. This approach involves generating robot plans one by one, allowing for greater robustness and adaptability in handling individual service requests. It proves to be highly practical in scenarios where service requests arrive sequentially, enabling efficient planning for each robot without unnecessary dependencies on the plans of others.

4.7.1 Prioritizing a batch of service requests

While addressing individual service requests by planning robot paths upon their arrival is a viable approach, situations can arise where multiple service requests arrive simultaneously in batches. In such scenarios, prioritizing these requests before planning the paths can lead to more efficient solutions. This approach involves planning the execution paths of robots one by one, based on their priority among other robots in the batch. Once a robot's path is planned, the operator availability is updated, accounting for scheduled operator assistance times, and is then used for planning the subsequent robot's path. The prioritization process can be summarized as Alg. 4.6 below:

Algorithm 4.6 Batch Service Request Prioritization

```
1: Input:   Batch of service requests $M$
2: Initialize operator availability
3: while $M$ is not empty do
4:     Identify the robot with the highest priority from the batch
5:     Plan an optimal path for the selected robot under current operator availability
6:     Update operator availability based on the planned path
7:     Remove the selected robot from $M$
```

The challenge now lies in devising an effective approach to prioritize the service requests within the batch. One possible approach is to solve the planning problem for all possible sequences derived from the set M and selecting the one that maximizes the reward. However, as we will discuss next, we can employ a greedy algorithm that efficiently prioritizes the batch of service requests.

4.7.2 Submodularity

We aim to show that the reward functions, as given in equations 4.1 and 4.2, result in a sequential submodular problem, indicating that planning for robots in any arbitrary sequence yields diminishing returns (for more details on submodularity, refer to Chapter 3 Section 3.2). This property enables the use of an efficient greedy algorithm for prioritizing the batch of service requests.

To establish sequential submodularity, we first define a sequence $\mathcal{A} = \langle a_1, \ldots, a_K \rangle$ of K robots derived from the set $M := \{1, \ldots, K\}$, where $a_i \in M$ for each $i \in \{1, \ldots, K\}$. Let P^{a_i} be the optimal (fastest) path for robot a_i, given the operator availability after planning for the first a_{i-1} robots in the sequence. This allows us to define the rewards given in equations 4.1 and 4.2 as functions of the sequence $\mathcal{A}$.

We now establish that the reward function presented in equation 4.1 (total reduction in travel time) satisfies the three required properties:

1. **Normalized:** For an empty sequence $\mathcal{A} = \emptyset$, $r(\mathcal{A}) = 0$.

   Proof. With no robots in the sequence, $l(P^k) = l(\tilde{P}^k)$ for all $k \in \{1, \ldots, K\}$. Therefore, the travel time reduced by the planning process (and the total reward) is zero. □

2. **Monotone non-decreasing:** For a sequence $\mathcal{B}$ derived from the base set M and $\mathcal{A}$ a subsequence of $\mathcal{B}$, i.e., $\mathcal{A} \subseteq \mathcal{B}$, $r(\mathcal{A}) \leq r(\mathcal{B})$.

   Proof. Since the use of operator assistance is optional and utilized only when it leads to a faster path, any path planned with assistance P^k will be as fast as, or faster than, the path planned without any assistance $\tilde{P}^k$ for any robot k in the sequence. In other words, we have $l(\tilde{P}^k) \geq l(P^k)$.

   Now, consider the sequence $\mathcal{B}$ derived from the base set M, which includes the subsequence $\mathcal{A}$. Any path P^k for robots in $\mathcal{A}$ is also a valid path for $\mathcal{B}$ since $\mathcal{A} \subseteq \mathcal{B}$. This implies that expanding the sequence from $\mathcal{A}$ to $\mathcal{B}$ cannot decrease the reward value from $r(\mathcal{A})$, thus confirming $r(\mathcal{A}) \leq r(\mathcal{B})$. □

3. **Submodular:** For sequences $\mathcal{A}$ and $\mathcal{B}$ such that $\mathcal{A} \subseteq \mathcal{B}$, and $x \in M \setminus \mathcal{B}$,

$$r(\mathcal{A} \parallel \{x\}) - r(\mathcal{A}) \geq r(\mathcal{B} \parallel \{x\}) - r(\mathcal{B})$$

   Proof. Let $\mu_{\mathcal{A}}$ denote the operator availability after the execution paths of all robots in sequence $\mathcal{A}$ are planned and the operator allocation is scheduled. Also, let us denote $\mu_1 \subseteq \mu_2$ if $\mu_1(t) = 1 \Rightarrow \mu_2(t) = 1$ for all times t. This means that operator assistance under μ_2 is at least as available as under μ_1. Since $\mathcal{A} \subseteq \mathcal{B}$, we have $\mu_{\mathcal{B}} \subseteq \mu_{\mathcal{A}}$.

   Moreover, let us define the length of path planned for robot x under operator availability μ as $l(P^x|\mu)$. As discussed earlier, planning a robot's path can optionally use operator assistance. Therefore, after a robot's plan has been created, operator availability can only decrease. Furthermore, higher operator availability results in equal or shorter path lengths for a robot, i.e., if $\mu_1 \subseteq \mu_2$, then $l(P^x|\mu_1) \geq l(P^x|\mu_2)$.

   Now, consider the reward function for sequences $\mathcal{A} \parallel \{x\}$ and $\mathcal{B} \parallel \{x\}$:

$$\begin{aligned} r(\mathcal{A} \parallel \{x\}) - r(\mathcal{A}) &= \sum_{k \in \mathcal{A} \parallel \{x\}} \Big(l(\tilde{P}^k) - l(P^k) \Big) - \sum_{k \in \mathcal{A}} \Big(l(\tilde{P}^k) - l(P^k) \Big) \\ &= l(\tilde{P}^x) - l(P^x|\mu_{\mathcal{A}}). \end{aligned} \tag{4.3}$$

Similarly,

$$
\begin{aligned}
r(\mathcal{B} \parallel \{x\}) - r(\mathcal{B}) &= \sum_{k \in \mathcal{B} \parallel \{x\}} \Big(l(\tilde{P}^k) - l(P^k) \Big) - \sum_{k \in \mathcal{B}} \Big(l(\tilde{P}^k) - l(P^k) \Big) \\
&= l(\tilde{P}^x) - l(P^x | \mu_{\mathcal{B}}). \qquad (4.4)
\end{aligned}
$$

Since $\mu_{\mathcal{B}} \subseteq \mu_{\mathcal{A}}$, we have $l(P^x|\mu_{\mathcal{B}}) \geq l(P^x|\mu_{\mathcal{A}})$. Also, recall that $l(\tilde{P}^x)$ is the path length without using any operator assistance and is therefore constant in both cases. This implies that $r(\mathcal{A} \parallel \{x\}) - r(\mathcal{A}) \geq r(\mathcal{B} \parallel \{x\}) - r(\mathcal{B})$. □

These three properties imply that, for the above reward function, the problem is submodular. Similarly, we can demonstrate that the reward given in equation 4.2 also satisfies the above three properties.

4.7.2.1 Greedy Approach to Sequencing

Having established the submodularity of the reward functions, we can now devise a greedy algorithm for efficiently prioritizing service requests when they arrive in batches. The essence of the greedy approach lies in adding one robot at a time to the sequence. At each step, we select the robot that offers the maximum reward increase, representing the largest reduction in travel time (or makespan), when added to the current sequence of robots. After incorporating the chosen robot, we update the operator availability accordingly and proceed to add the next robot to the sequence. This iterative process continues until all robots are included in the sequence, resulting in a prioritized batch of service requests. The advantage of the greedy approach is its computational efficiency while still providing good performance due to the monotone submodularity property. Pseudo-code of the greedy prioritization is given in Alg. 4.7.

Algorithm 4.7 Greedy Prioritization

```
1: Input: Batch of service requests $M := \{1, \ldots, K\}$, start and goal locations $s^k, g^k$
   Reward function $r : 2^M \to \mathbb{R}_{\geq 0}$, graph $G$, operator availability $\mu$, edge durations $\tau$,
   waiting limits $\overline{w}$
2: $\mathcal{A} \leftarrow \emptyset$ // Initialize empty sequence
3: for $i = \{1, \ldots, K\}$ do
4:     $k^* \leftarrow \arg\max_{k \in M \setminus \mathcal{A}} \ r(\mathcal{A} \parallel \{k\}) - r(\mathcal{A})$
5:     $P^k \leftarrow$ Budget-$A^*(G, \tau, \mu_{\mathcal{A}}, \overline{w}, s^{k^*}, g^{k^*})$
6:     $\mathcal{A} \leftarrow \mathcal{A} \parallel \{k^*\}$
7: Output: Paths $\{P^1, \ldots P^K\}$
```

Figure 4.10 illustrates a comparative analysis between the greedy and optimal approaches for prioritizing service requests, considering varying numbers of robots. The figure shows the ratio of the reward earned by the greedy prioritization (total reduction in travel time, as defined in equation 4.1) to the reward earned by the optimal approach. The optimal approach involves computing the total reward for each possible robot sequence and selecting the one with the maximum reward. The robot path planning, under a given operator availability, is accomplished using the Budget-A^* algorithm. From the figure, it is evident that the greedy approach performs remarkably well, consistently yielding above 80% of the optimal reward across different numbers of robots. The greedy approach's performance slightly decreases as the number of robots increases, but it remains highly competitive with the optimal method. The advantage of the greedy approach lies in its significantly faster computation time, being approximately two orders of magnitude faster than computing the optimal sequence. This substantial reduction in computation time makes the greedy prioritization an attractive choice in our system.

Figure 4.11 show the relative reward earned by the greedy prioritization for reward defined in equation 4.2 (reduction in system makespan). The plots show similar results to the previous case, and the greedy prioritization performs within 80% of the optimal while being significantly faster.

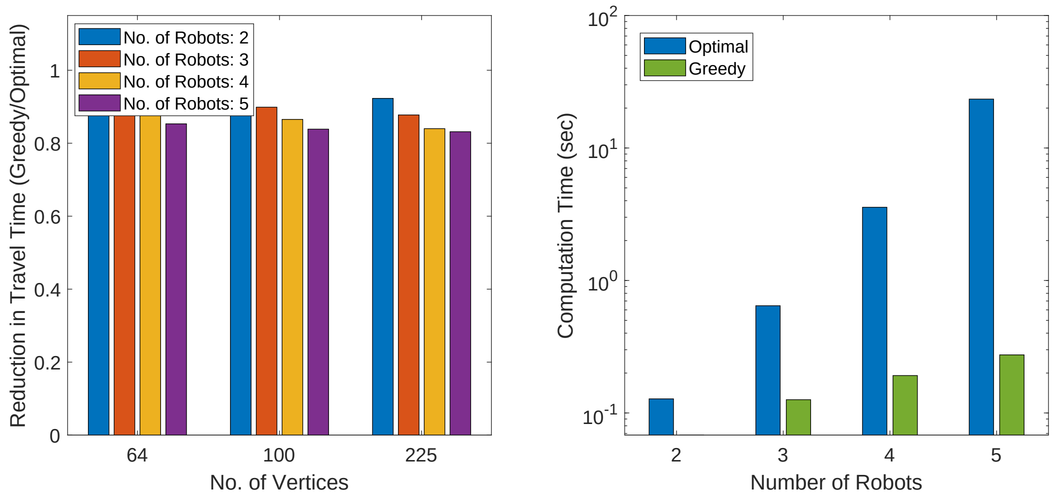


Figure 4.10: Left: Average reward earned (reduction in total travel time - equation 4.1) by the greedy prioritization relative to the optimal sequence for varying graph size and number of robots in the batch. Right: Computation time comparison of the greedy prioritization approach to the optimal one for different number of robots.

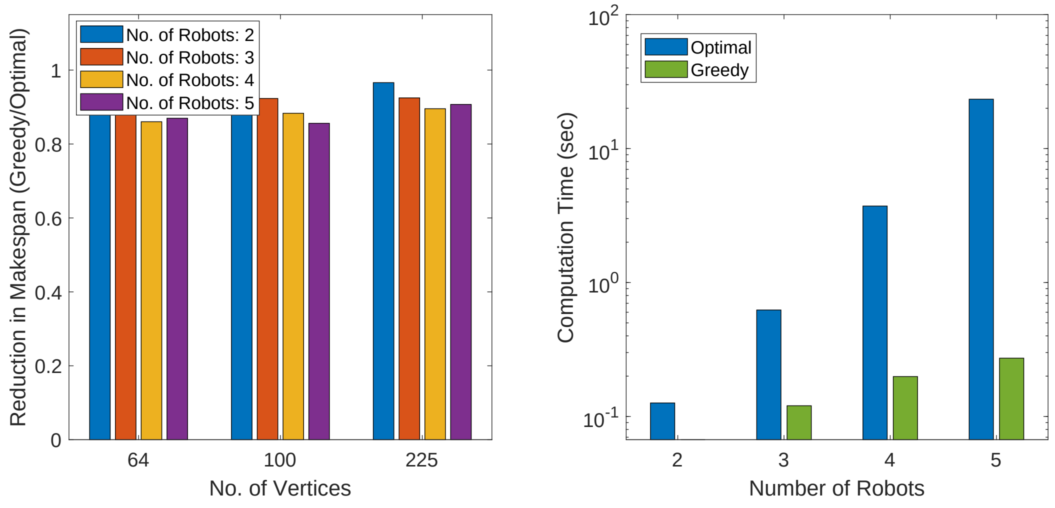


Figure 4.11: Left: Average reward earned (reduction in makespan - equation 4.2) by the greedy prioritization relative to the optimal sequence for varying graph size and number of robots in the batch. Right: Computation time comparison of the greedy prioritization approach to the optimal one for different number of robots.

4.8 Chapter Summary

In this chapter, we introduced Budget-A^*, a new algorithm to address the problem of collaborative robot planning with bounded waiting constraints and intermittent human availability. Our approach computes the optimal execution path, which specifies which path should the robot take, how much to wait at each location and when to use human assistance. Our simulations on a city road network demonstrate that Budget-A^* outperforms existing optimal methods, in terms of both computation time and number of nodes explored. Furthermore, we note that the greedy method performs well for the majority of test cases, which could potentially be utilized to further improve efficiency of the proposed algorithm.

While this chapter lays the groundwork for developing robot planning solutions for supervised robot fleets, and there are several avenues for enhancement and adaptation to different scenarios. Firstly, the Budget-A^* algorithm can be extended to accommodate additional constraints and complexities that may arise in real-world applications. For instance, in some scenarios, it is of importance to limit the human assistance provided to a robot. Investigating how to incorporate such constraints into the planning framework can lead to more versatile solutions. To make the approach scalable to even larger networks, there is room for optimizing the algorithm by exploring better heuristics and pruning techniques. While extending the planning for multiple robots, it may be interesting to explore and study other multi-robot reward functions.

Chapter 5

Operator Allocation in Robot Fleets

TL;DR: In this chapter, we define the operator allocation problem and explore the concepts of indexability and the Whittle Index. We discuss the application of an index policy for efficient operator allocation in robot fleets and validate its efficacy via extensive simulations.

In Chapter 2, we discussed how a robot fleet in remote supervision setting is a system where human supervisors monitor and interact with multiple robots using a group interface node. Such group nodes enable human supervisors to provide assistance to individual robots, either in event of a *fault* [37,129] or to further increase fleet performance in various industrial and social settings [38,46,193]. However, identifying which robot to assist in an uncertain environment is a challenging task for human operators [38,194]. Moreover, as the number of robots increases, it becomes challenging for the operators to maintain awareness of every robot, which cripples system's performance [155,156]. Therefore, human operators can benefit from having a group node that also acts as a decision support system (DSS) that advises which robots require attention and when [38,195].

In this chapter, we present such a DSS for a multi-robot multi-human system comprising a fleet of semi-autonomous robots with multiple human operators available for assistance if and when required. Figure 5.1 presents an overview of the problem setup, showing K robots navigating in a city block-like environment, moving from start to goal locations. While navigating, a robot passes through a series of waypoints, each characterized by a different probability of success in progressing to the next waypoint. There is also a possibility that the robot may fail at a task and get stuck in a fault (error) state from which assistance from a human operator is required to continue. There are M identical human operators available ($M \leq K$), each of whom can assist at most one robot at a time. While being assisted by an operator, robots have different probabilities of success and failure, and to get out of a fault state.

It is possible to solve the above problem by modelling it as a Markov Decision Process (MDP) [196]. However, such formulations suffer from the curse of dimensionality, making conventional MDP-based solution techniques scale poorly with problem size [197]. Moreover, the policy needs to be re-computed every time a robot or an operator enters or leaves the system. To tackle the scalability issue, researchers have proposed solutions based on simple myopic policies, sampling-based planner or an approximation algorithm [30,38,198]. However, these solutions either do not apply to stochastic settings, or they do not scale well with increasing number of robots and/or operators.

In this chapter, we show how an index-based policy can provide a scalable and better performing solution than the existing approaches for the given multi-robot multi-operator allocation problem with stochastic transitions. Specifically, our work makes the following contributions:

1) We show that the operator allocation problem with multiple independent robots can be formulated as an instance of the Restless Multi-Armed Bandit (RMAB) problem. We leverage this formulation to decompose the problem into several single-robot problems and computing the Whittle index heuristic (see Fig. 5.1). The resulting policy scales linearly with the number of robots and is independent of the number of operators.

2) We derive simple conditions to verify indexability of the model. These conditions can be checked independently for each state of each robot, thus providing a method that scales linearly with the size of the problem. This method can be applied to systems with any number of states and does not require the optimal policy to be of a threshold type.

3) We then implement our approach in two practical scenarios and present numerical experiments. The results show that the proposed method provides near-optimal solutions and outperforms existing efficient solution approaches, namely the reactive policy, 1- and 2-step myopic policies, and the benefit-maximizing policy.

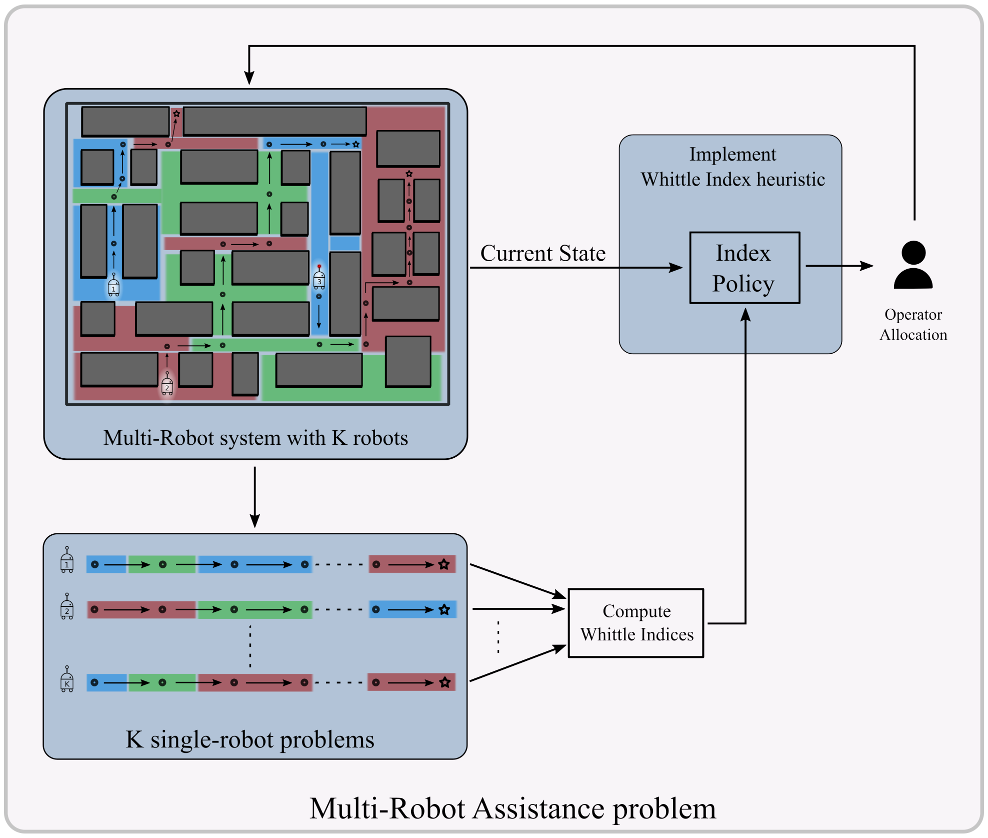


Figure 5.1: Overview of the multi-robot assistance problem for robots navigating in an environment. A number of mobile robots are tasked to navigate through a series of waypoints. Operators are allocated to the robots when required. This is done by decomposing the complete K-robot problem into K single robot problems and computing the Whittle index heuristic. Given the current state of the system, this heuristic can be used to efficiently compute the operator allocation.

5.1 Related Work

The problem of allocating operators in a multi-robot team bears similarities with the disciplines of multi-robot supervision, task scheduling and queuing theory. In this section, we briefly review the related research, followed by an introduction of Restless-Multi Armed Bandits.

In the literature, several studies discuss the problem of enabling human operators to assist multiple robots such as a team of navigating robots, a fleet of multiple UAVs, or a team performing search and rescue operations [191, 199, 200]. To understand and improve human supervision, researchers have used frameworks such as sliding autonomy to incorporate various human-robot team capabilities (like coordination and situational awareness) [201, 202]. Some studies also present interaction interfaces to facilitate and improve such supervision [203, 204].

The most closely related work to our problem is presented in [38], where the authors discuss single-operator multi-robot supervision systems. An advising agent guides the operator on which robot they should assist. The problem is solved using an l-step look-ahead (myopic) approach, which provides an efficient and practical solution, but suffers from scalability issues with increasing number of operators and the look-ahead steps. Researchers have also discussed deterministic versions of the problem, where exact outcomes of robots' actions and times for fault occurrences are known, and the allocation policy is determined using a sampling-based planner [198]. In [30], the authors present an approximation algorithm for a similar scheduling problem. These approaches however are not applicable in a stochastic setting.

The problem of assisting a number of independent robots, has also been studied under a learning framework. The approach presented in [37] learns the decision-making model of human operator from recorded data and tries to replicate that behaviour, optimizing based on the operator's internal utility function. In contrast, the problem presented in this chapter is designed to optimize a global performance metric assuming the knowledge of success and failure rates of robots with and without an operator allocated to them. Such knowledge can be estimated using recorded data similar to the work presented in [38]. For the scope of this chapter, we will assume this knowledge takes the form of known transition probabilities.

In the queuing discipline, several studies have investigated the effects of different queuing techniques [135] or threshold-based strategies [205] to prioritize operator's attention to the robots. However, the model that we study is different from a queuing model as it is possible for the robots to complete their tasks without the help of operators, and for the operators to be allocated to robots not stuck in a fault state.

The multi-target–multi-agent problems form another class of problems similar to the operator allocation problem. These problems deal with allocation of multiple agents to a number of targets aiming to detect or follow the targets under certain constraints [206,207]. However, our problem setup is different because the behaviour of the targets (robots) changes with the allocation of agents (operators) and it is not possible to allocate multiple agents to a single target at once. Moreover, our problem presents a collaborative task, where both the robots and operators are working to achieve a common goal.

