Hierarchical MPC Task Allocation for Heterogeneous Search and Rescue Robots with Safe Return Constraints

A Multi-Criteria Decision-Making Approach

Berkay Yazıcıoğlu

Delft University of Technology

TU Delft

Hierarchical MPC Task Allocation for Heterogeneous Search and Rescue Robots with Safe Return Constraints

A Multi-Criteria Decision-Making Approach

by

Berkay Yazıcıoğlu

to obtain the degree of Master of Science
at the Delft University of Technology

Student Number:	5564395	
Thesis Committee:	Dr. Manuel Mazo Espinosa,	TU Delft, Chair, Supervisor
	Dr. Anahita Jamshidnejad,	TU Delft, Examiner, Daily Supervisor
	Dr. Matthijs Spaan,	TU Delft, Examiner
Research Group:	Delft Center for Systems and Control	
Project Duration:	February 2023 - August 2025	

Cover: Aerial view of the earthquake destruction in the Hatay province of Turkey, February 7, 2023. (IHA via AP, modified)

TU Delft

Preface

This thesis project titled "Hierarchical MPC Task Allocation for Heterogeneous Search and Rescue Robots with Safe Return Constraints: A Multi-Criteria Decision-Making Approach" marks the conclusion of a challenging and long period of research as a graduate student within the faculty of Electrical Engineering, Mathematics, and Computer Science at the Delft University of Technology.

During my studies, I aimed to take on challenges from various fields that are related to my major, and this work is a culmination of everything I learned, as well as the new frontiers that my research led me towards. This project has undergone numerous iterations, during which it was often exciting and enjoyable to explore novel ideas. Having said that, there have also been times when I struggled a lot and making progress was difficult. Looking back, I've grown significantly as a person, and I've learned that failure is not an outcome but a part of the process, which resulted in a work that I can say I am proud of. I want to extend my sincerest gratitude to my supervisors, especially Dr. Anahita Jamshidnejad, for believing in me during this entire period. Our discussions about the topic were always immensely inspiring, where her guidance and expertise at every point of the research made me more motivated to not only produce a meaningful work as an engineer, but also to seek what has not been explored as a researcher.

I wish to thank my family for their unwavering support and trust in every decision I made. Without their love, sacrifices, and encouragements, this phase of my life would not even be possible in the first place. My partner in everything, Müge, thank you for becoming my home here and being there for me with every up and down. You were the light during my struggles and the muse of my successes. I also want to thank my friends for becoming my extended family; knowing that I could count on all of you encouraged me to build a new life in a new country, together. Everyone I mentioned made my time as a student in Delft the most memorable and truly the most joyful period of my life, and I am looking forward to seeing what comes next. Finally, I want to thank you, the reader, for taking the time to read this work.

Berkay Yazıcıoğlu
Delft, August 2025

Summary

Urban search and rescue (USaR) missions take place in complex, volatile environments where time, information, and resources are limited. In such scenarios, the primary objective is to locate and assist victims as efficiently as possible, often amid unknown and uncertain conditions. Traditional approaches rely heavily on human teams that must adapt in real-time, reassigning roles, shifting priorities, and coordinating actions under uncertainty. While robots have been increasingly introduced to assist in these efforts, their autonomy has been largely restricted to lower-level tasks, such as local navigation, mapping, and obstacle avoidance. These capabilities, while valuable, fall short when robots are expected to make mission-level decisions about what to do next, how to prioritize actions, or how to collaborate as part of a team. To achieve higher levels of autonomy, robotic systems must be able to reason about dynamically evolving tasks, heterogeneous capabilities, and long-term operational constraints. Existing task allocation frameworks often assume that tasks are predefined and utility functions, which quantify the value of taking actions with regard to mission goals, are fixed. This limits their adoption in real-world applications where task definitions may emerge from the environment and change over time. Furthermore, few systems adequately support continuous decision-making where robots must periodically return for charging and resume operation without predefined sequences. These gaps in adaptability, coordination, and mission continuity present a significant challenge for the autonomous deployment of robots in realistic USaR scenarios.

This thesis proposes a hierarchical decision-making and coordination architecture that addresses these challenges by unifying human-inspired reasoning with structured, optimization-based control. A generalized system model is introduced in which robots are described not by fixed roles but by their capabilities of action and perception, enabling the assessment of their individual suitability to tackle the wide range of dynamically emerging tasks resulting from the robot-environment interactions. Subsequently, the optimal control of sequential decision-making is modeled as a nonlinear model predictive control (MPC) problem with safe return constraints, enabling informed decision-making over a finite horizon with an evolving set of tasks. The assessment of task suitability is initially evaluated by a fuzzy inference system that combines the robot's capabilities with the task-specific prioritization measures. Then, a multi-criteria decision-making (MCDM) framework assigns mission-level utilities to task-related robot actions by weighing factors such as urgency, information gain, and inter-task preferences using the fuzzy system outputs, establishing flexibility and robustness when task definitions are ambiguous or unexpected and mission priorities shift over time, as hard-coded heuristics often fail to adapt. The decision-making framework is further integrated into a two-tier hierarchical control architecture. The first tier involves local controllers, where each robot solves a linearization of the MPC formulation by first filtering independent actions using utility estimates and observation overlaps, and then solving a mixed-integer linear programming (MILP)-based scheduling problem subject to safe return constraints in a computationally tractable way. In the second tier, the multi-robot extension of the MILP-based MPC architecture resolves conflicts between robot plans by integrating their local solutions into a globally consistent allocation. This hierarchical structure balances the adaptability of distributed control with the consistency of centralized planning, while also considering the remaining energy of the robots.

The proposed architecture was implemented in realistic computer-based USaR simulations involving dynamically evolving tasks, heterogeneous robot models, and uncertainty in observations. Compared to heuristic baselines, the proposed method achieves faster victim identification and area mapping while maintaining higher overall utility scores in the majority of cases within seventeen randomized single-robot scenarios, and produces comparable outputs to existing optimization-based solutions using significantly fewer computational resources. Moreover, the multi-robot experiments with different team compositions demonstrate that the method successfully allocates high-priority tasks to the most suitable robots in terms of their operational capabilities and task urgency, achieving faster mission completion compared to non-cooperative methods in all scenarios. These results confirm the potential of a fuzzy-MCDM-based, hierarchical control architecture to enhance autonomous behavior in robotic USaR teams under real-world operational constraints, while maintaining an abstraction layer for representing most types of heterogeneity present in the field of USaR robotics.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

Abbreviation	Definition
ACO	Ant Colony Optimization
DoF	Degrees of Freedom
EM-DAT	Emergency Events Database
FIS	Fuzzy Inference System
FLC	Fuzzy Logic Control
GA	Genetic Algorithm
GNN	Graph Neural Network
GNSS	Global Navigation Satellite System
IMU	Inertial Measurement Unit
LiDAR	Light Detection and Ranging
LTL	Linear Temporal Logic
MILP	Mixed-Integer Linear Programming
MPC	Model Predictive Control
MRTA	Multi-Robot Task Allocation
OP	Orienteering Problem
PI	Priority Index
PID	Proportional–Integral–Derivative
PSO	Particle Swarm Optimization
RADAR	Radio Detection and Ranging
RL	Reinforcement Learning
SaR	Search and Rescue
SLAM	Simultaneous Localization and Mapping
ToF	Time of Flight
TSP	Travelling Salesman Problem
UAV	Unmanned Aerial Vehicle
UGV	Unmanned Ground Vehicle
USaR	Urban Search and Rescue

1

Introduction

Disasters catastrophically disrupt communities and functions of society by damaging the environment, established structures, and humans, limiting the capabilities of local agencies for recovery. In the immediate aftermath of any disaster, the utmost important goal is gathering as much information as possible for focused efforts of extraction of the distressed individuals, commonly referred to as search and rescue (SaR). According to the most recent data provided in the EM-DAT database [1], which compiles standardized information on the reported disasters in literature, natural and man-made disasters claimed the lives of 1.4 million while affecting close to a billion people in total between 2000 and 2025. Most large-scale human and property loss occurs in urban settings, where prior information gathering and planning of a SaR mission are often difficult due to the high volatility and destruction that occurs within the environment, as well as the limited availability of time and resources [2]. In general, any type of SaR mission needs to consider how to efficiently allocate the resources at hand to the evolving situation by establishing a reliable and continuous operation, while maintaining effective search strategies within a limited time window.

Robotic platforms have long been considered tools to extend the capabilities of SaR efforts; however, their historical roles have been limited. Beginning with early deployments such as tethered unmanned ground vehicles (UGVs) in the World Trade Center collapse in 2001 [3], robots have mostly been proposed as teleoperated tools or semi-autonomous sensors. Since then, the diversity and capability of robotic platforms have expanded significantly, including the usage of unmanned aerial vehicles (UAVs), legged and tracked unmanned ground vehicles, and hybrid locomotion solutions adapted for specific capabilities. Most of the early research utilizing these types of robots has concentrated on enabling lower-level autonomy, solving problems such as local navigation and obstacle avoidance within bounded and well-defined environments. A more generalized description of lower-level autonomy can be achieved by formulating it as *path planning* problems, where either optimal traversal between given waypoints or reactive maneuvers to local changes are considered. The widely used approaches for the first type of path planning consider a graph-based description of the SaR environment and perform least-cost searches to reach a given goal. A more generalized version of such approaches considers the optimal ordering of multiple possible waypoints, commonly referred to as the traveling salesman problem (TSP), where an extension of it by considering a limited number of candidates to be selectable when budget constraints are present, is called the orienteering problem (OP) [4]. In contrast, reactive methods work by steering the robots within the vicinity of their current locations, considering cost functions that define the robot's goals. One of the more commonly used frameworks to enable this behavior is an optimization-based method called model predictive control (MPC), where the robots make predictions for future states within a limited look-ahead window to decide their next actions. MPC formulations are especially useful for navigating under dynamic and uncertain conditions where specific constraints need to be satisfied, making them suitable for SaR robots.

While these capabilities are crucial, they fall short when the robot is required to reason about *what* it should do next, *why* a specific action is valuable, or *how* to allocate its effort. In realistic SaR scenarios, robot actions must be guided not only by feasibility but by context-sensitive understanding of task relevance, expected utility, and collaboration potential. This necessitates autonomy at a higher

level, addressing decision-making under mission-level ambiguity and uncertainty, referred to as *task allocation* in this thesis. These demands for flexibility, context-aware reasoning, and adaptive task interpretation are only partially addressed by existing task allocation paradigms in the SaR literature. Many rely on fixed mission templates or explicit role hierarchies, limiting their ability to respond to evolving needs. The traditional approaches, such as auction and market-based methods, assume ad-hoc or static utility functions formulated via domain-specific heuristics, often failing when unexpected tasks or robot availabilities are introduced that were not considered during the system design. Conversely, learning based methods show promising adaptability; however, they require consistent training, which is not always possible or feasible within the SaR domain. These limitations motivate more human-like reasoning in how robot capabilities are matched to emerging tasks, an approach reflected in state-of-the-art human SaR coordination systems, which often use fuzzy decision-making to prioritize actions based on situational features [5]. Some more recent works in the robotic SaR literature incorporate such human expertise in formulating an autonomous robotic team, by integrating optimization-based control architectures into fuzzy evaluations of the robot actions [6]. Optimization-based autonomy is an emerging field within the SaR literature, particularly when combined with human knowledge and a structured method, such as MPC. However, meeting the demands of different types of SaR missions requires more than ad-hoc behavior strategies; it calls for a principled framework that can generalize across robot types, respond to newly emerging tasks, and reason over mission-level priorities in a structured, interpretable way. Moreover, the longer-term structure of the mission is mostly unexplored within the SaR domain, such as the need for energy-constrained robots to return for charging and resume operation as part of a continuous planning loop.

Given the heterogeneity of robot platforms, each differing in mobility, sensing, and endurance, no single robot type is ideally suited to meet the full spectrum of needs in generalized SaR operations. A scalable and adaptive approach is to deploy autonomous teams capable of flexibly assigning responsibilities and re-evaluating them throughout the mission. In this context, a *task* is not a fixed goal but a semantically meaningful opportunity for action, such as inspecting a collapsed structure, scanning a hotspot, or delivering aid, that emerges from incoming observations and changing mission priorities. Rigid or role-specific behavior design quickly falls short in such settings, as it assumes tasks are fully known in advance and tightly coupled to the assigned identity of each robot involved. Enabling robots to reason over dynamically evolving tasks and evaluate their relevance in light of partial knowledge and mission objectives is essential for true mission-level autonomy in robotic SaR teams. Moreover, such autonomy must operate continuously, not in isolated decision rounds, but as an ongoing process in which robots act, observe, recharge, and re-prioritize as the situation develops. This requires architectural support for persistent operation and adaptive behavior, extending beyond fixed sequences or one-shot task assignments. Multi-criteria decision-making (MCDM) provides a principled foundation for this need [7]. Rather than relying on single-objective optimization or handcrafted priorities, MCDM integrates multiple, often conflicting criteria, such as comparing target-oriented actions to coverage-oriented ones, into a unified utility representation. This allows robots to make modular, interpretable trade-offs that adapt to changing mission demands. While widely used in strategic decision systems, MCDM remains underexplored in robotic SaR, where its ability to integrate mission-level priorities and heterogeneous actions could directly address the core limitations of current task allocation paradigms.

1.1. Research Questions

Having highlighted the limitations of existing frameworks in autonomous SaR robotics, alongside the operational demands of SaR missions such as time sensitivity, mission continuity, and optimal resource allocation, this thesis sets out to design and evaluate a unified control architecture for multi-robot teams operating in environments with uncertain and evolving task demands. Central to this objective is the integration of observation-driven task emergence with fuzzy prioritization and MCDM, enabling robots to assess task relevance in a context-aware manner across heterogeneous capabilities. Building on this foundation, the research aims to develop a computationally tractable predictive planning approach that supports energy-constrained, continuous operation at the robot level, complemented by a hierarchical coordination mechanism that resolves task conflicts and maintains team-wide consistency in dynamically evolving mission contexts. In line with this overarching objective, the following research questions are formulated:

- How can dynamic prioritization of emergent and ambiguous tasks in SaR scenarios be improved using human-inspired fuzzy reasoning integrated with MCDM models of mission goals, compared to static or heuristic-based methods in terms of responsiveness, decision quality, and flexibility?
- How can predictive control derived from a generalized task allocation model be made tractable for real-time, energy-aware decision-making in autonomous systems using a mobile recharging agent with safe return constraints, and how does it compare to state-of-the-art approaches?
- How does integration of a global hierarchical layer to local controllers affect coordination, scalability, and conflict resolution in a heterogeneous multi-robot SaR team, and how does it improve consistency and performance in task allocation?

1.2. Research Contributions

Answering the research questions and the overall objectives, the main contributions of this work include:

1. Introducing a novel generalization of an abstract system model to account for a wide range of SaR robot types and their capabilities, with robots modeled through their available heterogeneous actions and the resulting environmental observations, decoupling robot roles from static capabilities. Tasks are treated as emergent, observation-dependent entities rather than fixed inputs to the SaR mission, supporting dynamically evolving priorities. To evaluate the suitability of each action, the thesis proposes a two-layer decision-making framework novel in its implementation to the SaR domain: first, fuzzy prioritization encodes human-interpretable reasoning under uncertainty, and second, a mission-level utility function is constructed through an MCDM architecture accounting for the time-sensitivity as a shared criterion for each task attempted by an action.
2. Formulating a nonlinear MPC architecture for each robot at the robot level to support sequential decision-making under dynamic task evolutions and limited foresight, with safe return constraints to enable planning according to the remaining energy of the robots. The model enables each robot to predict and optimize a future sequence of actions over a finite horizon, taking into account evolving action utilities through task completions and energy constraints. A novel two-stage linearization strategy is proposed for implementing the local nonlinear MPC formulation to reduce intractability and improve real-time performance. In the first stage, a subset of approximately independent candidate actions is selected using partial utility estimations and observation overlap filtering. In the second stage, the selected actions are scheduled using a mixed-integer linear programming (MILP) formulation inspired by the OP solutions in the literature, with time and energy constraints ensuring that the robots can always safely return for charging, thereby guaranteeing long-term mission continuity. At the global level, a conflict resolution controller extends the local MILP formulations by combining the local solutions to a globally optimal allocation.
3. Implementation and evaluation of the proposed methodology in a realistic urban search and rescue (USaR) scenario with evolving task sets and different types of robot models to demonstrate the effectiveness of the MILP-based controllers compared to other heuristic and state-of-the-art solutions that are compatible with the formalization proposed in this thesis. The USAR scenario criteria are set to find as many victims as possible and map the area in the shortest amount of time. The proposed methodology is compared with respect to these metrics, as well as task prioritization, victim health states, computational performance, and utility generation, in combination with qualitative behavioral analyses.

1.3. Thesis Structure

The thesis is structured as follows. First, Chapter 2 presents the literature in the SaR domain, while keeping the focus on USaR as it relates to the case study of this thesis. In Chapter 3, the specific methodology proposed in this thesis is given. The implementation details of the proposed methodology and the related case study simulation are given in Chapter 4. The results and related discussions of the case study simulations are presented in Chapter 5, where the thesis is concluded with future research suggestions in Chapter 6.

2

Literature Review

The structuring of the literature review begins by specifically defining the domain characteristics of search and rescue (SaR), particularly in the context of urban settings, to properly define the requirements of a robotic team designed to operate within this field. Then, existing robotic solutions are introduced by outlining the specific robot and actuator types, sensing capabilities, and control architectures proposed historically and in the current state-of-the-art. In general, the SaR operations are commonly divided into three categories [2]:

1. Urban search and rescue (USaR) refers to efforts aimed at locating an unknown number of stationary victims within a confined area that may be difficult to traverse and navigate due to the abrupt distortion caused by a disaster. The most ubiquitous and highly unpredictable urban disasters are earthquakes, where entrapment due to structural collapses necessitates USaR operations. Other similar disasters include blast effects due to terrorism or gas leaks, industrial accidents, avalanches, ground failures, subsidence, and tornadoes [8].
2. Wilderness SaR deals with open-ended search areas and is defined by the localization of several mobile victims. It is characterized by target or surveillance-oriented missions. The main environments of interest include forests, fields, caves, and mountains. The layout is not necessarily mapped beforehand and may include dynamic hazards and constrained areas [9].
3. Air-sea SaR is similar to the wilderness case in mission taxonomy; however, the environments are defined by their vast open areas without expecting many static or dynamic obstacles. The search is usually conducted via aerial support and satellite assistance to the SaR efforts on the field.

Urban disasters disproportionately affect a larger number of people and cause widespread destruction compared to other types of disasters. One of the most prevalent attributions to the increased casualty rate in urban scenarios is the severance of local functions on a large scale, which prevents structured SaR missions. There have been urban disasters that resulted in at least city-wide destruction, claiming the lives of thousands; the recent largest ones include Turkey 1999, India 2001, Iran 2003, Pakistan 2005, China 2008, Haiti 2010, Japan 2011, Nepal 2015, and Turkey 2023 earthquakes [10]. The common denominator for the USaR efforts in these disasters is that international help usually can not arrive in time to make significant contributions while local efforts are lacking due to the insufficient number of SaR teams and resources left in the aftermath of the destruction [10]. Considering the technical challenges introduced by the volatile environment and the time-sensitive nature of urban disasters, the focus of the thesis is selected in the USaR domain with an emphasis on large-scale earthquake response.

2.1. Domain Characteristics

In the event of a disaster within an urban setting, it is essential to assess the immediate environmental and operational considerations to enable effective USAR missions. For the purpose of this thesis, the most relevant environmental factors and constraints are investigated. Then, the tasks of USaR

teams and the current margins for improvement, with an emphasis on robotics implementations, are justified. Building on this framework, the qualitative parameters that can improve contemporary USaR approaches are reached from the required autonomy perspective to formulate the design considerations.

2.1.1. USaR Environment

There are various distinctly different disasters that may result in USaR scenarios with entrapment from structural collapses. Therefore, the first step is to identify the so-called hot zones and the associated hazards. As defined by Murphy [3], the USaR operations occur exclusively in the *hot zones*, where only the essential personnel are allowed. The *warm zones* extend around the rescue areas with operational and logistic stations, where the remaining areas are called *cold zones*. Boundaries between these areas can be volatile and setting them throughout the mission depending on how the situation evolves is an essential task [9]. Furthermore, the hot zones are generally exposed to weather conditions, and the rescue efforts may extend through the night when the necessary infrastructures for lighting are insufficient. The USaR teams must also remember that dynamic hazards may occur after initial collapses emanating from unstable debris, explosions from leaks, or rapidly accelerating fires [8]. The communication and localization systems may become unreliable in a hot zone, if not completely lacking, therefore establishing coordination and information flow on the field becomes a priority [9].

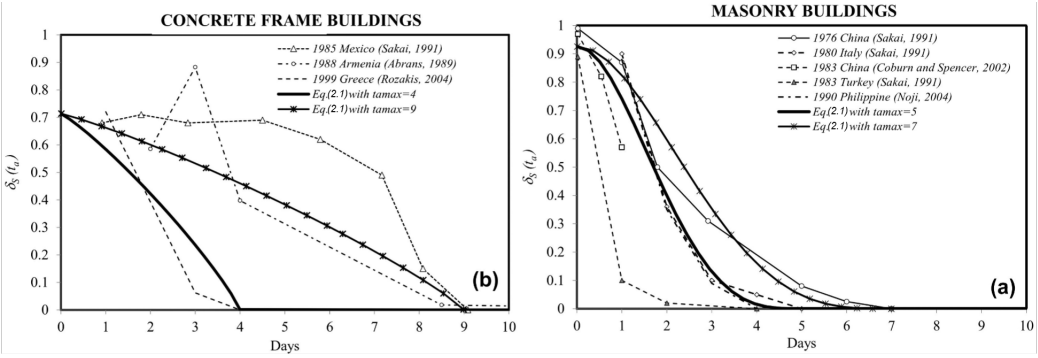


Figure 2.1: Survival likelihood with respect to the elapsed time after different types of structural collapses [11].

In general, confined spaces characterize the hot zones in disaster areas, where the regulations for safety create additional bottlenecks for rapid rescue efforts [3]. From a high-level perspective, the search must be conducted methodically before any rescue can be performed. As an example of utilizing the prior information from other disasters for establishing a prioritization model within USaR environments, Reinoso et al. [11] propose a mathematical model to predict the survival rates $\delta^{\mathsf{S}}(t_a)$ as a function of elapsed time t_a of different types of debris, as:

$$\delta^{\mathsf{S}}(t_a) = (1 - \delta F_0)(1 - F^{\mathsf{BE}}(t_a \ / \ t_a^{\mathsf{max}} \ |a, b)) \tag{2.1}$$

Where δF_0 denotes the factor of victims killed instantly depending on the structure type, t_a^{max} the maximum survival time, $F^{\mathsf{BE}}(.)$ the Beta cumulative distribution function, and a, b the related structure-dependent constants of the cumulative distribution. As shown in Figure 2.1, it is possible to model historical data from previous earthquakes using this formulation, informed by qualitative observations made during a mission, such as building types. Reinoso et al. further classify the following metrics that can be modeled and relevant to USaR mission goals:

- Expected occupancy rate at the time of the collapse.
- Qualitative vulnerability defined by the structural type (see Figure 2.1), local building codes, falling debris hazards around the structure, materials, and possible fire hazards both externally and within the infrastructure.
- Susceptibility to earthquake-induced secondary hazards such as tsunamis, subsidence, and landslides.
- Miscellaneous factors such as weather conditions and the efficiency of rescue and emergency plans in place.

From a more fine-grained perspective, USaR teams need to account for reduced visibility in the debris, environmental noise, secondary collapses, and inevitable deaths during or before the rescue; hence, they need to make quick decisions while ensuring personnel safety, stress, and fatigue management [3]. Combining all the environmental factors, the first 72-hour duration is generally the most important time interval for extended USaR missions to succeed [10].

2.1.2. Performance Factors

Disaster response in an earthquake area depends on many factors, a substantial number of which can be considered situational. However, the ultimate goal and, hence, metric for success is rescuing as many victims as possible within the minimum amount of time to ensure their survival and well-being. Therefore, one of the most important factors that affect USaR performance is overall situational awareness, including risk and safety assessment for the rescuers [8]. There are margins for improvement in the current methodologies listed by Statheropoulos et al [8], determined by surveying USaR personnel on the field. The initial analysis is conducted externally on a collapsed building through a thorough stability evaluation by specialized structural engineers. Unsafe buildings and uncertainties prolong these efforts before any detection of victims can be performed. Additionally, as new information is gathered, some buildings may need to be re-searched after debris removal, which can negatively affect the efficiency of USaR operations, as human resources and equipment are always limited. A case study surveying different spatial task allocation strategies for the USaR teams on the field under uncertainties [12] indicates that the most successful strategies are the ones that allocate resources for decreasing the uncertainties during the operation, where the most notable sources of uncertainty can be listed as:

- The number of injured people and the severity of their condition.
- Overall operation and individual task durations.
- Infrastructure priorities and routing within a partially unknown environment.
- Risk levels during rescue and agent fatigue.

Comparing the odds of USaR personnel with high and low situational awareness, respectively, for locating a victim, the former is nine times more likely to achieve success than the latter, depending on the mission time [13]. For the purpose of this thesis, the overarching performance factor in an earthquake scenario is therefore highlighted as improving situational awareness to enable informed decisions on the part of the rescuer. This improvement can be made efficiently by considering which uncertainty should be addressed first in terms of search priority. Compiled from expert interviews and earthquake response field data, the priority-index (PI) model [5] can be used to categorize parameters that define the likelihood of finding victims. Presented in Table 2.1, these categories are further divided into sub-categories that define respective degrees of severity in practice. Incorporating these decision parameters into the initial search task can provide the groundwork for improved situational awareness of rescue efforts and quantify the factors that affect search performance.

Rank	Prioritization of areas in USaR	Average importance weight (1 to 5)	Rank	Prioritization of areas in USaR	Average importance weight (1 to 5)
1	Destruction level of the building	4.60	7	Risk level in the damaged areas	3.10
2	Population density	4.57	8	Vulnerable population	2.30
3	Number of injured people	4.53	9	Access time	1.90
4	Severity level of injuries	4.50	10	Number of stories of buildings	1.77
5	Number of trapped people	4.37	11	Type of building	1.70
6	Occupancy level of buildings	3.37	12	Time of the day - weather conditions	≤ 1.50

Table 2.1: The average importance of area prioritization based on USaR case studies [5].

2.1.3. Design Considerations

The current areas for improvement, as explained in Sections 2.1.1 and 2.1.2, clearly necessitate the development of new tools for the USaR task to enable more possibilities in the field. Data on the operation of current earthquake response teams indicate that visual contact with victims and their localization is deemed to be of the highest importance; while improving the detection sensors and introducing ways of structural 3D mapping, combining communication with victims is pointed as the second and third most important requirements, respectively [14]. Furthermore, reviews based on previous disasters show that focusing on enriching topological navigation and mapping may be sufficient for a new tool to accomplish situational awareness to make enough difference during the mission [15].

Due to increased awareness requirements and the limited number of USaR personnel, autonomous robots offer valuable promise for the USaR toolbox. It is estimated that teleoperated solutions would require at least a 2:1 ratio of robots to operators on the field due to regulatory constraints; hence, the system must be designed to gather a meaningful amount of intelligence that could not have been achieved with human teams given the same mission duration [15]. To increase this ratio effectively, the need for more reliable autonomy becomes evident. Robots can ideally be used to remove debris and access risky or cluttered environments for retrieval and payload delivery; however, the autonomy constraints would be substantially high for safety purposes and thus not applicable in practice. Based on a European Union study on the inter-relation of tools in practice for USaR, modular systems are sought the most for robustness [16]. Furthermore, this work highlights the importance of end-user integration and well-defined behaviors during a USaR mission. Having said that, the systems must not attempt to do everything simultaneously due to overflowing requirements and ease of deployability [17]. For these reasons, the focus of this thesis is highlighted as aiding only the victim search rather than rescuing.

Robots can eliminate the performance bottlenecks associated with the human element, such as risk assessment, fatigue management, and resources allocated for the initial search mission [3], as well as improve the information flow. Navigational autonomy is a promising field of research since it allows for informed decision-making by constrained area survey, hot zone assessment, victim identification/localization, mitigation activities, and stay-behind monitoring, which are increasingly valuable tasks in an earthquake scenario [17]. The assumptions and requirements in the selected domain are summarized as follows, constituting the high-level design goals to improve situational awareness via autonomous robots:

(A) The USaR environment is assumed to be (partially) unknown, and the number of stationary entrapped victims within the hot zone boundaries is uncertain.

  (i) The system surveys the designated hot zone area while tracking the evolving situation within and around it rather than directly interacting with the rescue efforts.
  (ii) Fundamental features include mapping the area, locating victims, and continuously updating the current information on areas where USAR efforts cannot operate yet, while evaluating the risks involved.
  (iii) The considered scenarios do not involve sudden and drastic environmental changes. At the same time, the system design should be modular enough to enable the later incorporation of factors that are outside the scope of this thesis.

(B) The main selected metric of operation is ensuring the minimum mission time while finding as many victims as possible.

  (i) Tasks for this mission design include sensing the status of the victims while operating with navigational autonomy in the field and utilizing intelligent decision-making to determine the next course of action.
  (ii) Although sensor fusion is not prioritized and is assumed to be a black box for this thesis, perception methodologies must be detailed and modeled regarding their effect on the system.
  (iii) The level of allowed autonomy in USaR is an active discussion. Still, the system should be fully autonomous in lower-level control (following a trajectory, climbing stairs, and avoiding obstacles), whereas the higher-level decision-making autonomy should enable possible human input. The system's outputs and behavior should be human-interpretable and center the end-user in the design.

2.2. Robotics in USaR

Robots have been proposed and utilized in a USaR scenario for 20 years now, where some of the first versions were deployed after the World Trade Center disaster in 2001 for reaching to victims that otherwise would not be possible [18]. These initial attempts were all controlled by USAR personnel on the field outside the debris, where the relatively smaller ones were tethered for power and data streaming. In comparison, the bigger ones could operate wirelessly. Their primary purpose was to establish a visual and auditory connection with the victims, as well as the situational hazards inside the collapsed building [18]. Since then, the available computing power, control strategies, sensor quality, and communication networks have substantially increased, creating new opportunities for more intelligent and autonomous robots.

Since the working environment of USaR robots is unpredictable and unstructured, Liu et al. define the five key performance factors as survivability, mobility, sensor, communication, and operation [19]. Survivability is closely correlated with the reliability, durability, and adaptability of robots in volatile situations. These metrics directly translate into the ability of an intelligent system to succeed robustly in its task, even in changing environments. Mobility and, by extension, fault tolerance add to this argument by considering how an intelligent robot can stabilize itself and overcome a wide range of potential obstacles during a mission. Sensors are fundamental for gathering environmental information, not only to assess a victim's situation but also to avoid threats to themselves as well as to prevent causing further damage. Communication encapsulates the system's ability to gather and convey essential, time-sensitive information between victims and rescuers, where the operation combines all these metrics to determine the tasks the robot should perform during a USaR mission.

This section presents state-of-the-art robotics solutions, along with their respective strengths and weaknesses, for operating under the assumptions outlined in Section 2.1.3. The literature is structured in a modular approach, in the order of the types of robots and specialized actuators for USaR, sensors, perception, and finally, the approaches in the autonomous control of singular and teams of robots.

2.2.1. Robot and Actuator Types

(a) Hybrid wheeled-tracked locomotion [20].

(b) Hybrid legged-tracked locomotion [21].

Figure 2.2: Hybrid locomotion examples for a USaR UGV.

Following the successful field applications after the World Trade Center disaster, USaR robot designs with operators had gained traction with further integration in disasters such as the 2004 Japan earthquake, the 2005 USA Katrina hurricane, and again the 2011 Japan earthquake/tsunami [5]. Since the situational requirements can occur with a large variance in a USaR mission, the initial works mainly focus on different shapes and mobility systems for teleoperated ground robots with similar sensors and ranges [18]. The types of robots can be divided into two general categories: unmanned ground vehicles (UGVs) and unmanned aerial vehicles (UAVs). In the context of USaR, unmanned refers not to the control of the robots but to the designs that do not require any physical human presence on board; hence, it is not a notion of autonomy.

UGVs in USaR are tied to the faster localization of victims, resulting in shorter rescue times and a more reliable assessment of building damage compared to UAVs [22]. Additionally, Berns et al. sepa-

rate small and large UGVs functionally; a large UGV can be used for longer-range information collection close to the danger zones, while a small UGV can be used to enter tight places to look for victims to assess their status more in detail [22]. Hence, the operational goal is important for determining the physical architecture of the robots, on which task allocation, control, and perception strategies can be developed. For hot zone assessment and navigation, large UGVs are generally proposed with wheeled, tracked, or legged locomotion systems and their hybrid combinations [18][19]. The reasoning behind these choices can be pointed out as enabling generalized robust mobility options for the environmental uncertainties rather than specializing in certain use cases [3]. Especially in a USaR environment, certain features such as stairs, extremely uneven or unstable grounds, soft terrain, and dynamic obstacles are fundamentally more difficult to navigate [23]. The capabilities of an autonomous low-level controller, in essence, rely on the choice of fault-tolerance strategies such as tip-over correction and locomotion systems. Figure 2.3 presents performance comparison between the most common configurations for large UGVs, where some examples of the hybrid approaches are given in Figure 2.2. The comparison metrics are qualitatively presented, where the mobility in unstructured environments, energy efficiency, and overall speed of robots are compared across different locomotion types [21]. Although enabling more range through speed, wheeled robots are generally unsuitable for a USaR deployment when many irregularities within the debris fields are present. Combining tracks with either legs or wheels is preferable to using only one locomotion method, as it enables robustness while keeping control complexity and energy requirements moderately low. Specific variations within the same category may also lead to improvements in performance; for instance, the wheel-track configuration suggested in [20] outperforms other hybrid solutions for stair climbing. To account for the shortcomings of the locomotion systems, low-level autonomous controllers are proposed and implemented for various USaR robots [23].

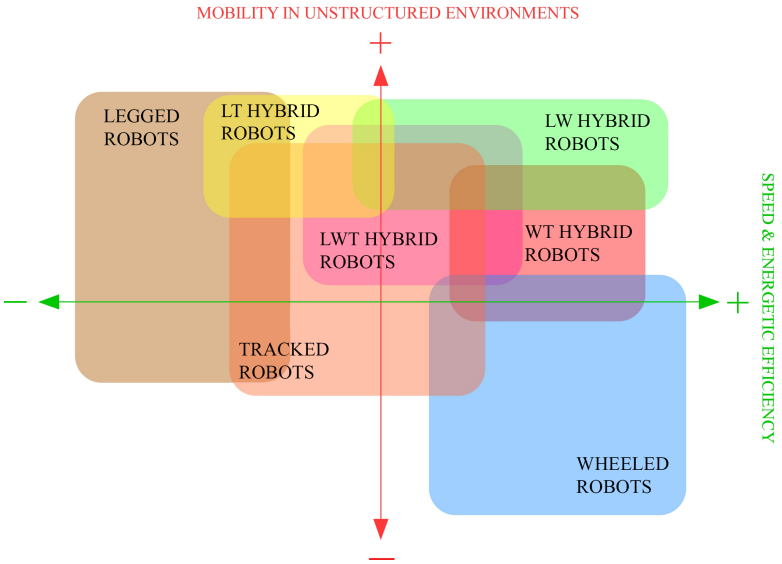


Figure 2.3: Qualitative locomotion comparison of large UGV types (for legged, tracked, wheeled, LT (legged-tracked), LW (legged-wheeled), WT (wheeled-tracked), and LWT (legged-wheeled-tracked) robots) [21].

In contrast to large UGVs, small UGVs are designed to crawl under the debris and navigate within close range of the victims to establish direct information and communication streaming [22]. In addition to the general locomotion types available for large UGVs, the functional purpose of small UGVs enables more specialized designs for close-quarter environments. Soft robotics and serialized tracked segments to slide into the cracks in the rubble are proposed uniquely for USaR purposes [18]. Furthermore, state-of-the-art snake robots are configured with passive wheels or interlocking motorized joints to enable high degrees of freedom (DoF) motion necessary for their operational goal [24]. The trade-offs, however, can be noted as reduced mission duration and considerably increased control complexity, which are inversely proportional to the gain in mobility. Nevertheless, snake-type locomotion can be effective for locating victims if deployed close to the debris site that is to be searched and equipped with the necessary tools for this task. Similarly to the hybrid locomotion of large UGVs, modular and shape-shifting mechanisms are also proposed for small UGVs for a USaR mission. One of the

initial such designs is called a variable geometry tracked vehicle because the shape of the robot can be altered during operation [18]. The initial shape of this robot resembles a crawler with tracks. When in a raised configuration, the tracks assume the shape of a triangle, allowing the design to conform to the terrain. An improvement on this strategy is suggested by adding modular connections between the tracked segments of the robot for docking and separating according to the needs of the terrain [25].

Some unique approaches for USaR robots consider different types of actuators in addition to the presented locomotion strategies. Han et al. [26] propose a snake robot with a gripper module for digging and interacting with the rubble on the way to potential victims, equipped with sensors and a camera at the fingertips. Integration of snake-like actuators into the current USaR toolbox is explored in the literature, with designs such as connecting them to other UGVs as sensory limbs [18], or a teleoperated deployment unit containing a specialized snake robot attached to USaR dogs [27]. This idea is further conceptualized as a shape-shifting robot carrying a smaller snake-like peripheral that can detach for a USaR scenario in the early stages of the field [3]. Teleoperated UGV and snake robot hybrids are successfully implemented and field tested by allowing the operator to detach and control using the main robot as a connection hub [28]. A similar design is proposed for USaR operations by configuring the snake as a detachable limb consisting of interlocked gears, which the main UGV can use to interact with the rubble on demand or extend it with a cord as a standalone unit [29]. Utilizing UAVs instead can eliminate the inherent mobility constraints emerging from UGVs. With UAV technologies on the rise, they have been used in USaR environments in two primary modes: on-site and logistic chain [30]. On-site UAVs operate in the hot zone, mostly for visual reconnaissance over a wide area. Functionalities of mapping for risk assessment, victim search, target observation, and surveillance are the main benefits of incorporating teleoperated UAVs during a mission [31]. Furthermore, the advancements in micro UAVs substantiate the arguments for their utilization by entering confined places, ventilation ducts, and underground sewer channels [30]. Logistic chain UAVs, in contrast, operate in warm and cold zones for cargo transportation. In addition to increased mobility, UAVs offer vertical exploration as a means to enhance situational awareness in disaster areas. Especially partially collapsed buildings can usually only be explored from above due to the unstable structure, making UAVs invaluable to the USaR task [3].

Although UAVs are superior for navigating the environment quickly, the robots must be close and within the debris to detect victims under collapsed structures. To that end, smaller UAVs are more desirable for exploration at the cost of short mission duration and limited computing capabilities [2]. In this sense, UGVs enable closer sensing and better assessment of victim status, while UAVs can eliminate the mobility problem on unstructured terrain and, therefore, increase the range of operation. Several designs are proposed to combine these strengths of UGVs and UAVs in the form of air-ground hybrid locomotion. A differential drive cylindrical robot that can rotate itself on the side by expanding its propellers located horizontally inside the chassis for traversing steep terrain is proposed and demonstrated to be feasible [32]. The downside of this approach is that the robot needs to change its orientation between transitions, and the dynamic components that expand into propellers are prone to failure. Premachandra et al. solve this problem by keeping the propellers stationary in the chassis and using them to assist lateral movement on tough terrain instead [33]. In this work, propellers attached to a suspension system are used to fly over obstacles and allow for quick maneuvers in support of the separate ground locomotion system. Finally, a ground-air USaR robot with a victim detection and positioning system and the capability to carry cargo is proposed [34]. Still, without compromising on reducing the sensory capabilities, these solutions suffer from substantial power requirements to support flight.

2.2.2. Sensing, Perception, and Mapping

Sensing methodologies are essential for USaR missions, allowing human and robotic teams to detect and assess victims, establish structural analysis, evaluate risks, and achieve overall situational awareness. The traditional search in an urban disaster zone without any technological tools can be performed by visual inspection or knocking into the debris and aiming to detect a response from underneath it [8]. Unfortunately, without having multiple points of view, visual obstruction due to dust and rubble may prohibit effective victim localization. In contrast, weak or unconscious victims are impossible to detect with the latter method. Contemporary USaR efforts include various sensory tools to overcome the issues

of the traditional methods.

		Feature	Field of View	Resolution	Robustness	Size	Victim Detection
Ultrasonic		distance	-	-	-	++	- -
LiDAR	**Rotating**	distance	++	++	+	o	-
	Solid State	distance	+	+	+	+	-
Camera	**Visible**	vision / distance	++	++	++	++	++
	Infrared	vision / heat	++	+	o	++	++
	ToF	distance	+	o	++	++	o
Microphone		sound	o	-	- -	o	+
Thermopile		heat	-	- -	-	++	+
Pyroelectric		heat	- -	-	o	++	++
RADAR		distance	+	o	-	-	- -
Chemical		metabolic byproducts	- -	-	o	o	++
Wireless Sniffer		WiFi	+	+	+	+	- -

Table 2.2: Sensor comparison for USaR (- -: very low, -: low, o: medium, +: high, ++: very high) [35][36].

The sensory tools for USaR can be listed under *penetrating* and *non-penetrating* sensors. Non-penetrating sensors only measure within the direct line of sight of the measured features, while obstructions in the field of view do not constrain penetrating sensors. The non-penetrating sensors include ultrasonic sensors, rotating light detection and ranging (LiDAR), solid-state LiDAR, and cameras [35]. Ultrasonic sensors use the time of flight (ToF) of generated sound waves reflecting from surfaces and are relatively unaffected by adverse environmental conditions. Disadvantages include noise due to bouncing, resolution, and blind zones in the field of view. Conversely, LiDARs achieve ToF measurements through pulsed light emitted from a laser diode, where the generated 3D data points are called point clouds. Although LiDARs also suffer from the same problems as ultrasound sensors, their accuracy and precision have been significantly improved. Furthermore, the ability to measure via individual rays enables different approaches to the LiDAR system, primarily through rotating a LiDAR array or generating dense rays using a solid-state setup, albeit at the cost of a more limited field of view. The final and most widely adopted non-penetrating sensors are cameras of three overarching types: visual, infrared, and ToF. Visual cameras capture between 400 and 780 nm light, similar to the human eye, with higher potential resolution. Additionally, multiple visual cameras are used to perceive depth and construct 3D features. Variations in lighting and weather conditions limit the use of visual cameras; hence, they are mostly utilized in combination with other sensors. Infrared cameras measure between 780 and 1000 nm wavelengths, mostly utilized in the near-infrared spectrum and as a replacement or in support of visual cameras. At the expense of resolution, infrared cameras are not susceptible to lighting conditions and can measure thermal bodies. Finally, ToF cameras are active sensors that use near-infrared light pulses to construct a 3D representation of the environment by measuring the phase difference between the emitted and received signals.

Penetrating sensors for USaR include microphones, thermopiles, pyroelectric sensors, radio detection and ranging (RADAR), chemical sensors, and wireless mobile phone sniffers [2] [36]. Microphones mimic the traditional way of finding humans, often coupled with actuators to knock into the debris. Directional microphones are used to process the location of potential victims since the processing of audio data is generally noisy. Thermopiles and pyroelectric sensors measure heat signatures, where

the former is used to detect the absolute temperature of a point, and the latter is used to detect thermal gradients. RADARs use radio waves for long-distance motion measurement, unaffected by adverse conditions. The drawbacks are the resolution and power requirements. Chemical sensors are mainly used to detect CO_2 for detecting the breathing cycles of victims and SpO_2 for blood oxygen measurements. Generally, they require close proximity to have reliable estimations of victims and are heavily dependent on conditions such as humidity, temperature, and dust. Having said that, there are recent developments in chemical sensor arrays specifically for USaR victim detection by also measuring the trace chemical changes in the environment related to human metabolism, making chemical sensing feasible for finding victims inside entrapment [37]. Finally, wireless sniffing of mobile phone WiFi networks has been suggested and tested to detect potential victims to narrow down the potential area of measurements for other accompanying sensors [2].

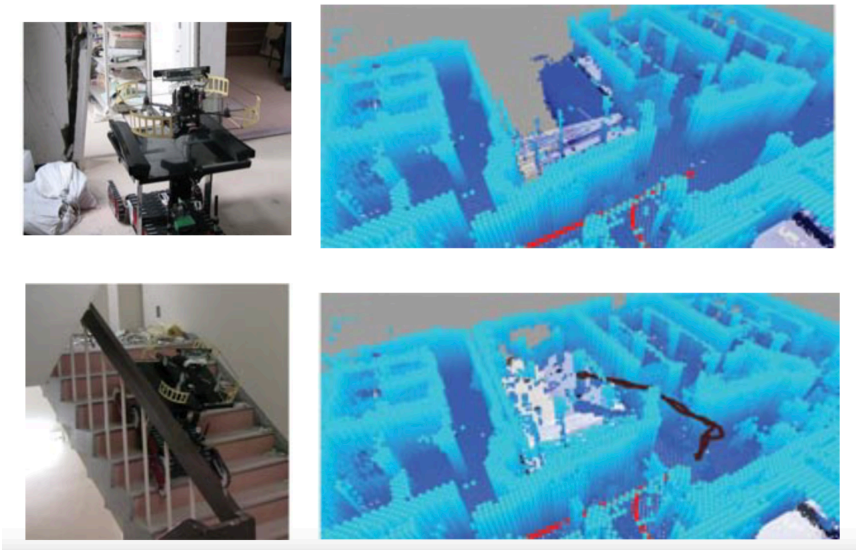


Figure 2.4: UGV and UAV team for mapping a damaged building. On the left, UGV carrying the UAV is given. On the right, mapping with only UGV (top) and combined mapping (bottom) with followed trajectories are given [38].

USaR necessitates specialized perception methodologies to process and draw meaning from the sensor measurements. Queralta et al. examine such methods under active and passive perception [39]. Active perception requires the robot to interact with the environment to obtain measurements, such as the signals emitted by LiDARs and RADARs. Passive perception does not require such interactions, and the most mature method is the use of cameras. Camera perception mainly relies on neural networks, where the input data is semantically segmented as a basis of object detection [39]. Semantic segmentation is often computationally costly and requires large datasets for accuracy; however, in recent years, both aspects have been significantly improved. Infrared and visual camera data can be processed to detect life signs in enclosed places [40]. Coupled with stereo vision, infrared fusion is further developed for USaR scenarios with relative baseline matching to detect humans with an accuracy of around 85%, depending on the environmental factors [41]. In general, multiple sensors, as listed in Table 2.2 are fused together to perceive varying features with specific combinations, called *multi-modal information*. The multi-modal information fusion can be achieved with different sensors on the same robot or distributively between the measurements of multiple robots [39]. Borges et al. present the terrain traversability perception available to ground vehicles using cameras, LiDARs, and RADARs, where cost-efficient methods exist for most robot types [42]. To achieve navigational autonomy on both lower and higher levels, terrain sensing is integral to USaR robots. Furthermore, specialized terrain analysis is imperative for first classifying damaged building areas and then quantifying the collapse intensity, expected structural damage, and finally, the life vulnerability estimates of the perceived environmental segments proposed by Reinoso et al. [11]. Pollino et al. provide a LiDAR and camera-based fusion algorithm to detect rubble features for environmental assessment using spectral analysis and machine learning tools, complementing the theoretical framework Reinoso et al. suggest [43]. Frameworks for integrating digital perception strategies into the USaR mission, considering structural triage methods

such as estimating the potential number of victims, access times, and risk metrics to the rescuers, are also suggested for manual USaR missions [44].

The final component for environmental awareness in a USaR setting is mapping. Mapping the disaster area during victim search is essential, as the landscape is expected to be different. Locating points of interest with respect to the traversable area increases search efficiency. The most ubiquitous method of mapping in robotics is called Simultaneous Localization and Mapping (SLAM), where the robots are localized within the unknown environment during mapping, and the map assists in the localization correction in the next steps [23]. There are many SLAM algorithms using this basic strategy for mapping unknown environments; however, in general, they are separated into 2D and 3D SLAM categories. Picard et al. provide a comprehensive survey for 3D real-time SLAM algorithms for embedded systems, which are the most relevant approaches for USaR robots [45]. These algorithms typically utilize camera and LiDAR-based perception algorithms to detect landmarks and approximate the environment as a semantic network. The mapping strategy may range from city-sized points of interest-based representation to high-fidelity, close counters 3D mapping inside the rubble, depending on the capabilities of the autonomous robot. In general, higher fidelity SLAM algorithms trade computational power in favor of resolution, where robust SLAM approaches are suggested using simpler sensors for small UGVs in a USaR mission [46].

For USaR robotics, SLAM approaches take more specific forms to satisfy the navigational autonomy requirements. UAVs are generally utilized to take pictures and videos, and using them offline for 3D reconstruction of disaster zones via structure-from-motion techniques has been suggested [47]. Online and faster methods for aerial 3D mapping of the USAR area have been surveyed by Verykokou et al., where pipelines for integrating autonomous pathing are discussed [14]. Zhang et al. propose a multi-layered mapping framework for detecting features such as terrain roughness, slope, elevation, and step height to build maps for ground robots using UAVs to efficiently explore a USaR area [48]. Similar strategies of combining UAVs and UGVs for collaborative mapping exist; for instance, a team of UAVs is used to extract a 2.5D occupancy grid map for a humanoid search robot to traverse a USaR area, detecting victims [49]. Qin et al. propose a collaborative perception scheme in which both types of robots can perform mapping individually, with UGVs creating a coarse LiDAR-based map and UAVs creating a finer map using cameras. Both types of maps are transferable between each other; specifically detected features enable heterogeneous systems to utilize each other's maps with respect to their own representation of the environment. To map indoor and enclosed areas, Michael et al. suggest teleoperated robots in an earthquake-damaged building where the UGV carries the UAV on a launching pad attached on top until the location of interest is reached [38]. The UAV can autonomously take off and land, while navigation is performed semi-autonomously with the assistance of teleoperation. Shown in Figure 2.4, the sensor readings are merged to generate a 3D map and its manifold into 2D features during operation. Finally, Azpurua et al. propose a mapping framework based on the Riemannian manifold from 3D SLAM data to construct a mesh representation of a confined environment and then extract traversable faces as the final map [50]. A more recent work specifically for SaR scenarios introduced a vision-based mapping framework that combines aerial and ground robots to enhance scene understanding in complex environments [51]. The system integrates video feeds from both robot types to perform three core tasks: detecting victims, tracking their movement, and mapping terrain elevation. By utilizing state-of-the-art computer vision approaches, their proposed framework estimates 2D keypoints of human bodies to reconstruct partial poses, even when certain body parts are occluded, by matching detections from different viewpoints. Victim tracking is modeled as a state estimation problem, where fused detections from both robots are processed through a Kalman filter to estimate the victim's trajectory over time. Additionally, terrain elevation is derived through homography-based projection of visual data, allowing the system to infer object placement and ground layout for safer navigation. The approach was further validated in real-world experiments, demonstrating its effectiveness in dynamic and partially occluded conditions, which highlights the potential of coordinated multi-robot vision in heterogeneous teams for addressing key perception challenges outlined in the SaR literature.

2.2.3. Control Systems

The environmental conditions in USAR hot zones are generally extreme and noisy, as outlined in Section 2.2.1, specialized robot types are often recommended. The control scheme for most of these solutions had traditionally been via teleoperation [18]. Most control architectures for teleoperated robots focus on increasing stability in performing certain actions, such as traversing stairs, uneven terrain, self-correction, and perception stabilization [23]. In the literature, low-level controllers exist for various types of actuators, which are mature enough for both USAR robotics and robotics in general. For the sake of scrutiny, such low-level controllers are not investigated in this thesis. Instead, the main focus is on higher-level control structures related to the increased autonomy requirements of USaR robots, which encapsulate the intelligent decision-making and efficient task-completion capabilities of state-of-the-art solutions. Furthermore, the relevant literature for the assumptions and goals outlined in Section 2.1.3 is found to mainly focus on multi-robot solutions or systems that can be extrapolated as such; hence, actuator-specific single-robot controllers are not investigated in this section.

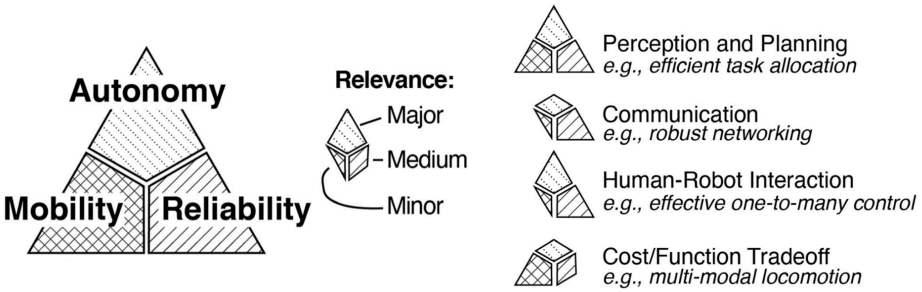


Figure 2.5: Relevance of functionality metrics for robots in USaR [52].

Regarding high-level control strategies, Drew defines the functionality metric trade-offs of robot teams with respect to mobility, autonomy, and reliability in their survey of multi-robot applications in USaR [52]. As shown in Figure 2.5, effective perception and planning are highlighted as the optimal combination of these metrics, encapsulating features such as cooperative mapping, formation, and task allocation. Locomotion specialization, mission duration, and environmental manipulation are categorized under cost-function trade-offs, which are essential for the design of individual robots initially and for using the teaming approach to account for the shortcomings of respective design choices. Communication link reliability and scalability are important constraints for the success of robot teams on the field, regardless of the level of autonomy. Finally, human-robot interaction considers the end-user of these approaches, with examples including single-operator control of the team and supervised autonomy. Perception and planning autonomy are estimated to be more important than other aspects of the current research in USaR robotics, with two main components: Path planning and task allocation. In the context of this thesis, path planning refers to the specific trajectory a robot is planned to take to reach a selected goal position. This strategy may be local and potentially include the vehicle dynamics, or it may be global, involving the identification of feasible waypoints towards the goal while avoiding obstacles and collisions. Task allocation, in its simplest form, involves generating the goal points for the path planners and determining the actions to be taken for each robot. Task allocation is more abstract in the sense that simply traversing to a goal location at a given time can be a task on its own, or the task structure might be more complex considering area coverage, ensuring communication pipelines, searching a specific area, and interacting with the environment in a cognitive way to fulfill semantic goals [39].