Restless Multi-Armed Bandit

Restless Multi-Armed Bandits (RMAB), first introduced in [173], is a generalization of Multi-Armed Bandits (MAB) [208] which has been previously used in problems like assisting human partners [209] and distributing resources among human teammates [58]. RMAB is a class of scheduling problems where limited resources have to be allocated among several alternative choices. Each choice, referred to as an *arm*, is a discrete-time controlled Markov process which evolves depending on the resource allocated to it. RMAB framework has been applied to problems in stochastic scheduling, patrol planning, sensor management and resource allocation in general [210].

Finding the optimal policy for RMAB suffers from the curse of dimensionality as the state space grows exponentially with the number of arms. In general, obtaining the optimal policy in an RMAB is PSPACE-hard [211]. However, the *Whittle index policy* offers a simpler and scalable alternative to the optimal policy. Even though the Whittle index policy does not guarantee an optimal solution, it minimizes expected cost for a relaxed problem under time-averaged constraint [173]. This approach is shown to work quite well for several scheduling and resource allocation problems [212–214]. A few studies have also implemented index-based methods to solve a sensor scheduling problem [215] or to serve a number of users transmitting a queue of data packets through a channel [213]. Therefore, it is a reasonable approach to solve an RMAB given that the problem satisfies a technical condition known as indexability. For more details, see Section 3.3. Unfortunately, it is difficult to verify this condition in general and there is no universal framework that applies to all problems. Existing methods proposed for verifying indexability have been investigated for specific systems such as two state restless bandits [216, 217] or restless bandits with optimal threshold-based policy [214, 216, 218].

5.2 Multi-Robot Assistance Problem

Consider a decision support system (DSS), consisting of a team of M human operators and a fleet of K semi-autonomous robots. Each robot $k \in \mathcal{K} := \{1, \dots, K\}$ is required to complete a sequence of N^k tasks to reach its goal. We will use a fleet of robots delivering packages in a city as a running example but similar interpretations hold for other applications mentioned in previous sections (e.g., robots reaching a sequence of configurations [198]). In this case, the robot's trajectory would correspond to a series of waypoints that a robot needs to navigate to reach its destination (goal location). At each waypoint, a robot can either operate autonomously or be teleoperated/assisted by one of the human operators. We assume that all human operators are identical in the way they operate the robots and that a human operator can assist at most one robot at a time.

Remark on notation: Throughout this chapter, we use calligraphic font to denote sets and roman font to denote variables. Uppercase letters are used to represent random variables and the corresponding lowercase letters represent their realizations. Bold letters are used for variables pertaining to multi-robot system while light letters represent corresponding single-robot variables.

We now provide a mathematical model for different components of the system.

5.2.1 Model of the robots

It is assumed that when operating autonomously, each robot uses a pre-specified control algorithm to complete its task. For the delivery robot example, this could be, for instance, a SLAM-based local path planner that the robot uses for navigating between the waypoints. We will not model the details of this control algorithm but simply assume that this control is imperfect and occasionally causes the robot to enter a fault state while doing a task (e.g., delivery robot getting stuck in a pothole or losing its localization). We model this behaviour by assuming that while completing each task, the robot may be in one of the two internal states: a *normal* state (denoted by $s = 0$) or a *fault* state (denoted by $s = 1$). When a robot is being assisted, it may still be possible for it to enter into a fault state.

The operating state of robot $k \in \mathcal{K}$ at time t, denoted by $x_t^k = (n_t^k, s_t^k)$, is tuple of its current task and internal state. The state space for robot k is given by

$$\mathcal{X}^k := \bigcup_{n=1}^{N^k} \{(n, 0), (n, 1)\} \cup \{(G, 0)\},$$

where the terminal state $(G, 0)$ indicates that all tasks have been completed. The state space for all robots is denoted by $\boldsymbol{\mathcal{X}} = \mathcal{X}^1 \times \cdots \times \mathcal{X}^K$.

The state of a robot evolves differently depending on whether it is operating autonomously (denoted by mode $a^k = 0$) or assisted (denoted by $a^k = 1$). Given robot $k \in \mathcal{K}$ in state $(n, s) \in \mathcal{X}^k$ operating in mode $a \in \{0, 1\}$, let p^{ka}_{ns} denote the probability of successfully completing the current task at the current time step and let q^{ka}_{ns} denote the probability of toggling the current internal state (i.e. going from normal to fault state and vice-versa). A diagram describing these transitions is shown in Fig. 5.2. Note that the terminal state $(G, 0)$ is an absorbing state, so $p^{ka}_{G0} = 0$ and $q^{ka}_{G0} = 0$.

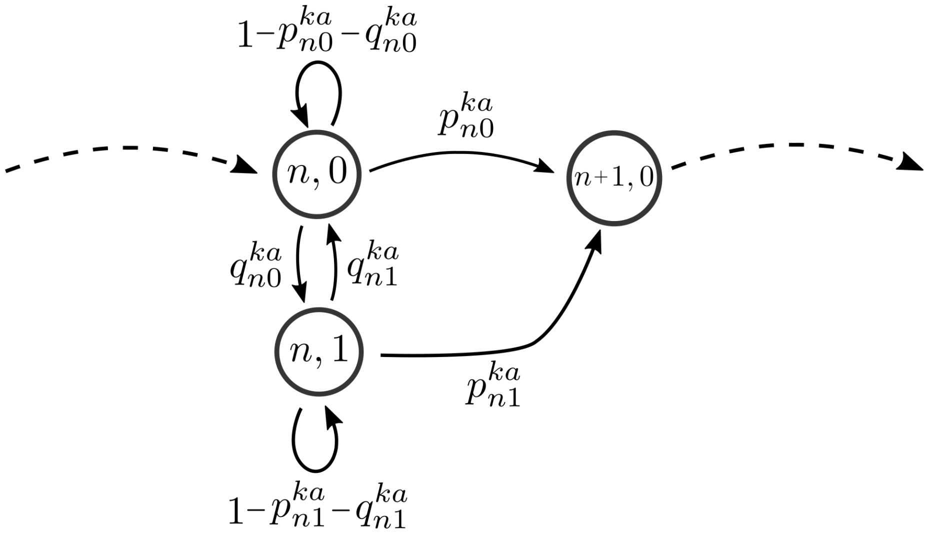


Figure 5.2: State-transition diagram for robot k working on task n, where in $(n, 0)$ the robot is in the normal state $s = 0$, and in $(n, 1)$ the robot is in the fault state $s = 1$. Transitions can occur between $(n, 0), (n, 1)$ and $(n + 1, 0)$, and the probabilities change with operating mode a.

There is a per-step cost $C^k \colon \mathcal{X}^k \times \{0, 1\} \to \mathbb{R}_{\geq 0}$, where $C^k((n^k, s^k), a^k)$ denotes the cost of operating robot $k \in \mathcal{K}$ in mode a^k when the robot is in state (n^k, s^k). Note that the per-step cost is zero in the terminal state, i.e, $C^k((G, 0), a) = 0$.

5.2.2 Model of the decision support system (DSS)

There is a decision support system that helps to allocate operators to the robots. At each time the decision support system observes the operating state $\boldsymbol{X}_t \coloneqq (X^1_k, \ldots, X^K_t)$

of all robots and picks at most M robots to assist. We capture this by the allocation $\boldsymbol{A}_t = (A_t^1, \dots, A_t^K) \in \mathcal{A}$, where

$$\mathcal{A} = \left\{ \boldsymbol{a} \coloneqq (a^1, \dots, a^K) \in \{0,1\}^K : \sum_{k=1}^{K} a^k \leq M \right\}. \tag{5.1}$$

The allocation is selected according to a time-homogeneous Markov policy $\pi : \boldsymbol{\mathcal{X}} \to \mathcal{A}$. The expected total cost incurred by any policy π is given by

$$J(\pi) = \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \sum_{k=1}^{K} C^k(X_t^k, A_t^k) \;\middle|\; \boldsymbol{X}_0 = \boldsymbol{x}_0 \right], \tag{5.2}$$

where $\gamma \in (0,1)$ is the discount factor and $\boldsymbol{x}_0 = (x_0^1, \dots, x_0^K)$ is the initial state with $x_0^k = (1,0)$ for every $k \in \mathcal{K}$.

5.2.3 Problem objective

We impose the following assumptions on the model:

(A1) Given an allocation $\boldsymbol{a} = (a^1, \dots, a^K)$ by the DSS, the operating states of the robots evolve independently of each other.

(A2) For every robot $k \in \mathcal{K}$, the probability of getting out the faulty internal state when assisted is strictly greater than 0, i.e., $p_{n1}^{k1} + q_{n1}^{k1} > 0$.

(A3) Under autonomous operation, a robot stays in the fault state, i.e., $p_{n1}^{k0} = q_{n1}^{k0} = 0$.

The design objective is to solve the following optimization problem:

Problem 5.1. *Given the set $\mathcal{K}$ of robots, the system dynamics and the per-step costs, the number M of human operators, and the discount factor $\gamma \in (0,1)$, choose a policy $\pi : \boldsymbol{\mathcal{X}} \to \mathcal{A}$ to minimize the total discounted cost $J(\pi)$ given by* (5.2).

Optimal solution for Problem 5.1 can be found by modelling it as a Markov decision process and solving using dynamic programming [196]. However, the sizes of state and action spaces of the resulting model grows exponential with the number of robots and operators. Thus, solving Problem 5.1 using dynamic programming becomes intractable for larger systems. To address this, we model Problem 5.1 as a restless multi-armed bandit (RMAB) problem and use the notion of indexability to find an efficient and scalable policy. We start by providing an overview of RMAB in the next section.

5.3 Indexability of the assistance problem

Problem 5.1 can be formulated as an instance of RMAB, where each robot corresponds to an arm. Under such a formulation, the state $\tilde{Z}_t^k$ of arm k corresponds to operating state $x_t^k = (n_t^k, s_t^k)$ of robot k. The transition function $\tilde{T}^k$ corresponds to the robot state evolution shown in Fig. 5.2 and the cost function $\tilde{C}^k$ corresponds to the associated per-step cost C^k. In addition, allocating an operator to a robot corresponds to choosing the active action for that arm while autonomous operation corresponds to choosing the passive action. This motivates using the Whittle index policy to solve Problem 5.1. However, before we can implement this approach, we must check for indexability of the problem. As discussed earlier, there is no universal framework to verify indexability of a problem. Moreover, the optimal policy for the given problem does not show any threshold-based behaviour. Therefore, we determine sufficient conditions for indexability from first principles by using properties of the value function of each individual arm.

Since indexability has to be checked for each arm separately, for this analysis, we drop the superscript k from all variables.

Let $V_\lambda : \mathcal{X} \to \mathbb{R}$ be the unique fixed point of the following equation

$$V_\lambda(x) = \min_{a \in \{0,1\}} Q_\lambda(x, a),$$

where

$$Q_\lambda(x, a) = C(x, a) + \lambda a + \gamma \sum_{x' \in \mathcal{X}} T(x'|x, a) V_\lambda(x'), \tag{5.3}$$

represents the Q-value of taking action a in state x. Here the transition function $T(x'|x, a)$ denotes the probability of transition from state x to state x' under action a and is represented by Fig. 5.2. Let $\pi_\lambda : \mathcal{X} \to \{0, 1\}$ be the corresponding optimal policy

$$\pi_\lambda(x) = \arg\min_{a \in \{0,1\}} Q_\lambda(x, a).$$

To ensure uniqueness of the $\arg\min$, we follow the convention that when $Q_\lambda(x, 0) = Q_\lambda(x, 1)$, the passive action $a = 0$ is chosen. Let $\mathcal{P}(\lambda)$ be the passive set given penalty λ and $w(x)$ be the Whittle index of state x for the problem of operator allocation in a single-robot system. Furthermore, define the *benefit function* as

$$B_\lambda(x) = Q_\lambda(x, 1) - Q_\lambda(x, 0). \tag{5.4}$$

Then, a sufficient condition for indexability is as follows:

Lemma 3. *A sufficient condition for Problem 5.1 to be indexable is that the benefit function $B_\lambda(x)$ for each robot is monotonically increasing in λ for all states $x \in \mathcal{X}$.*

Proof. The result follows from the observation that using (5.4) and Def. 3.11, we can re-write the passive set as

$$\mathcal{P}(\lambda) = \{x \in \mathcal{X} : B_\lambda(x) \geq 0\}. \tag{5.5}$$

Thus, monotonicity of the benefit function $B_\lambda(x)$ implies that the condition for indexability given in Def. 3.12 is satisfied.

□

We verify the monotonicity of $B_\lambda(x)$ by finding bounds on the value function and establish the following:

Theorem 2. *Let $r^a_{ns} \triangleq 1 - p^a_{ns} - q^a_{ns}$ denote the probability of repeating a task n under mode a with internal state s. Define $\alpha_1(n)$ and $\beta_0(n)$ as follows:*

$$\alpha_1(n) = 1 + \frac{\gamma q^1_{n0}}{1 - \gamma\, r^1_{n1}} + \frac{\gamma q^0_{n0}\left(\gamma\, r^1_{n0} + \dfrac{\gamma^2 q^1_{n0} q^1_{n1}}{1 - \gamma\, r^1_{n1}} - 1\right)}{1 - \gamma\, r^1_{n1} - \gamma\, r^0_{n0} + \gamma^2 r^1_{n1} r^0_{n0} - \gamma^2 q^0_{n0} q^1_{n1}},$$

and

$$\beta_0(n) = \frac{\gamma(p^1_{n0} - p^0_{n0}) + \gamma^2(p^0_{n0} r^1_{n0} - p^1_{n0} r^0_{n0})}{1 - \gamma\, r^0_{n0}}.$$

Then, the single-robot problem is indexable if for all $n \in \{1, 2, \ldots, N\}$:

$$\alpha_1(n) \geq 0 \quad \textit{and} \quad \frac{\beta_0(n)}{1-\gamma} \geq -1. \tag{5.6}$$

Proof. See Section 5.7. □

The multi-robot problem is indexable if the conditions given in Theorem 2 hold true for all robots. In the next section, we present specific instances of the general model described in Section 5.2 which are indexable and discuss their relevance in practical assistance problems for (semi)autonomous delivery robots.

5.4 Special Cases: Robot transitions in the city

This section presents two specific classes of transition functions which represent two types of faults commonly occurring in systems with remote navigating robots.

5.4.1 Transition Type-1 : Faults with continuation

Consider the following transition behaviour along a robot's waypoints. At each time step, the robot moves to its next waypoint with a probability representing, for example, the crowd in the area. There is also a probability of getting into a fault state such as encountering an unidentifiable obstacle. A human operator can assist the robot to its next waypoint both from a normal or fault state. Such transitions represent faults where the robot is functioning properly but is unsure about how to proceed due to uncertainty in its surroundings. Thus the probability of success when being assisted is the same regardless of whether the robot is in its normal state or stopped in the fault state, i.e., $p^1_{n0} = p^1_{n1}$ and $q^1_{n0} = q^1_{n1} = 0$. The corresponding transition dynamics are shown in Figure 5.3. Note that in this case $r^1_{n0} = r^1_{n1} = 1 - p^1_{n0}$.

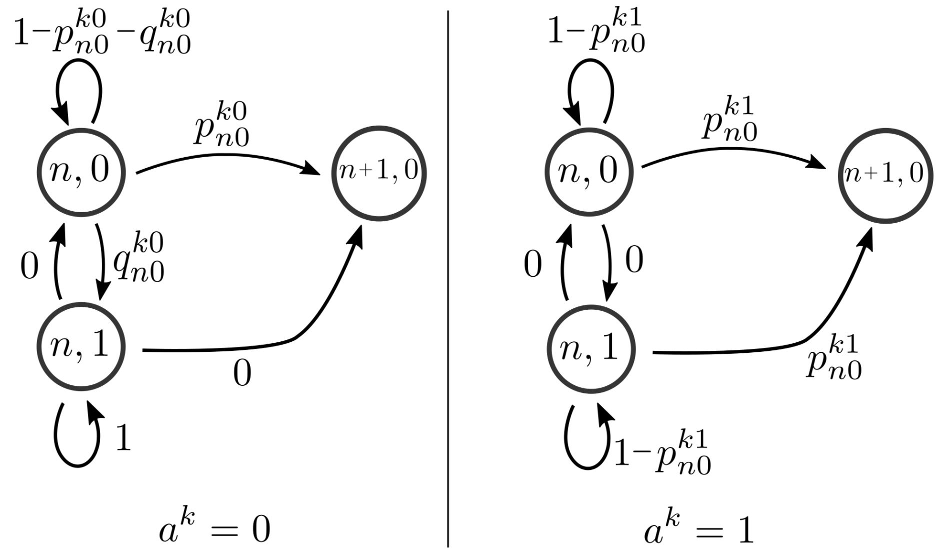


Figure 5.3: State-transition probabilities under the autonomous operation and teleoperation for type-1 transitions.

In this case, the coefficients $\alpha_1(n)$ and $\beta_0(n)$ can be simplified to the following expressions:

$$\alpha_1(n) = 1 - \frac{\gamma q^0_{n0}}{1-\gamma r^0_{n0}},$$
$$\beta_0(n) = \frac{\gamma(1-\gamma)(r^0_{n0} - r^1_{n0}) + \gamma q^0_{n0}(1-\gamma\, r^1_{n0})}{1-\gamma\, r^0_{n0}}. \tag{5.7}$$

Note that

$$\alpha_1(n) = \frac{1-\gamma+\gamma p^0_{n0}}{1-\gamma r^0_{n0}} \geq 0,$$
$$\frac{\beta_0(n)}{1-\gamma} + 1 \geq \frac{\gamma(r^0_{n0} - r^1_{n1})}{1-\gamma r^0_{n0}} + 1 = \frac{1-\gamma r^1_{n1}}{1-\gamma r^0_{n0}} \geq 0.$$

Thus, $\alpha_1(n)$ and $\beta_0(n)$ satisfy the sufficient condition of Theorem 2 for all allowed values of transition probabilities and the discount factor γ. Therefore, any robot following the Type-1 transitions is indexable.

5.4.2 Transition Type-2 : Faults with reset

Consider another type of transition where the robot can get into a fault state and needs error fixing while staying at its next waypoint. This includes scenarios such as losing localization or getting stuck in a minor obstacle. The human operator can try to assist the robot out of that situation by fixing the fault, resetting it back to its current waypoint (assuming the system is equipped with means to do so). Such transitions will mean that the probabilities $q^1_{n0} = p^1_{n1} = 0$ and the corresponding transition dynamics are shown in Fig. 5.4:

Substituting the values of transition probabilities from Fig. 5.4 to the expressions of $\alpha_1(n)$ and $\beta_0(n)$, the coefficients can be simplified to the following:

$$\alpha_1(n) = 1 - \frac{\gamma q^0_{n0}(1-\gamma r^1_{n0})}{(1-\gamma r^0_{n0})(1-\gamma(1-q^1_{n1})) - \gamma^2 q^0_{n0} q^1_{n1}},$$
$$\beta_0(n) = \frac{\gamma(1-\gamma)(r^0_{n0} - r^1_{n0}) + \gamma q^0_{n0}(1-\gamma\, r^1_{n0})}{1-\gamma\, r^0_{n0}}. \tag{5.8}$$

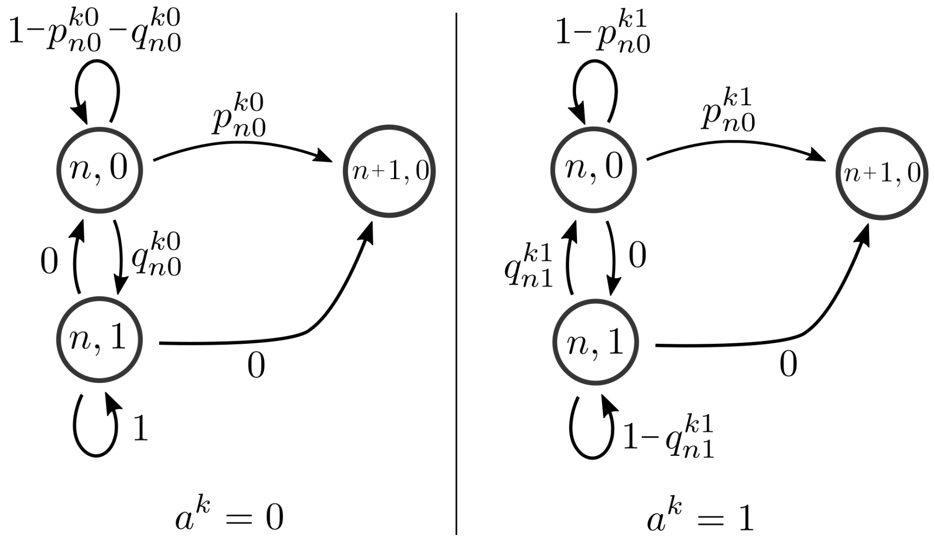


Figure 5.4: State-transition probabilities under the autonomous and teleoperate/assist actions for Type-2 transition dynamics.

Note that $\beta_0(n)$ here is the same as (5.7) and, therefore, satisfies (5.6). For $\alpha_1(n)$ to satisfy (5.6), the condition $\alpha_1(n) \geq 0$ results in the following condition on q_{n1}^1:

$$q_{n1}^1 \geq 1 - \frac{1}{\gamma} + \frac{\gamma q_{n0}^0 p_{n0}^1}{1-\gamma r_{n0}^0 - \gamma q_{n0}^0}. \tag{5.9}$$

As $q_{n1}^1 \leq 1$, (5.9) also yields the following condition on q_{n0}^0:

$$q_{n0}^0 \leq \frac{1-\gamma r_{n0}^0}{\gamma(1+\gamma p_{n0}^1)}. \tag{5.10}$$

Therefore, any robot state following the Type-2 transitions will satisfy the condition of indexability in Theorem 3 if (5.9) and (5.10) are satisfied, i.e., the probability q_{n0}^0 that the robot transitions from a normal state to fault state during autonomous operation is not too high and the probability q_{n1}^1 that the operator brings the robot from a fault state to a normal state is not too small.

As an example, consider a robot following Type-2 transitions with $p_{n0}^0 = q_{n0}^0 = p_{n0}^1 = 0.3$ and $\gamma = 0.95$. In this setting, any $q_{n0}^0 \in [0,1]$ satisfies (5.10) and any $q_{n1}^1 \in [0.1462, 1]$ satisfies (5.9). Thus, the model is indexable if there is at least a 14.62% chance that teleoperation successfully resets the robot from the fault state to a normal state.

5.5 Computation of Whittle Index

As discussed in Section 5.1, once the indexability of the problem instance has been verified, we can compute Whittle indices for all robots and determine the Whittle index policy.

General approaches of computing Whittle indices are either based on adaptive greedy algorithm [214, 219] or binary search [218]. In this section, we briefly provide details on adaptive greedy algorithm and describe how the Whittle index policy works. Readers are encouraged to refer to [214] for a detailed explanation and validation of the algorithm. The algorithm is presented in Alg. 5.1 for computing Whittle indices for a single robot.

Algorithm 5.1 Adaptive Greedy Algorithm for Whittle Index Computation

```
1: Input: Robot $(\mathcal{X}, \mathcal{A}, T, C, \gamma, x_0)$.
2: Initialize $\mathcal{P} = \emptyset$.
3: while $\mathcal{P} \neq \mathcal{X}$ do
4:     Compute $\mu_y^*$, $\forall y \in \mathcal{X} \setminus \mathcal{P}$ using Eq. (5.11).
5:     $\lambda^* \leftarrow \min_{y \in \mathcal{X} \setminus \mathcal{P}} \ \mu_y^*$
6:     $\mathcal{Y} \leftarrow \arg\min_{y \in \mathcal{X} \setminus \mathcal{P}} \ \mu_y^*$
7:     $w(y) \leftarrow \lambda^*$, $\forall y \in \mathcal{Y}$
8:     $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{Y}$
```

The algorithm operates as follows: For any subset $\mathcal{Z} \subseteq \mathcal{X}$, define the policy vector[1] $\pi^{(\mathcal{Z})} : \mathcal{X} \to \{0, 1\}$ as

$$\pi^{(\mathcal{Z})}(x) = \begin{cases} 0, & \text{if } x \in \mathcal{Z} \\ 1, & \text{if } x \in \mathcal{X} \setminus \mathcal{Z}. \end{cases}$$

Also define, $C_\pi = \left[C(x, \pi(x))\right]_{x \in \mathcal{X}}$, the cost vector for all states under a policy π, and $T_\pi = \left[T(x'|x, \pi(x))\right]_{x, x' \in \mathcal{X}}$, the transition matrix under policy π.

Then, in each iteration of the while loop, compute μ_y^* as follows:

$$\mu_y(x) = -\frac{D_{\pi^{(\mathcal{P})}}(x) - D_{\pi^{(\mathcal{P} \cup \{y\})}}(x)}{N_{\pi^{(\mathcal{P})}}(x) - N_{\pi^{(\mathcal{P} \cup \{y\})}}(x)}, \ \forall x \in \mathcal{X}$$

$$\mu_y^* = \min_{x \in \Lambda_y} \mu_y(x), \tag{5.11}$$

[1]In the following expressions π is used as a vector of size $|\mathcal{X}|$, constructed as a mapping from each state to corresponding action $a \in \{0, 1\}$.

where

$$D_\pi(x) = \Big[(I - \gamma T_\pi)^{-1} C_\pi\Big](x) \ \ ,$$
$$N_\pi(x) = \Big[(I - \gamma T_\pi)^{-1} \pi\Big](x) \ \ ,$$
$$\Lambda_y = \{s \in \mathcal{X} : N_{\pi(\mathcal{P})}(x) - N_{\pi(\mathcal{P} \cup \{y\})}(x) \neq 0\}.$$

The minimum value of μ_y^* calculated in Line 5 in Alg. 5.1 corresponds to the Whittle indices of the minimizing states (Line 6). These states are then taken out of consideration in the next iteration of the while loop by including them in the passive set $\mathcal{P}$. When the Alg. 5.1 exits the while loop, the Whittle indices for all states of that robot are calculated. This procedure is then repeated for all the robots in the system.

Once the Whittle indices for all states of all robots are obtained, the Whittle index policy can be implemented as given in Alg. 5.2. In Line 2 of the algorithm, the function `arg_top_M`$(\{w^k(x^k)\})$ returns indices of top M positive elements in a set, where ties are broken randomly. As determined in [214], the computational complexity of this method is $\mathcal{O}(K|\mathcal{X}|^3)$. In contrast, the computational complexity of finding the optimal policy for Problem 5.1 is $\mathcal{O}(\binom{K}{M}|\mathcal{X}|^{2K})$ using value iteration, where $|\mathcal{X}|$ is the size of state space of individual robot.

Algorithm 5.2 Whittle Index Policy π^I

```
1: Input: Set of Whittle indices $w^k(x^k)$ for all $k \in \{1, \ldots, K\}$ and $x^k \in \mathcal{S}^k$, No. of Operators $M$
2: $\mathcal{M} \leftarrow$ arg_top_M$(\{w^k(x^k)\})$
3: $a^k \leftarrow 0$ for all $k \notin \mathcal{M}$
4: $a^k \leftarrow 1$ for all $k \in \mathcal{M}$        // Allocate operators
5: return $(a^1, \ldots, a^K)$
```

5.6 Simulations and Results

In this section, we present performance results for a simulated multi-robot assistance problem under the following policies (described later): 1) Optimal policy, 2) Index policy, 3) Benefit maximizing Policy, 4) Myopic Policy, and 5) Reactive Policy. The problem and the solution frameworks for all policies were implemented using POMDPs.jl library in Julia [220].

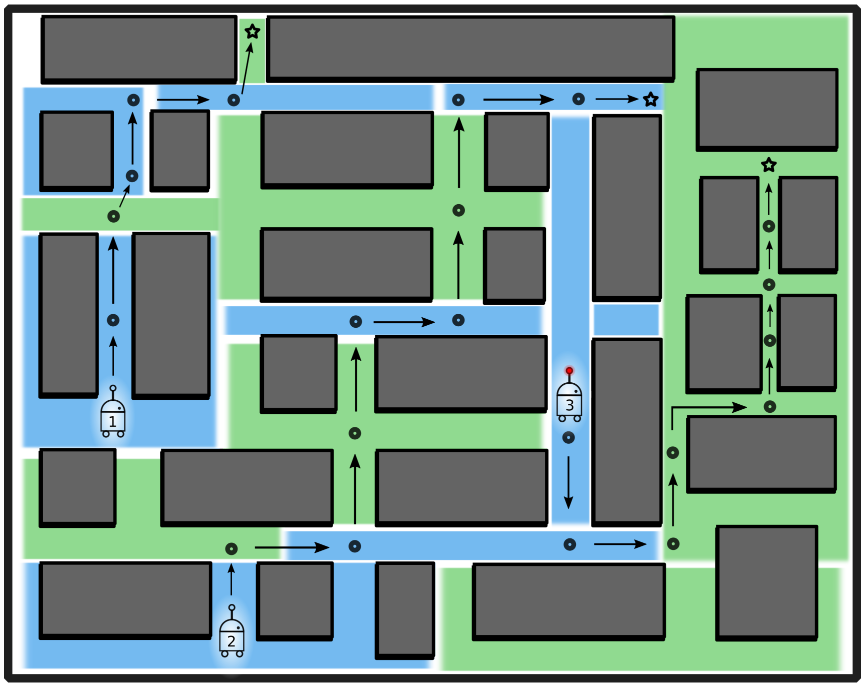


Figure 5.5: An instance of multi-robot assistance problem for robots navigating in a city block-like environment. Transition zones are marked by different color shadings, representing type-1 and type-2 transitions, as described in Section 5.4. Three robots are shown navigating to their corresponding goal locations, via a sequence of waypoints shown as black circles. These waypoints may lie in different transition zones resulting in varied performance for the robots.

5.6.1 Simulation Setup

For the simulations, a city map is generated as shown in Fig. 5.5 where the map is randomly divided into different zones corresponding to one of the two transition types presented in Section 5.4.

The exact values of transition probabilities at different locations in the map are sampled randomly from a uniform distribution, according to Table 5.1. The bounds on transition probabilities $\overline{q}_{n1}^{k1}$ and $\overline{q}_{n0}^{k0}$ for transition type-2 are determined by (5.9) and (5.10) respectively.

Table 5.1: Probabilities values used for simulations for the two types of transitions.

Probability	Type-1	Type-2
r_{n0}^{k0}	$[0.2, 0.5]$	$[0.2, 0.5]$
q_{n0}^{k0}	$[0.2, 0.5]$	$[0.1, \min\{\overline{q}_{n0}^{k0}, 1 - r_{n0}^{k0}\}]$
r_{n0}^{k1}	$[0.1, 0.4]$	$[0.1, 0.4]$
q_{n0}^{k1}	0.0	0.0
r_{n1}^{k1}	r_{n0}^{k1}	$1 - q_{n1}^{k1}$
q_{n1}^{k1}	0.0	$[\max\{\overline{q}_{n1}^{k1}, 0.1\}, 0.9]$

For the teleoperation problem, we use the following cost structure:

$$C^k\left((n,s),a\right) = \begin{cases} 0 & \text{if } n = G \\ \rho_n^k & \text{if } a = 0, s = 0 \\ \phi_n^k & \text{if } a = 0, s = 1 \\ \rho_n^k + \rho_T^k & \text{if } a = 1, s = 0 \\ \phi_n^k + \rho_T & \text{if } a = 1, s = 1, \end{cases} \tag{5.12}$$

with $\rho_n^k, \phi_n^k, \rho_T^k \in \mathbb{R}_{\geq 0}$ for any $n \in \{1, \dots, N^k\}$. This cost function captures the time that a robot takes to reach its goal, i.e., zero cost on reaching goal, non-negative costs for intermediate states, and an additional cost while being assisted.

Values of the different costs and the discount factor used are specified in Table 5.2.

Table 5.2: Parameter values used in the simulated operator allocation problem.

Parameter	ρ_T	ρ	ϕ	γ
Value	0.75	2.0	4.0	0.99

Separate tests were performed to test the validity, performance and scalability of the Index policy. At the beginning of each simulation, a number of robots are placed on the map (ranging from 1 to 50) with randomly generated start and goal locations, and 7 waypoints uniformly placed between the two. In practice, these waypoints are generated by a separate robot path planner for each individual robot, and are considered as an input for the operator allocation problem.

5.6.2 Baseline Policies

We consider the following baseline policies to assess the performance of the Index policy.

Optimal policy

The Optimal policy $\pi^* : \boldsymbol{\mathcal{X}} \to \mathcal{A}$, as defined by (5.2), is found by encoding the complete problem with all robots as an MDP and solving it using the *Sparse Value Iteration Solver* from the POMDP.jl library.

Reactive policy

The reactive policy allocates an operator to any robot stuck in a fault state. If there are more such robots than operators, a random subset of those robots is selected.

Myopic policy

Myopic/Greedy Policies are commonly used to obtain fast (but sub-optimal) solutions to intractable problems. For a comparison, we implement an l-step myopic policy presented in [38] for $l \in \{1, 2\}$. Define $V^{\mathbf{0}}(\boldsymbol{x}_{t+1})$ as the expected cost incurred by the system from current time step to infinity under passive actions. The l-step myopic policy $\pi^{G\text{-}l} : \boldsymbol{\mathcal{X}} \to \mathcal{A}$ is then defined as

$$\pi^{G\text{-}l}(\boldsymbol{x}_t) = \arg\min_{\boldsymbol{a} \in \mathcal{A}} \; g(\boldsymbol{x}_t, \boldsymbol{a}, l), \tag{5.13}$$

where the l-step look-ahead cost $g(\boldsymbol{x}_t, \boldsymbol{a}, l)$ is given by

$$g(\boldsymbol{x}_t, \boldsymbol{a}, l) = \begin{cases} C\left(\boldsymbol{x}_t, \boldsymbol{a}\right) + \sum\limits_{\boldsymbol{x}_{t+1}} \gamma \, T(\boldsymbol{x}_{t+1} | \boldsymbol{x}_t, \boldsymbol{a}) \, g(\boldsymbol{x}_t, \boldsymbol{a}, l-1), & \text{if } l \neq 0, \\ V^{\mathbf{0}}(\boldsymbol{x}_{t+1}), & \text{otherwise,} \end{cases}$$

where

$$C(\boldsymbol{x}, \boldsymbol{a}) = \sum_{k=1}^{K} C^k\left(x^k, a^k\right),$$

is the cost incurred in the current time step.

Benefit maximizing policy

For comparison, we also propose a heuristic policy which we call *benefit maximizing policy* that tries to exploit the independence of the robots' transitions. This policy is inspired by the *advantage function* used in reinforcement learning (for example, see [221]). The policy considers the *benefit* or *advantage* of taking the active action over the passive action for each robot and picks the robots with highest benefit at each time step, i.e.,

$$\pi^B(\boldsymbol{x}_t) = \arg\min_{\boldsymbol{a}\in\mathcal{A}} \sum_{k=1}^{K} a^k\ B_0(x_t^k), \tag{5.14}$$

where $B_0(x)$ corresponds to $B_\lambda(s)$ defined in (5.4) with $\lambda = 0$.

5.6.3 Comparison with the Optimal policy

First, the Index policy is compared against the Optimal policy to validate its applicability for our problem. Due to its poor scalability, the Optimal policy cannot be computed for larger problem instances, therefore this test is limited to smaller problem size (up to 4 robots and 2 operators). The relative performance (ratio of the cost incurred under Index policy to that under Optimal policy) is shown in Fig. 5.6. For comparison, 100 problem instances were tested under both policies and were simulated through Monte Carlo rollouts. Each problem instance is run until all robots reach their respective goal locations. This is repeated for 10^6 iterations and average cost is recorded.

It is observed that the Index policy performs quite close to the optimal policy for all test cases. As the ratio of number of robots to number of operators increases, the Index policy starts to degrade in comparison to the optimal. However, the relative cost for most cases still remains within 5% of the optimal.

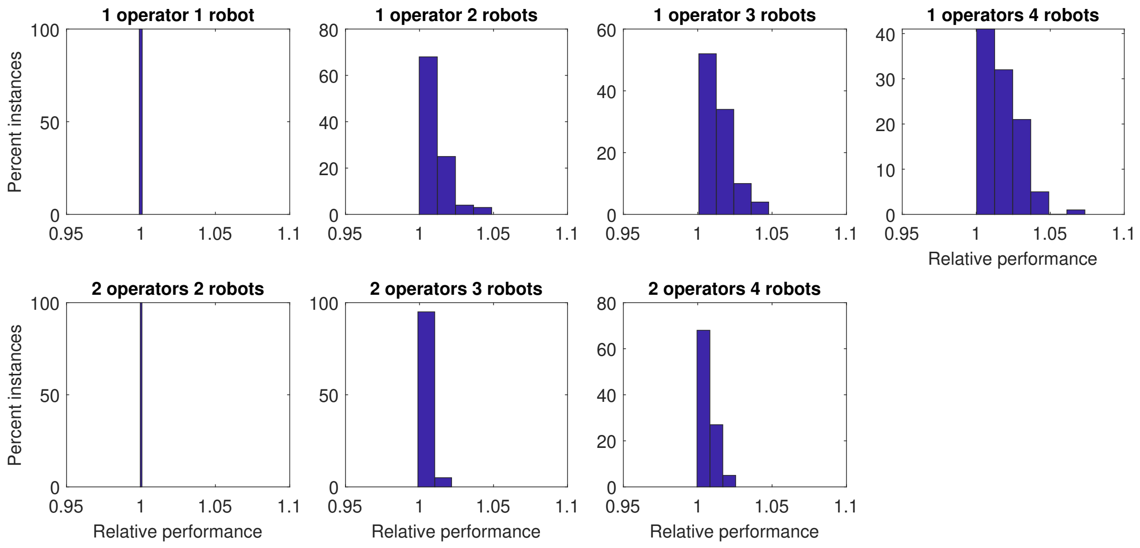


Figure 5.6: Relative performance of Index policy compared to the optimal policy. The plots show distribution of 100 indexable problem instances based on their performance under the two policies. Relative performance is calculated as the ratio of the average cost incurred under Index policy to that under Optimal policy.

5.6.4 Comparison with other baseline policies

Next, we compare the performance (measured as average cost incurred per robot before reaching its goal) of the Index policy with the three baseline policies on larger problem instances. For the comparison, a set of 100 problem instances is created, each with a set of 7 waypoints with randomly sampled transition probabilities according to Table 5.1. Each instance of the problem is then simulated separately under the different policies using Monte Carlo rollouts until all robots reach their goal states, repeated for 500 iterations. Each simulation iteration (rollout) is timed out at 10 seconds for each policy. If an iteration takes longer than this time, the simulations are interrupted and the policy's result for that test condition is not reported.

Figure 5.7 shows performance comparison of the four policies. The Index policy performs best out of the four policies, followed by the benefit maximizing policy (π^B) and the myopic policy (π^G). The Reactive policy performed the worst as expected. As a side note, the average cost incurred per robot under any policy is strongly correlated with the ratio of number of robots to the number of available operators. This observation supports the intuition that as human operators are required to distribute their assistance among more robots, their effectiveness decreases.

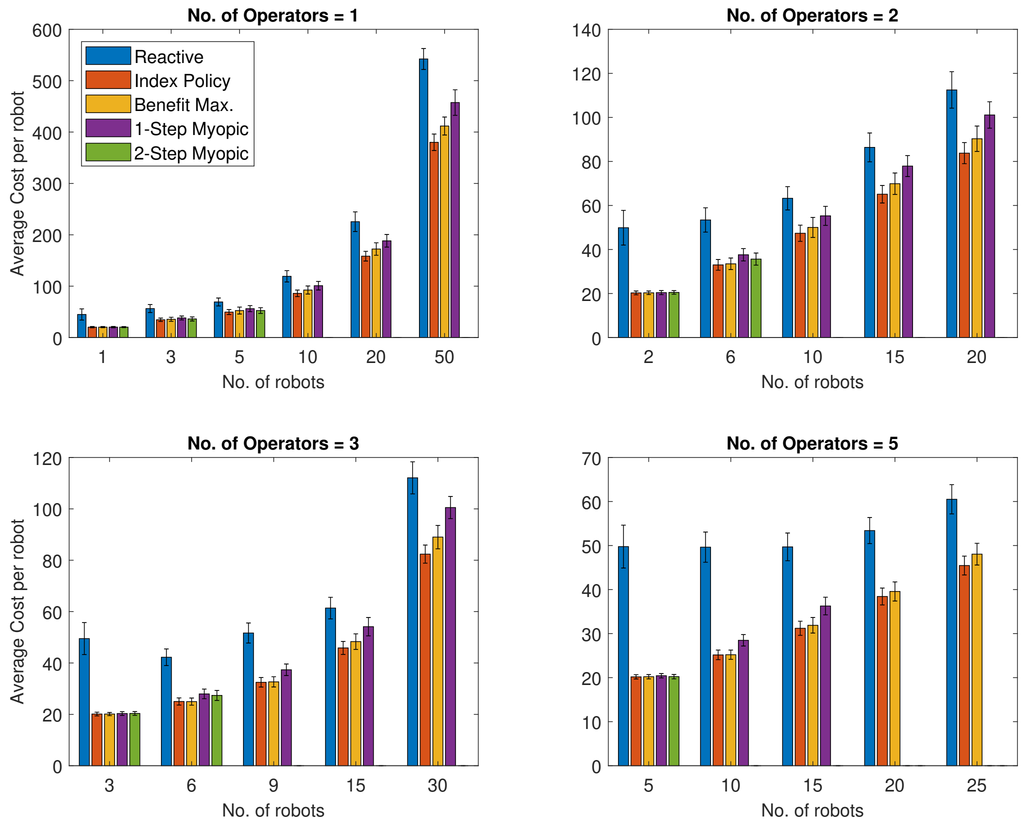


Figure 5.7: Performance comparison of the four policies for different number of operators available for allocation. Error bars in the plots show one standard-deviation above and below the average. Note that in larger problem instances, the simulations for myopic policies timed out and could not be completed in the specified time limit (10 sec per rollout).

5.6.5 Scalability

Table 5.3 shows the time that each policy takes to compute operator the allocation under different problem sizes. For these simulations, each robot is set to have 7 waypoints. As observed from the table, the computation times of the two Myopic policies scale exponentially with both the number of robots and the number of operators, with the time for 2-step Myopic policy growing at a much higher rate.

The computation times for the index and benefit maximizing policies scale linearly with the number of robots and are independent of the number of operators. Also, note that the Whittle index computation for one robot is independent from the rest. Therefore robots can be added/removed without re-computation of already computed indices. Furthermore, if the number of operators changes to $M > 1$, the policy simply allocates operators to the robots with the M highest Whittle indices. As a result, the policy is efficiently scalable with the number of robots and operators. For reference, the simulations were run on a Desktop PC with a 4 core, 4.20 GHz processor and 32 GB of RAM.

Table 5.3: Computation times of different policies (seconds).

Operators/ Robots	Index Policy	1-step Myopic	2-step Myopic	Benefit Maximizing
2/6	1.2e−6	3.8e−3	2.4e−1	3.4e−6
3/6	1.2e−6	6.2e−3	4.8e−1	3.4e−6
4/6	1.2e−6	8.1e−3	6.6e−1	3.4e−6
1/9	1.7e−6	9.6e−2	4.8e+0	4.9e−6
2/9	1.7e−6	2.6e−1	2.2e+1	4.9e−6
3/9	1.7e−6	6.4e−1	6.2e+1	4.9e−6

5.7 Proofs and Preliminary Results

For any fixed value of λ, the value function $V_\lambda(x)$ can also be written as

$$V_\lambda(x) = \min_{\pi\in\Pi} \mathbb{E}\Big[\sum_{t=0}^{T}\Big[C(X_t, A_t) + \lambda A_t\Big]|X_0 = x\Big],$$

where Π denotes the set of all Markov policies from $\mathcal{X}$ to $\{0,1\}$. Since the state space $\mathcal{X}$ is finite, so is Π. Thus, $V_\lambda(x)$ is the minimum of a finite number of functions, each of which is linear in λ. Therefore, $V_\lambda(x)$ is continuous and piecewise linear, with a finite number of corner points. This means $V_\lambda(x)$, and therefore $Q_\lambda(x,a)$ and $B_\lambda(x)$, are non differential w.r.t. λ at a finite number of points. Therefore, $B_\lambda(x)$ is monotonically increasing if $\partial B_\lambda(x)/\partial\lambda$, wherever it exists, is non-negative. Let $\Lambda^*(x)$ denote the finite set of values where $B_\lambda(x)$ is non-differentiable. Let $\Lambda^* = \cup_{x\in\mathcal{X}}\Lambda^*(x)$, which is also finite.

The main idea for the proof of Theorem 2 is to show that if (5.6) is satisfied then $\partial B_\lambda(x)/\partial\lambda$ is non-negative for $\lambda \notin \Lambda^*$. Then, Lemma 3 implies the indexability of the problem.