Another relevant approach to investigating the literature is to categorize the multi-robot target detection problem in a more general manner. Robin and Lacroix propose a taxonomy of controllers for autonomous multi-robot teams in terms of how the overall mission is formulated [53]. In their proposal, the control architecture types are split under *target detection* and *target tracking*, denoting whether the targets are previously known or not, respectively. In the specific context of this thesis, a similar categorization is also referred to as *coverage-oriented* and *target-oriented* SaR where the former assumes an unknown environment and thus aims to improve the exploration coverage, while the latter

approaches consider prior knowledge to be utilized, enabling strategies that can directly attempt to find mission targets. When *mobile search* is considered, as in the case of autonomous robots in a USaR environment, the search mission can be separated with respect to any guarantees that can be made for finding the victims. If the area is known in advance, the search can be defined as a capture mission, in which worst-case duration can be shown to be bounded and, therefore, optimized. In a more realistic scenario where the area is partially known and a probability distribution can be formulated in the search, the mission is regarded as a probabilistic search. When this task is cyclic, as it is often necessary to revisit previously mapped areas for victim assessment and surveillance, it is referred to as *patrolling* , considering the statistical performance over time between two visits to the same point. In the case of no guarantees being made for victim detection, the task is regarded as *hunting*. For the purposes of this thesis, as laid out under Section 2.1.3, probabilistic search formulation is considered, with which the literature on path planning and task allocation is investigated.

Path Planning

Path planning refers to the strategy employed by robot controllers to ensure safe, feasible, and efficient traversal between specified waypoints. In the work of Ibanez et al., a comprehensive taxonomy of state-of-the-art path planners is constructed [54]. The categories are laid out on the axis of local and global planners, where local planners usually implement lower-level obstacle avoidance between consecutive waypoints and generally execute faster. In contrast, global planners require more information from the environment, which is generally computationally expensive, to construct a better approximation of an optimal look-ahead pathing strategy between multiple given and calculated waypoints. For both categories, a path planner must be provided with a representation of the environment in which to operate. If the path planners need to optimize for additional criteria as well, such as the energy and time of traversal, modeling the mechanical interaction with the environment surface also becomes relevant. Building upon the sensors and mapping capabilities of USaR robots as described in Section 2.2.2, algorithm-specific environmental representations are utilized for most control architectures. These include cell decompositions by tesselation using triangles, hexagons, squares, or irregular-sized grids denoting regions of interest at different fidelity levels [54]. Instead of superimposing sensed features into distinct regions, another way of representing the environment is through map construction. Roadmaps are usually represented by graphs, where nodes represent a robot state and edges represent the transitions. The methods for roadmaps include Voronoi decomposition, visibility graphs, and state-lattice graphs [39][54]. In terms of robot-surface interaction modeling, the relevant information for path planners is generally in the form of stability, slippage, time, and energy. With the details outside the scope of this thesis, further information can be found in the literature cited in Section 2.2.1.

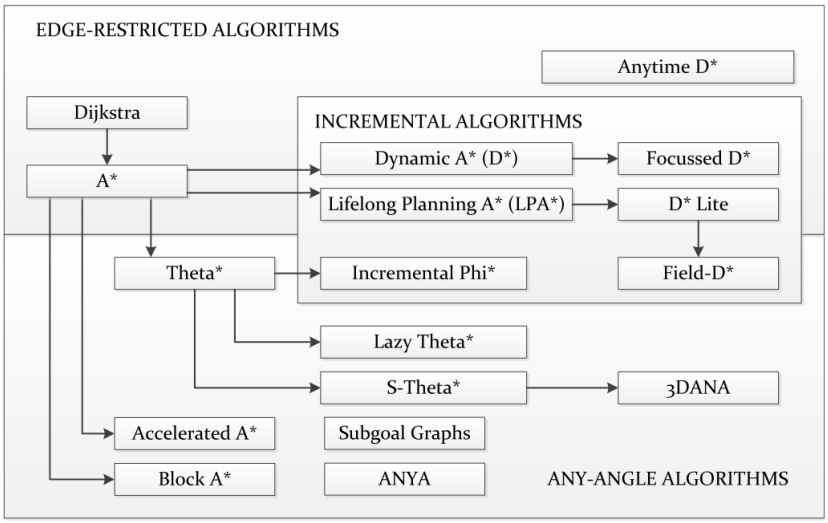


Figure 2.6: Dijkstra based graph search algorithms overview. [54].

Some of the most prevalent methods in global path planning are graph search methods. These methods discretize the c-space, also called the working space of robots, in the form of generalized

graphs. Denoting the waypoints as goal states, graph search algorithms aim to find the optimal node order between them. Regardless of whether the sampled environment is based on cell decomposition or roadmaps, a graph search can be conducted in an *edge-restricted* or *any-angle* fashion. Edge-restricted algorithms strictly use the edges on the initially given graphs to conduct the search, meaning a state transition can only happen to its immediate neighbors. Alternatively, any-angle algorithms are not constrained by the initially provided edges and can construct paths consisting of states that may not be neighboring each other. The majority of both types of algorithms are developed based on the basic idea of Dijkstra's shortest path update rule. Briefly, Dijkstra's method considers a propagating search from a source in which, for each reached node u and an unreached neighbor v with an edge weight $w(u, v)$, the following rule is applied at step k to update the cost of $d_{k+1}(v)$:

$$d_{k+1}(v) = \begin{cases} d_k(u) + w(u,v) & d_k(v) > d_k(u) + w(u,v) \\ d_k(v) & \text{otherwise} \end{cases} \tag{2.2}$$

This approach is majorly improved using domain-specific heuristics, which provide the basis of A* algorithms. The heuristics must be admissible, meaning that within the mathematical formulation of the graph, they never overestimate the true cost of ensuring optimality. The basic A* algorithm improves the rule given in Equation (2.2) by adding an admissible heuristic function to the updated cost, usually in the form of Euclidean or Manhattan norm for spatial graphs. Dynamic A*, also known as D*, implements a similar strategy but allows the previous computations to serve as incremental steps, enabling fast re-planning. Other improvements on these edge-restricted algorithms exist, such as D* Lite, Lifelong Planning A*, and Hybrid A*, on which further details can be found in the work of Gonzalez et al. [55]. Any-angle algorithms also draw on an A* formulation at their core, relying on graphs with defined norms to enable search without relying on only the edges. Such algorithms include Theta*, Field D*, and Accelerated A*, the taxonomy of which is given in Figure 2.6 , and further information is available in the work of Ibanez et al. [54]. The optimality of graph search methods depends on the heuristics they use and is ultimately influenced by the approximation of the environment. Instead, in the most general case, partial derivative equations of the search space on which the path planning is done can be solved with respect to a source node, called a boundary, in the form of Eikonal equations. Eikonal equations model how a wave propagates from a boundary in a given space, with a formulation similar to Dijkstra's in that both extend the search from the source and update the states reached according to the minimum cost. When used for path planning, gradient descent is the most prevalent approach to calculate the optimal path to a waypoint. The Fast Marching Method (FMM) employs this approach to generate smooth and continuous optimal paths. Gomez et al. list a family of algorithms based on Eikonal solvers and improved versions of the original FMM [56]. These methods have been further integrated with heuristics to handle anisotropic environment models, dynamic re-planning, and non-linear constraints, such as the Fast Iterative Method, Fast Sweeping Method, and Ordered Upwind Method [54].

When the environmental model and path planning objectives are in higher dimensions, sampling-based methods are widely used to approximate the global solution asymptotically. In general, sampling-based path planning methods are either in the form of single or multiple queries, meaning only one goal state is present in the former, while intermediary goals are allowed in the latter [54]. The overarching approach in many single-query methods in this category stems from the Rapidly Random Tree (RRT) algorithm, where the search space tree is dynamically created by sampling sequential branches until a feasible path to the goal state is found. Since the tree is built incrementally, further iterations can be performed to increase the optimality of the found path. Different improvements on this approach have been proposed, such as adding heuristics to speed up the convergence rate, similar to the A* family of algorithms. When expanded to the multiple-query problem, the most prominent method is the Probabilistic Roadmap Method, where samples in the search state are generated from the given goal points, and trees are expanded from each sample to create a continuous path. Finally, Fast Marching Tree algorithms leverage RRT to create samples similar to those of the FMM approaches detailed previously. More information on single- and multiple-query methods, as listed here, can be found in the review article by Gonzalez et al. [55].

Aside from the single- and multiple-query sampling methods, metaheuristic optimization techniques also aim to find near-optimal solutions for complex problems, often utilizing state space sampling for exploration. These methods yield suboptimal results, requiring problem-specific tuning to achieve better

performance; however, they offer scalable and flexible general solutions in exchange. The most commonly used examples for path planning include Genetic Algorithms (GAs), Simulated Annealing, Ant Colony Optimization (ACO), and Particle Swarm Optimization (PSO) [57]. Most such algorithms take inspiration from mathematically modeled natural phenomena. GAs sample the state space randomly and represent them as a population, of which, at each iteration, the best candidates are combined, similar to the cross-over of genes. Random mutations are introduced between each generation and the offspring to explore the state space more effectively, thereby simulating evolution during optimization. The ACO algorithm models the current iteration of solutions as ants operating in the search space, where the ideal solution is marked by artificial pheromones that attract the subsequent iterations. The pheromones are usually temporal to enable robustness in the search space in the presence of disturbances, since having timeouts on the established pheromones enables the correction of erroneous decisions as the overall system information increases with time, and individuals also take random actions for exploration. PSO, in contrast, models how a flock of animals behaves when searching for food. It is similar to genetic optimization techniques in that the population of candidates is used through iterations; however, in the case of PSO, the population is not altered but rather regarded as particles moving in the state space. An artificial speed function determines the movement of the particles, and a distance measure between particles in the update function models their spreading. Additional heuristics can be added specific to the problem to the generic update function. Simulated Annealing is based on the annealing process of metals, where the method cools down the solution space by considering worse solutions first to explore, and then reduces randomness to refine the result. More metaheuristic global optimization techniques exist, similar to the ones included here, for which the review paper of Zafar et al. can be referred to in detail in [57].

In the context of USAR path planning, Li et al. propose a hybrid method that utilizes a team of UAVs and UGVs, employing GAs for path planning. In their work, the non-autonomous UAVs are used strictly for providing aerial images to the UGVs, where with each updated map, the existing population of the algorithm is utilized to re-plan the paths [58]. Furthermore, considering their feature extraction methodology, the genetic algorithm results are locally optimized continuously to ensure obstacle-free and smooth pathing. In the work of Alenazi et al., a new metaheuristic method called the Spiral Dynamic Optimization algorithm is suggested by combining the basic Probabilistic Roadmap approach with metaheuristics modeling the logarithmic spirals found in nature, such as currents, to guide the expansion of the sampled state space where the authors argue that the convergence is improved compared to other methods when a cluttered indoor space is considered with hazardous obstacles [59]. When the continuous path planning of a swarm of robots is considered, the original PSO algorithm is enhanced by Couceiro et al. called Robotic PSO [60]. The Robotic PSO algorithm characterizes each robot with its position and performance, where the main difference lies in how the obstacle avoidance is formulated into the update function of the PSO approach. Furthermore, they incorporate social inclusion and exclusion mechanisms, whereby better-performing robots tend to stay together, while poor performers are directed towards unknown areas through the use of rewards and punishments, utilizing an update function called Darwinian Robotic PSO. They conclude that for search missions, the improved PSO approaches outperform other metaheuristic methods.

While the path planning approaches outlined so far consider the global optimization problem, artificial intelligence-based approaches can be used for both local and global path planning. The most generic approaches in this category are neural networks and fuzzy controllers [57]. Neural networks essentially model how neurons interconnect and fire with respect to input signals in nature to control the actions of the robots. Depending on how the input-output relationship is configured, such systems are proposed as a base for local and global path planners. Alternatively, fuzzy control systems operate based on if-then rules provided by human experts on specific control inputs to generate robust robot actions. This provides a powerful tool for constructing well-defined behaviors on planned paths, which can be reactive to immediate inputs for the local case. To deal with more complex environments or if the immediate behavior of a robot can not be sufficiently formulated beforehand, deep learning approaches are proposed for USaR robots. Hu et al. use a Voronoi-based cell decomposition for individual robots, where the path planning to a select number of sampled potential paths is done through deep reinforcement learning [61]. In this work, the deep-learning model guides the robot around obstacles where the observation and action spaces are configured with respect to the velocity, time, and targets of individual robots. Although not an artificial intelligence method, another generic technique that can be used

for both local and global planning is Model Predictive Control (MPC). The generalized MPC approach minimizes the cost function of a system using a dynamic model by predicting the feature behavior to optimize the control actions along a prediction horizon. An example of this method used for local path planning of a UGV in the rough terrain of a USaR area is given by Shin et al. [62]. They model the non-holonomic kinematics of a UGV with passivity constraints, allowing for improved energy preservation and resulting in faster and more efficient movements in the path planner. The optimization along the horizon is performed by employing a PSO solver. Scheffe et al. propose a more generalized approach for nonlinear and nonconvex MPC problems for robotic path planning, where the robots are modeled as finite-state automatons and their possible trajectories from a given state are assumed to be feasible motion primitives that can be sequentially appended to each other [63]. Then, an MPC problem is formulated via the stacked control inputs and the accumulated penalty of the states and inputs along the prediction horizon. They use the finite state automaton to describe the discretized system, where a receding horizon graph search is employed for the optimization. The graph search expands from the initial state until the accepting state is reached, if applicable, and a lazy look-ahead prediction of the next accumulated costs is applied to decide which branches of the graph to explore in the successive stages.

Most reactive computing methods model the environment as a field to guide the robot. These methods include artificial potential fields, vector field histograms, and virtual impedance algorithms [54][57]. Artificial potential fields model the obstacles as sources of repulsive forces acting on the robots with respect to distance, and the goal locations do the same by acting as attractive forces. The following dynamic action of the robot is calculated by summing all artificial forces acting on it, constrained by its kinematics. Improvements to this idea have been proposed, especially for overcoming the issue of getting stuck in local minima, such as escape heuristics, genetic methods, particle swarm look-ahead algorithms, and via the usage of scalar electrostatic force modeling instead of the sum of directional forces [54]. A relevant version of potential fields, especially for USaR applications, has been proposed by Park et al., called the advanced fuzzy potential field method [64]. This work uses a Takagi-Sugeno model that ties control actions to the repulsive forces with fuzzy rules. They make abstractions for any number of inputs that may be relevant from expert perspectives; in their case, the measured distances d_n, the angular difference between the obstacle and robot trajectory ψ_n, and the angular difference between the obstacle and the goal ϕ_n are modeled by membership functions. Such a framework enables scalability, where metrics related to USaR goals can easily be incorporated into the controller. The drawback of their method is that all simultaneous measurements are regarded as independent, distributed fuzzy systems, which results in an exponential number of possible rules, and the number of considered measurements cannot change on the fly. In addition to the potential fields, vector field histograms evaluate the obstacle density around the robot and select control actions accordingly, rather than modeling individual forces. Virtual impedance algorithms operate similarly to potential fields, where instead of attractive and repulsive forces, the robot is guided by virtual mechanical impedance models such as stiffness and damping. Ibanez et al. [54] also lists other reactive computing methods, such as bug algorithms and the dynamic window approach. Bug algorithms, as exhaustively surveyed by McGuire et al. [65], are the most straightforward methods, where the robot follows a path along the sides of the obstacle until an exit condition is met. The dynamic window approach searches the input space for the velocity of the robots constrained by feasible speed values and timings. This approach is locally optimal but struggles with local minima as well. On the other side of the reactive computing category, local optimization encapsulates planners that take a pre-existing path as input and optimize it for the vehicle's dynamics and the occurrence of obstacles during operation. These methods usually employ constrained interpolation techniques, using clothoid, polynomial, Bézier, or spline curves to generate paths between waypoints sampled from the original path. An introduction to these methods can be found in the work of Gonzalez et al. [55] for further reference.

Task Allocation

Task allocation refers to the strategy employed by an autonomous robotic team to identify, assign, and plan the actions required to achieve its mission goals. Having defined path planning as the control structures that navigate robots between given locations with a set of constraints and mission metrics, task allocation considers how such locations can be selected, in what order, and how robot assignments can be made. In most literature, the separation of operational components is not made, where the

selection of waypoints is considered part of the path planning architecture. For the purpose of this thesis, such components of the autonomous controllers are regarded as an instance of the overall *exploration* task in the USaR area. Exploration is defined as reaching states in the operational environment of the robots that result in the acquisition of new measurements of features of interest in the mission, often achieved by employing frontier expansion. Frontier expansion is one of the most popular strategies for robotic teams that explore unknown areas, where the frontier is defined as the threshold between the known and unknown segments of the operational space. In the literature presented thus far, the works that utilize frontier-based exploration strategies include Azpurua et al. using 3D frontier extraction and mapping with traversability extraction [50], Zhang et al. proposing UAVs to detect clustered frontiers for the UGVs during mapping [48], Qin et al. using frontiers as 3D viewpoint generators with selecting the path with the most information gain from the path planner outputs to them [66], Niroui et al. k-means clustering the grid locations between obstacles to select frontiers [67], and Hu et al. proposing robots that place information nodes that act as communication relays that the radius of determines the points on the map to be regarded as a frontier at each time step.

Exploration-based task allocation is sometimes referred to as area coverage algorithms, especially if the geometry of the area can be estimated during operation. Queralta et al. note a complete algorithm can only be constructed when only convex and the joint regions exist in the environment, where efficient task allocation can occur via area decompositions [39]. When the area is unknown or assumptions about the environment cannot be made, multi-robot area coverage algorithms are generally NP-hard, with nondeterministic polynomial time complexity. For that reason, probabilistic components have been the focus of the literature for USaR task allocation. As defined by Lacroix et al., probabilistic modeling uses a probability distribution in the search space, sometimes by modeling the movement of the targets if applicable [53]. Since USaR missions should minimize the time to reach as many victims as possible, a probabilistic framework can maintain a level of guarantee to approximate optimal behavior in worst-case scenarios. Sarmiento et al. propose one of the earlier works with such formulation, where they define the mission goal as a single robot finding an object in a known environment by minimizing the expected time to find it [68]. The environment E is first decomposed into polygons, where their size is determined based on the maximum view field possible for the robot. Then, the general formulation of the problem is finding an optimal task allocation S^* consisting of an ordering of the decomposed areas that minimizes the expected time $E[T|S]$ given a candidate area S:

$$S^* = \underset{S}{\text{arginf}}(E[T|S]) = \inf_{S} \left(\sum_j t_j \frac{A(S_j) \setminus \bigcup_{k<j} A(S_k)}{A(E)} \right) \tag{2.3}$$

Where in Equation (2.3) t_j denotes the intermediary time for traveling to the j^{th} region, and the operator $A(.)$ is the area of a given region. It is essential to note that there is a subtle distinction between minimizing the expected value of the time required to find a target and minimizing the worst-case time it would take. For the latter, the shortest path that entirely covers the environment must be found, where the prioritization of areas or the rate of reaching new regions is not considered. In contrast, minimizing the expected time can be achieved by gaining the probability mass of detecting the target as quickly as possible, corresponding to the second term in the summation given in Equation (2.3). They employ a graph-based search for task allocation, similar to Equation (2.2), where the nodes represent the decomposed regions and the dynamic weights are formed by multiplying the elapsed time by the probability mass value at the child node. To favor higher probabilities earlier, they postulate that the region j strictly dominates k if the probability mass function for the region j is larger than that of k and the marginal time to reach from the current search node to the region j is smaller than that of k. Although this work provides a probabilistic guarantee, it is not generalizable to the case of an unknown environment and an unknown task count.

A similar formulation for heterogeneous teams searching for an uncertain number of targets in a known environment is proposed by Meuth et al., in which an arbitrator is designed to detect changes in the environment and vehicle constraints for re-planning [69]. Their work first employs a probabilistic decomposition method that is initiated by creating a Voronoi diagram of a point set in the environment. Then, relative to the facet anchors, the vertices of each facet are examined under a given set of constraints, such as distance and observability. If a facet does not meet the constraints, it is divided into smaller facets until all decomposed regions satisfy them. Afterward, at each time step, a probability

of observing the state of a location during the search is generated by considering the sensing capability of a robot, the likelihood that the area will be visible, and the number of times the area is visited. Using this, they propose a method called equilibrium task allocation, in which a set of robots is first randomly assigned to a set of regions. If multiple robots are assigned to the same region, a gradient descent-based approach is used to disperse the robots by employing an equilibrium error defined by the deviation of a robot's loading from the average team load. Once equilibrium is achieved, vehicles are swapped between regions, and the algorithm is repeated until no more swaps are possible. An improvement in the form of a Hybrid Binary PSO task allocation algorithm is then designed, where each particle consists of a set of vehicles and regions. Bit masks are applied to filter out undesirable mappings of robots to regions, making the position update function a binary decision that combines both binary and real-valued positions under a single fitness function.

The main bottleneck for optimal task allocation in an unknown area is the lack of completeness in an algorithm that provides a probabilistic guarantee for finding an unknown number of victims. These tasks can be regarded as dynamically occurring during the mission progression, meaning the final number and type of tasks may be unknown to controllers during operation. To address this problem, receding horizon task allocation algorithms are suggested in the literature. Riehl et al. put forward a receding horizon cooperative search algorithm that jointly optimizes the waypoints and sensor orientations of a team of robots to find a moving target [70]. In their formulation, the states are represented by $\boldsymbol{x}(k)$ and $\boldsymbol{q}(k)$ vectors for position/weights and sensor tasks as field of view orientations respectively, where a path P_i for robot i is defined by a sequential ordering of its states with respect to the connected graph vertices in the environmental representation. To employ a receding horizon approach, given a fixed horizon, a path reward $R_i(P_i)$ is proposed as follows:

$$R_i(P_i) := \sum_{j=k}^{k+T_{P_i}} g_i(\boldsymbol{x}(j), \boldsymbol{q}(j)) \left(1 - \sum_{l=0}^{j} r(l)\right) \tag{2.4}$$

In Equation (2.4) k represents the current time step, $g_i(.)$ the conditional probability of finding the target, and $r(.)$ is the accumulated team reward until the given timestep. The conditional probability of finding a victim at a corresponding future timestep is defined by the aggregation of the weighted sum of *agent reward factors* across the robotic team, where the agent reward factor measures the probability of the sensor configuration $\boldsymbol{q}(j)$ finding a target at the position $\boldsymbol{x}(j)$. Accordingly, they separate the set $A^{\text{plan}}(k)$ of robots that need re-planning and normalize path rewards by dividing the total duration of paths. Then, the predictive control optimization problem is proposed as finding the optimal paths for each robot in the set $A^{\text{plan}}(k)$ that maximizes the cumulative normalized path rewards at a given timestep. Finally, they propose a cooperative graph-based model predictive search algorithm to solve this optimization problem. At every step that A^{plan} is non-empty, an environmental graph representation is dynamically created so that only the states of interest are kept. Following that, using the remaining path information of other robots, the optimization problem is solved by fixing a robot at a time to create an optimal task allocation schedule. This work demonstrates that, with optimal filtering of graph states, the probability of finding the target is one with sufficient mission time, which is minimized by the optimization approach. The main setback in this formulation is that the team is assumed to be homogeneous, and the guarantee can only be made for the single-target case.

Jamshidnejad and Koning tackle the model predictive control problem for exploration task allocation considering non-homogeneous and imperfect sensors of USaR robots by incorporating fuzzy logic in decision-making [6]. This work assumes an unknown environment and an unknown number of targets, modeling their health state and the probability of moving to a neighboring cell in the environment grid. To combine coverage and target-oriented task allocation goals, a hierarchical control architecture is proposed, featuring local fuzzy controllers on each robot and a supervisory model predictive controller that oversees the team-based optimality of the local controller outputs. Locally, the robots keep a map of their immediate environment only, and a combined global map is kept at the centralized supervisory level. Formulating $e^{\text{v}}(x, y, k)$ as the probability of the existence of a victim in position (x, y) at timestep k, $c(x, y, k)$ as the scan certainty, and $h_v(k)$ as the perceived health level of the victim v if detected at the timestep k, a fuzzy rule set is proposed to assign priorities $\rho(x, y, k)$ to positions (x, y) at the timestep k. These three input values are fed to the fuzzy rules with the appropriate fuzzy sets denoting linguistic terms adopted from human expertise in a USaR scenario. Using the local maps of robots, a

set of paths $P_i(k)$ consisting of ordered locations starting from the current timestep k is constructed to warm start the optimization, of which the task allocation problem becomes equivalent to assigning the best paths to the robots. The potential paths are compared by their grade, expressed as:

$$g(P_i(k)) = -c_1 \ell(P_i(k)) + c_2 \sum_{j=k}^{k+\ell(P_i(k))-1} \lambda^j \rho(x_i^{\mathsf{a}}(k), y_i^{\mathsf{a}}(k), j) \tag{2.5}$$

where in Equation (2.5), the term multiplied by the constant c_1 denotes the length of the path, and the term multiplied by the constant c_2 denotes the degree of exploration; incorporating a discount factor $\lambda \in [0, 1]$ to the aggregation of priority values along the path. The discount factor intends to reduce the potential effects of errors in future state estimations, and the constants c_1, c_2 enable the trade-off tuning. The potential path $P_i(k)$ with the highest grade $g(.)$ is selected by the local fuzzy controllers for every robot at each control step. The centralized supervisory controller is then responsible for resolving task allocation conflicts; in this specific work, it is approximated by a threshold on the intersection of the perception fields of robots. When this threshold is exceeded, the considered paths of local controllers of the respective robots are given to the receding horizon optimization solver to determine the best combination of them using the following formulation:

$$\max_{\mathbb{P}(k)} w_1 \sum_{i=1}^{N} g(P_i(k)) + w_2 \sum_{(x,y)\in E} c(x, y, N^{\mathsf{P}}(k)) \tag{2.6}$$

$$\text{s.t.} \quad (x_v^{\mathsf{v}}(k), y_v^{\mathsf{v}}(k)) \notin P_i(k) \cap P_j(k) \quad \text{where} \quad i, j = 1, \ldots, N \quad i \neq j \quad v = 1, \ldots, N^{\mathsf{v}}(k)$$

The optimization variable $\mathbb{P}(k)$ in Equation (2.2.3) is the set of paths for N robots of which the task allocation conflict is detected, where the first term denotes the sum of the path grades and the second is the total predicted scan certainty of the environment at the end of the prediction horizon $N^{\mathsf{P}}(k)$ corresponding to the largest length of the candidate paths. Effectively, this latter term acts to scatter the robots efficiently. The constraint restricts multiple robots from visiting the same victims, aiming to address the underlying reason for task allocation conflicts in this formulation. They benchmark the proposed hierarchical control architecture by comparing simulations that use only local controllers, only the supervisory controller, a randomized task allocator, and an ACO controller, considering the scan certainties in the environment as pheromones. Two primary metrics consist of the rise time for total scan certainty and victim search efficiency, which is measured by the number of detected victims per time step and their corresponding health states. They conclude that the proposed architecture improves all these metrics compared to the baseline and distributed selfish controllers, while approximating the purely centralized case sufficiently with substantially less complexity.

While some works in the literature handle only the exploration task by formulating it as a path optimization or waypoint selection problem, the necessities in a USaR scenario may require auxiliary tasks and goals to be successful, such as establishing communication networks, delivering goods, and creating a base of operations during the mission [12]. Although connectivity and charging are mostly regarded as constraints in optimization problems, there are works that formulate these requirements as tasks. Hollinger et al. propose a periodic connectivity task for USaR scenarios in which communication is assumed to be restricted by the relative locations of the robots [71]. Even though for most applications, ensuring continuous connectivity is desired, the central hypothesis claimed in their work state that breaking it intentionally may lead to more efficient information gathering. They explore and formalize a framework of path selection-based task allocation with periodic connectivity constraints, effectively behaving as scheduled tasks. The environment representation as a graph $G = (N, E)$ is time-extended for planning, denoted by $G' = (N', E')$, which is a directed graph with time-stamped nodes and edges that substantiate the time increments between locations on the original graph. Additionally, a connectivity graph $G_C = (N, E_C)$ is constructed describing which nodes have established communications. Given K robots, $A = \bigcup_{k=1}^{K} A_k$ the set of potential time-stamped paths, a disconnection interval, and a mission ending time T, the task allocation problem is expressed by:

$$A^* = \max_{A_k \subseteq N'} F(\cup_{k=1}^{K} A_k) \tag{2.7}$$

$$\text{s.t.} \quad A(\tau T_I) \text{ is connected on } G_C \quad \forall \tau \in \{0, \ldots, \lfloor T/T_I \rfloor\} \tag{2.8}$$

Where in Equation (2.8), $F(.)$ is a known deterministic function of possible paths, and $C(.)$ is the time-cost of a given path. Showing that for a fully connected G_C, this optimization problem is NP-hard, Hollinger et al. propose an approximate algorithm called implicit coordination, guaranteeing connection at every T_I interval. The synchronous online algorithm optimizes the objective one variable at a time in a round-robin fashion for each robot until convergence. Although they show that the algorithm converges fast and is scalable, it lacks guarantees for the execution of other possible tasks that may exist, and the efficiency relies on accurate synchronization. In a similar manner, Li et al. tackle the charging constraints of robotic teams in SaR missions as practical tasks, which are presented as an essential bottleneck in the efficiency of autonomous searching schemes [72]. They consider optimizing the area coverage by placing static charging stations in a known environment with an offline genetic optimization-based algorithm. The exploration allocation and the charging station locations are calculated together prior to deployment. To that end, each chromosome in the population represents ordered exploration task locations per robot, with the locations that require the robot to charge marked. Then, the cost of each gene is formulated as $J = c_1T + c_2P + c_3L + c_4H$, where T denotes the mission completion time, P the cost of the time taken to cover prioritized tasks, L the penalty for violating energy limitations, as well as the number of required charging stations, and H the distance between the charging points.

Contrary to the aforementioned works, it is also possible to assign semantical tasks to robots instead of specifying their arrival at a location to perform an action. Most SaR literature, in this regard, considers hierarchies in the robotic team and heuristic approaches that define such tasks. Hayat et al. consider two main tasks for multi-objective task allocation in SaR scenarios by minimizing the time for area coverage and setting up communication paths [73]. Their defined mission consists of a search phase followed by a response phase to find a stationary target. In the first phase, the search paths are planned centrally on a base station, where the total time to cover the area and return to the base station is denoted by τ_s. Upon detecting the target at the location l with a robot m at time t_1, it is added to the set S denoting surveilling robots and removed from the set of searching robots U. Afterward, the response phase begins, in which any robot coming into the communication range of the set S is labeled as a relay added to the set R and removed from U. Whenever the set R is populated, the relay labeling time t_{rl} is updated. Finally, the robots are required to form a communication chain through the robots in R to the base station, for which the time it takes is denoted t_n, effectively completing the total duration of the second phase as $\tau_R = t_{rl} + t_n$. The main optimization goal in the formulation of this work is therefore minimizing the total time $\tau = \lambda\tau_S + (1 - \lambda)\tau_R$, with $\lambda \in [0, 1]$ representing a trade-off tuning constant. Hayat et al. then propose three heuristic strategies that define t_n; namely inform-first, connect-first, and simultaneous inform and connect. In the inform-first strategy, the first robot added to R acts as a data mule and goes directly to the base station, while a relay chain for continuous monitoring is established whenever the set R contains enough robots. In the connect-first strategy, the first robot in R plans the shortest path to reach enough robots in the set U to form a relay chain back to the base station, where each alerted robot immediately starts forming the chain. Finally, the simultaneous strategy combines the first two by having the first robot in R to go directly to the base station, while adjusting its path if any robots in U are close by on its path to add them to the set R to start forming the relay chain while it is on the way. The total mission time τ is minimized with all three strategies using a genetic optimization approach, with the paths the robots take through both phases represented as chromosomes. Similarly, Hoog et al. propose a role-based task hierarchy in the robotic team by assigning robots as either explorers or relays [74]. Their formulation differs in that relay robots are also mobile and move between explorers and the base station, where they meet with explorers periodically in dynamically defined rendezvous points. Then, they propose a dynamic change in the roles with a role swap rule, considering two robots A and B each having destinations D_A and D_B. Denoting the path cost between two locations in the environment as $\gamma(u, v)$, the role swap condition is satisfied when:

$$\max\{\gamma(A, D_A), \gamma(B, D_B)\} > \max\{\gamma(A, D_B), \gamma(B, D_A)\} \tag{2.9}$$

If Equation (2.9) holds, the robot A assumes the role of robot B and they swap the hierarchies within the team, upon which this information is propagated whenever possible. Explorers search the far reaches of the environment using frontier exploration, where upcoming rendezvous points are dynamically determined and set whenever explorers gain enough information, thresholded by the remaining area to be searched. This heuristic effectively prevents the hierarchy from becoming too imbalanced. This work demonstrates that role swaps allow shorter travel paths and, therefore, faster exploration. The

decision-making architecture of predefined heuristic tasks is also formulated from the perspective of reinforcement learning as a generalized solution, as proposed in the work of Liu et al. [23]. In their hierarchical reinforcement learning model, a MAXQ task prioritization scheme is utilized in which the root task represents the overall mission, i.e., finding victims while exploring a cluttered USaR environment. This task is further decomposed into four individual sub-tasks, namely, exploration, navigation, victim identification, and human control. Accordingly, the robots select their lower-level controls based on the learned parameters and rewards they receive through performing higher-priority tasks earlier. With the human control option, the learning is aimed to continue during actual deployment.

Although the literature indicates that specific task allocation methods utilizing heuristics and predefined behavioral algorithms are effective, they often lack scalability and robustness in their decision-making processes. For that reason, generalized task allocation approaches are suggested for the USaR missions, where the primary focus is on optimally assigning any number and type of tasks, provided the metrics are formulated correctly. Basilico and Amigoni define the USaR goal for robotic teams as achieving good global performance through local decisions with partial information [7]. Since the goodness of a local decision can be measured with respect to multiple criteria that may change dynamically in their global prioritization, an abstract methodology for task allocation is necessary. To that end, they propose the evaluation of tasks as a Multi-Criteria Decision-Making (MCDM) optimization problem. The formalization of the problem starts with considering a set N of n criteria, where each candidate task p can be associated with a vector of utilities $u_p = (u_1(p), \dots, u_n(p))$. When dealing with an arbitrary number of potential criteria, the optimality involves selecting tasks on the Pareto frontier. Although it is possible to estimate the Pareto frontier through heuristics similar to the work of Sarmiento et al. [68], the rigorous approach includes maximizing a global utility function $u(p) = f(u_p)$ that aggregates the different utility criteria. Basilico and Amigoni show that if this function is non-decreasing in every one of its arguments, the optimal p^* is guaranteed to be on the Pareto frontier. They continue by arguing that most mainstream approaches, including those laid out thus far, combine utilities in an ad-hoc manner and implicitly seek to approximate the Pareto optimal results. One crucial element missing from ad-hoc methods is regarding different dependencies between criteria, such as redundancy and synergy. While the global utility of redundant criteria is less than the sum of their individual utilities, synergetic situations arise when such criteria are very different from each other and hence contribute more to the global utility than the sum of their individual utilities. The MCDM formulation addresses these aspects through the aggregation method, utilizing Choquet integrals. To achieve that, a function that maps the power set of N as $\mu : \mathcal{P}(N) \rightarrow [0, 1]$ is defined with properties $\mu(\{\emptyset\}) = 0$, $\mu(N) = 1$, and given that $A \subseteq B \subseteq N$ then $\mu(A) \leq \mu(B)$. In practice, μ represents a fuzzy measure on the criterion set N that associates weights. Criteria $G \subseteq N$ are considered redundant if $\mu(G) < \sum_{i \in G} \mu(i)$, synergetic if $\mu(G) > \sum_{i \in G} \mu(i)$, and independent otherwise. The discrete Choquet integral denoting the global utility is thus as follows:

$$f(u_p) = \mathcal{C}(u_p) = \sum_{j=1}^{n} (u_{(j)}(p) - u_{(j-1)}(p)) \mu(A_{(j)}) \tag{2.10}$$
$$A(j) = \{i = N | u_{(j)}(p) \leq u_i(p) \leq u_{(n)}(p)\}$$

Where in Equation (2.10), $(j) \in N$ denotes the j^{th} criteria in the increasing ordering of utilities given as $u_{(1)}(p) \leq \dots \leq u_{(n)}(p) \leq 1$, with the assumption that $u_{(0)}(p) = 0$ holds. Accordingly, the Pareto optimization is shown to be guaranteed by the maximization of $\mathcal{C}(.)$ over all candidate tasks. Using this framework, they implement the task allocation of a robotic team in an unknown environment for finding an unknown number of targets in a USaR scenario. Three different approaches are included to demonstrate the scalability and robustness of MCDM. The first one includes the amount of free area beyond the location p as $A(p)$, the probability of connectivity to the base station $P(p)$, and the distance from the current location r as $d(p, r)$. The set of weights μ is determined by assigning more importance to A than to P and d for faster exploration, where P and d are shown to have redundancy while d and A enjoy a synergy. This analysis guides them to assign feasible μ values. The second method adds another criterion, the remaining battery of robots as b. The final approach dynamically changes the weights μ through the mission to alter the priorities of the robots as time progresses from exploring to exploiting. Finally, noting that for n criteria, $2^n - 2$ weights need to be defined, a more principled method is proposed by considering two aspects. The first one is the overall importance is called Shapley's value, defined as the average marginal contribution of a criterion i to each subset that

does not include it:

$$\phi_\mu(i) = \sum_{T \subseteq N \setminus \{i\}} \frac{(|N| - |T| - 1)!|T|!}{|N|!} (\mu(T \cup \{i\}) - \mu(T)) \tag{2.11}$$

A higher value of $\phi_\mu(i) \in [0, 1]$ means more importance i has in the task allocation. The second aspect is the interaction between two criteria i and j, named Shapley's interaction index $I_\mu(i, j) \in [-1, 1]$:

$$I_\mu(i,j) = \sum_{T \subseteq N \setminus \{i,j\}} \frac{(|N| - |T| - 2)!|T|!}{(|N| - 1)!} (\mu(T \cup \{i,j\}) - \mu(T \cup \{i\}) - \mu(T \cup \{j\}) - \mu(T)) \tag{2.12}$$

Accordingly, if $I_\mu(i, j) < 0$ then i, j are redundant where the value -1 representing full redundancy, the opposite is true for $I_\mu(i, j) > 0$ demonstrating synergy and $I_\mu(i, j) = 0$ implies mutual independence.

The review article from Khamis et al. distinguishes the problem modeling of generalized multi-robot task allocation from the types of methods used to solve it; namely given as discrete fair division, optimal assignment, alliance efficiency, and multiple traveling salesman problems (TSP) [4]. The discrete fair division assumes a given set of N robots and a set of tasks S, and divides the task set so that each robot gets a fair share in the sense that the utility of the assigned subset of tasks is worth $1/N$ the total value of S. This problem can be extended to include indivisible and divisible tasks, as well as the heterogeneity of robots. The optimal assignment problem approach encapsulates maximizing the aggregate utility of task assignments to robots, where all the works laid out so far fall into this category, including the generalization of Basilico and Amigoni. The alliance efficiency problem is a mono-objective optimization formulation in which several tribes with specific skills aim to exploit the search space regarding modeled resources necessary for survival. A tribe is defined as a tuple $t := \langle x_t, s_t, r_t, a_t \rangle$ where x_t is a point in the solution space, s_t the set of skills depending on the objective function values, r_t the set of resource demands as constraints, and a_t the alliance identification. Finally, the multiple TSP considers the tasks as a set of cities to be visited by m salesmen, and the objective becomes determining a tour for each salesman, such that each city is visited with minimal time or distance. There exist many formulations of this problem, generally differing in the means of resolving sub-tours. Most TSP formulations can be classified under those that rank the cities, explicitly utilize time-indexed variables for the cities, or construct multi-commodity flow models. Kant and Mishra lay out a new type of TSP-like formulation, called the Orienteering Problem (OP), where not only does the order of visited cities need to be determined, but also which ones should be selected similar to a knapsack-problem need to be decided to maximize collected utility before a global budget expires [75]. They define profits at each city as ϕ_x, travel time as $T_{xy}$and a binary decision variable ψ_{xy} for selecting transitions. Then, the formalization takes the following form, which establishes a feasible connected path and adherence to the time budget through transition and city selections:

$$\begin{aligned}
&\max \sum_{x=2}^{n-1} \sum_{y=2}^{n} \phi_x \, \psi_{xy} \\
&\text{s.t.} \\
&\sum_{x=2}^{n} \psi_{1x} = 1, \quad \sum_{x=1}^{n-1} \psi_{xn} = 1 \\
&\sum_{x=1}^{n-1} \psi_{xk} = \sum_{y=2}^{n} \psi_{ky} \leq 1 \; \forall k = 2, \ldots, n-1 \\
&\sum_{x=1}^{n-1} \sum_{y=2}^{n} T_{xy} \, \psi_{xy} \leq T^{\max}
\end{aligned} \tag{2.13}$$

Kant and Mishra also survey the variants of the canonical OP; as team OP, time-window OP, time-dependent OP, stochastic/probabilistic OP, clustered OP, multi-objective OP, and links to generalized TSP. They introduce various ways of implementing these problems, ranging from MILP approximations to metaheuristics with different evolution strategies for genetic algorithms, ant colonies, iterated local searches, and adaptive large-neighborhood searches as promising approaches in the literature. A

state-of-the-art generalized solution for the OP-type problems is introduced by Kobeaga et al. [76], proposing a specialized genetic algorithm. Each candidate route is encoded as a permutation of all nodes, where the visited nodes form a closed tour and every non-visited node is treated as a fixed point, ensuring a fixed-length chromosome that can represent any subset of nodes. This approach notably allows infeasible individuals during the search to enhance exploration, then relies on dedicated repair and improvement operators to restore feasibility. A custom crossover operation, adapted from the edge recombination method often used for TSP problems, is designed to preserve as much parental structure as possible by probabilistically including nodes that appear in only one parent, and reusing a high proportion of parent edges to keep the cost of offspring low. The mutation operator further refines diversity by randomly selecting a node and either dropping it from a candidate route or adding it, where in the latter case, the node is inserted at its best position according to a fast heuristic that approximates the minimal cost increase. To polish each new solution, the algorithm applies an intensive post-optimization phase: first a tour-improvement local search reorders the nodes to minimize travel distance, which often converts borderline infeasible tours into feasible ones; if the route still violates the cost budget, a feasibility repair via a drop operator iteratively removes the least beneficial nodes until all constraints are satisfied. Once a tour is feasible, an insertion heuristic greedily tries to add any remaining high-profit nodes using a custom cost evaluation to identify low-cost insertion points. This robust framework is thoroughly benchmarked on medium and large graphs, consistently demonstrating its superiority over other approaches in terms of solution quality with minimal computational time.

Having defined the task allocation schemes as single tasks versus multi-tasks, single robot versus multi-robot, instantaneous versus scheduled assignments, and centralized and distributed approaches, Khamis et al. define two overarching methods for solving the aforementioned problems. The first is optimization-based approaches, which consider nonlinear and nonconvex integer optimization for most task allocation problems. These approaches can be solved by methods such as genetic optimization, simulated annealing, and PSO. The second approach is market-based, drawing inspiration from real-life auctions that utilize explicit communication and the expression of capabilities in the form of bids for overtaking tasks. Most auctions are designed utilizing the contract net protocol, which can be defined as decentralized competitive bidding. The four stages of this protocol start with the announcement stage, broadcasting the tasks to be allocated to the robots. Then, in the submission stage, each robot calculates its capability and utility for overtaking tasks and submits a bid accordingly. The following selection stage encapsulates the coordinator reviewing the bids using an optimization strategy to accept the best bids. Finally, in the contract stage, the winning agents are allocated tasks via a contract. Although this approach is flexible and robust, it may prioritize individual utilities more than the overall outcome.

In the review article of Felix et al. the market-based task allocation methods are compiled under auction-based and consensus-based categories [77]. Similar to the contract net protocol, robots submit bids representing the utilities they gain from completing auctioned tasks. Given a task t, the utility of a robot $U_r(t)$ is generally formulated as the difference between the expected rewards $R_r(t)$ and costs $C_r(t)$. Sequential single-item auctions extend this concept by considering dependencies between tasks, where each bid is adjusted based on previous allocations of the robot. To account for the cost of adding a task t to the currently scheduled plan P_r of a robot, the marginal cost $M_{r,t}$ is utilized and defined by the difference between the cost of the proposed plan $C_r(P_r + t)$ and the current plan $C_r(P_r)$. Thus, this approach prioritizes task synergies and sequential dependencies, enabling further constraints to be employed. Parallel single-item auctions also exist to allow simultaneous bidding, where the auctioneer assigns tasks to the bidders with the highest bids on a rolling basis, allowing for faster deployment at the expense of decreased dependency management. In contrast, consensus-based approaches apply cooperative strategies to task allocation, requiring all robots to reach an agreement over task distribution. The Consensus-Based Bundle Algorithm is one of the most prominent examples in this category for robotic task allocation, in which each robot r constructs a bundle of tasks $B_r = \{t_1, t_2, \ldots, t_k\}$ in descending order of task value based on its local utility function. Here, the marginal utility $MU_{r,t}$ of adding task t to a robot's current bundle is represented as:

$$MU_{r,t} = U_r(t) - \sum_{t' \in B_r} U_r(t') \tag{2.14}$$

Conflict prevention is substantiated by robots exchanging their bundles iteratively, where comparisons are performed, and outbid tasks are dropped incrementally. This way, conflict-free task allocation is proposed to be achieved globally, while also ensuring local optimality. Finally, hybrid methods that combine auction and consensus procedures are presented. One such method is clustered combinatorial auctions, where tasks are first grouped into clusters, and robots bid on task bundles rather than single tasks. For task allocation, clusters are assigned based on the overall marginal benefit, allowing for the efficient handling of inter-dependencies. Given a total set of tasks T, this method optimizes the search through subsets $S \subset T$ to calculate the overall bundle utility $U(S)$ as follows:

$$U(S) = \sum_{t \in S} (R_r(t) - C_r(t)) \tag{2.15}$$

In general, market-based approaches excel at scalability and ease of deployment for task allocation schemes; however, for robotic applications similar to USaR scenarios, they fall behind in optimality compared to optimization-based methods such as genetic algorithms and simulated annealing [78].

Finally, some works in the literature consider generalized task allocation from the perspective of finite-state automata. Tsalatsanis et al. introduce a dynamic allocation scheme in which every robot's suitability for a task is evaluated through a fuzzy-logic utility function, and decisions are taken by a limited-look-ahead discrete-event supervisor [79]. They propose finite-state automatons for robot actions and task executions, accounting for the possible failures and task abilities of different robots as well as globally defined inter-task dependencies. Although the method is robust and able to reallocate tasks online, it is not scalable since the global joint finite-state automaton of all robots and tasks grows rapidly. Bai et al. push formal task allocation of finite-state automatons further by casting both individual and collaborative requirements as Linear Temporal Logic (LTL) formulas [80]. They assume that each robot has its own individual task specification, where the heterogeneous robotic team must satisfy a collaborative goal through a sequence of task executions. They propose a centralized automaton that decomposes the global LTL definition into robot sub-tasks, where task hierarchies and completion orders are encoded. Their hierarchical split between global sequence extraction and local schedule optimization demonstrates how LTL decomposition can maintain task-level optimality without expanding the state space; however, this method requires all tasks to be determined beforehand, which conflicts with the uncertain environment in which USaR missions take place. Guo et al. add a final layer of refinement to this approach by introducing safe-return constraints, formalized as a second LTL formula, into probabilistically labeled Markovian motion primitives of individual robots [81]. They synthesize an outbound policy that maximizes task success probability and cost efficiency, and a return policy that guarantees a high-probability path back to a charging platform on demand. All of the LTL and automata-based solutions require global solvers and definitive tasks from the start, which is unsuitable for the USaR requirements laid out in this thesis; however, their abstraction of robot capabilities and task requirements is a powerful approach for heterogeneous autonomous robotic teams.

3

Methodology

This chapter presents the formalization and implementation of the proposed decision-making and coordination architecture for autonomous robot teams in USaR environments. The methodology builds directly on the conceptual foundation laid out for the need for flexible, interpretable, and mission-adaptive task allocation, identified as a key gap in the literature. To address this, Section 3.1 introduces a generalized system model that abstracts robot capabilities through their heterogeneous action sets and the observations these actions yield. This abstraction enables an observation-driven definition of tasks and grounds a two-stage decision-making framework that combines fuzzy reasoning and MCDM. Then, Section 3.2 introduces the hierarchical control architecture for optimally tackling the defined MPC optimization problem in a computationally tractable way for SaR applications, by applying MILP-based linearization.

3.1. Problem Formulation

In essence, heterogeneous robot actions can be modeled as environmental interactions, such as actuation and sensing, that take a certain *time* and expend a certain amount of *energy*, while obtaining *observations* about the outcome of the action from the environment. Building on this framework, the robots in a SaR environment also need to store a *local map* that holds long-term information, enabling individual decision-making. At the system level, a *global map* and a *task set* are the minimum requirements for the proposed structure in this thesis, as the main goal of SaR robots is to increase environmental awareness for human teams. As a high-level description of the system formalization, each robot operates on its own local map, which may have different representations of the environment. Upon taking an action, a robot obtains a set of observations and updates its own map accordingly, while also updating the global map. These updates, resulting from action observations, are then used to construct the next task set, determining which tasks are assigned to the corresponding robot. Whenever a robot plans an action, it synchronizes its local map with the global map to access the most recent information flow in the environment. This framework is proposed as the most general representation of an asynchronous robotic team with sporadic synchronizations, accounting for all types of heterogeneous representations. Section 3.1.1 first introduces the mathematical formalization of the environment, robots, and the task sets. Then, Section 3.1.2 introduces the framework of heterogeneous task allocation through robot actions, and concludes with the MPC optimization formulation of the proposed generalized robotic SaR missions.

Remark 3.1.1 (Notation Design.)**.** *In this thesis, a framework of notation design is used for overall consistency and clarity.* ***Calligraphic letters*** *are employed to denote the general sets used as domains and for variables configured as the operational space of the robots, such as maps, ground truth sets, and observations.* ***Uppercase letters*** *are generally used for sets that are direct subsets of the sets denoted by the calligraphic letters and the representation of the time-dependent sets. Another use of uppercase letters is for the binary decision variables associated with the specific controllers introduced in the control architecture. The notable exceptions to these rules are the letters* H N*, and* M *which are used to represent scheduling*

horizons, small constants, and big constants, respectively. ***Lowercase letters*** *have a variety of functions throughout the thesis; however, they are consistently utilized for representing atomic members of corresponding sets. The exceptions to this logic are the usage of the letter z for binary charge constraints, f for robot state-related functions, u for utilities, and t for time.* ***Subscripts*** *are always lowercase, where k is the timestep counter, i is the robot index, j is the charger schedule index, l is the task or criteria index depending on the context, and m, n are the indices used for iterating over other sets defined wherever applicable.* ***Superscripts*** *are also generally strings of lowercase letters denoting the context of the associated variable. If a variable is tied to another variable, that models a unique member of a set, then the letter chosen for the latter set is used as a superscript to indicate the relationship. The* ***Greek letters*** *are used for modeling the crucial design elements and functions, such as energy and tasks. Some examples include the letter ν for value functions, δ for time-energy depletion functions, σ for fuzzy system input functions, and the letters ϕ, Φ for task-related functions or binary functions. The important exceptions for this design choice include the letters π, Π for schedules and control input-output vectors, and the letter κ for iterating from an already existing timestep, in the form of a subscript $k + \kappa$. Finally,* ***bold letters*** *are used to denote vectors, and the accent $\hat{}$ is used to denote predictions.*

3.1.1. Environment & Robot Modeling

The generalized representation of the ground truth environment is modeled by a temporal attributed graph $\mathcal{G}$:

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{F}, \ell^{\text{env}}, \omega^{\text{env}}, \nu^{\text{env}}) \tag{3.1}$$

where $\mathcal{V}$ is a set of spatial vertices, $\mathcal{E}$ is a set of undirected edges $\mathcal{E} \subseteq \{(u, v) \mid u, v \in \mathcal{V}, u \neq v\}$ representing the distances between connected vertices, and $\mathcal{F}$ is a set of attributable semantic features. The time-dependent vertex labeling function $\ell^{\text{env}} : \mathcal{V} \to 2^{\mathcal{F}}$ maps the vertices $v \in \mathcal{V}$ to an attribute tuple as $\ell^{\text{env}}(v) = F(v) \subseteq \mathcal{F}$ that contains relevant semantic features of the vertex v. The edge weight function $\omega^{\text{env}} : \mathcal{E} \to \mathbb{R}$ denotes a distance metric between connected vertices. Finally, the ground truth value function $\nu^{\text{env}} : \bigcup_{v \in \mathcal{V}} \{v\} \times \mathbb{R} \to \mathbb{R}^{|F(v)|}$ maps each vertex and time instance t to real values corresponding to the entries of the feature set $F(v)$ of vertex v at time t. In practice, human USaR teams look for interpreted and/or measured indicators from the environment to decide where the search efforts should be concentrated at a given time. An example framework from the literature is given in Table 2.1. Therefore, the semantic features $F(v)$ of a vertex v may be attributed as fuzzy features relating to the USaR prioritization criteria provided in Section 2.1. Generally, the feature set contains all of the relevant labels of a unique vertex that denotes an input to the decision-making architecture of the robotic system.