Now fix an $n \in \{1, \dots, N\}$. Define $z = (n, 0)$, $z' = (n+1, 0)$ and $e = (n, 1)$. Then using Eq. (5.3) and Fig. 5.2, we have

$$Q_\lambda(z, a) = C(z, a) + \lambda a + \gamma q^a_{n0} V_\lambda(e) + \gamma p^a_{n0} V_\lambda(z') + \gamma r^a_{n0} V_\lambda(z), \tag{5.15}$$

$$Q_\lambda(e, a) = C(e, a) + \lambda a + \gamma q^a_{n1} V_\lambda(z) + \gamma p^a_{n1} V_\lambda(z') + \gamma r^a_{n1} V_\lambda(e). \tag{5.16}$$

Then we have the following results:

Lemma 4. *For all $\lambda \notin \Lambda^*$,*

$$0 \le \frac{\partial V_\lambda(x)}{\partial \lambda} \le \frac{1}{1-\gamma}, \quad \forall x \in \mathcal{X}.$$

Proof. Under an optimal policy π^*, we have:

$$V_\lambda(x) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t C_\lambda(X_t, \pi^*(X_t)) \middle| X_0 = x\right].$$

Therefore, we get

$$\begin{aligned}\frac{\partial V_\lambda(x)}{\partial \lambda} &= \frac{\partial}{\partial \lambda}\mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t C_\lambda(X_t, \pi^*(X_t)) \middle| X_0 = x\right] \\ &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \frac{\partial}{\partial \lambda} C_\lambda(X_t, \pi^*(X_t)) \middle| X_0 = x\right].\end{aligned}$$

Since $\frac{\partial}{\partial \lambda} C_\lambda(x, \pi^*(x)) \in [0, 1]$ for all $x \in \mathcal{X}$, we can write

$$0 \le \frac{\partial V_\lambda(x)}{\partial \lambda} \le \sum_{t=0}^{\infty} \gamma^t \implies 0 \le \frac{\partial V_\lambda(x)}{\partial \lambda} \le \frac{1}{1-\gamma}.$$

□

Define

$$\alpha_0(n) = 1,$$

$$\beta_0(n) = \frac{\gamma(p^1_{n0} - p^0_{n0}) + \gamma^2(p^0_{n0}r^1_{n0} - p^1_{n0}r^0_{n0})}{1 - \gamma\, r^0_{n0}},$$

$$\alpha_1(n) = 1 + \frac{\gamma q^1_{n0}}{1 - \gamma\, r^1_{n1}} + \frac{\gamma q^0_{n0}\left(\gamma\, r^1_{n0} + \dfrac{\gamma^2 q^1_{n0} q^1_{n1}}{1 - \gamma\, r^1_{n1}} - 1\right)}{1 - \gamma\, r^1_{n1} - \gamma\, r^0_{n0} + \gamma^2 r^1_{n1} r^0_{n0} - \gamma^2 q^0_{n0} q^1_{n1}},$$

$$\beta_1(n) = \gamma p^1_{n0} + \frac{\gamma^2 q^1_{n0} p^1_{n0}}{1 - \gamma r^1_{n1}} + \frac{(\gamma p^0_{n0} - \gamma^2 p^0_{n0} r^1_{n1} + \gamma^2 q^0_{n0} p^1_{n0})\left(\gamma r^1_{n0} + \dfrac{\gamma^2 q^1_{n0} q^1_{n1}}{1 - \gamma r^1_{n1}} - 1\right)}{1 - \gamma r^1_{n1} - \gamma r^0_{n0} + \gamma^2 r^1_{n1} r^0_{n0} - \gamma^2 q^0_{n0} q^1_{n1}}.$$

Also define $b_{00}(n) = b_{10}(n) = 1$ and

$$b_{01}(n) = \frac{1 - \gamma r^0_{n0}}{1 - \gamma r^1_{n0}},$$

$$b_{11}(n) = \frac{1 - \gamma r^1_{n1} - \gamma r^0_{n0} + \gamma^2 r^1_{n1} r^0_{n0} - \gamma^2 q^1_{n1} q^0_{n0}}{1 - \gamma r^1_{n1} - \gamma r^1_{n0} + \gamma^2 r^1_{n1} r^1_{n0} - \gamma^2 q^1_{n1} q^1_{n0}}.$$

Lemma 5. *Let $\pi_\lambda(z) = i$ and $\pi_\lambda(e) = j$, then for $\lambda \notin \Lambda^*$,*

$$\frac{\partial B_\lambda(z)}{\partial \lambda} = b_{ij}(n)\left[\alpha_j(n) + \beta_j(n)\frac{\partial V_\lambda(z')}{\partial \lambda}\right].$$

Proof. The result follows from considering the four cases $(i, j) \in \{(0,0), (0,1), (1,0), (1,1)\}$ separately and simplifying

$$\frac{\partial B_\lambda(z)}{\partial \lambda} = \frac{\partial Q_\lambda(z,1)}{\partial \lambda} - \frac{\partial Q_\lambda(z,0)}{\partial \lambda}. \tag{5.17}$$

Example Case: $(i, j) = (0, 1)$:

Since $\pi_\lambda(z) = 0$, we have $V_\lambda(z) = Q_\lambda(z,0)$. Therefore using (5.15), we get

$$Q_\lambda(z,0) = \frac{C(z,0) + \gamma q^0_{n0} V_\lambda(e) + \gamma p^0_{n0} V_\lambda(z')}{1 - \gamma r^0_{n0}}, \tag{5.18}$$

$$Q_\lambda(z,1) = C(z,1) + \lambda + \gamma q^1_{n0} V_\lambda(e) + \gamma p^1_{n0} V_\lambda(z') + \gamma r^1_{n0} Q_\lambda(z,0). \tag{5.19}$$

Since $\pi_\lambda(e) = 1$, we have $V_\lambda(e) = Q_\lambda(e,1)$. From (5.16), we get

$$V_\lambda(e) = \frac{C(e,1) + \lambda + \gamma q^1_{n1} V_\lambda(z) + \gamma p^1_{n1} V_\lambda(z')}{1 - \gamma r^1_{n1}}. \tag{5.20}$$

Differentiating w.r.t. λ, we get

$$\begin{aligned}
\frac{\partial Q_\lambda(z,0)}{\partial \lambda} &= \frac{1}{1-\gamma r^0_{n0}} \left(\gamma q^0_{n0} \frac{\partial V_\lambda(e)}{\partial \lambda} + \gamma p^0_{n0} \frac{\partial V_\lambda(z')}{\partial \lambda} \right), \\
\frac{\partial Q_\lambda(z,1)}{\partial \lambda} &= 1 + \gamma q^1_{n0} \frac{\partial V_\lambda(e)}{\partial \lambda} + \gamma p^1_{n0} \frac{\partial V_\lambda(z')}{\partial \lambda} + \gamma r^1_{n0} \frac{\partial Q_\lambda(z,0)}{\partial \lambda}, \\
\frac{\partial V_\lambda(e)}{\partial \lambda} &= \frac{1}{1-\gamma r^1_{n1}} \left(1 + \gamma q^1_{n1} \frac{\partial Q_\lambda(z,0)}{\partial \lambda} + \gamma p^1_{n1} \frac{\partial V_\lambda(z')}{\partial \lambda} \right).
\end{aligned} \tag{5.21}$$

Therefore, $\dfrac{\partial Q_\lambda(z,0)}{\partial \lambda}$ can be written as:

$$\begin{aligned}
\frac{\partial Q_\lambda(z,0)}{\partial \lambda} &= \frac{1}{1-\gamma r^0_{n0}} \left(\frac{\gamma q^0_{n0}}{1-\gamma r^1_{n1}} \left(1 + \gamma q^1_{n1} \frac{\partial Q_\lambda(z,0)}{\partial \lambda} + \gamma p^1_{n1} \frac{\partial V_\lambda(z')}{\partial \lambda} \right) + \gamma p^0_{n0} \frac{\partial V_\lambda(z')}{\partial \lambda} \right), \\
&= \frac{\gamma q^0_{n0}}{(1-\gamma r^0_{n0})(1-\gamma r^1_{n1})} \\
&\qquad + \frac{\gamma^2 q^0_{n0} q^1_{n1}}{(1-\gamma r^0_{n0})(1-\gamma r^1_{n1})} \frac{\partial Q_\lambda(z,0)}{\partial \lambda} + \frac{\gamma^2 q^0_{n0} p^1_{n1} + \gamma p^0_{n0}(1-\gamma r^1_{n1})}{(1-\gamma r^0_{n0})(1-\gamma r^1_{n1})} \frac{\partial V_\lambda(z')}{\partial \lambda}, \\
\frac{\partial Q_\lambda(z,0)}{\partial \lambda} &= \frac{1}{(1-\gamma r^0_{n0})(1-\gamma r^1_{n1})} \left(\gamma q^0_{n0} + \gamma^2 q^0_{n0} q^1_{n1} \frac{\partial Q_\lambda(z,0)}{\partial \lambda} + \left(\gamma^2 q^0_{n0} p^1_{n1} + \gamma p^0_{n0}(1-\gamma r^1_{n1}) \right) \frac{\partial V_\lambda(z')}{\partial \lambda} \right), \\
\frac{\partial Q_\lambda(z,0)}{\partial \lambda} &= \frac{1}{(1-\gamma r^0_{n0})(1-\gamma r^1_{n1}) - \gamma^2 q^0_{n0} q^1_{n1}} \left(\gamma q^0_{n0} + \left(\gamma^2 q^0_{n0} p^1_{n1} + \gamma p^0_{n0}(1-\gamma r^1_{n1}) \right) \frac{\partial V_\lambda(z')}{\partial \lambda} \right).
\end{aligned} \tag{5.22}$$

Substituting the above equations in (5.17), we get

$$\begin{aligned}\frac{\partial B_\lambda(z)}{\partial \lambda} &= \frac{\partial Q_\lambda(z,1)}{\partial \lambda} - \frac{\partial Q_\lambda(z,0)}{\partial \lambda} \\ &= 1 + \gamma q^1_{n0}\frac{\partial V_\lambda(e)}{\partial \lambda} + \gamma p^1_{n0}\frac{\partial V_\lambda(z')}{\partial \lambda} - (1-\gamma r^1_{n0})\frac{\partial Q_\lambda(z,0)}{\partial \lambda} \\ &= 1 + \Big(\gamma q^1_{n0} - \frac{1-\gamma r^1_{n0}}{1-\gamma r^0_{n0}}\gamma q^0_{n0}\Big)\frac{\partial V_\lambda(e)}{\partial \lambda} + \Big(\gamma p^1_{n0} - \frac{1-\gamma r^1_{n0}}{1-\gamma r^0_{n0}}\gamma p^0_{n0}\Big)\frac{\partial V_\lambda(z')}{\partial \lambda}.\end{aligned}$$

Substituting the value of $\frac{\partial V_\lambda(e)}{\partial \lambda}$ using equations (5.21) and (5.22), we get

$$\frac{\partial B_\lambda(z)}{\partial \lambda} = b_{01}\Big(\alpha_1(n) + \beta_1(n)\frac{\partial V_\lambda(z')}{\partial \lambda}\Big).$$

Results for the remaining cases: $(i,j) \in \{(0,0),(1,0),(1,1)\}$ can be obtained in a similar way as above. □

Lemma 6. *For all $\lambda \notin \Lambda^*$, $\partial B_\lambda(e)/\partial\lambda \geq 0$.*

Proof. We consider two cases:

1) Case I: $\pi_\lambda(e) = 0$:
From (5.16), we have

$$Q_\lambda(e,0) = V_\lambda(e) = \frac{C(e,0)}{1-\gamma},$$

which is independent of λ. Therefore, we get

$$\frac{\partial B_\lambda(e)}{\partial \lambda} = \frac{\partial Q_\lambda(e,1)}{\partial \lambda} = 1 + \gamma\ q^1_{n1}\ \frac{\partial V_\lambda(z)}{\partial \lambda} + \gamma\ p^1_{n1}\ \frac{\partial V_\lambda(z')}{\partial \lambda}.$$

From Lemma 4, we get that $\dfrac{\partial V_\lambda(z)}{\partial \lambda} \geq 0$ and $\dfrac{\partial V_\lambda(z')}{\partial \lambda} \geq 0$. This gives us $\partial B_\lambda/\partial\lambda \geq 0$.

2) Case II: $\pi_\lambda(e) = 1$:
As a result, we have

$$Q_\lambda(e,0) = C(e,0) + \gamma\ V_\lambda(e).$$

Therefore, using (5.17), we get

$$\frac{\partial B_\lambda(e)}{\partial \lambda} = (1-\gamma)\frac{\partial V_\lambda(e)}{\partial \lambda}.$$

From Lemma 4, we get that $\partial B_\lambda / \partial \lambda \geq 0$. □

5.7.1 Proof of Theorem 2

Lemma 6 shows that the Benefit function $B_\lambda(x)$ is always monotonically increasing for all fault states.

Since $q^1_{n1} \leq 1 - r^1_{n1}$ and $q^0_{n0} \leq 1 - r^0_{n0}$, we can write

$$\gamma^2 q^1_{n1} q^0_{n0} \leq (\gamma - \gamma r^1_{n1})(\gamma - \gamma r^0_{n0}). \tag{5.23}$$

Moreover, since $\gamma \in (0,1)$, we have

$$(\gamma - \gamma r^1_{n1})(\gamma - \gamma r^0_{n0}) < (1 - \gamma r^1_{n1})(1 - \gamma r^0_{n0}). \tag{5.24}$$

From (5.23) and (5.24), we can write

$$1 - \gamma r^1_{n1} - \gamma r^0_{n0} + \gamma^2 r^1_{n1} r^0_{n0} - \gamma^2 q^1_{n1} q^0_{n0} > 0.$$

Similarly,

$$1 - \gamma r^1_{n1} - \gamma r^1_{n0} + \gamma^2 r^1_{n1} r^1_{n0} - \gamma^2 q^1_{n1} q^1_{n0} > 0.$$

This implies that $b_{11} > 0$.

Lemma 4 gives us bounds on the derivative of the value function w.r.t. λ. These bounds are then used with results of Lemma 5 to show that the function $B_\lambda(x)$ is monotonically increasing for all $x \in \mathcal{X}$ if for all $n \in \{1, 2, \ldots, N\}$ and $j \in \{0, 1\}$ [2]:

$$\alpha_j(n) \geq 0 \text{ and } \alpha_j(n) + \beta_j(n)\frac{1}{1-\gamma} \geq 0.$$

We observe that $\alpha_0(n) = 1 \geq 0$. Also, $\alpha_1(n)$ and $\beta_1(n)$ can be simplified as:

[2]To obtain this result, we make use of Lemma 4, which establishes bounds on the derivative of the value function. Since transition probabilities at each state are independent, monotonicity of the Benefit function B_λ is guaranteed if $\alpha_j(n) + \beta_j(n)\partial V(n+1)/\partial\lambda$ is non-negative for both lower and upper bounds of $\partial V(n+1)/\partial\lambda$.

$$\alpha_1(n) = 1 + \frac{\gamma q^1_{n0}}{1-\gamma\, r^1_{n1}} + \frac{\gamma q^0_{n0}\left(\gamma\, r^1_{n0} + \dfrac{\gamma^2 q^1_{n0} q^1_{n1}}{1-\gamma\, r^1_{n1}} - 1\right)}{1-\gamma\, r^1_{n1} - \gamma\, r^0_{n0} + \gamma^2 r^1_{n1} r^0_{n0} - \gamma^2 q^0_{n0} q^1_{n1}}$$

$$= \frac{1-\gamma\, q^0_{n0} + \gamma\, q^1_{n0} - \gamma\, r^0_{n0} - \gamma\, r^1_{n1} - \gamma^2\, q^0_{n0}\, q^1_{n1} + \gamma^2\, q^0_{n0}\, r^1_{n0} - \gamma^2\, q^1_{n0}\, r^0_{n0} + \gamma^2\, r^0_{n0}\, r^1_{n1}}{1-\gamma\, r^0_{n0} - \gamma\, r^1_{n1} + \gamma^2\, r^0_{n0}\, r^1_{n1} - \gamma^2\, q^0_{n0}\, q^1_{n1}},$$

$$\beta_1(n) = \gamma p^1_{n0} + \frac{\gamma^2 q^1_{n0} p^1_{n0}}{1-\gamma r^1_{n1}} + \frac{(\gamma p^0_{n0} - \gamma^2 p^0_{n0} r^1_{n1} + \gamma^2 q^0_{n0} p^1_{n0})\left(\gamma r^1_{n0} + \dfrac{\gamma^2 q^1_{n0} q^1_{n1}}{1-\gamma r^1_{n1}} - 1\right)}{1-\gamma r^1_{n1} - \gamma r^0_{n0} + \gamma^2 r^1_{n1} r^0_{n0} - \gamma^2 q^0_{n0} p^1_{n1}}$$

$$= \frac{\gamma\,(1-\gamma)\begin{pmatrix} q^0_{n0} - q^1_{n0} + r^0_{n0} - r^1_{n0} + \gamma\, q^0_{n0}\, q^1_{n1} - \gamma\, q^1_{n0}\, q^1_{n1} \\ -\gamma\, q^0_{n0}\, r^1_{n0} + \gamma\, q^1_{n0}\, r^0_{n0} - \gamma\, r^0_{n0}\, r^1_{n1} + \gamma\, r^1_{n0}\, r^1_{n1} \end{pmatrix}}{1-\gamma\, r^0_{n0} - \gamma\, r^1_{n1} + \gamma^2\, r^0_{n0}\, r^1_{n1} - \gamma^2\, q^0_{n0}\, q^1_{n1}}$$

Therefore,

$$\alpha_1(n) + \frac{\beta_1(n)}{1-\gamma} = \frac{1-\gamma r^1_{n0} - \gamma r^1_{n1} + \gamma^2 r^1_{n0} r^1_{n1} - \gamma^2 q^1_{n0} q^1_{n1}}{1-\gamma r^0_{n0} - \gamma r^1_{n1} + \gamma^2 r^0_{n0} r^1_{n1} - \gamma^2 q^0_{n0} q^1_{n1}} = \frac{1}{b_{11}}.$$

Since, $b_{11}(n) > 0$, we get $\alpha_1(n) + \beta_1(n)/(1-\gamma) > 0$.

Therefore, the single-robot problem is indexable if:

$$\alpha_1(n) \geq 0 \quad \text{and} \quad \frac{\beta_0(n)}{1-\gamma} \geq -1,\ \forall n \in \{1, \dots, N\}.$$

□

5.8 Chapter Summary

In this chapter, we provide an analysis of operator allocation problem for a multi-robot assistance task and demonstrate the effectiveness of Restless Bandit framework to obtain a scalable policy. This policy is based on Whittle index heuristic and performs close to the optimal and significantly better than other efficient solution approaches. We also provide an analysis of indexability of such problems and give a simplified condition to quickly verify if a problem instance is indexable. These results can also be used to specify required transition behavior in the form of bounds on transition probabilities.

There are, however, a few limitations of the proposed approach. When a problem instance is not indexable, the Whittle indices are not defined and the methods of computing these indices may not give meaningful values. Therefore, the proposed approach is not applicable in such cases. Also, note that the conditions for indexability identified in Theorem 2 are sufficient but not necessary. However, the assistance problem presented here is indexable in most instances. This was verified by randomly generating problem instances without any bounds and numerically verifying monotonicity of the passive set $\mathcal{P}(\lambda)$. Out of the 1000 random instances, $999, 992$ and 940 instances were found to be indexable for discount factors $\gamma = 0.9, 0.95, 0.99$ respectively. However, the sufficient conditions were satisfied for $700, 607$ and 452 instances for $\gamma = 0.9, 0.95, 0.99$. This suggests that model is, in general, indexable and the Whittle index heuristic is applicable. It also suggests that the system may benefit from improved sufficient conditions for indexability. It is worth noting that this analysis can provide us with class of transitions for which there is no need to check the conditions. For instance, in case of transition type-1, the sufficient conditions are satisfied for all values of transitions probabilities and thus the requirements for indexability are always met.

There are several research directions for future work that can lead to a richer modelling of such operator allocation problems. Incorporating the notion of uncertain transition probabilities instead of assuming the knowledge of exact values will result in a more robust model. The probabilities can be estimated using recorded data [37, 38] or, by modelling performance parameters such as operators' expertise [222]. Shifting the system definition from discrete to continuous space (or just adding the time dimension to the actions) can represent a more practical scenario. Overall, this work presents a starting point to a wide variety of human-robot collaborative systems with multiple agents and provides a promising framework to solve large instances of such problems.

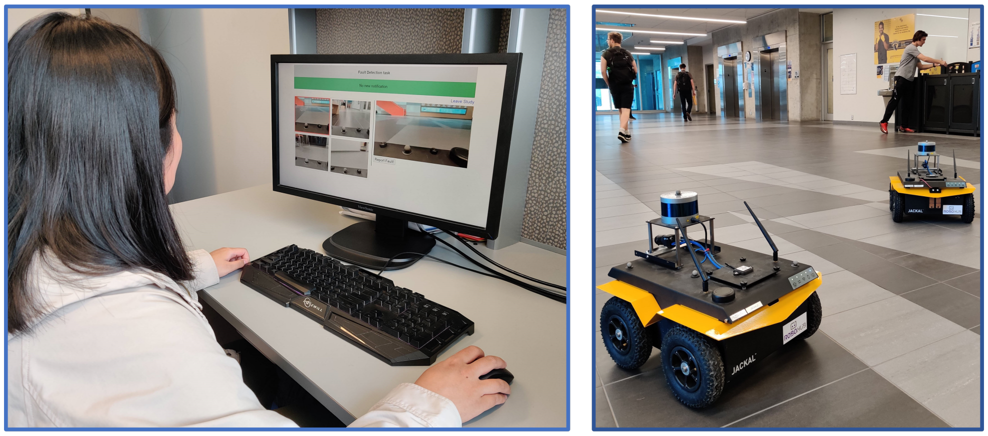
Chapter 6

Interruptions in Remote Supervision of Robot Fleets

TL;DR: In this chapter, we discuss the design of the user study and the development of the experiment interface used to evaluate the impact of interruptions in a multi-robot supervision system, alongside the resulting outcomes.

Many of the systems built for assisting humans in remote supervision of multiple robots are developed around the assumption that human operators will be solely working on the supervision task [11, 223]. However, in a common implementation of such systems, there are several different tasks that a supervisor can be working on (or switching between) at a given time. Their primary task is to monitor the robots looking for fault status in their operation. As a secondary task, they may be responsible for resolving faults when a robot's automatic correction procedure fails [135]. Additionally, in a practical scenario, a supervisor may need to work on tasks unrelated to active robot monitoring, such as coordinating with colleagues. However, even though these secondary tasks are parts of supervisor's job, they can act as interruptions as they take the supervisor's attention away from the primary task of monitoring the robots. This can be a problem, especially in the case of time-critical systems (e.g., robots navigating on a road network).

In the multi-robot supervision literature, it is well-established that controlling a large



(a) (b)

Figure 6.1: Multi-robot supervision user study setup: a) A participant monitoring multiple robots using the web-based interface, b) Clearpath Jackal robots used for the study.

number of robots negatively affects supervisor's attention and increases their workload [26, 223]. Researchers have used approaches like implementing different communication strategies [224], using task coordinators [194] or adjusting robot behaviors [193] to tackle this problem. Several studies from outside the robotics literature have compared the impact of interruptions on workload based on their relation to the primary task [225–227]. However, there is a gap in research in understanding the role of interruptions in multi-robot supervision systems, and the significance of differentiating interruptions based on their relation to the primary task of robot supervision.

In this chapter, we investigate the effects of two types of *interruptions* in a multi-robot supervision system. We consider a system where users primarily work on a monitoring task (reporting faults in robot behaviour) and intermittently face either of the two types of interruptions: 1) Intrinsic: ones related to the primary task, and 2) Extrinsic: ones unrelated to the primary task. Given a primary task, we define intrinsic interruptions as the ones that are closely related to the primary task. In a robot supervision task, intrinsic interruptions can include teleoperating the robot or resolving robot failures. We define extrinsic interruptions as those where the supervisor works on something completely unrelated to the primary task environment. This can either be another part of their job or simply an unexpected distraction.

We investigate the effects of the two types of interruptions using a user study with a simulated robot supervision task (Fig. 6.1). The main findings of this work are as follows.

We found the number of robots monitored by the participants to be a good predictor of their performance, both in terms of percentage fault reported ($F = 20.23, p < 0.001$) and average response time ($F = 852, p < 0.001$). Even though the interruptions do not significantly affect performance, they do result in an increase in participants' perceived workload on most of the NASA-TLX scales. Pairwise comparison of different test conditions reveal a significant increase in participants' workload, with extrinsic interruptions resulting in higher workload than intrinsic ones.

The rest of the chapter is organized as follows: In Section 6.1, we present some of the existing work on interruptions in workplace, and in multi-robot supervision tasks. In Section 6.2, we describe different elements of our study design and in Section 6.3, the results are presented. The chapter ends with Section 6.4 discussing the findings and implications.