There is a wide range of heterogeneity in USaR robotics in terms of robot actuators and traversability (Section 2.2.1), as well as sensing capabilities (Section 2.2.2). In most general terms, all robots in the USaR domain can be characterized by their state, containing a ground truth vertex and a current map. The heterogeneity is then modeled by a set of actions that determine the transitions between states. Actions constitute a relationship between robot-specific measurable features and robot-specific capabilities in terms of actuation and sensing. To model a closed-loop system, a set of observations related to unique actions is included in the model of a generalized USaR robot. In practice, the observation set contains assigned values for relevant features, serving as a short-term memory, while the local maps contain distilled information related to robot decision-making, acting as a long-term memory. For each robot $i \in \{1, \ldots, N^{\text{robot}}\}$, where $N^{\text{robot}} \in \mathbb{N}^+$ denotes the number of robots in the autonomous robot team, the most general representation consists of a robot-specific state space $\mathcal{S}_i$, action space $\mathcal{A}_i$, and observation space $\mathcal{O}_i$. Consequently, the robots transition between states only through taking actions, while collecting relevant observations tied to the taken action from the current state. To properly define the action space, in this thesis, a robot-specific unique action type set $\mathcal{C}_i = \{\alpha_1, \ldots, \alpha_{N_i^{\text{action}}}\}$ is formulated, where $N_i^{\text{action}} \in \mathbb{N}^+$ denotes the number of unique action types attributed to the robot i.

Then, the action space of a robot can be defined by:

$$\mathcal{A}_i \subseteq \mathcal{C}_i \times \mathcal{V} \tag{3.2}$$

This thesis formulates an abstract action $a = (\alpha, v)$ as a tuple with $a \in \mathcal{A}_i$, $\alpha \in \mathcal{C}_i$, and $v \in \mathcal{V}$ for the robot i. Upon executing an action from a given state $s \in \mathcal{S}_i$, a robot i spends $\delta_i^{\mathsf{time}}(s,a) \in \mathbb{R}_{\geq 0}$ time and consumes $\delta_i^{\mathsf{energy}}(s,a) \in \mathbb{R}_{\geq 0}$ energy. Furthermore, each robot has its own state transition function $f_i^{\mathsf{s}} : \mathcal{S}_i \times \mathcal{A}_i \to \mathcal{S}_i$ and observation function $f_i^{\mathsf{o}} : \mathcal{S}_i \times \mathcal{A}_i \to 2^{\mathcal{O}_i}$. In the context of the proposed formulation, an observation is defined by a tuple:

$$o = (v, F^{\mathsf{o}}(v), \nu^{\mathsf{o}}(v)), \quad o \in \mathcal{O}_i \tag{3.3}$$

where $v \in \mathcal{V}$ is the vertex associated with the observation, $F^{\mathsf{o}}(v) \subseteq F(v)$ is a set of observed features, and their corresponding measurement values, and $\nu^o(v)$ is constructed by sampling the ground truth values $\nu^{\mathsf{env}}(v,t)$ at the time instance t of the action execution. The sampling method is specified by the unique observation functions f_i^{o} and the selected action type α, therefore enabling a substantial amount of flexibility in defining robot-environment interactions and heterogeneous features. Each action generates a set of bounded observations $O_i = \{o_1, \ldots, o_{|f_i^o(s,a)|}\}$, where specific action types are mapped to a subset of environmental features associated with particular vertices, enabling the representation of various field of view types and sensor modeling.

A robot $r_i(t_k)$, is modeled as a discrete-time tuple, where each robot $i \in \{1, \ldots, N^{\mathsf{robot}}\}$ evolves asynchronously over its own local timeline. At a global timestep t_k, a USaR robot is modeled as a time-dependent state and time-invariant robot-specific features:

$$r_i(t_k) = \left(s_i(t_k),\ \epsilon_i^{\mathsf{crit}},\ \mathcal{C}_i, f_i^{\mathsf{o}},\ f_i^{\mathsf{s}},\ \delta_i^{\mathsf{time}},\ \delta_i^{\mathsf{energy}}\right) \tag{3.4}$$

where $s_i(t_k) \in \mathcal{S}_i$ is the robot state at the global time t_k and $\epsilon_i^{\mathsf{crit}} \in \mathbb{R}_{\geq 0}$ is the critical energy level of the robot. Robots operate asynchronously, as actions may take different durations to complete. For that reason, two distinct time indices are used within the thesis: a global time t_k for coordinating system-wide events, and a robot-local time $t_{i,\tilde{k}_i}$ to track each robot's asynchronous action execution and state transitions. As given in the robot state description, at time $t_{i,\tilde{k}_i}$, robot i begins executing action $a \in \mathcal{A}_i$. Although this action is not explicitly included in the robot's state $s_i(t_k)$, it is used to determine when the next state transition will occur. Specifically, the next local timestep for the robot transitioning from its current state is denoted by the time it takes to complete the latest selected action:

$$t_{i,\tilde{k}_i+1} = t_{i,\tilde{k}_i} + \delta_i^{\mathsf{time}}\left(s_i(t_{i,\tilde{k}_i}), a\right), \quad a \in \mathcal{A}_i \tag{3.5}$$

As an extension of the asynchronous robot modeling, at any global time t_k, the state transition of a robot $r_i(t_k)$ is defined as:

$$s_i(t_k) = \begin{cases} s_i\left(t_{i,\tilde{k}_i}\right) & \text{if } t_k < t_{i,\tilde{k}_i+1} \\ f_i^{\mathsf{s}}\left(s_i(t_{i,\tilde{k}_i}), a\right) & \text{if } t_k = t_{i,\tilde{k}_i+1} \end{cases}, \quad a \in \mathcal{A}_i \tag{3.6}$$

This establishes the robot states as piecewise constant during action execution, with transitions occurring only at completion points. In contrast, the global time evolution is characterized by the times at which at least one robot completes an action, denoted by:

$$t_{k+1} = \min_{i \in \{1,\ldots,N^{\mathsf{robot}}\}} \left\{ t_{i,\tilde{k}_i+1} \mid t_{i,\tilde{k}_i+1} > t_k \right\} \tag{3.7}$$

Thus, at each global time t_k, only the robots whose actions are completed at that instant will update their state, while others remain unchanged, effectively synchronizing the overall robotic team at the system level. Similarly, for robot i, the observation function f_i^{o} maps the robot's state and action at execution start to an observation set. At global time t_k, the resulting observation set $O_i(t_k)$ is defined as:

$$O_i(t_k) = \begin{cases} f_i^{\mathsf{o}}\left(s_i(t_{i,\tilde{k}_i}), a\right) & \text{if } t_k = t_{i,\tilde{k}_i+1} \\ \emptyset & \text{otherwise} \end{cases}, \quad a \in \mathcal{A}_i \tag{3.8}$$

This models the feedback the robots get by performing their actions, ranging from a set of local measurements to interactions with the SaR environment. The robot state $s_i(t_k)$ denotes the robot's most recently updated state as observed at global time t_k. It is defined by the tuple:

$$s_i(t_k) = \left(v_i^{\mathsf{r}}(t_k),\ \epsilon_i(t_k),\ \mathcal{M}_i(t_k),\ A_i(t_k),\ t_{i,\tilde{k}_i} \right) \tag{3.9}$$

where $v_i^{\mathsf{r}}(t_k) \in \mathcal{V}$ is the vertex that the robot is positioned at time t_k, $\epsilon_i(t_k) \in \mathbb{R}_{\geq 0}$ is the available energy, $\mathcal{M}_i(t_k)$ is the local map, $A_i(t_k) \subseteq \mathcal{A}_i$ is the set of feasible actions, and $t_{i,\tilde{k}_i}$ is the local time of the robot at a given time t_k. The local maps denote a robot-specific approximation of the ground truth:

$$\mathcal{M}_i(t_k) = (V_i^{\mathcal{M}}(t_k),\ E_i^{\mathcal{M}}(t_k),\ O_i^{\mathcal{M}}(t_k)) \tag{3.10}$$

where $V_i^{\mathcal{M}}(t_k) \subseteq \mathcal{V}$ is the set of mapped vertices, $E_i^{\mathcal{M}}(t_k) \subseteq \mathcal{E}$ is the set of mapped edges, and $O_i^{\mathcal{M}}(t_k)$ is the set of observation sets related to the mapped vertices and relevant robot-specific features. Robots operate and make decisions based solely on their local maps, which may differ in spatial coverage and detail depending on robot type. To establish this behavior, at each time step t_k, each robot-specific action type $\alpha \in \mathcal{C}_i$ is assigned a set of allowed vertices $V_i^{\alpha}(t_k) \subseteq V_i^{\mathcal{M}}(t_k)$. For most actions, this set consists of vertices on the local map such that the robot can traverse to from its current location and obtain a non-empty observation set. However, for some action types, there may be further constraints, such as the availability of a charging station at a candidate vertex when $\alpha = \texttt{charge}$. Then, the set of feasible actions is defined by:

$$A_i(t_k) = \{(\alpha, v) \mid \alpha \in \mathcal{C}_i,\ v \in V_i^{\alpha}(t_k)\} \tag{3.11}$$

It is further assumed that each action $(\alpha, v) \in A_i(t_k)$ is unique. As a design constraint, no action may be taken if the available energy is below the robot's critical energy threshold. All energy-related feasibility and updates are evaluated relative to the robot's most recent local state at time $t_{i,\tilde{k}_i}$. The energy constraint and update rule are therefore expressed as:

$$\epsilon_i(t_{i,\tilde{k}_i}) < \epsilon_i^{\mathsf{crit}} \Rightarrow A_i(t_{i,\tilde{k}_i}) = \emptyset \tag{3.12}$$

$$\epsilon_i(t_{i,\tilde{k}_i+1}) = \epsilon_i(t_{i,\tilde{k}_i}) - \delta_i^{\mathsf{energy}}\left(s_i(t_{i,\tilde{k}_i}), a\right), \quad a \in A_i(t_{i,\tilde{k}_i}) \tag{3.13}$$

The direct interpretation of the energy constraint provided in Constraint (3.12) yields that without the ability to replenish the individual energy levels $\epsilon_i(t_{i,\tilde{k}_i})$, the robots quickly become nonoperational, and the continuous autonomy of mission execution diminishes. To address this issue, an action type dedicated to charging is assumed to be included for every robot within the team:

$$\alpha^{\mathsf{charge}} \in \mathcal{C}_i,\ \forall i \in \{1, \ldots, N^{\mathsf{robot}}\} \tag{3.14}$$

$$\exists (\alpha^{\mathsf{charge}}, v) \in A_i(t_k), \quad v \in V_i^{\mathsf{charge}}(t_k), \forall i \in \{1, \ldots, N^{\mathsf{robot}}\} \tag{3.15}$$

where the Constraint (3.15) reformulates the critical energy constraint in terms of including a feasible charging action at every feasible action set $A_i(t_k)$, denoted by the feasible vertex set $V_i^{\mathsf{charge}}(t_k)$, which contains the vertices enabling charging that the robot can reach with sufficient energy at the time step t_k. In the formulation of this thesis, charging actions can only occur upon docking on the *central agent*. The central agent is configured as the base of operations for the robotic system, enabling charging for all robots, coordinating the global map, and overseeing mission objectives. Therefore, the set $V_i^{\mathsf{charge}}(t_k)$ at the time step t_k contains a subset of vertices that the central agent is scheduled to be in. In practice, this central agent can be a static hub that the USaR personnel set up along the hot zones; however, in the context of this thesis, it is regarded as another autonomous unit that evolves the task sets algorithmically and moves within the USaR zone by scheduling paths to improve search efficiency. There are two benefits to this approach: the first is that it allows for a completely autonomous system description, where the robots do not always assume that the charging location is static and can incorporate evolving situations into their decision-making, enabling the maximum amount of flexibility for the proposed system. The second benefit is that by intentionally relocating the central agent through the hot zone to get closer to areas of interest, a faster completion time for the mission can be achieved. Formally, the central agent state $s^{\mathsf{c}}(t_k)$ that is capable of coordinating the information of the robotic system is modeled as:

$$s^{\mathsf{c}}(t_k) = (v^{\mathsf{c}}(t_k), \mathcal{M}^{\mathsf{c}}(t_k), \mathcal{T}(t_k), \boldsymbol{\pi}^{\mathsf{c}}(t_k), H^{\mathsf{c}}(t_k)) \tag{3.16}$$

The state $s^{\mathsf{c}}(t_k)$ contains the vertex of the central agent as $v^{\mathsf{c}}(t_k)$, the global map $\mathcal{M}^{\mathsf{c}}(t_k)$, the set of available tasks $\mathcal{T}(t_k)$, the remaining schedule $\boldsymbol{\pi}^{\mathsf{c}}(t_k)$, and the number of scheduled waypoints left $H^{\mathsf{c}}(t_k)$ at time t_k. The global map $\mathcal{M}^{\mathsf{c}}(t_k)$ has the same formulation as the local maps. The local maps of robots may differ from one another, depending on the physical and computational requirements of each robot. Therefore, whenever they receive new observations, they update both their local maps and the global map. Conversely, as new areas are mapped, the central agent selectively sends updates to the local robot maps, enabling information flow between robots. Therefore, whenever a new action is taken at a time t_k, the central agent state is asynchronously updated to generate the new global map $\mathcal{M}^{\mathsf{c}}(t_{k+1})$ and the task set $\mathcal{T}(t_{k+1})$. Another source of asynchronicity is the central agent's schedule, $\boldsymbol{\pi}^{\mathsf{c}}(t_k)$ which consists of time and vertex pairings on a limited horizon that the central agent will follow. Therefore, the vertex of the central agent is updated whenever this schedule is met, expressed as follows at time t_k:

$$\boldsymbol{\pi}^{\mathsf{c}}(t_k) = \left[(t_1^{\mathsf{c}},\ v_1^{\mathsf{c}}) \quad \dots \quad (t_{|H^{\mathsf{c}}(t_k)|}^{\mathsf{c}},\ v_{|H^{\mathsf{c}}(t_k)|}^{\mathsf{c}})\right] \tag{3.17}$$

$$(t_1^{\mathsf{c}}, v_1^{\mathsf{c}}) = (t_k, v^c(t_k)) \tag{3.18}$$

Where t_j^{c} denotes the time the central agent plans to reach vertex v_j^{c}, where its state is updated accordingly, and $H^{\mathsf{c}}(t_k)$ denotes the remaining horizon in the schedule it follows at time t_k, for $j \in \{1, \dots, H^{\mathsf{c}}(t_k)\}$. When the central agent finishes its schedule, it generates a new one, details of which will be introduced in the following section. As an underlying assumption, robots can only charge if they are at the same vertex as the central agent simultaneously. To establish this behavior, t_{j+1}^{c} denotes the latest possible time for any robot to meet with the central agent at the vertex v_j^{c}. For completeness, $t_{H^{\mathsf{c}}(t_k)+1}^{\mathsf{c}} = \infty$ is considered by all robots to enable them to return to the final vertex of the schedule if they have sufficient energy. As a result, it is assumed that when the central agent reaches $v_{H^{\mathsf{c}}(t_k)}^{\mathsf{c}}$, the next scheduling time is calculated based on the last time any robot plans to reach this vertex, ensuring there are no scheduling conflicts that prevent the robots from returning to the central agent.

From this formulation of the central agent, the construction of the set $V_i^{\mathsf{charge}}(t_k)$ depends on which vertices $v_j^{\mathsf{c}} \in \pi^{\mathsf{c}}(t_k)$ can be reached before their corresponding deadlines t_{j+1}^{c} with sufficient energy, such that $\epsilon_i(t_{i,\tilde{k}_i}) \geq \epsilon_i^{\mathsf{crit}}$ always holds true for some $j \in \{1, \dots, H^c(t_k)\}$. These conditions can be formalized by:

Time and energy thresholds of the feasibility of $s_i(t_k)$ for safely reaching the charger waypoint j

$$\epsilon_{i,j}^{\mathsf{charge}}(s_i(t_k)) = \epsilon_i^{\mathsf{crit}} + \delta_i^{\mathsf{energy}}(s_i(t_k), (\alpha^{\mathsf{charge}}, v_j^{\mathsf{c}})), \quad \forall j \in \{1, \dots, H^{\mathsf{c}}(t_k)\} \tag{3.19}$$

$$t_{i,j}^{\mathsf{charge}}(s_i(t_k)) = t_{j+1}^{\mathsf{c}} - \delta_i^{\mathsf{time}}(s_i(t_k), (\alpha^{\mathsf{charge}}, v_j^{\mathsf{c}})), \quad \forall j \in \{1, \dots, H^{\mathsf{c}}(t_k)\} \tag{3.20}$$

Then, at the time step t_k for the robot i, the set $V_i^{\mathsf{charge}}(t_k)$ is defined as:

$$V_i^{\mathsf{charge}}(t_k) = \left\{v_j^{\mathsf{c}},\ j \in \{1, \dots, H^{\mathsf{c}}(t_k)\} \mid \epsilon_i(t_k) \geq \epsilon_{i,j}^{\mathsf{charge}}(s_i(t_k)),\ t_k \leq t_{i,j}^{\mathsf{charge}}(s_i(t_k))\right\} \tag{3.21}$$

The set of possible actions and their related observations model how the robots interact with the environment and participate in completing the USaR mission. This approach models the robots in the most generalizable way, and is congruent to how actual humans operate. Often in real-life scenarios, the observable outcomes of approaching a problem in different ways are considered and compared to find the best option for achieving a goal. Although the overall objective of a USaR mission usually does not change from initialization, the specific tasks assigned to achieve it dynamically evolve as the mission progresses. The set of tasks that are available at time t_k is denoted by $\mathcal{T}(t_k)$, and a task $\tau_l(t_k) \in \mathcal{T}(t_k)$, $l \in \{1, \dots, |\mathcal{T}(t_k)|\}$ is defined as a tuple:

$$\tau_l(t_k) = (v_l^{\mathsf{task}}, \phi_l^{\mathsf{task}}, F_l^{\mathsf{task}}) \tag{3.22}$$

where $v_l^{\mathsf{task}} \in \mathcal{V}$ is the vertex at which the task is located, $\phi_l^{\mathsf{task}} : \mathcal{S}_i \times \mathcal{A}_i \to \{0, 1\}$ is a Boolean indicator function that denotes whether the task can be successfully completed by robot i performing

action $a \in A_i(t_k)$ from its current state $s_i(t_k)$, and $F_l^{\text{task}} \subseteq F(v_l^{\text{task}})$ is the subset of environmental features associated with the task. The feature set F_l^{task} effectively denotes the type of its attached task, where two tasks m and n are considered of the same type if $F_m^{\text{task}} = F_n^{\text{task}}$, meaning task categories are specified by their associated semantic features. The task set $\mathcal{T}(t_k)$ is defined as the set of all attemptable tasks at the time t_k:

$$\mathcal{T}(t_{k+1}) = \Phi(\mathcal{T}(t_k), a), \quad a \in \{A_i(t_k) \mid i \in \{1, \dots, N^{\text{robot}}\}\} \tag{3.23}$$

Here, $\Phi(\cdot)$ denotes the task set update function, which modifies $\mathcal{T}(t_k)$ by removing any tasks successfully completed by actions that finished execution at time t_k. This ensures that only uncompleted and newly spawned tasks remain available in the system.

System Modeling Summary

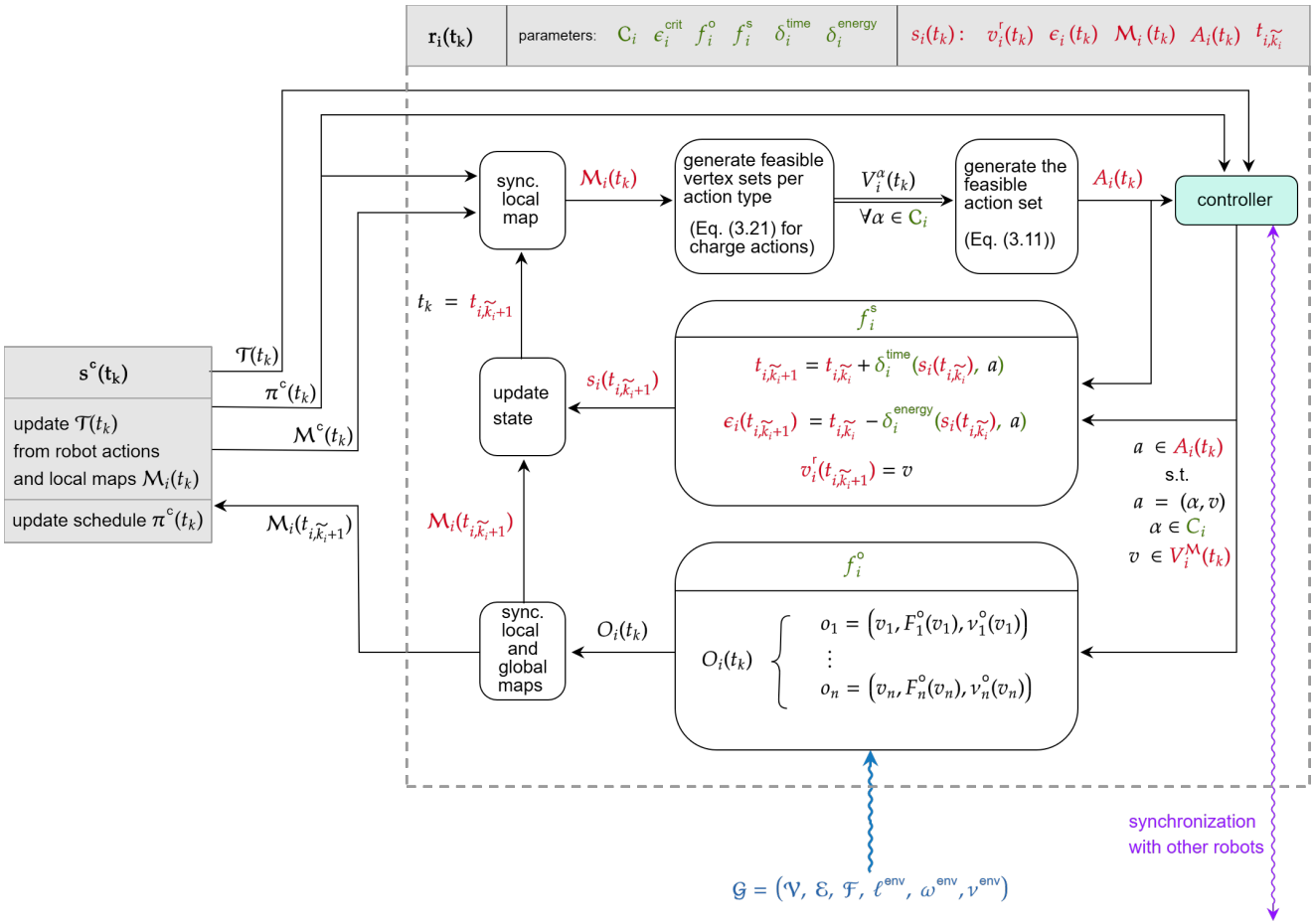


Figure 3.1: Summary of the mathematical modeling of SaR robots and the generalized system.

The summary of the interactions between a robot and the central agent, as well as the modules defined for the operation of the robots, are given in Figure 3.1. The single-line arrows represent inputs and outputs to the corresponding modules, where the thick line arrows represent transitions as a bundle. The wavy lines demonstrate environmental sensing and asynchronous communication with other robots. The flow of operations begins at the stage when the local time equals the global time, at which point the robot synchronizes its own local map with the global map. From the updated local map, sets of feasible vertices are generated for each action type to construct the feasible action sets. Then, the controller, which will be defined further in Section 3.2 takes the current task set, central agent schedule,

and the feasible action sets as the inputs to generate the next action of the robot. The robot state is updated for the next local time while simultaneously obtaining an observation set from the environment, and updating its local map as a result. When the local map is updated, the global map is also updated accordingly, enabling other robots to make up-to-date decisions. Furthermore, the central agent relies on these synchronization operations to update the current tasks, overseeing which ones are completed and which tasks need to be spawned, as well as updating its own schedule based on the states of the robots in the environment.

The summary of notations introduced in this section is given in Table 3.1 for reference, where each column contains the related notation for the ground truth variables, robot parameters, robot state, and central agent state, respectively. Within these columns, contextually relevant notations are clustered with consecutive cells in a gray background.

t_k	global time	$t_{i,\tilde{k}_i}$	local time of a robot i	$s_i(t_k)$	state of a robot i	$s^{\mathsf{c}}(t_k)$	central agent state
$\mathcal{G}$	environment graph	$r_i(t_k)$	robot i	$v_i^{\mathsf{r}}(t_k)$	vertex of a robot i	$v^{\mathsf{c}}(t_k)$	central agent vertex
$v \in \mathcal{V}$	environment vertices	$\mathcal{S}_i$	state space of a robot i	$\mathcal{M}_i(t_k)$	local map of a robot i	$H^{\mathsf{c}}(t_k)$	remaining schedule length of the central agent
$\mathcal{E}$	environment edges	$\alpha \in \mathcal{C}_i$	action types of a robot i	$V_i^{\mathcal{M}}(t_k)$	local map vertices of a robot i	$\boldsymbol{\pi}^{\mathsf{c}}(t_k)$	remaining schedule of the central agent
$\mathcal{F}$	environment feature set	$a = (\alpha, v) \in \mathcal{A}_i$	action and action space of a robot i	$E_i^{\mathcal{M}}(t_k)$	local map edges of a robot i	$\boldsymbol{\pi}_j^{\mathsf{c}}(t_k) = (t_j^{\mathsf{c}}, v_j^{\mathsf{c}})$	time and vertex pair for a waypoint j of the central agent
ℓ^{env}	vertex feature label function	f_i^{s}	state transition function of a robot i	$O_i^{\mathcal{M}}(t_k)$	local map vertex observations of a robot i	$\mathcal{M}^{\mathsf{c}}(t_k)$	global map
ω^{env}	edge weight function	f_i^{o}	observation function of a robot i	$\epsilon_i(t_k)$	energy level of a robot i	$\tau_l(t_k) \in \mathcal{T}(t_k)$	task l and the available task set
ν^{env}	vertex feature value function	$o \in \mathcal{O}_i$	observation and observation space of a robot i	$A_i(t_k)$	feasible action set of a robot i	v_l^{task}	vertex of a task l
N^{robot}	robot count	$F^{\mathsf{o}}(v)$	feature set of an observation on the vertex v	$V_i^{\alpha}(t_k)$	feasible vertex set per action type of a robot i	ϕ_l^{task}	completion condition of a task l
$\epsilon_{i,j}^{\text{charge}}$	energy feasibility threshold of a robot i to reach charger waypoint j	$\nu^{\mathsf{o}}(v)$	feature value set of an observation on the vertex v	δ_i^{time}	time expenditure of a robot i when an action is taken	F_l^{task}	feature set associated with a task l
$t_{i,j}^{\text{charge}}$	time of arrival deadline of a robot i to reach charger waypoint j	ϵ_i^{crit}	critical energy value of a robot i	δ_i^{energy}	energy loss of a robot i when an action is taken	Φ	transition function of the available task set

Table 3.1: Summary of notations included in the autonomous robot system model for SaR applications.