6.1 Related Work

In this section, we discuss existing work on interruptions in workplace and their role in human–multi-robot systems, as well as the relevance of intrinsic and extrinsic interruptions.

6.1.1 Human Supervision of Multiple Robots

Even though robotic systems are rapidly increasing their autonomous capabilities, human supervision is still considered necessary to ensure that task goals are met during unanticipated events [16, 228]. There exists motivation to decrease the number of supervisors required in large multi-robot systems [229], but doing so negatively impacts supervisors' workload and performance [223, 230, 231].

The existing research primarily attempts to address this problem from the robotics side of the system, for example, by implementing different communication strategies [224], using task coordinators [194], adjusting robot behaviors [193] or using frameworks such as sliding autonomy [201, 202]. However, human factors also play an important role in governing system performance, and it is crucial to design the system in a way that results in effective human operation.

Interruptions are one of the common issues that can disrupt the ability of operators to maintain attention on a given task. Studies have shown that interruptions during a task can negatively impact user workload and increase error rates [232, 233]. However, the role and impact of interruptions in human–multi-robot systems has not been studied adequately in the literature. This is especially true for remote multi-robot supervision tasks where we naturally have different kinds of interruptions based on their relation to the monitoring task.

6.1.2 Interruptions in the Workplace

Studying interruptions faced by humans is an important area of research in many applications as they can negatively affect the performance and workload of workers [232, 233]. Interruptions can be simply defined as unanticipated disruption in one's primary task and diversion of attention to a related or unrelated secondary task [234]. Interruptions can originate *externally*, from the environment (noise, notifications or other external factors), or they can arise *internally*, from within the human (due to, for example, boredom or non-task-related thoughts). They can also be characterized based on their timing, relevance and attentional requirement [235–237].

In the literature on workplace interruptions (mostly in healthcare applications), researchers have studied different types of interruptions based on their relation and relevance to the primary task that workers are performing [225, 226]. Researchers have used the term *extraneous interruptions* to describe those that do not directly pertain to the primary task, and such interruptions are found to be one of the most common types faced by workers [227].

6.1.3 Interruptions in Multi-Robot Supervision

When looking at the multi-robot supervision literature, we notice a gap in the research on how different types of interruptions affect system performance and user workload. With this work, we aim to bridge this gap by differentiating different secondary tasks that supervisors need to perform in a multi-robot system into intrinsic and extrinsic interruptions, as ones related and unrelated to the primary task respectively.

This distinction also bears similarities with notion of different types of cognitive loads studied under the Cognitive Load Theory (CLT). The theory distinguishes between three different types of cognitive load: Intrinsic (load from the task itself), Extrinsic/Extraneous (load not related to the task but induced by its design), and Germane (load from learner's deliberate use of cognitive strategies) [238, 239]. This distinction provides further motivation to explore significance of differentiating interruptions based on their relationship with the primary task.

6.2 Methodology

In this section, we provide details of our user study, including the application designed for robot monitoring, different tasks that participants encounter, and our study hypotheses. The study adopts a mixed factorial design in which the type of interruption (intrinsic or extrinsic) is the *within-participant* factor, while the number of robots to be monitored (4 or 9) is the *between-participant* factor. These numbers are selected based on a pilot study, which ensures that the difficulty level of the monitoring task ranges from easy to moderately difficult, and aligns with the existing understanding of human psychological attention limit. Moreover, similar numbers have been used in prior studies on human-multi-robot systems across various applications [195]. We recruited 39 participants in total distributed evenly between 4 and 9 robots cases. The participants of the study consisted of university students and individuals who were recruited via personal networks. None of the participants had prior experience using a robot monitoring interface. The study has been reviewed and received ethics clearance through a University of Waterloo Research Ethics Committee (ORE#43628).

6.2.1 Study Design

A web-based application is designed to conduct the study. The application replicates a basic setup of a remote-monitoring interface, with camera feed from multiple robots in one half of the screen, and an enlarged view of a single robot in the other half. Participants can select any one robot for an enlarged view for detailed inspection and for reporting faults in that robot. The application also displays notifications for any interruption that may arrive, which in our case are the prompts for secondary tasks based on the test condition.

Each robot's *camera feed* shows a pre-recorded video from a camera mounted on the robot, navigating in an indoor building environment with light foot traffic (see Fig. 6.1(b)). The robot navigation was intentionally corrupted to include faults, which were designed to appear as one of three behaviors in robots' movements: 1) Stops moving, 2) Moving in circles at a spot, and 3) Turning side-to-side without moving forward. The faults were randomly introduced during robot navigation with their start and end times determined randomly as well. Each fault lasted for at least 20 seconds and there was at least 30 seconds between two faults. In the videos used for the study, the robots experienced faults between 1 and 5 times, with an average fault duration of 30 seconds. A robot was in a fault state for about 29.5% of the total duration.

At any given time during the experiment, a participant can be working on one of the

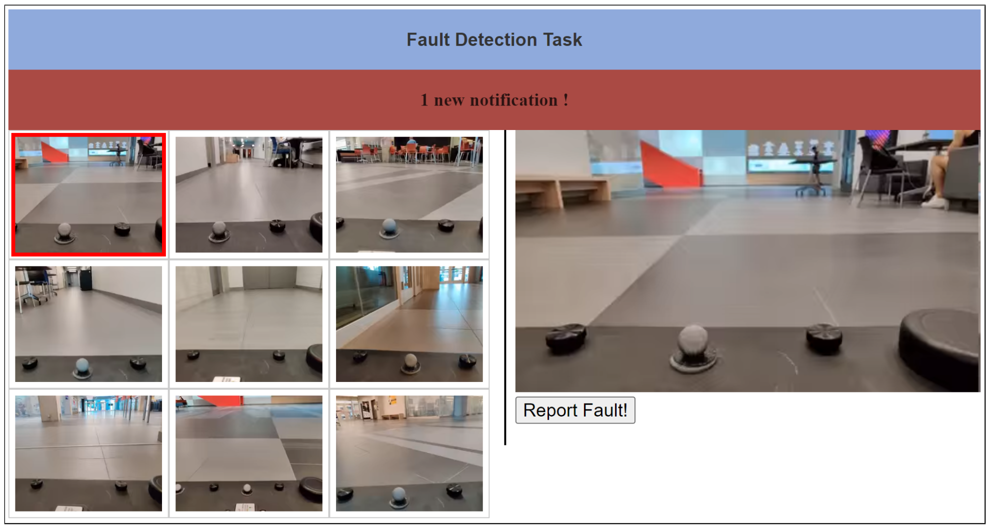


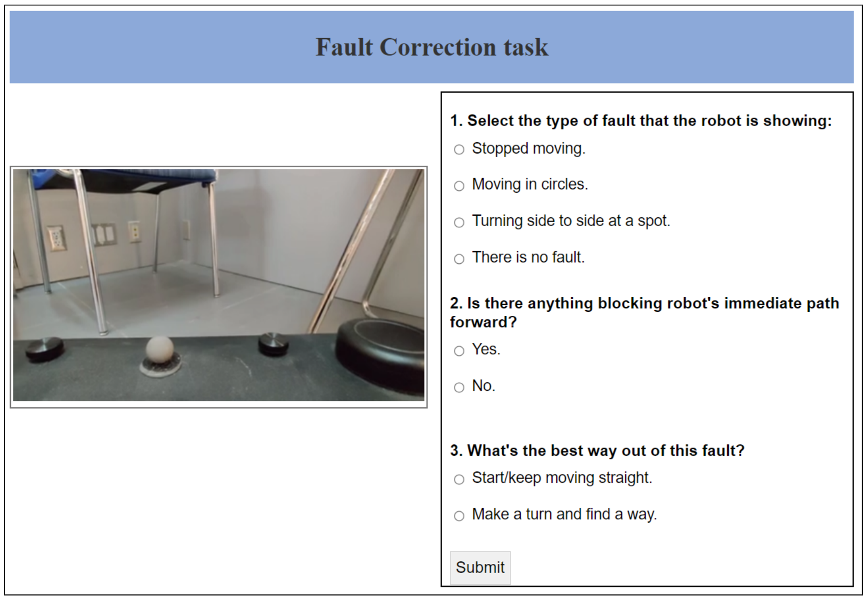
Figure 6.2: Interface for robot monitoring task. The grid of robot cards on the left side shows video information from all robots. On the right, there is an enlarged visual of the selected robot. On the top, there is a notification panel for incoming secondary tasks.

following three tasks:

1) Robot monitoring: This is the primary (default) task during the experiment, which requires participants to monitor all the robots shown in the interface (Fig. 6.2), and detect if any of the robots is in a fault state. Once a fault is detected, participants need to select that robot and press the 'Report Fault' button. This action is programmed to *fix* the fault, and its camera feed is refreshed to show the robot navigating normally again. Participants are required to report all the faults that appear during the experiment and as soon as possible from their appearance.

2) Fault Correction: During this secondary task, participants are shown a video feed of a potentially faulty robot and are required to answer questions about it (see Fig. 6.3). This acts as an intrinsic interruption closely related to the primary monitoring task while not requiring any technical knowledge about the robot operation. Once all questions are answered correctly, the participant is taken back to the monitoring task. For the study, videos for this task are randomly selected from a pool of 15 videos each showing a different type of fault (or no fault at all).

3) Messaging Task: This secondary task represents the extrinsic interruption during which participants are required to write a message to their colleagues. The message is already displayed on the screen and participants need to type it again in the space provided


Fault Correction task

1. Select the type of fault that the robot is showing:
- Stopped moving.
- Moving in circles.
- Turning side to side at a spot.
- There is no fault.

2. Is there anything blocking robot's immediate path forward?
- Yes.
- No.

3. What's the best way out of this fault?
- Start/keep moving straight.
- Make a turn and find a way.

Figure 6.3: Interface for fault correction task. On the left, participants see a robot potentially in a fault state. On the right, there are several questions to characterize the fault.

(see Fig. 6.4). Once the message is typed, the participant can press the 'Send message' button and is then taken back to the monitoring task. For the study, messages for this task are randomly selected from a pool of 15 messages, each with 85 or fewer characters.

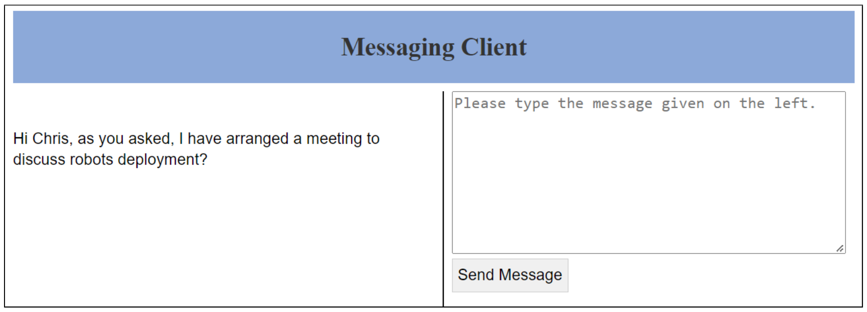
6.2.2 Procedure

Each participant first goes through a training session and then completes the experiment under three different test conditions, which decide the type of interruption they will be facing. The order of these three conditions is counterbalanced. Therefore participants see these conditions in a different order, with a minute-long break between conditions. The three conditions are described below.

1) Condition-0 (No interruption): In this condition, no interruption occurs and participants work on the robot monitoring task for the whole duration.

2) Condition-1 (Intrinsic interruption): In this condition, participants are shown a notification after they spend certain amount of time on the monitoring task. In this condition, clicking a notification takes the participant to the fault correction task screen.

3) Condition-2 (Extrinsic interruption): Under this condition, the experiment



Hi Chris, as you asked, I have arranged a meeting to discuss robots deployment?

Figure 6.4: Interface for the messaging task. On the left, participants see a pre-written message to be sent. On the right is a text box to type the message and the send button.

is conducted in a similar way as Condition-1, except that clicking notifications takes the participant to the messaging task screen.

Under each condition, participants work on the monitoring task for 2 minutes, during which they may receive notifications for interruptions in the form of secondary tasks. These notifications are randomly presented after the participant spends between 15 and 40 seconds on the monitoring task, with the exact time sampled from a uniform distribution. These durations were selected based on pilot testing to ensure a balance between time spent on the primary and secondary tasks.

6.2.3 Metrics

After a participant is finished with the assigned tasks, they are asked to fill out the NASA-TLX questionnaire [240] (once after each test condition). Once they complete the task under all conditions, they fill a post-experiment questionnaire and the procedure is finished. Additionally, the application also records participant's performance parameters, such as faults reported and response time of identifying faults. If a robot gets into fault when a participant is working on an interruption, the response time only starts to count when they resume the monitoring task. This allows us to avoid counting the time a participant spends on an interruption task towards their response time.

6.2.4 Hypotheses

This study seeks to learn how the intrinsic and extrinsic interruptions can affect supervisor's performance and workload in a multi-robot remote supervision system. However, NASA-TLX scores are highly influenced by individual differences especially when using the unweighted scores[1] [240]. To eliminate the effects of individual differences, we analyze change in scores of participants across test conditions.

For this study, we propose the following null hypotheses:
***H_0*-1:** Task performance does not differ across test conditions (type of interruptions shown).
***H_0*-2:** Perceived workload does not differ across test conditions.

6.3 Results

We analyze the experiment data under three categories: users' performance, perceived workload, and responses to post-experiment questionnaire.

6.3.1 Results on Performance

For the presented task, we use the following metrics as a measure of performance: First is the percentage of faults reported, calculated as the ratio of faults reported to total faults appeared during a task. Second is the response time, calculated as the average time it took a user to report a fault (time from appearance of a fault to its reporting[2]).

Figure 6.5 shows the percentage of faults reported by participants under each test condition. From the graph, we observe that when monitoring 4 robots, most of the participants were able to detect the majority of the faults ($> 70\%$) under all three test conditions. As the number of robots increases to 9, the percentage of faults reported decreases under all three test conditions. A 2-Way ANOVA confirms that number of robots is a very strong predictor of percent fault reported ($F = 20.23, p < 0.001$). This is expected as participants are required to keep attention over a larger stream of information. The conditions themselves do not show any significant main effect on the outcome ($F = 0.21, p = 0.81$).

[1]We do not use category weights for two reasons: First, it is unclear how using category weights affect sensitivity of the score across different systems [240]. Second, administering ranking questions for the weighting step requires a fair amount of effort from the participants which, in our case, is comparable to the effort required for the task itself.

[2]If a participant fails to report a fault, the fault duration is considered to be the response time for that fault.

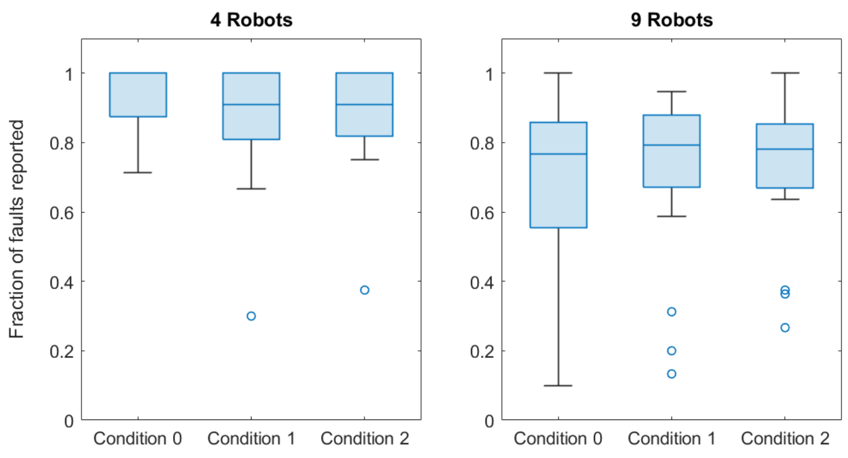


Figure 6.5: Percentage Fault Reported under different test conditions for participants monitoring 4 robots (left) and 9 robots (right).

Figure 6.6 shows the average amount of time a participant took to report faults under different conditions. These results are similar to the case of percent fault reported, where number of robots are a strong predictor of outcome ($F = 852, p < 0.001$) while interruption type does not have a significant effect ($F = 0.05, p = 0.95$). This may be partially because the response time only starts to count when participants resume the monitoring task. This allows us to avoid counting the time a participant spends on an interruption task towards their response time.

Given these results, **the null hypothesis H_0-1 cannot be rejected**: task performance does not differ significantly with test conditions (type of interruptions shown).

6.3.2 Results on Workload

Figure 6.7 shows the participants' perceived workload during different test conditions, measured as NASA-TLX scores.

From the figure, we observe that participants reported higher workload, on average, under Condition-2 (extrinsic interruptions) on most of the workload categories followed by Condition-1 and then Condition-0, regardless of the number of robots they monitored.

Since we are using unweighted scores, it is more relevant for the study to compare change in scores between test conditions for individual participants rather than taking the average. Therefore, we present pairwise comparisons of participants' scores for each workload category.

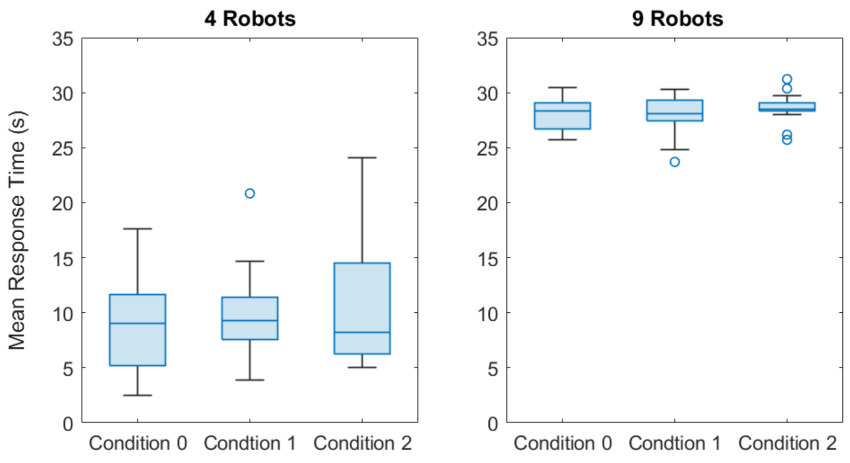


Figure 6.6: Average response time (time from fault appearance to fault reporting) for all users under different conditions.

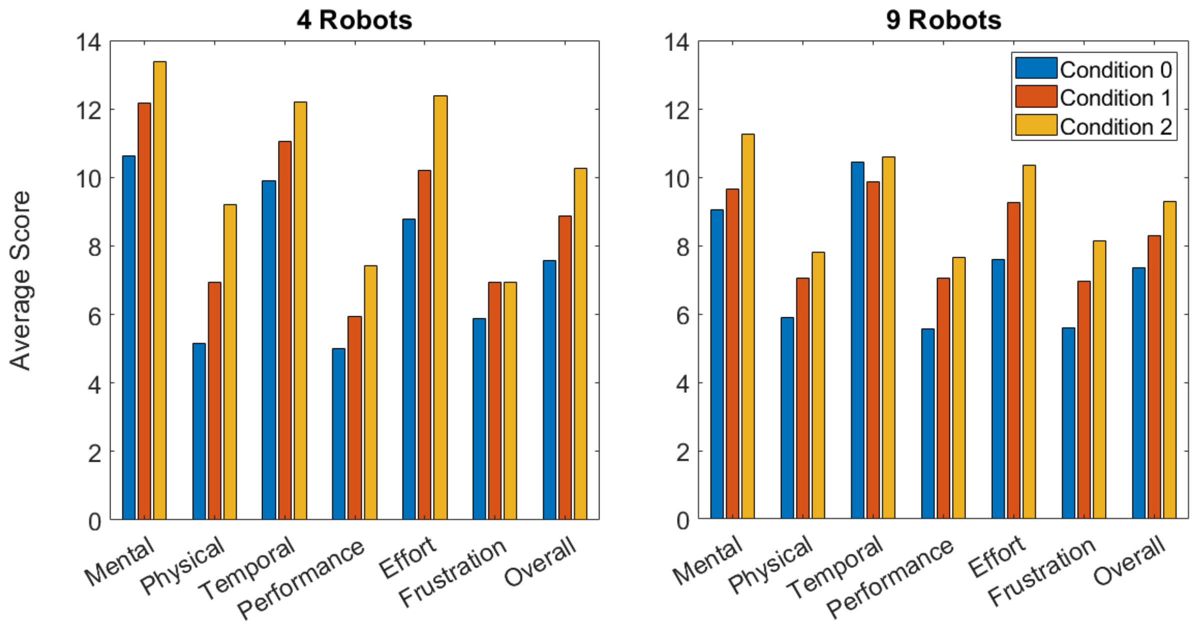


Figure 6.7: Average Rating for TLX Questions for different conditions for 4 and 9 robots.

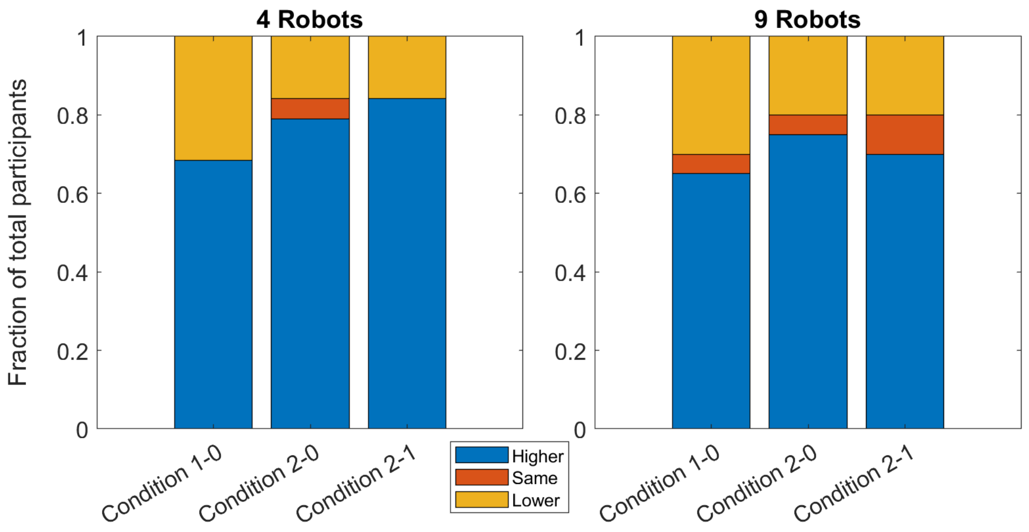


Figure 6.8: Fraction of participants who reported higher/lower/same scores between two conditions. For example, for pair 1-0, blue represents that participants' reported higher workload in condition-1, orange means scores were same for both conditions and yellow means a higher workload reported in condition-0.

Figure 6.8 shows how individual participants' workload scores changed between conditions. These are shown as the percentage of participants who reported higher, lower, or the same workload score. Comparisons are performed in going from Condition-0 to 1, Condition-0 to 2, and Condition-1 to 2. A one sample t-test reveals that differences between most of the test conditions are significantly different from a zero-mean distribution, except for the difference between condition-1 and condition-0 for the 9-robot case. Table 6.1 shows the result of these t-tests.

To provide a more comprehensive analysis, we also present the distribution of changes in workload scores for participants across all workload categories and pairwise comparisons in Figures 6.9 and 6.10. The results indicate that a majority of participants reported higher workload scores under condition-2 compared to both condition-0 and condition-1.