3.1.2. Task Allocation Modeling

Thus far, the system model is provided as a generalizable heterogeneous robotic team with asynchronous timings. This section aims to provide a decision-making model for assigning *utilities* to actions based on the tasks they attempt at a given state of the SaR mission. The designed relationship between actions, their resulting observations, and the relevant tasks in the current task set is demonstrated in Figure 3.2 for better contextualization of the proposed system. Here, five different tasks are depicted, with two distinct types corresponding to F_1 and F_2 feature sets. The complete set of actions the robot can take is denoted by $\mathcal{A}_i$ for each robot i, however, only actions one through three can attempt any task, therefore only these actions are considered within the current set of possible actions, $A_i(t_{i,\tilde{k}_i})$. In this specific example, some actions are able to make observations related to other types of features, such as a_n for F_3, however, since it does not relate to any task in the current timestep, this action is redundant. Having said that, the task set evolves with what other tasks are completed beforehand, and this specific action may be needed if a task related to F_3 spawns in a later timestep. The observations are local to the vertex of the taken action; therefore, only the tasks that spatially fall within the area of effect of the actions can be completed. This behavior is modeled by the indicator functions $\phi_l^{\text{task}}(s_i(t_k), a)$ of each task. Depending on the nature of each action, the resulting observation set may attempt multiple actions, as illustrated in the cases of a_1 and a_2, modeling the multitasking behavior of a robot. Similarly, a task can be attempted by multiple actions, as given in the case for τ_3, modeling different ways to approach the same task. These two behavior models cover all bases for the generalizable task allocation structure proposed in this thesis without relying on heuristic formulations.

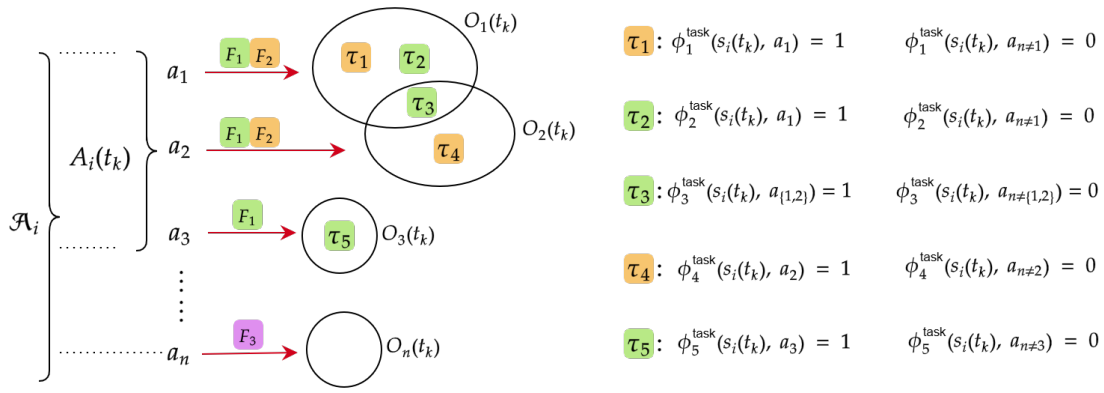


Figure 3.2: Representation of the relationship between robot actions, observations and available tasks.

As a result of this formulation, the robots need to rank every action in their current action set $A_i(t_{i,\tilde{k}_i})$ based on the overall utility it produces for the global mission goals. This utility depends on two independent factors, since the tasks are not attempted individually but through action observations. The first factor, therefore, is how well a robot can perform a specific action corresponding to the feature sets that the action produces. This is a standalone property of the robots, regardless of the tasks or actions they may attempt, due to the heterogeneous capabilities of each robot. For example, the same action may produce different quality of observations, be prone to errors due to the environmental constraints present at that time, or cover different-sized areas between robots. The second factor is how important an attempted task is from the perspective of the global USaR mission. This metric is independent of which robot attempts those tasks; rather, it aims to rank the importance of the same type of tasks relative to one another. The separation of action and task evaluations into a multi-stage fuzzy system further serves to prevent the rule explosion problem inherent to such approaches, as well as to improve modularity in the formalization of the robotic team. Section 3.1.2 introduces the fuzzy action and task evaluation architecture, where Section 3.1.2 continues with the introduction of the MCDM formulation to generate action utilities based on how well the fuzzy system outputs satisfy mission-level criteria while considering interactions between action outputs. The problem formulation is then concluded with the introduction of the general nonlinear MPC problem in Section 3.1.2, which defines the optimal control of the robotic team using the generalized mathematical model.

Fuzzy Action & Task Evaluation

The proposed fuzzy action-task evaluation system begins by introducing a capability metric for a robot performing an action, which models the difference in confidence in the abilities of different robots performing similar actions. Similarly, the same action performed by the same robot at different states may yield varying results, where this behavior is generalized through the aggregation of the quantity and quality of observations upon the execution of the said action. The action evaluation is then modeled by the fuzzy rules as follows:

$$\begin{aligned}\mathcal{R}_m^{\text{action}}: &\textbf{If } \sigma_i^{\text{cap}}(s_i(t_{i,\tilde{k}_i}),a) \text{ is } B_{m,1}^{\text{action}} \textbf{ and } \sigma_i^{\text{qual}}(s_i(t_{i,\tilde{k}_i}),a) \text{ is } B_{m,2}^{\text{action}} \textbf{ and} \\ &\sigma_i^{\text{quan}}(s_i(t_{i,\tilde{k}_i}),a) \text{ is } B_{m,3}^{\text{action}} \textbf{ then } \rho_i^{\text{action}}(s_i(t_{i,\tilde{k}_i}),a) \text{ is } C_m^{\text{action}}\end{aligned} \quad , \quad a \in A_i(t_{i,\tilde{k}_i}) \tag{3.24}$$

Where $\sigma_i^{\text{cap}}(s_i(t_{i,\tilde{k}_i}),a)$ is the capability of the robot i, $\sigma_i^{\text{qual}}(s_i(t_{i,\tilde{k}_i}),a)$ is a measure of the quality of observations and $\sigma_i^{\text{quan}}(s_i(t_{i,\tilde{k}_i}),a)$ is a measure of the quantity of observations upon performing the action $a \in A_i(t_{i,\tilde{k}_i})$ at the state $s_i(t_{i,\tilde{k}_i})$. These general terms cover all aspects of any type of action that can be taken by a robot, and enables application specific implementations within a formal framework. The sets $B_{m,\{1,2,3\}}^{\text{action}}$ and C_m^{action} are fuzzy sets that adopt a relevant linguistic term, with m denoting the fuzzy rule index. Likewise, the priority of a task $l \in \{1,\dots,|\mathcal{T}(t_{i,\tilde{k}_i})|\}$ is determined by the state of the current task set $\mathcal{T}(t_{i,\tilde{k}_i})$ where the task needs to have a measure of comparison between other tasks in terms of its relevant features F_l^{task}. The generalized fuzzy system for this evaluation is as follows, with fuzzy sets configured similarly to the action evaluation:

$$\mathcal{R}_{m,l}^{\text{priority}}: \textbf{If } \left(\bigwedge_{j\in\{1,\dots,|F_l^{\text{task}}|\}} \sigma_{l,j}^{\text{priority}}(\mathcal{T}(t_{i,\tilde{k}_i})) \text{ is } B_{m,l,j}^{\text{priority}} \right) \textbf{ then } \rho_l^{\text{priority}}(\mathcal{T}(t_{i,\tilde{k}_i})) \text{ is } C_m^{\text{priority}} \tag{3.25}$$

The priority evaluation of each task $\tau_l(t_{i,\tilde{k}_i})$ depends on the predetermined fuzzy rules for each type of task set, denoted by their unique feature sets F_l^{task}. Types of tasks may vary quite significantly within the evolving task set, and incorporating all relationships between different types of tasks to create an absolute ranking of tasks is practically infeasible. For this reason, this fuzzy system only evaluates the priority of each task within the same type of tasks, which are subject to the same rule sets. This is once again done for increasing modularity. A comparison of different types of tasks is performed at an upper hierarchy to address the issue of fair evaluation among all tasks currently considered by the robot. Finally, the value of attempting a task through an action for a robot is determined by the combination of the outputs of these two systems by another fuzzy model:

$$\begin{aligned}\mathcal{R}_m^{\text{task}}: &\textbf{If } \rho_i^{\text{action}}(s_i(t_{i,\tilde{k}_i}),a) \text{ is } C_m^{\text{action}} \textbf{ and } \rho_l^{\text{priority}}(\mathcal{T}(t_{i,\tilde{k}_i})) \text{ is } C_m^{\text{priority}} \textbf{ then} \\ &\nu_l^{\text{task}}(\mathcal{T}(t_{i,\tilde{k}_i}),s_i(t_{i,\tilde{k}_i}),a) \text{ is } f_m^{\text{task}}(\rho_i^{\text{action}},\rho_l^{\text{priority}})\end{aligned} \quad , \quad a \in A_i(t_{i,\tilde{k}_i}) \tag{3.26}$$

This final stage of the fuzzy system is configured as a Takagi-Sugeno system, as opposed to the Mamdani systems selected for the initial stages. This selection is intentional to maintain a continuous output surface, as the task evaluation values still need to be compared with respect to time and other types of tasks. The complete fuzzy system for action-task evaluation is given in Figure 3.3.

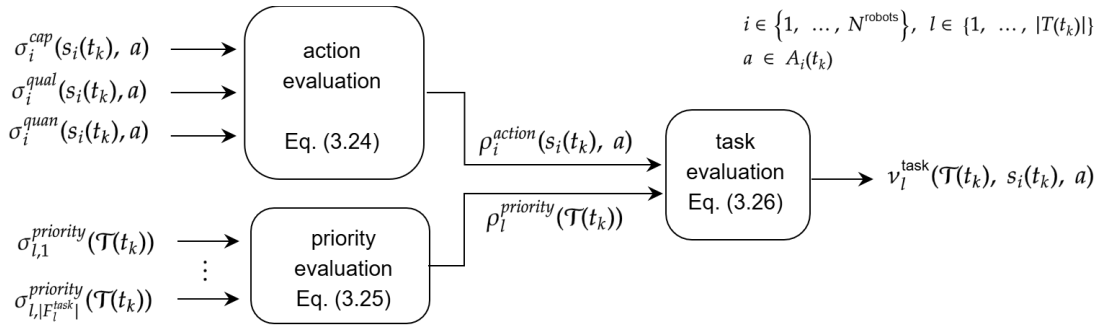


Figure 3.3: Fuzzy inference system determining the value of a task given a robot action

MCDM Formulation

The fuzzy inference system for action-task evaluation outputs a utility for each task attempted through an action. However, the final architecture must incorporate preferences across different types of tasks and account for time-related considerations. As previously addressed, these variables are often included as ad-hoc utility functions or within fuzzy systems in other works of the literature; however, this approach often fails when the types of spawned tasks and the amount of time it is feasible to assign to each task change dynamically throughout the mission. USaR missions often generate unforeseen or hybrid tasks; for example, a robot may need to place a communication beacon if the centralized task set considers it necessary during operation. If there is a robot capable of doing that, it needs to consider performing this task as well. Furthermore, tasks related to searching victims may be more time-sensitive than placing beacons, and mapping may be less important than searching for a victim at the later stages of the mission. Separating time costs from the fuzzy system additionally has the benefit of incorporating it as part of the hard constraints when the robot needs to return to a charging location safely, considering the charger's schedule. The MCDM method introduced by Basilico and Amigoni naturally accounts for all these factors without having to define an entire rule base from scratch, and avoids introducing additional heuristics as new types of tasks become available [7]. At a high level, MCDM first ranks all categories represented by the different types of task utilities present in the current task set, along with a time category introduced at this stage. Then, a Choquet integral is used to compute a weighted sum of incremental utility contributions, where each increment is weighted by the accumulated importance of the active subset of criteria. As a result, all task-related utilities obtained from the fuzzy inference system are aggregated into a single utility value that reflects system-wide priorities for robot decision-making at the local time $t_{i,\bar{k}_i}$.

Remark 3.1.2 (Choquet Integral and Utility Aggregation)**.** *The Choquet integral is a fuzzy aggregation operator used to capture interactions between criteria when multiple criteria are present in a decision-making problem. It can be viewed as a generalization of the Lebesgue integral, designed for situations where the underlying measure is not additive, but instead captures interactions between subsets of a finite set of variables. In a classical Lebesgue integral, the total contribution of a function is obtained by integrating over the values it takes, weighted by the additive measure of the domain regions exceeding those values. The Choquet integral replaces the additive measure with a monotonic set function that allows the weight assigned to a group of variables to reflect more than just the sum of their parts. This distinction is crucial when the criteria are not independent, for example, when the value of achieving two objectives simultaneously is not equal to the sum of their individual contributions. In such cases, the Choquet integral enables the modeling of synergistic or redundant interactions. This makes the Choquet integral especially suited for multi-criteria decision problems where mutual influence among criteria must be explicitly taken into account. The precise formulation of a generalized Choquet integral for decision-making problems is proposed by Khamis et al. [4] for further reference, and briefly introduced in the Section* **??**.

To aid understanding of the application of the Choquet integral in utility calculations, a small example is provided next. Let the individual utility values for some criteria $C^{\text{criteria}} = \{\alpha_1, \alpha_2, \alpha_3\}$ *be* $u_1 = 0.2$*,* $u_2 = 0.7$*, and* $u_3 = 0.5$ *respectively. Then, let the importance of the unique interactions between these criteria be defined as:*

$$\mu(\{\alpha_1\}) = 0.2, \quad \mu(\{\alpha_2\}) = 0.3, \quad \mu(\{\alpha_3\}) = 0.4,$$
$$\mu(\{\alpha_1, \alpha_2\}) = 0.6, \quad \mu(\{\alpha_1, \alpha_3\}) = 0.7, \quad \mu(\{\alpha_2, \alpha_3\}) = 0.8, \quad \mu(\{\alpha_1, \alpha_2, \alpha_3\}) = 1.$$

Then, the utilities are sorted in ascending order, where the subscript $()$ *denotes the ordering. The ordering and the corresponding criteria interaction sets are:*

$$u_{(1)} = 0.2, \;\; u_{(2)} = 0.5, \;\; u_{(3)} = 0.7$$
$$C_{(1)} = \{\alpha_1, \alpha_2, \alpha_3\}, \;\; C_{(2)} = \{\alpha_2, \alpha_3\}, \;\; C_{(3)} = \{\alpha_2\}$$

Finally, the discrete Choquet integral is calculated to find the aggregated MCDM utility, where $a_{(0)} = 0$*:*

$$\begin{aligned} u^{mcdm} &= \sum_{i=1}^{3} (u_{(i)} - u_{(i-1)}) \cdot \mu(C_{(i)}) \\ &= (0.2 - 0) \cdot \mu(\{\alpha_1, \alpha_2, \alpha_3\}) + (0.5 - 0.2) \cdot \mu(\{\alpha_2, \alpha_3\}) + (0.7 - 0.5) \cdot \mu(\{\alpha_2\}) \\ &= 0.2 \cdot 1 + 0.3 \cdot 0.8 + 0.2 \cdot 0.3 \\ &= 0.5 \end{aligned}$$

In the formulation of the thesis, a feature set $F \subseteq F(v)$ for a vertex $v \in \mathcal{V}$ consists of semantical labels that the corresponding vertex is associated with. As an example, depending on the capabilities present in the robotic team in terms of detecting information relevant to the unique labels, a feature set $F = \{\texttt{terrain}, \texttt{charger}, \texttt{victim}\}$ can denote the most basic functionalities a USaR robot may have; namely detecting terrain information from the associated vertex, whether or not a charger exists in that vertex and the capability of detecting victims from that vertex. The same approach is utilized in all formulations whenever a variable contains a unique feature set that defines it. A feature set F_l^{task} associated with a task $\tau_l(t_{i,\tilde{k}_i}) \in \mathcal{T}(t_{i,\tilde{k}_i})$ denotes the minimal set of features that an observation set $O_i(t_{i,\tilde{k}_i})$ needs to contain to be attempted by the robot i by performing the action $a \in A_i(t_{i,\tilde{k}_i})$. Following this interpretation, each unique feature set F_l^{task} at a given task set can be said to correspond to a unique *criterion* that each evaluated action needs to satisfy in terms of the detected features generated by the related observation. An overarching criterion in any type of decision-making during a SaR mission is the assessment of time to ensure the efficient allocation of resources in time-critical settings. To ensure the inclusion of time as a singular criterion on its own for the assessment of allocated tasks at a given decision-making step, an additional feature set $F^{\text{time}} = \{\texttt{time}\}$ is included for each task $\tau_l(t_{i,\tilde{k}_i})$ along with its own feature set F_l^{task}. Formally, the proposed MCDM stage begins by generating the unique criteria set $C(\mathcal{T}(t_{i,\tilde{k}_i}))$ of the current task set, with the addition of the time criterion:

$$C(\mathcal{T}(t_{i,\tilde{k}_i})) = F^{\text{time}} \cup \left\{ F_l^{\text{task}} \mid l \in \{1, \dots, |\mathcal{T}(t_{i,\tilde{k}_i})|\} \right\} \tag{3.27}$$

Then, a fuzzy measure $\mu : 2^{C(\mathcal{T}(t_{i,\tilde{k}_i}))} \to [0,1]_{\mathbb{R}}$ within the power set of the criteria set assigns an importance to each group of criteria, satisfying the following conditions:

$$\mu(\emptyset) = 0, \quad \mu(C(\mathcal{T}(t_{i,\tilde{k}_i})) = 1, \quad C_m \subseteq C_n \subseteq C(\mathcal{T}(t_{i,\tilde{k}_i})) \Rightarrow \mu(C_m) \leq \mu(C_n) \tag{3.28}$$

Considering that the fuzzy measure μ describes the dependencies between the criteria, the set $C_m \subseteq C(\mathcal{T}(t_{i,\tilde{k}_i}))$ is independent if $\mu(C_m) = \sum_{c \in C_m} \mu(c)$ holds. Then, for each individual criterion in this set, the action utility vector $\boldsymbol{u}_i^{\text{action}}(t_{i,\tilde{k}_i}, a)$ for the robot i with the candidate action $a \in A_i(t_{i,\tilde{k}_i})$ is calculated by summing all fuzzy utilities that the action considers corresponding to that criterion:

$$\boldsymbol{u}_i^{\text{action}}(t_{i,\tilde{k}_i}, a) = \left\langle \sum_{l=1}^{|\mathcal{T}(t_{i,\tilde{k}_i})+1|} \phi_l^{\text{task}}\left(s_i(t_{i,\tilde{k}_i}), a\right) \cdot \mathbf{1}_c(F_l^{\text{task}}) \cdot \nu_l^{\text{task}}\left(\mathcal{T}(t_{i,\tilde{k}_i}), s_i(t_{i,\tilde{k}_i}), a\right) \right\rangle_{c \in C(\mathcal{T}(t_{i,\tilde{k}_i}))} \tag{3.29}$$

Where the operator $\langle\rangle_{x \in X}$ denotes building an ordered set as a vector by iterating a given set X, the indicator function $\mathbf{1}_c : 2^{\mathcal{F}} \to \{0,1\}$ denotes if a given task has a matching label to the criteria $c \in C(\mathcal{T}(t_{i,\tilde{k}_i}))$. The action utility vector effectively contains the aggregated sum of all tasks that correspond to the same criteria, with a one-to-one correspondence to each entry $c \in C(\mathcal{T}(t_{i,\tilde{k}_i}))$ considered as a result of a candidate action. While this is a mathematically rigorous way of representing the action utility vector, in practice, only calculating the tasks that fall within the action observation set is sufficient. The additional entry for the time criteria F^{time} needs to be formulated in terms of the generalized task variables to complete the formalization. To that end, the time utility is incorporated as a meta-task, for which the robots do not directly attempt it; however, a utility value needs to be generated, similar to the tasks the robot considers. The following values are set to account for the time utility incorporation:

$$\phi^{\tau}_{|\mathcal{T}(t_{i,\tilde{k}_i})+1|}(s_i(t_{i,\tilde{k}_i}), a) = 1 \tag{3.30}$$

$$F^{\text{task}}_{|\mathcal{T}(t_{i,\tilde{k}_i})+1|} = F^{\text{time}} \tag{3.31}$$

$$\nu^{\tau}_{|\mathcal{T}(t_{i,\tilde{k}_i})+1|}(\mathcal{T}(t_{i,\tilde{k}_i}), s_i(t_{i,\tilde{k}_i}), a) = 1 - \min\left(1,\ \frac{\delta_i^{\text{time}}(s_i(t_{i,\tilde{k}_i}), a) - \delta t^{\min}}{\delta t^{\max} - \delta t^{\min}}\right) \tag{3.32}$$

where $\delta t^{\min}$ and $\delta t^{\max}$ are the normalization values for the time the robot spends performing the evaluated action. These normalization values depend on the context the utilities are generated in, for example, while local controllers only need to consider the time scaling between all currently evaluated actions $A_i(t_{i,\tilde{k}_i})$; global controllers may need to take into account the total elapsed time t_k and normalize according to an absolute time of performing the action. At the final stage of the MCDM utility calculation of a given action, the discrete Choquet integral over the fuzzy utility vector $\boldsymbol{u}_i^{\text{action}}(t_{i,\tilde{k}_i}, a)$ and corresponding criteria set $C(\mathcal{T}(t_{i,\tilde{k}_i}))$ is expressed as follows for the robot i:

MCDM utility calculation through robot actions

$$u_i(t_{i,\tilde{k}_i}, a) = \sum_{l=1}^{|C(\mathcal{T}(t_{i,\tilde{k}_i}))|} \mu(C_{i,(l)}) \cdot \left(\boldsymbol{u}^{\text{action}}_{i,(l)}(t_{i,\tilde{k}_i}, a) - \boldsymbol{u}^{\text{action}}_{i,(l-1)}(t_{i,\tilde{k}_i}, a)\right) \tag{3.33}$$

where $(l) \in \{1, \ldots, |C(\mathcal{T}(t_{i,\tilde{k}_i}))|\}$ indicates the criterion at the l^{th} order according to the sorted order that satisfies $\boldsymbol{u}^{\text{action}}_{i,(1)}(t_{i,\tilde{k}_i}, a) \leq \cdots \leq \boldsymbol{u}^{\text{action}}_{i,(|C(\mathcal{T}(t_{i,\tilde{k}_i}))|)}(t_{i,\tilde{k}_i}, a) \leq 1$ and $\boldsymbol{u}^{\text{action}}_{i,(0)} = 0$. Finally, the set $C_{i,(l)}$ indicates the aggregated criteria so far during the sum, denoted as:

$$C_{i,(l)} = \left\{ m \in \{1, \ldots, |C(\mathcal{T}(t_{i,\tilde{k}_i}))|\} \mid \boldsymbol{u}^{\text{action}}_{i,(l)}(t_{i,\tilde{k}_i}, a) \leq \boldsymbol{u}^{\text{action}}_{i,(m)}(t_{i,\tilde{k}_i}, a) \leq \boldsymbol{u}^{\text{action}}_{i,(|C(\pi^s(t_{i,\tilde{k}_i}))|)}(t_{i,\tilde{k}_i}, a) \right\} \tag{3.34}$$

fuzzy system		MCDM system		
$\rho_i^{\text{cap}}(s_i(t_k), a)$	capability rating of the robot i executing action a from state $s_i(t_k)$	$C(\mathcal{T}(t_k))$		criteria set of the current task set containing unique task feature sets
$\rho_i^{\text{qual}}(s_i(t_k), a)$	quality rating for the observations of the robot i executing action a from state $s_i(t_k)$	$\mu(C),\ C \subseteq C(\mathcal{T}(t_k))$		importance of the joint contribution of the criteria subset C
$\rho_i^{\text{quan}}(s_i(t_k), a)$	quantity rating for the observations of the robot i executing action a from state $s_i(t_k)$	$\boldsymbol{u}_i^{\text{action}}(t_k, a)$		aggregated utility vector per criteria $c \in C(\mathcal{T}(t_k))$ of using observations resulting from action a of the robot i, using the fuzzy system outputs
$\rho_{l,m}^{\text{priority}}(\mathcal{T}(t_k))$	priority rating of the task l within the current task set for each task feature $m \in \{1, \ldots, \lvert F_l^{\text{task}}\rvert\}$	$u_i(t_k, a)$		MCDM utility result of robot i taking action a, as the output of the discrete Choquet integral taken on $\boldsymbol{u}_i^{\text{action}}(t_k, a)$ over the criteria set $C(\mathcal{T}(t_k))$
$\nu_l^{\text{task}}(\mathcal{T}(t_k), s_i(t_k), a)$	output of the fuzzy system for the task l when the robot i performs the action a	**safe return constraints**	$z_{i,j}^{\text{charge}}(t_k)$	binary variables denoting the robot i can reach the central agent waypoint j from its current state
			$M^{\text{time}},\ M^{\text{energy}}$	big-M constants for time and energy constraints

Table 3.2: Summary of notations included in the fuzzy-MCDM decision-making framework.

Optimization Problem Formulation

Having established the mathematical models for the robot-environment interactions and the utility-based task allocation framework through possible robot actions, the generalized USaR problem can be expressed as an optimization problem with safe return constraints. By definition, the safe return capabilities are directly tied to the feasible charging vertex set $V_i^{\text{charge}}(t_k)$ being non-empty for all robots at all times. This condition can be formulated linearly by introducing a binary variable $z_{i,j}^{\text{charge}}(t_k) \in \{0,1\}$ to denote whether the vertex $v_j^{\text{c}} \in \pi^{\text{c}}(t_k)$ is included in the set $V_i^{\text{charge}}(t_k)$ at the time step t_k. Then, the overall USaR objective as an optimization problem, maximizing the total utility generated by the team of robots through allocating tasks via their actions across all global time steps, can be formulated as follows:

Generalized MCDM task allocation as an optimization problem formulation

$$\underset{\{a_i \in A_i(t_k)\}_{i=1}^{N^{\text{robot}}}}{\text{argmax}} \sum_{k \in \mathcal{K}} \sum_{i=1}^{N^{\text{robot}}} u_i(t_k, a_i) \tag{3.35}$$

s.t.

$$t_k - t_{i,j}^{\text{charge}}\left(s_i(t_k)\right) \leq M^{\text{time}} \cdot \left(1 - z_{i,j}^{\text{charge}}(t_k)\right), \qquad \forall i, \forall j, \forall k \tag{3.36}$$

$$\epsilon_{i,j}^{\text{charge}}(s_i(t_k)) - \epsilon_i(t_k) \leq M^{\text{energy}} \cdot \left(1 - z_{i,j}^{\text{charge}}(t_k)\right), \qquad \forall i, \forall j, \forall k \tag{3.37}$$

$$\sum_{j=1}^{H^{\text{c}}(t_k)} z_{i,j}^{\text{charge}}(t_k) \geq 1, \qquad \forall i, \forall k \tag{3.38}$$

$$t_k \neq t_{i,\tilde{k}_i} \Rightarrow a_i = \left(\alpha^{\text{idle}}, v_i^{\text{r}}\left(t_{i,\tilde{k}_i}\right)\right), \qquad \forall i, \forall k \tag{3.39}$$

$$\{\alpha^{\text{charge}}, \alpha^{\text{idle}}\} \in \mathcal{C}_i, \qquad \forall i \tag{3.40}$$

Where M^{time} and M^{energy} denote sufficiently large big-M constants, and $\mathcal{K} = \{k \in \mathbb{N} \mid t_k \leq t^{\text{end}}\}$ denotes the set of all possible global time steps before the end of mission time t^{end} is reached. Big-M constraints are a modeling technique in mixed-integer programming where a large constant is used to conditionally activate or deactivate parts of a constraint based on the value of a binary variable. This thesis considers the subscripts i, j, and l to denote robot, charger schedule, and task/criteria indices respectively; denoted by $i \in \{1, \ldots, N^{\text{robot}}\}$, $j \in \{1, \ldots, H^{\text{c}}(t_k)\}$, $l \in \{1, \ldots, |\mathcal{T}(t_k)|\}$ when used in the context of task iterations and $l \in \{1, \ldots, |C(\mathcal{T}(t_k))|\}$ when used in the context of MCDM criteria. The Constraints (3.36), (3.37), and (3.38) formulate that at least one safe return action is always feasible for each robot i at all time steps k. The Constraint (3.39) establishes the correctness of the expression in the Problem (3.35) by assigning `idle` actions whenever robots are out of sync with the global time t_k, such that the corresponding utility is always defined. Moreover, the Constraint (3.40) ensures both `charge` and `idle` types of actions are provided for all robots. In addition to the above considerations, the system dynamics follow the asynchronous update rules introduced in Equations (3.5) and (3.7). At each global time $t_k \in \mathcal{K}$, only the robots that $t_k = t_{i,\tilde{k}_i}$ will complete their actions, update their states via their transition function, and generate utility. The utility function $u_i(t_k, a_i)$ corresponds to the MCDM-evaluated utility of the action completed by the robot i at the global time t_k when the action $a_i \in A_i(t_k)$ is taken. Due to the asynchronous nature of robot actions, most robots will not complete an action at each global time step. For those robots, the utility is defined as $u_i(t_k, a_i) = 0$. Since robots plan their next action within their local times, they may attempt to tackle the same tasks simultaneously, or generate a mismatch in the expected utility before taking an action when another robot has already completed those tasks during parallel operation. Furthermore, a global solution to this optimization problem is both intractable and infeasible since the global task set dynamically evolves with each action. To address both aspects of the generalized optimization problem, a hierarchical MPC-based controller architecture is presented in the next section.

3.2. Control Architecture

The system model described in Section 3.1.1 requires a control architecture to generate robot actions that maximize the total MCDM utility at the end of the SaR mission, as described in the Problem (3.35). Directly solving this problem is not possible, as addressed, due to the task sets at each global timestep evolving as a result of the robot actions performed, and robots executing their individual actions asynchronously with one another. To address the issue of robot synchronization in terms of global optimality, a hierarchical control architecture is proposed to balance decentralized robot-level autonomy with global coordination, enabling robots to reason independently while maintaining coherent team behavior under dynamically evolving task conditions. This hierarchical control architecture in the context of the designed mathematical formulation of the robotic SaR team is demonstrated in Figure 3.4. Each robot employs its own local controller, which generates a candidate action sequence $\pi_i^{\mathrm{a}}(t_k)$ from the current task state $\mathcal{T}(t_k)$, the central agent's schedule $\pi^{\mathrm{c}}(t_k)$, and the feasible action set $A_i(t_k)$ of the robot i. The global controller then ensures that all current robot schedules are non-conflicting and optimal from the perspective of global mission goals, since the local controllers are unaware of the schedules of other robots. Both controllers, at their core, integrate fuzzy decision-making and multi-criteria utility estimation, as described in Section 3.1, into a model predictive optimization problem with safe return constraints. MPC formulations are naturally suited when the underlying problems are unpredictable, and a limited look-ahead into the future states of the system is necessary due to the sequential development of the state space. Therefore, a nonlinear MPC problem formulation is introduced in Section 3.2.1 at the local level for each robot, and then the conditions for linearizing this optimization problem are described as a MILP optimization problem for computational tractability. Section 3.2.2 describes the upper hierarchy as the global coordination controller of the robotic team, which utilizes the optimized schedules resulting from the local controllers to resolve conflicting task allocations under robot coordination conditions. This hierarchical division of control enables scalable operation across heterogeneous teams and supports mission continuity by explicitly incorporating safe return behaviors and energy-aware planning. Finally, a controller for the central agent scheduling is introduced in Section 3.2.3 to induce dynamic behaviors within the SaR environment and establish a fully autonomous closed-loop heterogeneous robotic system.

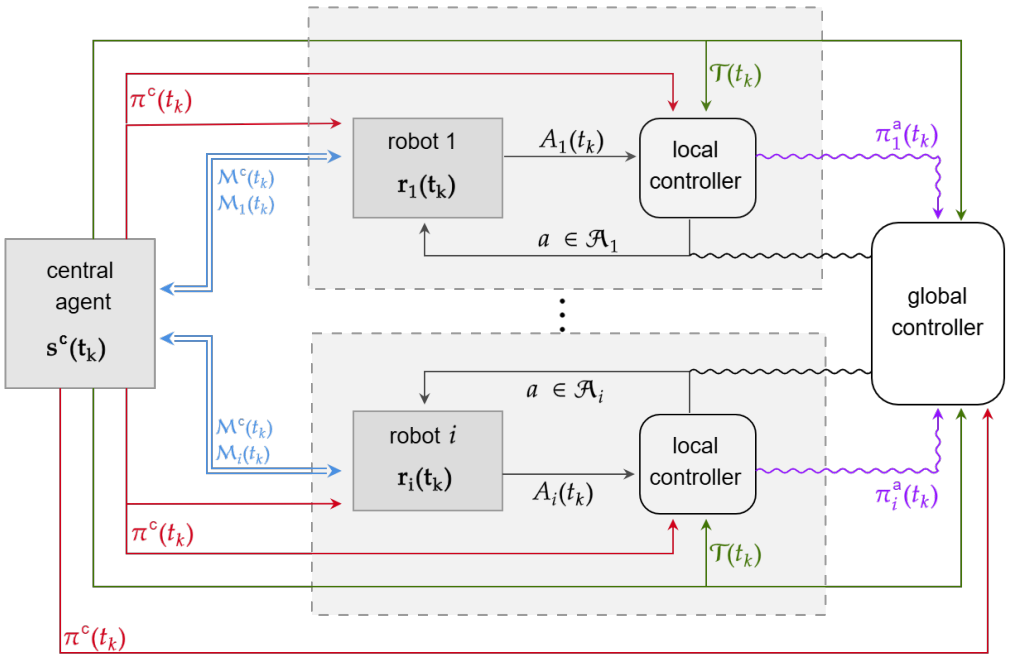


Figure 3.4: Diagram of the proposed hierarchical control architecture.

3.2.1. Local Robot Controllers

To solve the optimization problem defined in Section 3.1, each robot needs to be able to make its own decisions independently from other robots in the environment. As discussed earlier, an offline solution to this problem is infeasible due to the dynamically changing task sets throughout the mission. To address this point, an MPC reformulation of the original problem is proposed for each robot at the local control level. The finite-horizon optimization considers the current state of the system and aims to find the best sequence of actions by estimating the future states within the horizon $H_i^{\mathsf{pred}} \in \mathbb{N}^+$ of a robot. At each global time t_k, when a robot i completes an action, the robot triggers its local planner, effectively meaning at the planning stage, $t_k = t_{i,\tilde{k}_i}$ is satisfied. Additionally, the robots do not need to consider the entirety of the global task set since only the prediction horizon amount of actions can be considered at the local stage. Thus, $\mathcal{T}_i(t_k) \subseteq \mathcal{T}(t_k)$ denotes the subset of global tasks that the robot i considers for the finite-horizon optimization, thereby reducing the action space related to the task sets. As a result, each robot needs to estimate its future states $\hat{s}_i(t_{i,\tilde{k}_i+\kappa})$ and an approximation of the future task sets $\widehat{\mathcal{T}}_i(t_{i,\tilde{k}_i+\kappa})$, where $\kappa \in \{0, \dots, H_i^{\mathsf{pred}} - 1\}$ denotes the estimation step. Therefore, starting from the feasible task set at the initial timestep $\mathcal{T}_i(t_{i,\tilde{k}_i})$, the task set estimator takes the form of:

$$\widehat{\mathcal{T}}_i(t_{i,\tilde{k}_i+\kappa+1}) = \widehat{\Phi}_i\left(\widehat{\mathcal{T}}_i(t_{i,\tilde{k}_i+\kappa}),\, a\right), \quad a \in A_i(t_{i,\tilde{k}_i+\kappa}) \tag{3.41}$$

Where the function $\widehat{\Phi}_i(\cdot)$ approximates the global task set update function $\Phi(\cdot)$ by only considering the effects of the robot's actions to satisfy the locality of the controller design, likewise, the extended state $\boldsymbol{\pi}_i^{\mathsf{s}}(t_{i,\tilde{k}_i})$ and control inputs $\boldsymbol{\pi}_i^{\mathsf{a}}(t_{i,\tilde{k}_i})$ are formulated along the prediction horizon as follows:

$$\boldsymbol{\pi}_i^{\mathsf{s}}(t_{i,\tilde{k}_i}) = \left[\left(\hat{s}_i(t_{i,\tilde{k}_i}),\, \widehat{\mathcal{T}}_i(t_{i,\tilde{k}_i})\right) \quad \dots \quad \left(\hat{s}_i(t_{i,\tilde{k}_i+H_i^{\mathsf{pred}}-1}),\, \widehat{\mathcal{T}}_i(t_{i,\tilde{k}_i+H_i^{\mathsf{pred}}-1})\right)\right] \tag{3.42}$$

$$\boldsymbol{\pi}_i^{\mathsf{a}}(t_{i,\tilde{k}_i}) = \left[a \in \widehat{A}_i(t_{i,\tilde{k}_i}) \quad \dots \quad a \in \widehat{A}_i(t_{i,\tilde{k}_i+H_i^{\mathsf{pred}}-1})\right] \tag{3.43}$$

Then, the MPC optimization problem for local robot controllers can be expressed as:

MPC reformulation of the generalized task allocation problem for local robot controllers at the global time t_k

$$\underset{\boldsymbol{\pi}_i^{\mathsf{s}}(t_{i,\tilde{k}_i}),\boldsymbol{\pi}_i^{\mathsf{a}}(t_{i,\tilde{k}_i})}{\operatorname{argmax}} \sum_{\kappa=0}^{H_i^{\mathsf{pred}}-1} u_i\left(t_{i,\tilde{k}_i+\kappa}, \boldsymbol{\pi}_{i,\kappa+1}^{\mathsf{a}}(t_{i,\tilde{k}_i})\right) \tag{3.44}$$

s.t.

$$t_{i,\tilde{k}_i+\kappa} - t_{i,j}^{\mathsf{charge}}\left(\hat{s}_i(t_{i,\tilde{k}_i+\kappa})\right) \le M^{\mathsf{time}} \cdot \left(1 - z_{i,j}^{\mathsf{charge}}(t_{i,\tilde{k}_i+\kappa})\right), \qquad \forall j, \forall\kappa \tag{3.45}$$

$$\epsilon_{i,j}^{\mathsf{charge}}\left(\hat{s}_i(t_{i,\tilde{k}_i+\kappa})\right) - \epsilon_i(t_{i,\tilde{k}_i+\kappa}) \le M^{\mathsf{energy}} \cdot \left(1 - z_{i,j}^{\mathsf{charge}}(t_{i,\tilde{k}_i+\kappa})\right), \qquad \forall j, \forall\kappa \tag{3.46}$$

$$\sum_{j=1}^{H^{\mathsf{c}}(t_k)} z_{i,j}^{\mathsf{charge}}(t_{i,\tilde{k}_i+\kappa}) \ge 1, \qquad \forall\kappa \tag{3.47}$$

$$t_{i,\tilde{k}_i} = t_k \tag{3.48}$$

$$\{\alpha^{\mathsf{charge}}, \alpha^{\mathsf{idle}}\} \in \mathcal{C}_i \tag{3.49}$$

The evaluation of the objective function given in Problem (3.44) requires the nonlinear estimation of the state $\hat{s}_i(t_{i,\tilde{k}_i+\kappa})$, and the feasible task set $\widehat{\mathcal{T}}_i(t_{i,\tilde{k}_i+\kappa})$ at each timestep $t_{i,\tilde{k}_i+\kappa}$, where both depend on the previous action taken at the time $t_{i,\tilde{k}_i+\kappa-1}$. As a result, the objective function must be calculated sequentially, starting from the robot's current local time $t_{i,\tilde{k}_i} = t_k$. The objective function provided in Problem (3.44) is the reformulation of the MCDM utility aggregation of the robotic team presented in Problem (3.35) for establishing a prediction-based finite horizon problem to decide the next action of individual robots. The exact calculation of the utility function is provided in Equation (3.33). The Constraints (3.45), (3.46), (3.47) correspond to the safe return conditions such that there exists at least

one valid charging action throughout the prediction horizon, such that the robot can always rendezvous with the central agent, depending on its schedule. These constraints follow the exact same formulation of the generalized MCDM task allocation problem, specifically the Constraints (3.36) , (3.37), (3.38). Similarly, the Constraint (3.48) sets the starting time of the prediction horizon to the global time t_k since the decision-making times are synchronized through the local robot times, and the Constraint (3.49) ensures `charge` and `idle` actions must always be included in the capabilities of the robots in the team. To ensure correct and reliable operations, all of these constraints are regarded as *hard* constraints that always need to be satisfied. Since charging actions are always present, the problem remains feasible, allowing the robot to always return to the charger if no other action satisfies the first three constraints.

Remark 3.2.1 (Local Robot Controller Assumptions)**.** *The set of available actions at each time step is constrained by both the charging constraints and the task set, as only actions relevant to the current tasks generate a positive utility. For these reasons, some simplifying assumptions are made regarding the evolution of the task set. First, the task set update estimate function $\widehat{\Phi}_i(.)$ only models the evolution of the tasks that are present in the initial task set $\mathcal{T}(t_k)$, meaning it does not try to predict what new tasks may be generated along the prediction horizon. This is established due to the added complexity of estimating not only how the central agent creates tasks at a given time, but also to avoid predicting the states of all other robots within the team, which may result in the creation of new tasks. The robots may still consider the same tasks within their individual local horizons, and this aspect is addressed in the upper hierarchy, where the global cooperation optimization takes place. The second assumption is that the creation of the initial task set estimation $\widehat{\mathcal{T}}_i(t_{i,\tilde{k}_i})$ is treated as a preprocessing step, in which only a reasonable number of tasks are filtered for consideration (such as a constant number of tasks that are closest to the robot at the local time $t_{i,\tilde{k}_i}$). The charging constraints of the MPC optimizer are also enforced during the preprocessing step, ensuring that all tasks within the preprocessed set are related to at least one feasible action. If no such actions are taken, the robot must return to the central agent for charging.*

Despite its flexibility, the nonlinear MPC model poses challenges for real-time use due to its dynamic and nonconvex nature. The feasible actions and task sets evolve in response to predicted states, and repeated task attempts within the horizon necessitate bookkeeping to avoid redundant actions in terms of generated observational overlaps. Moreover, the sorting operation during MCDM calculations introduces further non-convexity via sorting and criteria aggregation. To address this, a two-stage MILP approximation is proposed, as provided in Figure 3.5. In the first stage, a subset of actions is generated across the prediction horizon by selecting those that are sufficiently independent in space and task scope, and achieve high estimated MCDM utility at the initial timestep. This approximation assumes that early utility estimates provide a reliable proxy for long-term contribution, which is reasonable under moderately dynamic task environments. In the second stage, a prediction horizon-length action sequence is selected from this set under time and energy constraints, utilizing a MILP-based linearization of the nonlinear problem. The advantages of this approach are twofold: First, it enables fast solution times while preserving the core features such as utility prioritization, task assignment, and safe return constraints. Second, modern MILP solvers effectively leverage the strengths of branch-and-bound and cutting planes algorithms to tackle highly branching sequential decision-making problems.

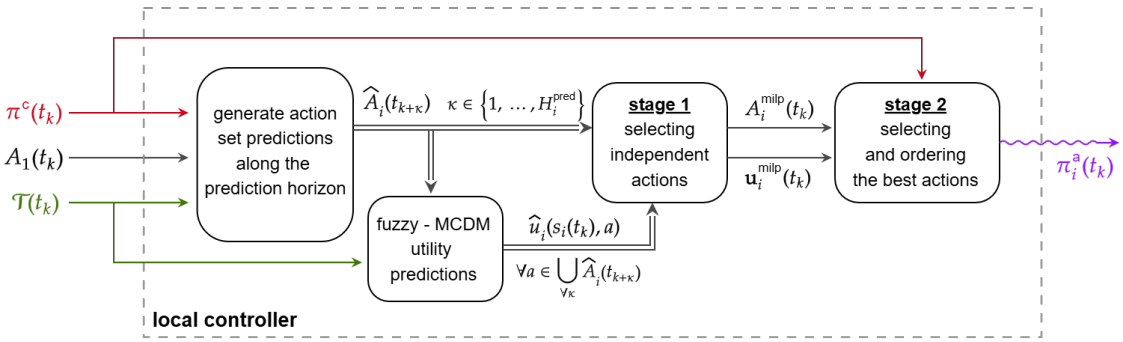


Figure 3.5: Diagram of the proposed local robot controllers.

First Stage: Independent Action Filtering

The first stage of the proposed local controllers starts by defining the dependence between the criteria across all possible actions determined from the initial prediction time $t_{i,\tilde{k}_i}$, denoted by the set $A_i^{\text{pred}}(t_{i,\tilde{k}_i}) \subseteq \mathcal{A}_i$:

$$A_i^{\text{pred}}(t_{i,\tilde{k}_i}) = \bigcup_{\kappa=0}^{H_i^{\text{pred}}-1} \hat{A}_i(t_{i,\tilde{k}_i+\kappa}) \tag{3.50}$$

where the sets $\hat{A}_i(t_{i,\tilde{k}_i+\kappa})$ denote the predicted feasible action sets along the prediction horizon. The actions generate task utilities through their predicted observations, therefore let two observation tuples satisfy $o_1 = o_2$ if and only if $v_1^{\text{o}} = v_2^{\text{o}}$ holds. Two observation vertices are considered equal if their locations and associated features $F(v^{\text{o}})$ are the same. Then, the observation sets related to two distinct actions $O_1 = f^{\text{o}}(s_i(t_{i,\tilde{k}_i}), a_1)$ and $O_2 = f^{\text{o}}(s_i(t_{i,\tilde{k}_i}), a_2)$ are independent if $O_1 \cap O_2 = \emptyset$. The vertex v^{o} contains the features that match the requirements of a task that the action is intended for, and therefore, if two vertices are equivalent between action measurement predictions, there is a dependence between them. With this approach, it is assumed that only one measurement from a specific location-feature pair is needed within one prediction horizon; therefore, completing the first action partially or fully satisfies the tasks associated with the shared vertex-feature pair, thereby reducing the marginal utility of the second.

Let $O_{i,m}^{\text{pred}}(t_{i,\tilde{k}_i}) = \hat{f}_i^{\text{o}}(s_i(t_{i,\tilde{k}_i}), a_{i,m})$ denote the predicted observation set associated with each action $a_{i,m} \in A_i^{\text{pred}}(t_{i,\tilde{k}_i})$, where $m \in \{1, \ldots, |A_i^{\text{pred}}(t_{i,\tilde{k}_i})|\}$. For each observation $o_n \in O_{i,m}^{\text{pred}}(t_{i,\tilde{k}_i})$, $n \in \{1, \ldots, |O_{i,m}^{\text{pred}}(t_{i,\tilde{k}_i})|\}$ of individual actions, the corresponding utility value $u_{i,m,n}^{\text{pred}}(t_{i,\tilde{k}_i})$ contains marginal MCDM utility predictions, computed as:

$$\bar{u}_{i,m,n}(t_{i,\tilde{k}_i}) = \frac{\hat{u}_i\left(s_i(t_{i,\tilde{k}_i}), a_{i,m}(t_{i,\tilde{k}_i})\right)}{|O_{i,m}^{\text{pred}}(t_{i,\tilde{k}_i})|} \tag{3.51}$$

for all observations n in $O_{i,m}^{\text{pred}}(t_{i,\tilde{k}_i})$. The utility predictions represented by $\hat{u}_i(s_i(t_{i,\tilde{k}_i}), a_{i,m})$ exclude the time-meta task term from Equation (3.33) since time-based preferences will be incorporated in the second stage, and this stage only aims to generate independent action sets and their predicted marginal MCDM utilities. Then, the full predicted observation set and corresponding aggregated utilities over unique observations are defined as:

MCDM utility prediction calculation per unique observation along the prediction horizon

$$\mathcal{O}_i^{\text{pred}}(t_{i,\tilde{k}_i}) = \bigcup_{m=1}^{|A_i^{\text{pred}}(t_{i,\tilde{k}_i})|} O_{i,m}^{\text{pred}}(t_{i,\tilde{k}_i}) \tag{3.52}$$

$$\boldsymbol{u}_i^{\text{pred}}(t_{i,\tilde{k}_i}) = \left\langle \frac{\sum_{m=1}^{|A_i^{\text{pred}}(t_{i,\tilde{k}_i})|} \sum_{n=1}^{|O_{i,m}^{\text{pred}}(t_{i,\tilde{k}_i})|} \bar{u}_{i,m,n}(t_{i,\tilde{k}_i}) \cdot \mathbf{1}_o(O_{i,m,n}^{\text{pred}}(t_{i,\tilde{k}_i}))}{\sum_{m=1}^{|A_i^{\text{pred}}(t_{i,\tilde{k}_i})|} \sum_{n=1}^{|O_{i,m}^{\text{pred}}(t_{i,\tilde{k}_i})|} \mathbf{1}_o(O_{i,m,n}^{\text{pred}}(t_{i,\tilde{k}_i}))} \right\rangle_{\forall o \in \mathcal{O}_i^{\text{pred}}(t_{i,\tilde{k}_i})} \tag{3.53}$$

where $\mathbf{1}_o(o') \in \{0, 1\}$ is an indicator function denoting if a given observation o' is equal to the reference observation $o \in \mathcal{O}_i^{\text{pred}}(t_{i,\tilde{k}_i})$. The total predicted observation set $\mathcal{O}_i^{\text{pred}}(t_{i,\tilde{k}_i})$ is constructed as the union of all predicted observation sets. The vector $\boldsymbol{u}_i^{\text{pred}}(t_{i,\tilde{k}_i})$ thus stores the average normalized MCDM utility for each unique observation across all actions. This aggregation reflects an expected utility interpretation, assuming that different actions generating the same observation contribute toward its predicted value.

Then, let $X_{i,m} \in \{0, 1\}$ be the binary decision variable for selecting the m^{th} action from $A_i^{\text{pred}}(t_{i,\tilde{k}_i})$ and $Y_{i,n} \in \{0, 1\}$ be the binary decision variable for selecting the n^{th} unique observation in $\mathcal{O}_i^{\text{pred}}(t_{i,\tilde{k}_i})$ for

robot i. Here, $m \in \{1, \ldots, |A_i^{\text{pred}}(t_{i,\tilde{k}_i})|\}$ indexes actions and $n \in \{1, \ldots, |\mathcal{O}_i^{\text{pred}}(t_{i,\tilde{k}_i})|\}$ indexes unique predicted observations. The linear optimization problem for selecting independent actions can then be expressed by relating the selections in X_i to the unique observations and their corresponding utilities within Y_i. To that end, let the set $I_{X_i}(n)$ contain the indices of X_i that correspond to actions relating to the observation $\mathcal{O}_{i,n}^{\text{pred}}(t_{i,\tilde{k}_i})$, and the set $I_{Y_i}(m)$ contain the indices of Y_i that correspond to the unique observations relating to the given action $A_{i,m}^{\text{pred}}(t_{i,\tilde{k}_i})$. The optimization goal is then selecting as many high-value actions as possible while maintaining N^{overlap} number of overlaps in their corresponding observations. Note that for the approximation of individual action utilities to be exact, there should be zero overlaps. However, in practice, some overlap in the observations for different actions may be preferable to keep the search space smoother, at the cost of overestimating the total utility, since repeated observation selections inflate their contribution up to N^{overlap} times. The associated MILP for the first stage of the task allocator is formulated as follows:

MILP formulation for the first stage of the local robot controllers to generate independent action sets and their corresponding utilities

$$\underset{X_i, Y_i}{\text{argmax}} \sum_{n=1}^{|\mathcal{O}_i^{\text{pred}}(t_{i,\tilde{k}_i})|} Y_{i,n}\, \boldsymbol{u}_{i,n}^{\text{pred}}(t_{i,\tilde{k}_i}) \tag{3.54a}$$

s.t.

$$\sum_{n \in I_{Y_i}(m)} Y_{i,n} \geq |I_{Y_i}(m)| - M^{\mathsf{Y}}(1 - X_{i,m}), \quad \forall m \tag{3.54b}$$

$$\sum_{n \in I_{Y_i}(m)} Y_{i,n} \leq |I_{Y_i}(m)| + M^{\mathsf{Y}}(1 - X_{i,m}), \quad \forall m \tag{3.54c}$$

$$\sum_{m \in I_{X_i}(n)} X_{i,m} \geq 1 - M^{\mathsf{X}}(1 - Y_{i,n}), \quad \forall n \tag{3.54d}$$

$$\sum_{m \in I_{X_i}(n)} X_{i,m} \leq 1 + N^{\text{overlap}} + M^{\mathsf{X}}(1 - Y_{i,n}), \quad \forall n \tag{3.54e}$$

In the above constraints, M^{X} and M^{Y} are sufficiently large constants used to deactivate constraints when the associated decision variables $Y_{i,n}$ and $X_{i,m}$ are not selected, respectively. There are two sets of constraints characterizing the relationship between X_i and Y_i. The first pair consisting of Constraints (3.54b) and (3.54c) ensures that all observations relating to a selected action are also chosen. The second pair consisting of Constraints (3.54d) and (3.54e) maintains that if an observation is selected, there must be at least one and at most $1 + N^{\text{overlap}}$ associated actions selected. The optimization is always feasible as long as $|\mathcal{O}_i^{\text{pred}}(t_{i,\tilde{k}_i})| > 0$ and $N^{\text{overlap}} > 0$ hold; at least one action will always satisfy the proposed constraints, because actions generate their corresponding observation sets, which, by definition, do not overlap. Using the single-step partial MCDM utilities of observations, the optimal selection of actions is steered towards the most non-overlapping total utility distribution. The selected actions and their utility are then given by the optimal X_i^* and Y_i^* where each utility for the selected action set is calculated by summing the utilities of all related observations and then dividing by the number of times each observation is associated with a selected action, ensuring that contributions are not overrepresented due to overlapping observations:

Outputs of the first stage of the local robot controllers

$$A_i^{\text{milp}}(t_{i,\tilde{k}_i}) = \left\{ A_{i,m}^{\text{pred}}(t_{i,\tilde{k}_i}),\ m \in \{1, \ldots, |A_i^{\text{pred}}(t_{i,\tilde{k}_i})|\} \mid X_{i,m}^* = 1 \right\} \tag{3.55}$$

$$\boldsymbol{u}_i^{\text{milp}}(t_{i,\tilde{k}_i}) = \left\langle \sum_{n \in I_{Y_i}(m)} \frac{\boldsymbol{u}_{i,n}^{\text{pred}}(t_{i,\tilde{k}_i})}{\sum_{m' \in I_{X_i}(n)} X_{i,m'}^*} \right\rangle_{m \in \{1, \ldots, |A_i^{\text{milp}}(t_{i,\tilde{k}_i})|\}} \tag{3.56}$$

i	robot iterator	m	action-prediction horizon iterator	n	unique observation iterator
$A_i^{\text{pred}}(t_k)$	set of predicted actions along the prediction horizon	$O_{i,m}^{\text{pred}}(t_k)$	predicted observation set as the result of an action m	$\mathcal{O}_i^{\text{pred}}(t_k)$	set of unique observations along the prediction horizon
$\hat{u}_i(s_i(t_k), a_m)$	partial MCDM estimation of an action m	$\bar{u}_{i,m,n}(t_k)$	normalized utility for a unique observation n as the result of an action m	$\boldsymbol{u}_i^{\text{pred}}(t_k)$	unique observation utility vector as the input to the first stage
$X_{i,m}$	binary decision variable for selecting an action m	$Y_{i,n}$	binary decision variable for selecting a unique observation n	N^{overlap}	maximum number of allowed action overlaps per observation
$I_{X_i}(n)$	index set of actions corresponding to a unique observation n	$I_{Y_i}(m)$	index set of observations corresponding to the unique action m	M^{Y} M^{X}	big-M constants for the decision variables
$A_i^{\text{milp}}(t_k)$	selected actions at the output	$\boldsymbol{u}_i^{\text{milp}}(t_k)$	utility of selected actions at the output	H_i^{pred}	prediction horizon

Table 3.3: Summary of notations included in the first stage of the local robot controllers.