These findings lead us to **reject the null hypothesis H_0-2**, i.e., perceived workload differs with test conditions (type of interruptions).

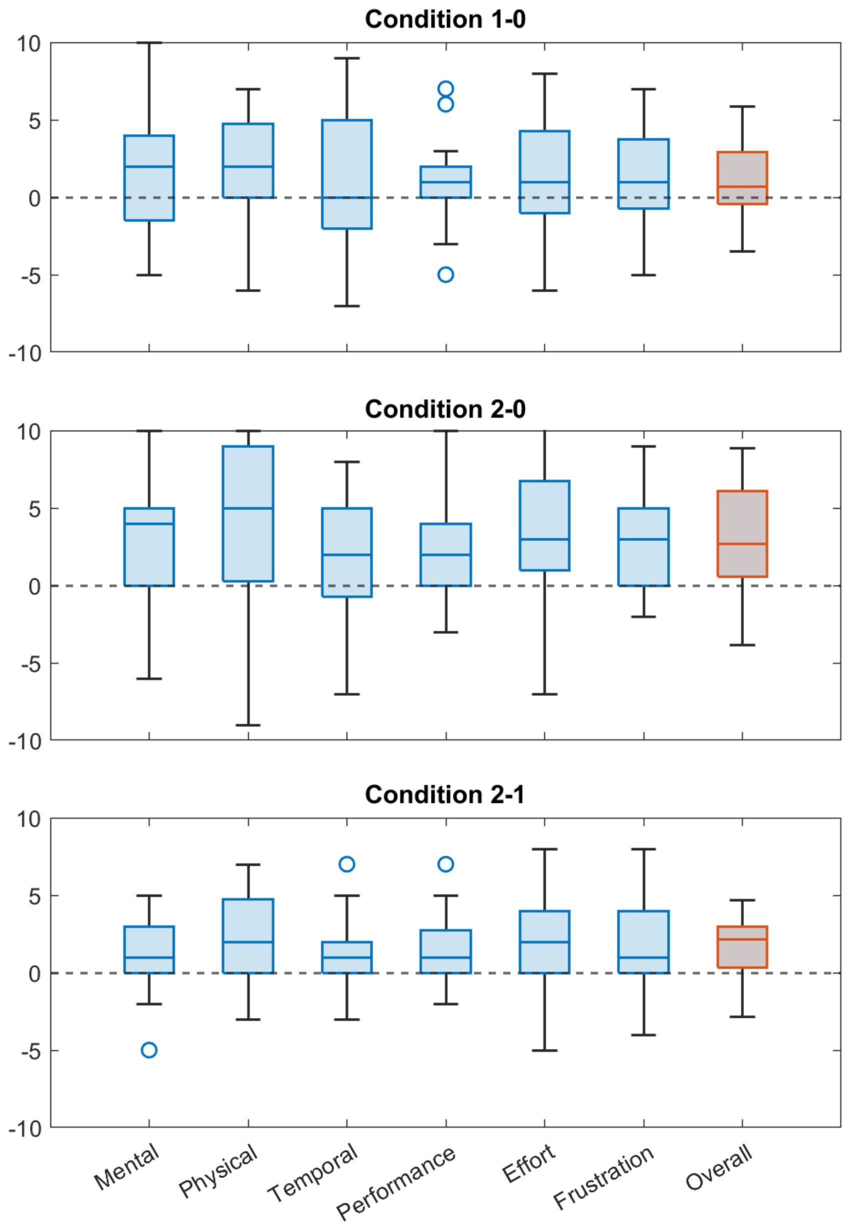


Figure 6.9: Difference in TLX scores between different conditions for 4 robots. For example, the Condition 1-0 plots show distribution of condition 1 minus condition 0 scores for individual participant.

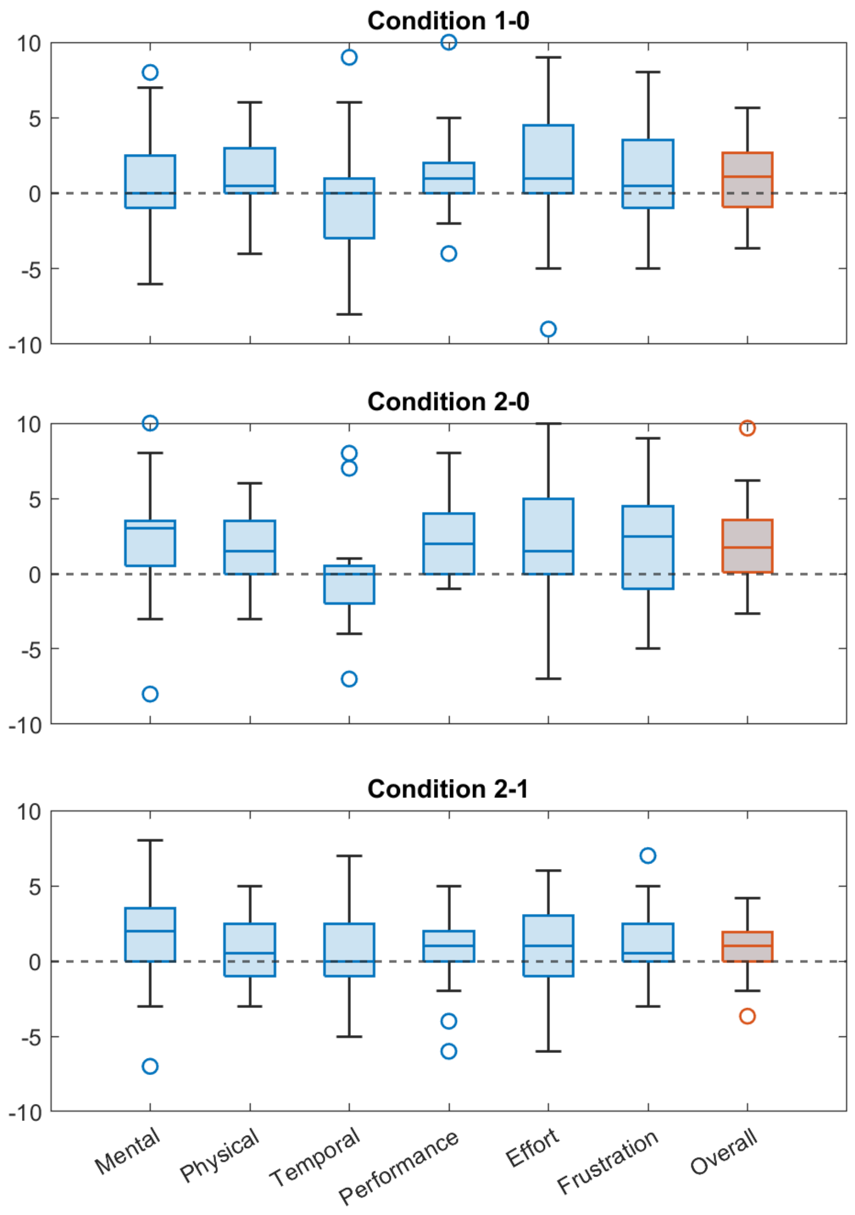


Figure 6.10: Difference in TLX scores between different conditions for 9 robots.

Table 6.1: One-sample t-test p-values for different pairwise comparison of TLX scores.

	Conditions 1-0	Conditions 2-0	Conditions 2-1
4 robots	0.0299	0.0009	0.0011
9 robots	0.1150	0.0109	0.0419
Overall	0.0071	0.0000	0.0001

Table 6.2: Prompts used in post-experiment questionnaire.

1	I found the fault correction tasks and messaging tasks disruptive while monitoring the robots.
2	I found it difficult to switch from the robot monitoring task to the correction task.
3	I found it difficult to switch from the robot monitoring task to the messaging task.
4	I found it difficult to resume the robot monitoring task after the correction task.
5	I found it difficult to resume the robot monitoring task after the messaging task.

6.3.3 Post-Experiment Questionnaire

Besides measuring performance and workload of the participants, we also asked them some further questions regarding their perception of different tasks they performed during the whole experiment. This post-experiment questionnaire is shown in Table 6.2, and participants' agreement with each statement was recorded on a 20-point scale (1 being Strongly Disagree and 20 being Strongly Agree).

From participants' responses, we observe that majority of participants found the interruption tasks disruptive while monitoring the robots, with higher average score among participants who monitored 9 robots (Question 1). Participants also reported more difficulty while switching from monitoring task to the messaging task compared to the fault correction task, with mean difference $\bar{\mu} = 2.79$ (Questions 2,3). A one sample t-test confirms the significance of difference ($p = 0.015$). We note similar results on perceived difficulty for resuming the monitoring task after an interruption (Questions 4,5) ($\bar{\mu} = 3.20$, $p = 0.001$).

6.4 Chapter Summary

The study presented in this chapter reveals some interesting features of multi-robot supervision systems where users monitor several independent mobile robots. From the results, we observe that while working on the robot monitoring task, any interruption, be it intrinsic (fault correction) or extrinsic (messaging), will result in an increase of user workload. The effects of extrinsic interruptions on workload are more severe than those of intrinsic ones for both four- and nine-robot cases.

However, the impact of these interruptions on the task performance is found to be insignificant. The number of robots being monitored is observed to be the major factor in a change of performance. One possible reason for this observation is that the monitoring task in our system requires a short working memory as faults in the system are determined solely based on the current state/short-term behaviour of the robots. Even though the type of interruption does not significantly affect performance, participants reported them to be disruptive while monitoring the robots. Participants also reported that switching to the extrinsic task and resuming the monitoring task afterward is more difficult compared to the intrinsic task.

These findings suggest that when designing such multi-robot supervision systems, it is important to prevent interruptions from extrinsic tasks while working on robot monitoring. It may be helpful to postpone such interruptions towards the end of the task. It may also be helpful to distribute the responsibility for robot monitoring and fault correction tasks among different operators to limit supervisors switching between the two tasks. In the future, we would like to expand this study to further characterize the effects of intrinsic and extrinsic interruptions by controlling the frequency of interruptions, changing the difficulty of the task, and having a larger number of robots to monitor. It is also interesting to explore how the results will change if the secondary tasks are introduced in a multi-tasking scenario instead of being separate interruption tasks, where users try to work on different tasks simultaneously.

Chapter 7

Conclusion and Future Work

The thesis has made significant contributions to addressing the challenges associated with supervised robot fleets, focusing on the planning of routes and missions, operator allocation, and human factors in these systems. Throughout the research, we have presented novel methodologies and algorithms, conducted simulations, and provided valuable insights into the dynamics of human-robot interaction. Collectively, these contributions enhance the efficiency, safety, and overall performance of supervised robot fleet systems, advancing the field of human-robot interaction and enabling effective collaboration between humans and robots in real-world applications. The Budget-A^* algorithm efficiently addresses collaborative robot planning with intermittency in the availability of human assistance, while the Restless Bandit framework provides an effective operator allocation policy. Additionally, the user study offers valuable insights for designing multi-robot supervision systems. These contributions advance the field of human-robot interaction, enabling effective collaboration between humans and robot fleets in various real-world applications. The outcomes of this thesis lay a foundation for further research and improvement in this evolving field.

7.1 Future Work

While this thesis has made contributions to various aspects of supervised robot fleet research, there are several opportunities for further improvement and directions for future work to extend the presented findings.

Optimizing costs and ensuring equitable allocation: The solutions presented in Chapter 4 aim to minimize path durations for the robots to reach their goals efficiently.

However, in certain applications, it may be more relevant to optimize different costs, such as the usage of operator assistance, waiting times, or specific routes. Including such costs in the planning process can provide finer control over the robot paths and assist in limiting the amount of assistance allocated to each robot. In a multi-robot setting, we may also be interested in making sure that the operator allocation is done in an equitable manner, distributing human assistance fairly among the robots.

Dynamic and unknown environments: In Chapter 4, we considered known and stationary travel durations for the robots. Similarly, in Chapter 5, we considered the state-transition probabilities to be known to the decision support system. To enhance the adaptability of our solutions, future research can explore dynamic and non-stationary environments where travel durations and operator availability may vary over time. Additionally, considering scenarios where task demands change and robots need to adapt their plans or replan for new service requests would enable more responsive fleet planning. Moreover, in Chapter 5, the state transition probabilities are defined such that the task durations are exponentially distributed random variables. While this assumption was appropriate for certain applications, it is worth investigating the impact of different distribution models on the performance of the proposed algorithm. Exploring the extension of our results to other distribution models for task durations could provide a deeper understanding of the robustness and applicability of our solution approach in a wider range of scenarios.

Risk-aware planning: Ensuring safety in robot fleet operations is crucial. Traditional approaches to safe planning often aim to mitigate the possibilities of encountering critical situations, such as entering hazardous zones or reaching states where further progress is uncertain. However, in robotic systems where human operators are available to provide assistance, the final layer of safety relies on the interaction between the robot and the operator, rather than solely on the critical situation itself. Therefore, to define safety requirements in this context, we can use the notion of *service loss*, capturing the times when a robot requires assistance but no operator is available. Under this notion, we can define different safety-related aspects such as the probability, duration, or cost of service loss associated with a given robot path and operator allocation. Additionally, it may be important to consider the robustness of planned routes, i.e., the ability to maintain or recover the original plan in the event of delays. By incorporating these considerations, we can design safer and more reliable routes for the robots in the fleet.

Improving study design: The research presented in Chapter 6 provides valuable insights into the effects of interruptions on robot supervision. However, there are various opportunities to further enhance the validity and applicability of the findings by improving the study design. One important aspect is the construction of secondary tasks, which were artificially created for the study. To better reflect real-world scenarios, using actual supervisors' tasks could provide more realistic and meaningful results. In addition, the duration of the study should be extended, as robot supervision often involves long periods of engagement with the system. While the short study allowed us to explore the effects of interruptions in a controlled environment, understanding how these interruptions impact performance over extended durations is essential.

Expanding scope of the user study: In the work presented in Chapter 6, the user study focused on a single modality of communication, the screen. Nevertheless, in applications such as search and rescue or subterranean surveillance, the incorporation of multi-modal communication could hold greater utility. Incorporating an exploration of interruption effects within these multi-modal communication systems could significantly broaden the scope of our work and yield valuable insights into different real-world contexts. Additionally, extending the scope of the research to investigate the effects of interruptions in different contexts and exploring strategies to minimize disruptions would be beneficial. For example, understanding how interruptions impact performance in safety-critical environments or high-stress situations could provide essential guidance for designing effective human-robot interaction strategies. Moreover, developing interruption management techniques to optimize task completion under interrupted conditions could prove valuable in enhancing overall system efficiency and user satisfaction.

References

[1] S. Srinivas, S. Ramachandiran, and S. Rajendran, "Autonomous robot-driven deliveries: A review of recent developments and future directions," *Transportation research part E: logistics and transportation review*, vol. 165, p. 102834, 2022. 1

[2] D. S. Terracciano, L. Bazzarello, A. Caiti, R. Costanzi, and V. Manzari, "Marine robots for underwater surveillance," *Current Robotics Reports*, vol. 1, pp. 159–167, 2020. 1

[3] G. Ferri*, A. Munafò*, A. Tesei, P. Braca, F. Meyer, K. Pelekanakis, R. Petroccia, J. Alves, C. Strode, and K. LePage, "Cooperative robotic networks for underwater surveillance: an overview," *IET Radar, Sonar & Navigation*, vol. 11, no. 12, pp. 1740–1761, 2017. 1

[4] C. Iclodean, N. Cordos, and B. O. Varga, "Autonomous shuttle bus for public transportation: A review," *Energies*, vol. 13, no. 11, p. 2917, 2020. 1

[5] M. M. Rahman and J.-C. Thill, "Impacts of connected and autonomous vehicles on urban transportation and environment: A comprehensive review," *Sustainable Cities and Society*, p. 104649, 2023. 1

[6] J. Rosenzweig and M. Bartl, "A review and analysis of literature on autonomous driving," *E-Journal Making-of Innovation*, pp. 1–57, 2015. 1

[7] R. Hoque, L. Y. Chen, S. Sharma, K. Dharmarajan, B. Thananjeyan, P. Abbeel, and K. Goldberg, "Fleet-dagger: Interactive robot fleet learning with scalable human supervision," in *Conference on Robot Learning*. PMLR, 2023, pp. 368–380. 1

[8] G. Datta, R. Hoque, A. Gu, E. Solowjow, and K. Goldberg, "Iifl: Implicit interactive fleet learning from heterogeneous human supervisors," *arXiv preprint arXiv:2306.15228*, 2023. 1

[9] A. Kamboj, T. Ji, and K. Driggs-Campbell, "Examining audio communication mechanisms for supervising fleets of agricultural robots," in *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN).* IEEE, 2022, pp. 293–300. 1, 20

[10] A. Hong, O. Igharoro, Y. Liu, F. Niroui, G. Nejat, and B. Benhabib, "Investigating human-robot teams for learning-based semi-autonomous control in urban search and rescue environments," *Journal of Intelligent & Robotic Systems*, vol. 94, pp. 669–686, 2019. 1

[11] A. Dahiya, A. M. Aroyo, K. Dautenhahn, and S. L. Smith, "A survey of multi-agent human–robot interaction systems," *Robotics and Autonomous Systems*, vol. 161, p. 104335, 2023. 1, 3, 14, 37, 99

[12] T. B. Sheridan, "Human–robot interaction: status and challenges," *Human factors*, vol. 58, no. 4, pp. 525–532, 2016. 1, 7

[13] A. Khamis, A. Hussein, and A. Elmogy, "Multi-robot task allocation: A review of the state-of-the-art," *Cooperative robots and sensor networks 2015*, pp. 31–51, 2015. 3, 63

[14] P. Gupta and V. Srivastava, "Optimal fidelity selection for human-in-the-loop queues using semi-markov decision processes," in *2019 American Control Conference (ACC)*, 2019, pp. 5266–5271. 3, 40

[15] A. Dahiya and S. L. Smith, "Optimal robot path planning in a collaborative human-robot team with intermittent human availability," *arXiv preprint arXiv:2307.04674*, 2023. 4

[16] A. Dahiya, N. Akbarzadeh, A. Mahajan, and S. Smith, "Scalable operator allocation for multi-robot assistance: A restless bandit approach," *IEEE Transactions on Control of Network Systems*, pp. 1397–1408, 2022. 4, 8, 13, 14, 22, 23, 40, 62, 63, 101

[17] A. Dahiya, Y. Cai, O. Schneider, and S. L. Smith, "On the impact of interruptions during multi-robot supervision tasks," in *2023 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 2023, pp. 9771–9777. 5

[18] O. Gvirsman, Y. Koren, T. Norman, and G. Gordon, "Patricc: A platform for triadic interaction with changeable characters," in *Proceedings of the 2020 ACM/IEEE International Conference on Human-Robot Interaction*, 2020, pp. 399–407. 7

[19] M. C. Gombolay, R. A. Gutierrez, S. G. Clarke, G. F. Sturla, and J. A. Shah, "Decision-making authority, team efficiency and human worker satisfaction in mixed human–robot teams," *Autonomous Robots*, vol. 39, no. 3, pp. 293–312, 2015. 7, 14

[20] A. M. Abrams and A. M. Rosenthal-von der Pütten, "I–c–e framework: Concepts for group dynamics research in human-robot interaction," *International Journal of Social Robotics*, pp. 1–17, 2020. 7, 15

[21] M. Selvaggio, M. Cognetti, S. Nikolaidis, S. Ivaldi, and B. Siciliano, "Autonomy in physical human-robot interaction: A brief survey," *IEEE Robotics and Automation Letters*, 2021. 7

[22] C. Bartneck, T. Belpaeme, F. Eyssel, T. Kanda, M. Keijsers, and S. Šabanović, *Human-robot interaction: An introduction.* Cambridge University Press, 2020. 7, 13

[23] A. Thomaz, G. Hoffman, and M. Cakmak, "Computational human-robot interaction," *Foundations and Trends in Robotics*, vol. 4, no. 2-3, pp. 105–223, 2016. 7

[24] X. V. Wang, Z. Kemény, J. Váncza, and L. Wang, "Human–robot collaborative assembly in cyber-physical production: Classification framework and implementation," *CIRP annals*, vol. 66, no. 1, pp. 5–8, 2017. 7

[25] H. A. Yanco and J. Drury, "Classifying human-robot interaction: an updated taxonomy," in *2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)*, vol. 3. IEEE, 2004, pp. 2841–2846. 7, 14, 16

[26] M. Lewis, "Human interaction with multiple remote robots," *Reviews of Human Factors and Ergonomics*, vol. 9, no. 1, pp. 131–174, 2013. 7, 14, 23, 100

[27] A. Kolling, P. Walker, N. Chakraborty, K. Sycara, and M. Lewis, "Human interaction with robot swarms: A survey," *IEEE Transactions on Human-Machine Systems*, vol. 46, no. 1, pp. 9–26, 2015. 7, 13, 14, 19, 23, 25

[28] S. Sebo, B. Stoll, B. Scassellati, and M. F. Jung, "Robots in groups and teams: A literature review," *Proceedings of the ACM on Human-Computer Interaction*, vol. 4, no. CSCW2, pp. 1–36, 2020. 8, 13

[29] R. Oliveira, P. Arriaga, and A. Paiva, "Human-robot interaction in groups: Methodological and research practices," *Multimodal Technologies and Interaction*, vol. 5, no. 10, p. 59, 2021. 8

[30] S. K. K. Hari, A. Nayak, and S. Rathinam, "An approximation algorithm for a task allocation, sequencing and scheduling problem involving a human-robot team," *Robotics and Automation Letters*, vol. 5, no. 2, pp. 2146–2153, 2020. 8, 22, 40, 71, 73

[31] M. IJtsma, L. M. Ma, A. R. Pritchett, and K. M. Feigh, "Computational methodology for the allocation of work and interaction in human-robot teams," *Journal of Cognitive Engineering and Decision Making*, vol. 13, no. 4, pp. 221–241, 2019. 8, 14

[32] X. Liu, P. Huang, and S. S. Ge, "Optimized control for human-multi-robot collaborative manipulation via multi-player q-learning," *Journal of the Franklin Institute*, 2021. 8

[33] H. Karami, K. Darvish, and F. Mastrogiovanni, "A task allocation approach for human-robot collaboration in product defects inspection scenarios," in *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2020, pp. 1127–1134. 8, 14

[34] G. Podevijn, R. O'grady, N. Mathews, A. Gilles, C. Fantini-Hauwel, and M. Dorigo, "Investigating the effect of increasing robot group sizes on the human psychophysiological state in the context of human–swarm interaction," *Swarm Intelligence*, vol. 10, no. 3, pp. 193–210, 2016. 8

[35] APA, "Social Interactions," https://dictionary.apa.org/social-interactions, Dictionary of Psychology, accessed: 2022-10-04. 8

[36] M. F. Jung, D. DiFranzo, S. Shen, B. Stoll, H. Claure, and A. Lawrence, "Robot-assisted tower construction—a method to study the impact of a robot's allocation behavior on interpersonal dynamics and collaboration in groups," *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 10, no. 1, pp. 1–23, 2020. 10, 13, 17

[37] G. Swamy, S. Reddy, S. Levine, and A. D. Dragan, "Scaled autonomy: Enabling human operators to control robot fleets," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 5942–5948. 11, 13, 16, 18, 22, 40, 70, 73, 98

[38] A. Rosenfeld, N. Agmon, O. Maksimov, and S. Kraus, "Intelligent agent supporting human–multi-robot team collaboration," *Artificial Intelligence*, vol. 252, pp. 211–231, 2017. 11, 18, 20, 23, 70, 71, 73, 87, 98

[39] M. R. Fraune, S. Sherrin, S. Sabanović, and E. R. Smith, "Rabble of robots effects: Number and type of robots modulates attitudes, emotions, and stereotypes," in *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, 2015, pp. 109–116. 12

[40] T. Fincannon, F. Jentsch, B. Sellers, and A. Talone, "Best practices in human operation of robotic/unmanned vehicles: A technical review of recommendations regarding the human-to-robot ratio," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 57, no. 1. SAGE Publications Sage CA: Los Angeles, CA, 2013, pp. 1268–1272. 12, 20

[41] T. Fincannon, J. R. Keebler, F. Jentsch, E. Phillips, and A. W. Evans III, "Team size, team role, communication modality, and team coordination in the distributed operation of multiple heterogeneous unmanned vehicles," *Journal of Cognitive Engineering and Decision Making*, vol. 5, no. 1, pp. 106–131, 2011. 12

[42] C. Breazeal, K. Dautenhahn, and T. Kanda, "Social robotics," *Springer handbook of robotics*, pp. 1935–1972, 2016. 13

[43] M. A. Goodrich and A. C. Schultz, *Human-robot interaction: a survey.* Now Publishers Inc, 2008. 13

[44] A. Bauer, D. Wollherr, and M. Buss, "Human–robot collaboration: A survey," *International Journal of Humanoid Robotics*, vol. 5, no. 01, pp. 47–66, 2008. 13

[45] B. Sellner, F. W. Heger, L. M. Hiatt, R. Simmons, and S. Singh, "Coordinated multiagent teams and sliding autonomy for large-scale assembly," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1425–1444, 2006. 13, 14, 15, 22

[46] A. Khasawneh, H. Rogers, J. Bertrand, K. C. Madathil, and A. Gramopadhye, "Human adaptation to latency in teleoperated multi-robot human-agent search and rescue teams," *Automation in Construction*, vol. 99, pp. 265–277, 2019. 13, 16, 20, 24, 70

[47] H. Yedidsion, J. Deans, C. Sheehan, M. Chillara, J. Hart, P. Stone, and R. J. Mooney, "Optimal use of verbal instructions for multi-robot human navigation guidance," in *International Conference on Social Robotics.* Springer, 2019, pp. 133–143. 13, 18

[48] J. Penders, L. Alboul, U. Witkowski, A. Naghsh, J. Saez-Pons, S. Herbrechtsmeier, and M. El-Habbal, "A robot swarm assisting a human fire-fighter," *Advanced Robotics*, vol. 25, no. 1-2, pp. 93–117, 2011. 13

[49] J. Saez-Pons, L. Alboul, and J. Penders, "Experiments in cooperative human multi-robot navigation," in *2011 IEEE International Conference on Robotics and Automation*. IEEE, 2011, pp. 1–4. 13

[50] X. Z. Tan, S. Reig, E. J. Carter, and A. Steinfeld, "From one to another: how robot-robot interaction affects users' perceptions following a transition between robots," in *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2019, pp. 114–122. 13, 14, 16, 17, 21, 23

[51] P. Khandelwal, S. Barrett, and P. Stone, "Leading the way: An efficient multi-robot guidance system," in *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems*, 2015, p. 1625–1633. 13

[52] I. Leite, M. McCoy, M. Lohani, D. Ullman, N. Salomons, C. Stokes, S. Rivers, and B. Scassellati, "Emotional storytelling in the classroom: Individual versus group interaction between children and robots," in *2015 10th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2015, pp. 75–82. 13, 19

[53] J. Swaminathan, J. Akintoye, M. R. Fraune, and H. Knight, "Robots that run their own human experiments: Exploring relational humor with multi-robot comedy," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 2021, pp. 1262–1268. 13

[54] D. J. Bruemmer, D. A. Few, R. L. Boring, J. L. Marble, M. C. Walton, and C. W. Nielsen, "Shared understanding for collaborative control," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 35, no. 4, pp. 494–504, 2005. 13

[55] R. R. Murphy, "Human-robot interaction in rescue robotics," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 34, no. 2, pp. 138–153, 2004. 13, 20

[56] R. R. Murphy, K. S. Pratt, and J. L. Burke, "Crew roles and operational protocols for rotary-wing micro-uavs in close urban environments," in *Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction*, 2008, pp. 73–80. 13, 14

[57] J. L. Drury, L. Riek, and N. Rackliffe, "A decomposition of uav-related situation awareness," in *1st ACM SIGCHI/SIGART conference on Human-robot interaction*, 2006, pp. 88–94. 13, 14, 20

[58] H. Claure, Y. Chen, J. Modi, M. Jung, and S. Nikolaidis, "Multi-armed bandits with fairness constraints for distributing resources to human teammates," in *2020 ACM/IEEE International Conference on Human-Robot Interaction*, 2020, pp. 299–308. 13, 17, 74

[59] Z. Carlson, T. Sweet, J. Rhizor, J. Poston, H. Lucas, and D. Feil-Seifer, "Team-building activities for heterogeneous groups of humans and robots," in *International Conference on Social Robotics*. Springer, 2015, pp. 113–123. 13

[60] S. Rosenthal, M. Veloso, and A. K. Dey, "Is someone in this office available to help me?" *Journal of Intelligent & Robotic Systems*, vol. 66, no. 1-2, pp. 205–221, 2012. 13, 18, 21

[61] E. Short and M. J. Mataric, "Robot moderation of a collaborative game: Towards socially assistive robotics in group interactions," in *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2017, pp. 385–390. 13, 16, 17

[62] M. Vázquez, A. Steinfeld, and S. E. Hudson, "Maintaining awareness of the focus of attention of a conversation: A robot-centric reinforcement learning approach," in *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2016, pp. 36–43. 13

[63] E. S. Kim, L. D. Berkovits, E. P. Bernier, D. Leyzberg, F. Shic, R. Paul, and B. Scassellati, "Social robots as embedded reinforcers of social behavior in children with autism," *Journal of autism and developmental disorders*, vol. 43, no. 5, pp. 1038–1049, 2013. 13, 21

[64] H. Kozima, C. Nakagawa, and Y. Yasuda, "Interactive robots for communication-care: A case-study in autism therapy," in *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005*. IEEE, 2005, pp. 341–346. 13

[65] K. Dautenhahn, "Roles and functions of robots in human society: implications from research in autism therapy," *Robotica*, vol. 21, no. 4, pp. 443–452, 2003. 13

[66] C. Fernández-Llamas, M. Á. Conde, F. J. Rodríguez-Sedano, F. J. Rodríguez-Lera, and V. Matellán-Olivera, "Analysing the computational competences acquired by K-12 students when lectured by robotic and human teachers," *International Journal of Social Robotics*, vol. 12, no. 5, pp. 1009–1019, 2020. 13, 23

[67] S. Chandra, P. Alves-Oliveira, S. Lemaignan, P. Sequeira, A. Paiva, and P. Dillenbourg, "Children's peer assessment and self-disclosure in the presence of an educational robot," in *2016 25th IEEE international symposium on robot and human interactive communication (RO-MAN)*. IEEE, 2016, pp. 539–544. 13, 14

[68] F. Tanaka, K. Isshiki, F. Takahashi, M. Uekusa, R. Sei, and K. Hayashi, "Pepper learns together with children: Development of an educational application," in *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2015, pp. 270–275. 13

[69] L. Fortunati, F. Cavallo, and M. Sarrica, "Multiple communication roles in human–robot interactions in public space," *International Journal of Social Robotics*, pp. 1–14, 2018. 13, 14, 15, 16, 19

[70] S. D. Ramchurn, J. E. Fischer, Y. Ikuno, F. Wu, J. Flann, and A. Waldock, "A study of human-agent collaboration for multi-uav task allocation in dynamic environments," in *Twenty-Fourth International Joint Conference on Artificial Intelligence*, 2015. 13, 19

[71] G.-J. M. Kruijff, M. Janíček, S. Keshavdas, B. Larochelle, H. Zender, N. J. Smets, T. Mioch, M. A. Neerincx, J. V. Diggelen, F. Colas, *et al.*, "Experience in system design for human-robot teaming in urban search and rescue," in *Field and Service Robotics*. Springer, 2014, pp. 111–125. 13, 14, 15, 21

[72] M. Lewis, H. Wang, S. Y. Chien, P. Velagapudi, P. Scerri, and K. Sycara, "Process and performance in human-robot teams," *Journal of Cognitive Engineering and Decision Making*, vol. 5, no. 2, pp. 186–208, 2011. 13

[73] P.-J. Lee, H. Wang, S.-Y. Chien, M. Lewis, P. Scerri, P. Velagapudi, K. Sycara, and B. Kane, "Teams for teams performance in multi-human/multi-robot teams," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 54, no. 4. SAGE Publications Sage CA: Los Angeles, CA, 2010, pp. 438–442. 13

[74] J. Patel and C. Pinciroli, "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)*. IEEE, 2020, pp. 1135–1142. 13, 18, 19

[75] F. Driewer, M. Sauer, and K. Schilling, "Design and evaluation of a teleoperation interface for heterogeneous human-robot teams," *IFAC Proceedings Volumes*, vol. 40, no. 16, pp. 113–118, 2007. 13, 20

[76] J. M. Bradshaw, A. Acquisti, J. Allen, M. R. Breedy, L. Bunch, N. Chambers, P. Feltovich, L. Galescu, M. A. Goodrich, R. Jeffers, *et al.*, "Teamwork-centered autonomy for extended human-agent interaction in space applications," in *AAAI 2004 Spring Symposium*, 2004, pp. 22–24. 13

[77] M. Lippi and A. Marino, "A mixed-integer linear programming formulation for human multi-robot task allocation," *arXiv preprint arXiv:2106.06772*, 2021. 14

[78] M. S. Malvankar-Mehta and S. S. Mehta, "Optimal task allocation in multi-human multi-robot interaction," *Optimization Letters*, vol. 9, no. 8, pp. 1787–1803, 2015. 14, 23

[79] R. Oliveira, P. Arriaga, F. Correia, and A. Paiva, "Looking beyond collaboration: Socioemotional positive, negative and task-oriented behaviors in human–robot group interactions," *International Journal of Social Robotics*, vol. 12, no. 2, pp. 505–518, 2020. 14

[80] F. Correia, S. Mascarenhas, R. Prada, F. S. Melo, and A. Paiva, "Group-based emotions in teams of humans and robots," in *2018 ACM/IEEE international conference on human-robot interaction*, 2018, pp. 261–269. 14, 18, 19, 21, 23, 24

[81] P. Alves-Oliveira, P. Sequeira, F. S. Melo, G. Castellano, and A. Paiva, "Empathic robot for group learning: A field study," *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 8, no. 1, pp. 1–34, 2019. 14, 18, 19, 23

[82] I. Leite, M. McCoy, M. Lohani, N. Salomons, K. McElvaine, C. Stokes, S. Rivers, and B. Scassellati, "Autonomous disengagement classification and repair in multiparty child-robot interaction," in *2016 25th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2016, pp. 525–532. 14, 24

[83] C. Chen, Y. Liu, S. Kreiss, and A. Alahi, "Crowd-robot interaction: Crowd-aware robot navigation with attention-based deep reinforcement learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6015–6022. 14

[84] M. E. Foster, A. Gaschler, M. Giuliani, A. Isard, M. Pateraki, and R. P. Petrick, "Two people walk into a bar: Dynamic multi-party social interaction with a robot agent," in *Proceedings of the 14th ACM international conference on Multimodal interaction*, 2012, pp. 3–10. 14

[85] H. Erel, D. Trayman, C. Levy, A. Manor, M. Mikulincer, and O. Zuckerman, "Enhancing emotional support: The effect of a robotic object on human–human support quality," *International Journal of Social Robotics*, pp. 1–20, 2021. 14

[86] A. Freedy, O. Sert, E. Freedy, J. McDonough, G. Weltman, M. Tambe, T. Gupta, W. Grayson, and P. Cabrera, "Multiagent adjustable autonomy framework (maaf) for multi-robot, multi-human teams," in *2008 International Symposium on Collaborative Technologies and Systems.* IEEE, 2008, pp. 498–505. 14, 19

[87] J. Wang and M. Lewis, "Assessing cooperation in human control of heterogeneous robots," in *3rd ACM/IEEE international conference on Human robot interaction*, 2008, pp. 9–16. 14

[88] J. Scholtz, "Theory and evaluation of human robot interactions," in *36th Annual Hawaii International Conference on System Sciences, 2003. Proceedings of the.* IEEE, 2003, pp. 10–pp. 14

[89] M. Li, M. Kwon, and D. Sadigh, "Influencing leading and following in human–robot teams," *Autonomous Robots*, vol. 45, no. 7, pp. 959–978, 2021. 14, 22

[90] S. Joshi and S. Šabanović, "Robots for inter-generational interactions: implications for nonfamilial community settings," in *14th ACM/IEEE International Conference on Human-Robot Interaction (HRI).* IEEE, 2019, pp. 478–486. 14, 21

[91] E. S. Short, K. Swift-Spong, H. Shim, K. M. Wisniewski, D. K. Zak, S. Wu, E. Zelinski, and M. J. Matarić, "Understanding social interactions with socially assistive robotics in intergenerational family groups," in *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN).* IEEE, 2017, pp. 236–241. 14, 21

[92] A. Taheri, A. Meghdari, M. Alemi, and H. Pouretemad, "Human–robot interaction in autism treatment: A case study on three pairs of autistic children as twins, siblings, and classmates," *International Journal of Social Robotics*, vol. 10, no. 1, pp. 93–113, 2018. 14

[93] H. Wang, M. Lewis, P. Velagapudi, P. Scerri, and K. Sycara, "How search and its subtasks scale in n robots," in *4th ACM/IEEE international conference on Human robot interaction*, 2009, pp. 141–148. 14

[94] M. A. Goodrich, M. Quigley, and K. Cosenzo, "Task switching and multi-robot teams," in *Multi-Robot Systems. From Swarms to Intelligent Automata Volume III*. Springer, 2005, pp. 185–195. 14

[95] J. A. Adams, "Multiple robot/single human interaction: Effects on perceived workload," *Behaviour & Information Technology*, vol. 28, no. 2, pp. 183–198, 2009. 14, 20

[96] C. M. Humphrey, C. Henk, G. Sewell, B. W. Williams, and J. A. Adams, "Assessing the scalability of a multiple robot interface," in *2007 2nd ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2007, pp. 239–246. 14

[97] J. Suh and C.-w. Woo, "Design and development of a social robot framework for providing an intelligent service," in *Proc. World Congress on Engineering and Computer Science*, vol. 2, 2009. 15

[98] Z. G. Saribatur, V. Patoglu, and E. Erdem, "Finding optimal feasible global plans for multiple teams of heterogeneous robots using hybrid reasoning: an application to cognitive factories," *Autonomous Robots*, vol. 43, no. 1, pp. 213–238, 2019. 15

[99] M. R. Rosa, "Adaptive synchronization for heterogeneous multi-agent systems with switching topologies," *Machines*, vol. 6, no. 1, p. 7, 2018. 15

[100] Z. G. Saribatur, E. Erdem, and V. Patoglu, "Cognitive factories with multiple teams of heterogeneous robots: Hybrid reasoning for optimal feasible global plans," in *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014, pp. 2923–2930. 15

[101] J. Feine, U. Gnewuch, S. Morana, and A. Maedche, "A taxonomy of social cues for conversational agents," *International Journal of Human-Computer Studies*, vol. 132, pp. 138–161, 2019. 15, 21

[102] B. Gromov, L. M. Gambardella, and G. A. Di Caro, "Wearable multi-modal interface for human multi-robot interaction," in *2016 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2016, pp. 240–245. 15, 21

[103] A. Rule and J. Forlizzi, "Designing interfaces for multi-user, multi-robot systems," in *Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction*, 2012, pp. 97–104. 15, 20

[104] N. Mavridis, "A review of verbal and non-verbal human–robot interactive communication," *Robotics and Autonomous Systems*, vol. 63, pp. 22–35, 2015. 15

[105] R. Stiefelhagen, C. Fugen, R. Gieselmann, H. Holzapfel, K. Nickel, and A. Waibel, "Natural human-robot interaction using speech, head pose and gestures," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, vol. 3. IEEE, 2004, pp. 2422–2427. 15

[106] P. Schermerhorn and M. Scheutz, "Disentangling the effects of robot affect, embodiment, and autonomy on human team members in a mixed-initiative task," in *Proceedings from the International Conference on Advances in Computer-Human Interactions*. Citeseer, 2011, pp. 236–241. 15

[107] J. Y. Chen, E. C. Haas, and M. J. Barnes, "Human performance issues and user interface design for teleoperated robots," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 37, no. 6, pp. 1231–1245, 2007. 15

[108] H. Hüttenrauch, K. S. Eklundh, A. Green, and E. A. Topp, "Investigating spatial relationships in human-robot interaction," in *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2006, pp. 5052–5059. 15

[109] T. Williams, P. Briggs, and M. Scheutz, "Covert robot-robot communication: Human perceptions and implications for human-robot interaction," *Journal of Human-Robot Interaction*, vol. 4, no. 2, pp. 24–49, 2015. 15

[110] J. Patel, T. Ramaswamy, Z. Li, and C. Pinciroli, "Direct and indirect communication in multi-human multi-robot interaction," *arXiv preprint arXiv:2102.00672*, 2021. 15, 19

[111] Y. Che, A. M. Okamura, and D. Sadigh, "Efficient and trustworthy social navigation via explicit and implicit robot–human communication," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 692–707, 2020. 15

[112] D. Rifinski, H. Erel, A. Feiner, G. Hoffman, and O. Zuckerman, "Human-human-robot interaction: robotic object's responsive gestures improve interpersonal evaluation in human interaction," *Human–Computer Interaction*, vol. 36, no. 4, pp. 333–359, 2021. 16

[113] S. Liu, P. Chang, Z. Huang, N. Chakraborty, K. Hong, W. Liang, D. L. McPherson, J. Geng, and K. Driggs-Campbell, "Intention aware robot crowd navigation

with attention-based interaction graph," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 12 015–12 021. 16

[114] T. Setter, A. Fouraker, M. Egerstedt, and H. Kawashima, "Haptic interactions with multi-robot swarms using manipulability," *Journal of Human-Robot Interaction*, vol. 4, no. 1, pp. 60–74, 2015. 18

[115] A. Nanavati, M. Doering, D. Brščić, and T. Kanda, "Autonomously learning one-to-many social interaction logic from human-human interaction data," in *2020 ACM/IEEE International Conference on Human-Robot Interaction*, 2020, pp. 419–427. 18, 19

[116] Y. Lim, K. Ranasinghe, A. Gardi, N. Ezer, and R. Sabatini, "Human-machine interfaces and interactions for multi uas operations," in *Proceedings of the 31st Congress of the International Council of the Aeronautical Sciences (ICAS 2018), Belo Horizonte, Brazil*, 2018, pp. 9–14. 18, 19

[117] R. C. Arkin and K. Ali, "Integration of reactive and telerobotic control in multi-agent robotic systems," in *Third International Conference on Simulation of Adaptive Behavior*, 1994, pp. 473–478. 18

[118] H. Jones and M. Snyder, "Supervisory control of multiple robots based on a real-time strategy game interaction paradigm," in *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat. No. 01CH37236)*, vol. 1. IEEE, 2001, pp. 383–388. 18

[119] L. H. Kim, D. S. Drew, V. Domova, and S. Follmer, "User-defined swarm robot control," in *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, 2020, pp. 1–13. 19, 25

[120] T. Belpaeme, J. Kennedy, A. Ramachandran, B. Scassellati, and F. Tanaka, "Social robots for education: A review," *Science robotics*, vol. 3, no. 21, 2018. 19

[121] N. Ayanian, A. Spielberg, M. Arbesfeld, J. Strauss, and D. Rus, "Controlling a team of robots with a single input," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2014, pp. 1755–1762. 19

[122] G. Podevijn, R. O'Grady, Y. S. Nashed, and M. Dorigo, "Gesturing at subswarms: Towards direct human control of robot swarms," in *Conference Towards Autonomous Robotic Systems*. Springer, 2013, pp. 390–403. 19

[123] L. Zhang and R. Vaughan, "Optimal robot selection by gaze direction in multi-human multi-robot interaction," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 5077–5083. 19

[124] A. D. Tews, M. J. Mataric, and G. S. Sukhatme, "A scalable approach to human-robot interaction," in *2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422)*, vol. 2. IEEE, 2003, pp. 1665–1670. 19

[125] S. A. Mostafa, M. S. Ahmad, and A. Mustapha, "Adjustable autonomy: a systematic literature review," *Artificial Intelligence Review*, vol. 51, no. 2, pp. 149–186, 2019. 19

[126] W.-Y. Hwang and S.-Y. Wu, "A case study of collaboration with multi-robots and its effect on children's interaction," *Interactive Learning Environments*, vol. 22, no. 4, pp. 429–443, 2014. 19

[127] C. Nam, P. Walker, M. Lewis, and K. Sycara, "Predicting trust in human control of swarms via inverse reinforcement learning," in *2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2017, pp. 528–533. 20

[128] J. Scholtz, J. Young, J. L. Drury, and H. A. Yanco, "Evaluation of human-robot interaction awareness in search and rescue," in *IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04. 2004*, vol. 3. IEEE, 2004, pp. 2327–2332. 20

[129] Y. Wang, Z. Shi, C. Wang, and F. Zhang, "Human-robot mutual trust in (semi) autonomous underwater robots," in *Cooperative Robots and Sensor Networks 2014*. Springer, 2014, pp. 115–137. 20, 22, 24, 70

[130] C. L. Gittens, "Remote HRI: A methodology for maintaining COVID-19 physical distancing and human interaction requirements in HRI studies," *Information Systems Frontiers*, pp. 1–16, 2021. 20

[131] H. Hastie, D. A. Robb, J. Lopes, M. Ahmad, P. L. Bras, X. Liu, R. Petrick, K. Lohan, and M. J. Chantler, "Challenges in collaborative HRI for remote robot teams," *arXiv preprint arXiv:1905.07379*, 2019. 20

[132] T. Rezvani, K. Driggs-Campbell, and R. Bajcsy, "Optimizing interaction between humans and autonomy via information constraints on interface design," in *2017*

IEEE 20th International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2017, pp. 1–6. 20

[133] J. J. Roldán, E. Peña-Tapia, A. Martín-Barrio, M. A. Olivares-Méndez, J. Del Cerro, and A. Barrientos, "Multi-robot interfaces and operator situational awareness: Study of the impact of immersion and prediction," *Sensors*, vol. 17, no. 8, p. 1720, 2017. 20

[134] A. Hussein, L. Ghignone, T. Nguyen, N. Salimi, H. Nguyen, M. Wang, and H. A. Abbass, "Towards bi-directional communication in human-swarm teaming: A survey," *arXiv preprint arXiv:1803.03093*, 2018. 20

[135] S. Y. Chien, Y. L. Lin, P. J. Lee, S. Han, M. Lewis, and K. Sycara, "Attention allocation for human multi-robot control: Cognitive analysis based on behavior data and hidden states," *International Journal of Human-Computer Studies*, vol. 117, pp. 30–44, 2018. 20, 73, 99

[136] M. Ostanin, R. Yagfarov, D. Devitt, A. Akhmetzyanov, and A. Klimchik, "Multi robots interactive control using mixed reality," *International Journal of Production Research*, vol. 59, no. 23, pp. 7126–7138, 2021. 20

[137] J. A. Frank, S. P. Krishnamoorthy, and V. Kapila, "Toward mobile mixed-reality interaction with multi-robot systems," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 1901–1908, 2017. 20

[138] A. Hong, D. G. Lee, H. H. Bülthoff, and H. I. Son, "Multimodal feedback for teleoperation of multiple mobile robots in an outdoor environment," *Journal on Multimodal User Interfaces*, vol. 11, no. 1, pp. 67–80, 2017. 20

[139] S. Pourmehr, V. Monajjemi, S. A. Sadat, F. Zhan, J. Wawerla, G. Mori, and R. Vaughan, ""You are green" a touch-to-name interaction in an integrated multimodal multi-robot HRI system," in *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, 2014, pp. 266–267. 21

[140] M. Faria, R. Silva, P. Alves-Oliveira, F. S. Melo, and A. Paiva, ""Me and you together" movement impact in multi-user collaboration tasks," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017, pp. 2793–2798. 21

[141] J. Rios-Martinez, A. Spalanzani, and C. Laugier, "From proxemics theory to socially-aware navigation: A survey," *International Journal of Social Robotics*, vol. 7, no. 2, pp. 137–153, 2015. 21

[142] F. Yamaoka, T. Kanda, H. Ishiguro, and N. Hagita, "A model of proximity control for information-presenting robots," *IEEE Transactions on Robotics*, vol. 26, no. 1, pp. 187–195, 2009. 21

[143] J.-Y. Yang and D.-S. Kwon, "The effect of multiple robot interaction on human-robot interaction," in *2012 9th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*. IEEE, 2012, pp. 30–33. 21, 23, 24

[144] H. Erel, Y. Cohen, K. Shafrir, S. D. Levy, I. D. Vidra, T. Shem Tov, and O. Zuckerman, "Excluded by robots: Can robot-robot-human interaction lead to ostracism?" in *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, 2021, pp. 312–321. 21

[145] D. Druckman, L. Adrian, M. F. Damholdt, M. Filzmoser, S. T. Koszegi, J. Seibt, and C. Vestergaard, "Who is best at mediating a social conflict? comparing robots, screens and humans," *Group Decision and Negotiation*, vol. 30, no. 2, pp. 395–426, 2021. 21

[146] M. Shiomi, H. Sumioka, and H. Ishiguro, "Survey of social touch interaction between humans and robots," *Journal of Robotics and Mechatronics*, vol. 32, no. 1, pp. 128–135, 2020. 21

[147] J. Berg, A. Lottermoser, C. Richter, and G. Reinhart, "Human-robot-interaction for mobile industrial robot teams," *Procedia CIRP*, vol. 79, pp. 614–619, 2019. 21

[148] A. Noormohammadi-Asl, A. Ayub, S. L. Smith, and K. Dautenhahn, "Task selection and planning in human-robot collaborative processes: To be a leader or a follower?" in *2022 31st IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2022, pp. 1244–1251. 22

[149] M. Fooladi Mahani, L. Jiang, and Y. Wang, "A bayesian trust inference model for human-multi-robot teams," *International Journal of Social Robotics*, pp. 1–15, 2020. 22

[150] C. Nam, P. Walker, H. Li, M. Lewis, and K. Sycara, "Models of trust in human control of swarms with varied levels of autonomy," *IEEE Transactions on Human-Machine Systems*, vol. 50, no. 3, pp. 194–204, 2019. 22

[151] C. J. Shannon, L. B. Johnson, K. F. Jackson, and J. P. How, "Adaptive mission planning for coupled human-robot teams," in *2016 American Control Conference (ACC)*. IEEE, 2016, pp. 6164–6169. 22

[152] A. Bera, T. Randhavane, E. Kubin, A. Wang, K. Gray, and D. Manocha, "The socially invisible robot navigation in the social world using robot entitativity," in *2018 ieee/rsj international conference on intelligent robots and systems (iros)*. IEEE, 2018, pp. 4468–4475. 22

[153] A. Bajcsy, S. L. Herbert, D. Fridovich-Keil, J. F. Fisac, S. Deglurkar, A. D. Dragan, and C. J. Tomlin, "A scalable framework for real-time multi-robot, multi-human collision avoidance," in *2019 international conference on robotics and automation (ICRA)*. IEEE, 2019, pp. 936–943. 22

[154] Y. Boussemart and M. L. Cummings, "Predictive models of human supervisory control behavioral patterns using hidden semi-markov models," *Engineering Applications of Artificial Intelligence*, vol. 24, no. 7, pp. 1252–1262, 2011. 22

[155] D. R. Olsen Jr and S. B. Wood, "Fan-out: measuring human control of multiple robots," in *SIGCHI conference on Human factors in computing systems*, 2004, pp. 231–238. 22, 70

[156] S. Y. Chien, M. Lewis, S. Mehrotra, and K. Sycara, "Imperfect automation in scheduling operator attention on control of multi-robots," in *Human Factors and Ergonomics Society Annual Meeting*, vol. 57, no. 1, 2013, pp. 1169–1173. 22, 70

[157] M. R. Fraune, S. Šabanović, E. R. Smith, Y. Nishiwaki, and M. Okada, "Threatening flocks and mindful snowflakes: How group entitativity affects perceptions of robots," in *2017 12th ACM/IEEE International Conference on Human-Robot Interaction*. IEEE, 2017, pp. 205–213. 23

[158] A. Bera, T. Randhavane, A. Wang, D. Manocha, E. Kubin, and K. Gray, "Classifying group emotions for socially-aware autonomous vehicle navigation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2018, pp. 1039–1047. 23

[159] M. Faria, F. S. Melo, and A. Paiva, "Understanding robots: Making robots more legible in multi-party interactions," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 2021, pp. 1031–1036. 23

[160] L. D. Riek, "Wizard of oz studies in HRI: a systematic review and new reporting guidelines," *Journal of Human-Robot Interaction*, vol. 1, no. 1, pp. 119–136, 2012. 23, 25

[161] M. Johansson, G. Skantze, and J. Gustafson, "Head pose patterns in multi-party human-robot team-building interactions," in *International conference on social robotics*. Springer, 2013, pp. 351–360. 23

[162] M. Shiomi, T. Kanda, D. F. Glas, S. Satake, H. Ishiguro, and N. Hagita, "Field trial of networked social robots in a shopping mall," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2009, pp. 2846–2853. 23

[163] S. Jiang and R. C. Arkin, "Mixed-initiative human-robot interaction: definition, taxonomy, and survey," in *2015 IEEE International Conference on Systems, Man, and Cybernetics*. IEEE, 2015, pp. 954–961. 24

[164] Y. Wang, L. R. Humphrey, Z. Liao, and H. Zheng, "Trust-based multi-robot symbolic motion planning with a human-in-the-loop," *ACM Transactions on Interactive Intelligent Systems (TiiS)*, vol. 8, no. 4, pp. 1–33, 2018. 24

[165] R. Liu, Z. Cai, M. Lewis, J. Lyons, and K. Sycara, "Trust repair in human-swarm teams+," in *2019 28th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN)*. IEEE, 2019, pp. 1–6. 24

[166] R. Savery, A. Rogel, and G. Weinberg, "Emotion musical prosody for robotic groups and entitativity," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 2021, pp. 440–446. 24

[167] E. W. Dijkstra, "A note on two problems in connexion with graphs," in *Edsger Wybe Dijkstra: His Life, Work, and Legacy*, 2022, pp. 287–290. 30

[168] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. 30

[169] X. Cai, T. Kloks, and C.-K. Wong, "Time-varying shortest path problems with constraints," *Networks: An International Journal*, vol. 29, no. 3, pp. 141–150, 1997. 32, 41, 52

[170] G. L. Nemhauser, L. A. Wolsey, and M. L. Fisher, "An analysis of approximations for maximizing submodular set functions—i," *Mathematical programming*, vol. 14, pp. 265–294, 1978. 33

[171] D. Golovin and A. Krause, "Adaptive submodularity: Theory and applications in active learning and stochastic optimization," *Journal of Artificial Intelligence Research*, vol. 42, pp. 427–486, 2011. 33

[172] S. Alaei, A. Makhdoumi, and A. Malekian, "Maximizing sequence-submodular functions and its application to online advertising," *Management Science*, vol. 67, no. 10, pp. 6030–6054, 2021. 33

[173] P. Whittle, "Restless bandits: Activity allocation in a changing world," *Journal of applied probability*, vol. 25, no. A, pp. 287–298, 1988. 35, 74

[174] L. Royakkers and R. van Est, "A literature review on new robotics: automation from love to war," *International journal of social robotics*, vol. 7, pp. 549–570, 2015. 37

[175] M. Mintrom, S. Sumartojo, D. Kulić, L. Tian, P. Carreno-Medrano, and A. Allen, "Robots in public spaces: Implications for policy design," *Policy Design and Practice*, vol. 5, no. 2, pp. 123–139, 2022. 37

[176] D. G. Riley and E. W. Frew, "Assessment of coordinated heterogeneous exploration of complex environments," in *IEEE Conference on Control Technology and Applications (CCTA)*, 2021, pp. 138–143. 38, 40

[177] Y. Wang, Y. Yuan, Y. Ma, and G. Wang, "Time-dependent graphs: Definitions, applications, and algorithms," *Data Science and Engineering*, vol. 4, pp. 352–366, 2019. 38, 40, 41

[178] C. Hickert, S. Li, and C. Wu, "Cooperation for scalable supervision of autonomy in mixed traffic," *arXiv preprint arXiv:2112.07569*, 2021. 40

[179] Y. Cai, A. Dahiya, N. Wilde, and S. L. Smith, "Scheduling operator assistance for shared autonomy in multi-robot teams," in *IEEE Conference on Decision and Control (CDC)*, 2022, pp. 3997–4003. 40, 61, 62

[180] F. Fusaro, E. Lamon, E. De Momi, and A. Ajoudani, "An integrated dynamic method for allocating roles and planning tasks for mixed human-robot teams," in *IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*, 2021, pp. 534–539. 40

[181] S. Mau and J. Dolan, "Scheduling for humans in multirobot supervisory control," in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2007, pp. 1637–1643. 40

[182] N. D. Powel and K. A. Morgansen, "Multiserver queueing for supervisory control of autonomous vehicles," in *2012 American Control Conference (ACC)*. IEEE, 2012, pp. 3179–3185. 40

[183] B. C. Dean, "Algorithms for minimum-cost paths in time-dependent networks with waiting policies," *Networks: An International Journal*, vol. 44, no. 1, pp. 41–46, 2004. 40

[184] L. Zhao, T. Ohshima, and H. Nagamochi, "A* algorithm for the time-dependent shortest path problem," in *WAAC08: The 11th Japan-Korea Joint Workshop on Algorithms and Computation*, vol. 10, 2008. 40, 55

[185] A. Orda and R. Rom, "Shortest-path and minimum-delay algorithms in networks with time-dependent edge-length," *Journal of the ACM (JACM)*, vol. 37, no. 3, pp. 607–625, 1990. 40

[186] B. Ding, J. X. Yu, and L. Qin, "Finding time-dependent shortest paths over large graphs," in *International Conference on Extending Database Technology: Advances in Database Technology*, 2008, pp. 205–216. 40, 41

[187] M. Bentert, A.-S. Himmel, A. Nichterlein, and R. Niedermeier, "Efficient computation of optimal temporal walks under waiting-time constraints," *Applied Network Science*, vol. 5, no. 1, pp. 1–26, 2020. 40

[188] E. He, N. Boland, G. Nemhauser, and M. Savelsbergh, "Time-dependent shortest path problems with penalties and limits on waiting," *INFORMS Journal on Computing*, vol. 33, no. 3, pp. 997–1014, 2021. 41

[189] L. Foschini, J. Hershberger, and S. Suri, "On the complexity of time-dependent shortest paths," in *ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 2011, pp. 327–341. 41

[190] L. R. Ford Jr and D. R. Fulkerson, "Constructing maximal dynamic flows from static flows," *Operations research*, vol. 6, no. 3, pp. 419–433, 1958. 41, 53

[191] T. Ji, R. Dong, and K. Driggs-Campbell, "Traversing supervisor problem: An approximately optimal approach to multi-robot assistance," *arXiv preprint arXiv:2205.01768*, 2022. 61, 73

[192] A. M. Elmogy, "Market_based framework for mobile surveillance systems," http://hdl.handle.net/10012/5324, 2010. 63

[193] K. Zheng, D. F. Glas, T. Kanda, H. Ishiguro, and N. Hagita, "Supervisory control of multiple social robots for navigation," in *2013 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2013, pp. 17–24. 70, 100, 101

[194] J. Y. Chen and M. J. Barnes, "Supervisory control of multiple robots: Effects of imperfect automation and individual differences," *Human Factors*, vol. 54, no. 2, pp. 157–174, 2012. 70, 100, 101

[195] J. Y. Chen and M. J. Barnes, "Human–agent teaming for multirobot control: A review of human factors issues," *IEEE Transactions on Human-Machine Systems*, vol. 44, no. 1, pp. 13–29, 2014. 70, 103

[196] M. L. Puterman, *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014. 71, 77

[197] M. L. Littman, T. L. Dean, and L. P. Kaelbling, "On the complexity of solving markov decision problems," in *Eleventh Annual Conference on Uncertainty in Artificial Intelligence*, 1995, pp. 394–402. 71

[198] S. A. Zanlongo, P. Dirksmeier, P. Long, T. Padir, and L. Bobadilla, "Scheduling and path-planning for operator oversight of multiple robots," *Robotics*, vol. 10, no. 2, p. 57, 2021. 71, 73, 75

[199] J. R. Peters, V. Srivastava, G. S. Taylor, A. Surana, M. P. Eckstein, and F. Bullo, "Human supervisory control of robotic teams: Integrating cognitive modeling with engineering design," *IEEE Control Systems Magazine*, vol. 35, no. 6, pp. 57–80, 2015. 73

[200] J. W. Crandall, M. L. Cummings, M. Della Penna, and P. M. De Jong, "Computing the effects of operator attention allocation in human control of multiple robots," *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, vol. 41, no. 3, pp. 385–397, 2010. 73

[201] S. Music and S. Hirche, "Control sharing in human-robot team interaction," *Annual Reviews in Control*, vol. 44, pp. 342–354, 2017. 73, 101

[202] M. B. Dias, B. Kannan, B. Browning, E. Jones, B. Argall, M. F. Dias, M. Zinck, M. Veloso, and A. Stentz, "Sliding autonomy for peer-to-peer human-robot teams," in *International conference on intelligent autonomous systems*, 2008, pp. 332–341. 73, 101

[203] D. Szafir, B. Mutlu, and T. Fong, “Designing planning and control interfaces to support user collaboration with flying robots,” *The International Journal of Robotics Research*, vol. 36, no. 5-7, pp. 514–542, 2017. 73

[204] E. A. Kirchner, S. K. Kim, M. Tabie, H. Wöhrle, M. Maurus, and F. Kirchner, “An intelligent man-machine interface—multi-robot control adapted for task engagement based on single-trial detectability of p300,” *Frontiers in human neuroscience*, vol. 10, p. 291, 2016. 73

[205] M. M. Raeissi, N. Brooks, and A. Farinelli, “A balking queue approach for modeling human-multi-robot interaction for water monitoring,” in *International Conference on Principles and Practice of Multi-Agent Systems*. Springer, 2017, pp. 212–223. 73

[206] J. Berger, N. Lo, and M. Noel, “A new multi-target, multi-agent search-and-rescue path planning approach,” *International Journal of Computer, Electrical, Automation, Control and Information Engineering*, vol. 8, no. 6, pp. 935–944, 2014. 74

[207] H. Pei, S. Chen, and Q. Lai, “Multi-target consensus circle pursuit for multi-agent systems via a distributed multi-flocking method,” *International Journal of Systems Science*, vol. 47, no. 16, pp. 3741–3748, 2016. 74

[208] J. C. Gittins, “Bandit processes and dynamic allocation indices,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 41, no. 2, pp. 148–164, 1979. 74

[209] L. Chan, D. Hadfield-Menell, S. Srinivasa, and A. Dragan, “The assistive multi-armed bandit,” in *ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2019, pp. 354–363. 74

[210] A. Mahajan and D. Teneketzis, “Multi-armed bandit problems,” in *Foundations and applications of sensor management*. Springer, 2008, pp. 121–151. 74

[211] C. H. Papadimitriou and J. N. Tsitsiklis, “The complexity of optimal queuing network control,” *Mathematics of Operations Research*, vol. 24, no. 2, pp. 293–305, 1999. 74

[212] G. Xiong, R. Singh, and J. Li, “Learning augmented index policy for optimal service placement at the network edge,” *arXiv preprint arXiv:2101.03641*, 2021. 74

[213] V. S. Borkar, G. S. Kasbekar, S. Pattathil, and P. Y. Shetty, “Opportunistic scheduling as restless bandits,” *IEEE Transactions on Control of Network Systems*, vol. 5, no. 4, pp. 1952–1961, 2017. 74

[214] N. Akbarzadeh and A. Mahajan, "Conditions for indexability of restless bandits and an $\mathcal{O}(K^3)$ algorithm to compute whittle index," *arXiv preprint arXiv:2008.06111*, 2020. 74, 83, 84

[215] S. Wu, K. Ding, P. Cheng, and L. Shi, "Optimal scheduling of multiple sensors over lossy and bandwidth limited channels," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 3, pp. 1188–1200, 2020. 74

[216] K. Avrachenkov, U. Ayesta, J. Doncel, and P. Jacko, "Congestion control of TCP flows in internet routers by means of index policy," *Computer Networks*, vol. 57, no. 17, pp. 3463–3478, 2013. 74

[217] K. Liu and Q. Zhao, "Indexability of restless bandit problems and optimality of whittle index for dynamic multichannel access," *IEEE Transactions on Information Theory*, vol. 56, no. 11, pp. 5547–5567, 2010. 74

[218] N. Akbarzadeh and A. Mahajan, "Dynamic spectrum access under partial observations: A restless bandit approach," in *Canadian Workshop on Information Theory*. IEEE, 2019, pp. 1–6. 74, 83

[219] J. Niño-Mora, "Dynamic priority allocation via restless bandit marginal productivity indices," *Top*, vol. 15, no. 2, pp. 161–198, 2007. 83

[220] M. Egorov, Z. N. Sunberg, E. Balaban, T. A. Wheeler, J. K. Gupta, and M. J. Kochenderfer, "POMDPs.jl: A framework for sequential decision making under uncertainty," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 831–835, 2017. 84

[221] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *4th International Conference on Learning Representations, ICLR*, 2016. 88

[222] P. Carreno-Medrano, A. Dahiya, S. L. Smith, and D. Kulić, "Incremental estimation of users' expertise level," in *International Conference on Robot & Human Interactive Communication (ROMAN)*. IEEE, 2019. 98

[223] C. Y. Wong and G. Seet, "Workload, awareness and automation in multiple-robot supervision," *International Journal of Advanced Robotic Systems*, vol. 14, no. 3, 2017. 99, 100, 101

[224] A. Rossi, M. Staffa, and S. Rossi, "Supervisory control of multiple robots through group communication," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 9, no. 1, pp. 56–67, 2016. 100, 101

[225] D. A. Wiegmann, A. W. ElBardissi, J. A. Dearani, R. C. Daly, and T. M. Sundt III, "Disruptions in surgical flow and their relationship to surgical errors: an exploratory investigation," *Surgery*, vol. 142, no. 5, pp. 658–665, 2007. 100, 102

[226] H. M. Herrick, S. Lorch, J. Y. Hsu, K. Catchpole, and E. E. Foglia, "Impact of flow disruptions in the delivery room," *Resuscitation*, vol. 150, pp. 29–35, 2020. 100, 102

[227] G. Fealy, S. Donnelly, G. Doyle, M. Brenner, M. Hughes, E. Mylotte, E. Nicholson, and M. Zaki, "Clinical handover practices among healthcare practitioners in acute care services: A qualitative study," *Journal of clinical nursing*, vol. 28, no. 1-2, pp. 80–88, 2019. 100, 102

[228] P. Zigoris, J. Siu, O. Wang, and A. T. Hayes, "Balancing automated behavior and human control in multi-agent systems: a case study in roboflag," in *Proceedings of the 2003 American Control Conference, 2003.*, vol. 1. IEEE, 2003, pp. 667–671. 101

[229] P. Squire and R. Parasuraman, "Effects of automation and task load on task switching during human supervision of multiple semi-autonomous robots in a dynamic environment," *Ergonomics*, vol. 53, no. 8, pp. 951–961, 2010. 101

[230] J. Y. Chen, M. J. Barnes, and M. Harper-Sciarini, "Supervisory control of multiple robots: Human-performance issues and user-interface design," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 41, no. 4, pp. 435–454, 2010. 101

[231] J. M. Riley and M. R. Endsley, "Situation awareness in hri with collaborating remotely piloted vehicles," in *proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 49, no. 3. SAGE Publications Sage CA: Los Angeles, CA, 2005, pp. 407–411. 101

[232] J. D. Oury and F. E. Ritter, *Building Better Interfaces for Remote Autonomous Systems: An Introduction for Systems Engineers.* Springer Nature, 2021. 101, 102

[233] K. R. Campoe and K. K. Giuliano, "Impact of frequent interruption on nurses' patient-controlled analgesia programming performance," *Human factors*, vol. 59, no. 8, pp. 1204–1213, 2017. 101, 102

[234] F. Sasangohar, B. Donmez, P. Trbovich, and A. C. Easty, "Not all interruptions are created equal: positive interruptions in healthcare," in *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, vol. 56, no. 1. SAGE Publications Sage CA: Los Angeles, CA, 2012, pp. 824–828. 102

[235] S. Addas and A. Pinsonneault, "The many faces of information technology interruptions: a taxonomy and preliminary investigation of their performance effects," *Information Systems Journal*, vol. 25, no. 3, pp. 231–273, 2015. 102

[236] A. J. Gould, "What makes an interruption disruptive? understanding the effects of interruption relevance and timing on performance," Ph.D. dissertation, UCL (University College London), 2014. 102

[237] J. Gluck, A. Bunt, and J. McGrenere, "Matching attentional draw with utility in interruption," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2007, pp. 41–50. 102

[238] J. Q. Young, D. M. Irby, M.-L. Barilla-LaBarca, O. Ten Cate, and P. S. O'Sullivan, "Measuring cognitive load: mixed results from a handover simulation for medical students," *Perspectives on medical education*, vol. 5, no. 1, pp. 24–32, 2016. 102

[239] J. Sweller, J. J. Van Merrienboer, and F. G. Paas, "Cognitive architecture and instructional design," *Educational psychology review*, vol. 10, no. 3, pp. 251–296, 1998. 102

[240] S. G. Hart, "NASA-task load index (NASA-tlx); 20 years later," in *Proceedings of the human factors and ergonomics society annual meeting*, vol. 50, no. 9, 2006, pp. 904–908. 106, 107