Remark 3.2.2 (Local Robot Controller First Stage Example)**.** *To contextualize the first stage of the local robot controllers, a simple example is demonstrated in Figure 3.6, and the summary of notations included in this stage is given in Table 3.3. In this example, three actions are predicted to be feasible along the prediction horizon of the robot, denoted by* a_1, a_2, a_3*, Each of these associate with a predicted observation set, namely* $O_{i,1}^{\textit{pred}}(t_k) = \{o_1, o_2, o_3\}$*,* $O_{i,2}^{\textit{pred}}(t_k) = \{o_2, o_3, o_4\}$*, and* $O_{i,3}^{\textit{pred}}(t_k) = \{o_3, o_4, o_5, o_6\}$*, while the combined set becomes* $\mathcal{O}_i^{\textit{pred}}(t_k) = \{o_1, o_2, o_3, o_4, o_5, o_6\}$*. To calculate an approximation of independent utilities for each action, the marginal utility contributions* $\hat{\boldsymbol{u}}_{i,m}$ *of each unique observation are calculated, using Equation* (3.51) *as shown in the diagram. Then, applying Equation* (3.53)*, the values per action are averaged to find the singular utility for each action, contained in the vector* $\boldsymbol{u}_i^{\textit{pred}}(t_k)$*. Using the binary decision variables* X_i *for selecting actions and* Y_i *for selecting observations, the cost function described in Problem* (3.54a) *maximizes the utility selection described by* Y_i *values. The optimal selections are shown in red in the diagram. Trivially, the direct maximization of the cost function is achieved by selecting all observations. Then, the Constraint* (3.54d) *establishes that for each observation* n*, at least one corresponding action must be selected, of which the relationships are expressed by the elements of the set* I_{X_i} *containing action indices per observation. The Constraint* (3.54a) *sets that at most* $N^{\textit{overlap}} + 1$ *actions per observation can be chosen. In this example,* o_3 *is observed by all actions, therefore it must be included by* Y_i*. However, since* $N^{\textit{overlap}} = 1$*, only two out of the three actions are feasible for selection by this constraint. To establish the inverse relationship, the Constraints* (3.54b) *and* (3.54c) *ensure that when an action is selected, all observations related to that action must also be selected. Therefore, the optimization problem here can be expressed as selecting two actions that cover the entirety of the set* $\mathcal{O}_i^{\textit{pred}}(t_k)$*, which is satisfied by selecting* a_1 *and* a_2*, as shown at the output stage of the diagram. Finally, the utilities corresponding to these actions are calculated by taking the weighted average of the included observations of corresponding actions, where the weights are the inverse of the number of selected actions related to the observation, as described in Equation* (3.56)*. With this operation, the overrepresented action* o_3 *is normalized in the final approximation of the independent utility of the reduced action set* $A_i^{\textit{milp}}(t_k)$*. The optimization problem defined in the first stage is always feasible, as it is always possible to select only one action that satisfies all constraints by definition.*

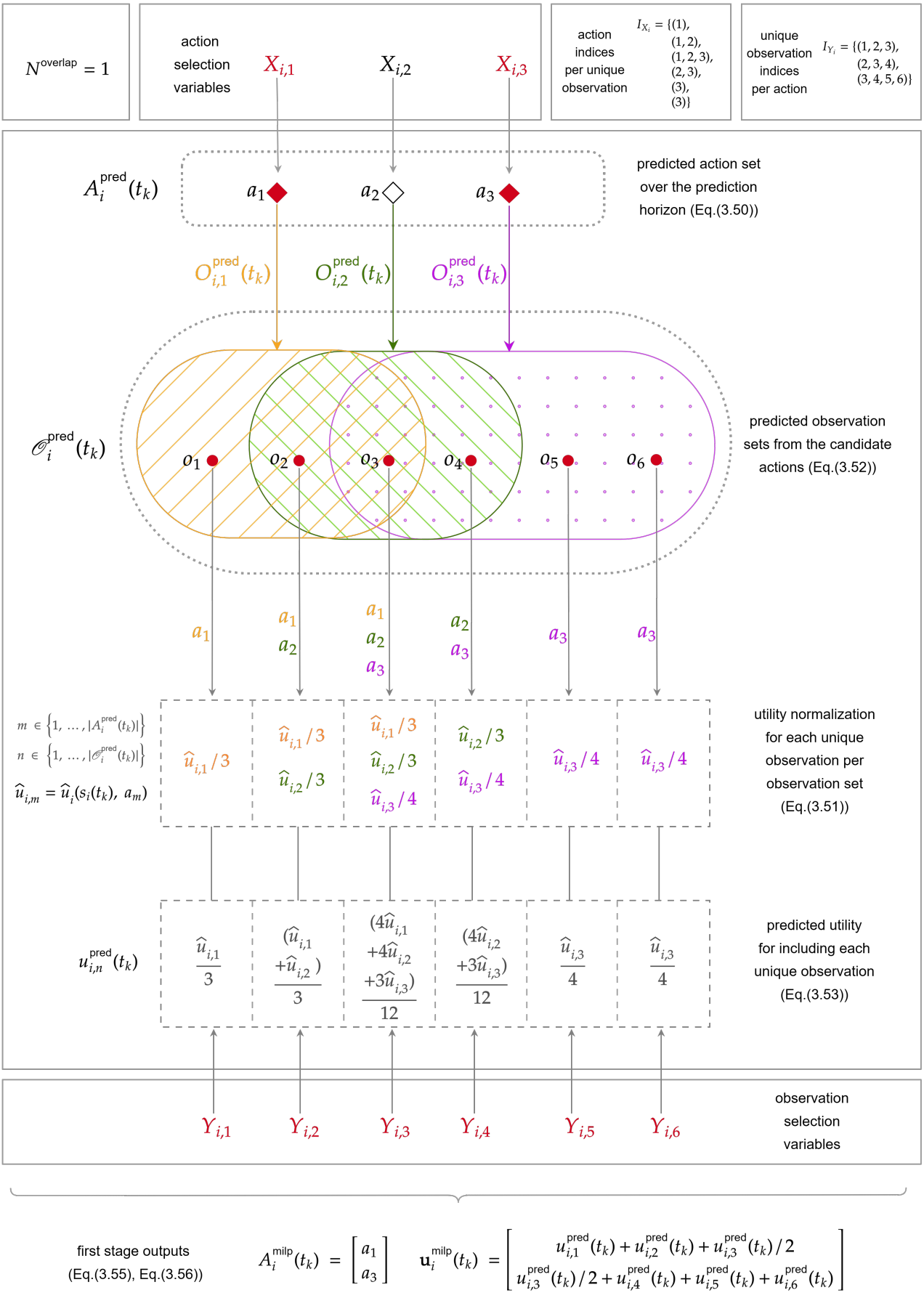


Figure 3.6: Operational diagram of the first stage for the linearization of the local MPC controllers.

Second Stage: Optimal Selection & Ordering of the Reduced Action Set

Having obtained a set of reduced actions $A_i^{\text{milp}}(t_{i,\tilde{k}_i})$ and the corresponding independent utility vector $\boldsymbol{u}_i^{\text{milp}}(t_{i,\tilde{k}_i})$ at the output of the first stage, the optimization problem on the second stage reduces to selecting the best ordering of at most H_i^{pred} actions that maximizes the total utility. Since the action set $A_i^{\text{milp}}(t_{i,\tilde{k}_i})$ consists of approximately independent actions, the states corresponding to taking each action can be estimated directly. To establish this, the estimation functions $\hat{\delta}_i^{\text{time}} : \mathcal{A}_i \times \mathcal{A}_i \to \mathbb{R}_{\geq 0}$ and $\hat{\delta}_i^{\text{energy}} : \mathcal{A}_i \times \mathcal{A}_i \to \mathbb{R}$ are introduced for defining the time and energy expenditure when the action $A_{i,n}^{\text{milp}}(t_{i,\tilde{k}_i})$ is taken right after the estimated state resulting from the action $A_{i,m}^{\text{milp}}(t_{i,\tilde{k}_i})$. Consequently, the pairwise transitional values of relative time, energy, and MCDM time utility can be represented by the square matrices T_i^{trans}, E_i^{trans} and T_i^{mcdm}, constructed as follows:

$$T_{i,m,n}^{\text{trans}} = \hat{\delta}_i^{\text{time}}(A_{i,m}^{\text{milp}}(t_{i,\tilde{k}_i}), A_{i,n}^{\text{milp}}(t_{i,\tilde{k}_i})) \tag{3.57}$$

$$E_{i,m,n}^{\text{trans}} = \hat{\delta}_i^{\text{energy}}(A_{i,m}^{\text{milp}}(t_{i,\tilde{k}_i}), A_{i,n}^{\text{milp}}(t_{i,\tilde{k}_i})) \tag{3.58}$$

$$T_{i,m,n}^{\text{mcdm}} = 1 - \min\left(1,\ \frac{\hat{\delta}_i^{\text{time}}(A_{i,m}^{\text{milp}}(t_{i,\tilde{k}_i}), A_{i,n}^{\text{milp}}(t_{i,\tilde{k}_i})) - \delta t_n^{\text{min}}}{\delta t_n^{\text{max}} - \delta t_n^{\text{min}}}\right) \tag{3.59}$$

where the normalization values δt_n^{max} and δt_n^{min} are considered as the maximum and minimum of the time transitions from all considered actions to the action n, respectively. This is done to correctly rate the sequential times spent on consecutive action transitions, where consistently high utility selections generate more utility when their consecutive timings are also low. Note that the subscripts $m, n \in \{1, \ldots, |A_i^{\text{milp}}(t_k)|\}$ iterate over the candidate actions. To maintain continuity from the current state of the robot, the candidate action set is padded from the starting action to include an `idle` action, enabling transitions from the current state.

i	robot iterator	m, n	action set iterators	j	central agent schedule iterator
$A_i^{\text{milp}}(t_k)$	selected actions from the first stage at the input	$\boldsymbol{u}_i^{\text{milp}}(t_k)$	utility of selected actions from the frst stage at the input	H_i^{pred}	maximum number of selected candidates
$T_{i,m,n}^{\text{trans}}$	transition time of taking action n after action m	$E_{i,m,n}^{\text{trans}}$	energy loss of taking action n after action m	$T_{i,m,n}^{\text{mcdm}}$	MCDM time utility of taking action n after action m
$\Pi_i^{\text{milp}}(t_k)$	tuple containing the decision variables	$X_{i,m,n}$	binary decision variable for transitioning from action m to action n	$V_{i,m}$	binary decision variable for selecting action m
$T_{i,m}$	continuous decision variable for the elapsed time on selecting action m	$E_{i,m}$	continuous decision variable for the remaining energy on selecting action m	$Z_{i,m,j}^{\text{charge}}$	binary decision variable denoting safe return to central agent schedule j on selecting action m
$U_{i,m}$	continuous decision variable for the utility gained by action m	$U_{i,m}^{\text{select}}$	continuous decision variable for the utility gained by selecting action m	$\boldsymbol{\pi}_i^{\text{a}}(t_k)$	output schedule of actions

Table 3.4: Summary of notations included in the second stage of the local robot controllers.

All decision variables for the MILP task allocation problem within a single set are represented by the following tuple:

$$\Pi_i^{\text{milp}}(t_{i,\tilde{k}_i}) = (X_i, V_i, T_i, E_i, U_i, Z_i^{\text{charge}}, U_i^{\text{select}}) \tag{3.60}$$

where X_i is a square matrix denoting the binary decision variables for selecting the transition from the candidate m to the candidate n, V_i is the vector of binary decision variables denoting the selection of the candidate m. For the state represented by each candidate action, T_i is the vector of continuous

decision variables denoting the normalized elapsed time, E_i is the vector of continuous decision variables denoting the remaining energy, U_i is the vector of continuous decision variables denoting the utility obtained by including the corresponding candidate at its elapsed time, and U_i^{select} is the vector of continuous decision variables denoting the utility of the selected actions supplied by the values in V_i. Finally, the binary decision variables encoding the safe return feasibility of each action are encoded within the $|A_i^{\text{pred}}(t_{i,\tilde{k}_i})| \times |\boldsymbol{\pi}^{\text{c}}(t_{i,\tilde{k}_i})|$ matrix Z_i^{charge}. The summary of notation in the formulation of the local controller second stage is provided in Table '3.4.

Traversal Constraints

Since the initial condition of the robot is encoded in the first candidate, a valid sequence of actions must always include it as the first element. Additionally, each selection must be unique; therefore, there can not be any loops encoded in the transition matrix X_i. While the order of selection matters for the calculation of E_i and T_i values, for the utility-related variables, the vector V_i becomes relevant to activate only the selected candidates. The traversal constraints maintaining this behavior are formulated as follows:

Traversal constraints of the local controller second stage, $\forall m, n = \{1, \ldots, |A_i^{\text{pred}}(t_{i,\tilde{k}_i})|\}$

- At most $H_i^{\text{pred}} + 1$ candidates must be selected.

$$\sum_m V_{i,m} \leq H_i^{\text{pred}} + 1 \tag{3.61}$$

- The total number of transitions must be equal to the number of transitions between selected candidates.

$$\sum_m V_{i,m} - \sum_m \sum_n X_{i,m,n} = 1 \tag{3.62}$$

- Candidate 1 must have exactly one outgoing transition.

$$\sum_n X_{i,1,n} = 1 \tag{3.63}$$

- Candidate 1 must have no incoming transitions.

$$\sum_m X_{i,m,1} = 0 \tag{3.64}$$

- No self-transitions can occur.

$$X_{i,m,m} = 0 \tag{3.65}$$

- Allow a transition $X_{i,m,n}$ if only both candidates m and n are selected.

$$X_{i,m,n} \leq V_{i,m} \ \wedge X_{i,m,n} \leq V_{i,n} \tag{3.66}$$

- The candidate n can not be selected if there is no incoming transition to it.

$$\sum_{m>1} X_{i,m,n} = V_{i,n} \tag{3.67}$$

- If there is an outgoing transition from candidate m, the candidate must be selected.

$$\sum_n X_{i,m,n} \leq V_{i,m} \tag{3.68}$$

Safe Return Constraints

The safe return constraints must be accounted for to further restrict the space of feasible action sequences regarding the formulation of the system described in this thesis. To that end, first, the bookkeeping of the elapsed time and energy values E_i and T_i defined by the selection and ordering encoded by X_i and Y_i needs to be performed. For establishing correctness, the initial conditions $T_{i,1} = t_{i,\tilde{k}_i}$ and $E_{i,1} = \epsilon_i(t_{i,\tilde{k}_i})$ are set at the start of the sequence. Then, the safe return constraints $Z^{\text{charge}}_{i,m,j}$ denoting whether the candidate m can feasibly meet the central agent waypoint j can be determined by comparing the corresponding values in E_i and T_i to the time and energy deadlines of each waypoint. To establish this condition as a hard constraint, the selection of infeasible candidates needs to be blocked. The set of constraints that ensures all these conditions is formulated as follows:

Safe return constraints of the local controller second stage, $\forall m, n = \{1, \ldots, |A^{\text{pred}}_i(t_{i,\tilde{k}_i})|\}$ and $\forall j \in \{1, \ldots, |\boldsymbol{\pi}^{\text{c}}(t_{i,\tilde{k}_i})|\}$

- Elapsed time along the set of selected actions must be sequential.

$$X_{i,m,n} = 1 \Rightarrow T_{i,n} - T_{i,m} = T^{\text{trans}}_{i,m,n} \tag{3.69}$$

- Remaining energy along the set of selected actions must be sequential.

$$X_{i,m,n} = 1 \Rightarrow E_{i,m} - E_{i,n} = E^{\text{trans}}_{i,m,n} \tag{3.70}$$

- If the time deadline for the central agent waypoint j is not met by the action m, then $Z^{\text{charge}}_{i,m,j}$ must be zero.

$$T_{i,m} - t^{\text{charge}}_{i,j}(A^{\text{milp}}_{i,m}(t_{i,\tilde{k}_i})) \leq M^{\text{time}}(1 - Z^{\text{charge}}_{i,m,j}) \tag{3.71}$$

- If the energy requirement for the central agent waypoint j is not met by the action m, then $Z^{\text{charge}}_{i,m,j}$ must be zero.

$$\epsilon^{\text{charge}}_{i,j}(A^{\text{milp}}_{i,m}(t_{i,\tilde{k}_i})) - E_{i,m} \leq M^{\text{energy}}(1 - Z^{\text{charge}}_{i,m,j}) \tag{3.72}$$

- If none of the central agent waypoints can be feasibly reached by the action m, it can not be selected.

$$\sum_j Z^{\text{charge}}_{i,m,j} = 0 \Rightarrow V_{i,m} = 0 \tag{3.73}$$

Although the Constraints (3.68) and (3.69) are written as indicator constraints, for generalized MILP solvers, the conversion to a big-M representation is straightforward. In this thesis, indicator constraints are provided wherever appropriate for conciseness. Moreover, a key observation in this formulation is that since utilities are always positive, the maximization of the cost function will always attempt to select as many actions as possible. By Constraint (3.73), at least one safe return constraint pair will be attempted to be satisfied, further steering the optimizer towards feasible regions.

Utility Constraints

The final component of the optimization framework is the linearization of the MCDM formula provided in Equation (3.33), contained by the variables in the utility vector U_i. Assuming that the time component of the utility function does not affect the individual utility of other criteria, the objective function can be separated into dependent (time) and independent (aggregation of the remaining criteria) components. As discussed, this stage separates the criteria related to the candidate actions, calculated similarly to Equation (3.27), but without including the timing meta-task in the utility calculation, which are extracted in the previous stage as $\boldsymbol{u}^{\text{milp}}_{i,n}(t_{i,\tilde{k}_i})$. Then, let the feature set F^{dom}_n denote the *dominating* features in the MCDM calculation of the utility value for the candidate n. The criteria for determining which fea-

tures should be included depend on the use case; however, including the features corresponding to the values above the median of the corresponding action utility vector (as introduced in Equation (3.29)) is proposed for this thesis. Accordingly, the reduced fuzzy measures for linearizing the MCDM problem are denoted by $\mu_{n,1} = \mu(F^{\mathsf{time}})$, $\mu_{n,2} = \mu(F_n^{\mathsf{dom}})$, and $\mu_{n,3} = \mu(F_n^{\mathsf{dom}} \cup F^{\mathsf{time}})$. This decomposition creates a utility function with two criteria; therefore, the necessary sorting operation can be represented by a conditional statement. Then, the calculated sequential utilities within U_i are masked by the selection denoted by V_i within the vector U_i^{select}. The constraints to establish this formulation are given as follows:

Utility constraints of the local controller second stage, $\forall m, n = \{1, \dots, |A_i^{\mathsf{pred}}(t_{i,\tilde{k}_i})|\}$

- If the MCDM value related to the elapsed normalized time aggregation is larger than the time-independent utility, linearize the discrete Choquet integral accordingly.

$$\sum_m X_{i,m,n} T_{i,m,n}^{\mathsf{mcdm}} \geq \boldsymbol{u}_{i,n}^{\mathsf{milp}}(t_{i,\tilde{k}_i}) \Rightarrow$$

$$U_{i,n} = \mu_{n,1} \cdot \left(\sum_m X_{m,n}^i T_{m,n}^{\mathsf{mcdm,\,i}} - \boldsymbol{u}_{i,n}^{\mathsf{milp}}(t_{i,\tilde{k}_i}) \right) + \mu_{n,3} \cdot \boldsymbol{u}_{i,n}^{\mathsf{milp}}(t_{i,\tilde{k}_i}) \tag{3.74}$$

- If the MCDM value related to the elapsed normalized time aggregation is smaller than the time-independent utility, linearize the discrete Choquet integral accordingly.

$$\sum_m X_{i,m,n} T_{i,m,n}^{\mathsf{mcdm}} \leq \boldsymbol{u}_{i,n}^{\mathsf{milp}}(t_{i,\tilde{k}_i}) \Rightarrow$$

$$U_{i,n} = \mu_{n,2} \cdot \left(\boldsymbol{u}_{i,n}^{\mathsf{milp}}(t_{i,\tilde{k}_i}) - \sum_m X_{i,m,n} T_{i,m,n}^{\mathsf{mcdm}} \right) + \mu_{n,3} \cdot \sum_m X_{i,m,n} T_{i,m,n}^{\mathsf{mcdm}} \tag{3.75}$$

- If the candidate action m is not selected, then it can not contribute to the total utility.

$$V_{i,m} = 0 \Rightarrow U_{i,m}^{\mathsf{select}} = 0 \tag{3.76}$$

- If the candidate action m is selected, then it contributes to the total utility.

$$V_{i,m} = 1 \Rightarrow U_{i,m}^{\mathsf{select}} = U_{i,m} \tag{3.77}$$

Optimization Problem

Optimization problem defined for the second stage of the local controllers.

$$\underset{\Pi_i^{\mathsf{milp}}(t_{i,\tilde{k}_i})}{\mathsf{argmax}} \sum_{m=1}^{|A_i^{\mathsf{milp}}(t_{i,\tilde{k}_i})|} U_{i,m}^{\mathsf{select}} \tag{3.78a}$$

s.t.

1. Traversal constraints: (3.61)(3.62)(3.63)(3.64)(3.65)(3.66)(3.67)(3.68) (3.78b)
2. Safe return constraints: (3.69)(3.70)(3.71)(3.72)(3.73) (3.78c)
3. Utility constraints: (3.76)(3.77)(3.74)(3.75) (3.78d)

$$\forall m, n \in \{1, \dots, |A_i^{\mathsf{milp}}(t_{i,\tilde{k}_i})|\},\ \forall j \in \{1, \dots, |\boldsymbol{\pi}^{\mathsf{c}}(t_k)|\}$$

The complete optimization problem for finding the optimal ordering of actions within the prediction horizon for the second-stage MILP controller of the robot i, with all formulated constraints, is provided in the Problem (3.44). Solving the combinatorial MILP formulation given in Equation (3.78a) exactly could be infeasible in practice since the solution space is factorial by the number of considered actions $|A_i^{\text{milp}}(t_{i,\tilde{k}_i})|$ in the worst case. To address this issue, the proposed approach is embedded in a heuristic improvement algorithm that employs a large neighborhood search (LNS). The rationale behind this approach is to produce an initial incumbent solution from the full problem and then iteratively attempt to improve it by fixing a portion of the candidate selections, allowing the remaining variables to be optimized as a smaller problem using the same model. If a better solution is found, then the initial incumbent solution is updated accordingly. The LNS process is applied for a number of iterations or until a maximum stall value is reached. With this, the final proposed algorithm for the local robot controllers is as follows:

Algorithm 1: Local robot control optimization using MILP with LNS

Generate an incumbent solution $\pi_i^{\text{a}^\star}(t_{i,\tilde{k}_i})$ by partially optimizing the model in Equation (3.78a)
if Optimal solution found **then**
    return
end if
$U_i^{\text{milp}^\star} \leftarrow$ Final utility of each candidate encoded by the ordering in $\pi_i^{\text{a}^\star}(t_{i,\tilde{k}_i})$
$n^{\text{iter}} \leftarrow 1$
$n^{\text{stall}} \leftarrow 0$
while $n^{\text{iter}} \leq N^{\text{iter}}$ and $n^{\text{stall}} < N^{\text{stall}}$ **do**
    $x^{\text{best}} \leftarrow$ Select randomly n^{best} candidates using $U_i^{\text{milp}^\star}$ as a probability distribution
    for $j = 1$ to n^{best} **do**
        Add the constraint $V_{i,x^{\text{best}}} = V^{\star}_{i,x^{\text{best}}}$ to the original problem
    end for
    $\pi_i^{\text{new}^\star}(t_{i,\tilde{k}_i}) \leftarrow$ Solve the smaller MILP problem in Equation (3.78a) with the added constraints
    $U_i^{\text{new}^\star} \leftarrow$ Final utility of each candidate encoded by the ordering in $\pi_i^{\text{new}^\star}(t_{i,\tilde{k}_i})$
    if $\sum_m U_{i,m}^{\text{new}^\star} > \sum_m U_{i,m}^{\text{milp}^\star}$ **then**
        $\pi_i^{\text{milp}^\star}(t_{i,\tilde{k}_i}) \leftarrow \pi_i^{\text{new}^\star}(t_{i,\tilde{k}_i})$
        $U_i^{\text{milp}^\star} \leftarrow U_i^{\text{new}^\star}$
        $n^{\text{stall}} \leftarrow 0$
    else
        $n^{\text{stall}} \leftarrow n^{\text{stall}} + 1$
    end if
    $n^{\text{iter}} \leftarrow n^{\text{iter}} + 1$
    Reset constraints for the next iteration
end while

3.2.2. Global Coordination Controller

The overall optimization problem defined in Equation (3.35) is locally addressed by the individual robot controllers, while the global optimality still needs to be accounted for due to the potential conflicts in robot behaviors. A key observation in the problem formulation is that the robots can individually contribute to the utility at a given time. Additionally, increasing the number of tasks attempted through collective actions will generate more utility than multiple robots attempting the same tasks in isolation. Furthermore, local controllers are incentivized to select actions that reduce the total time while also yielding more utility, as assessed by the robot's own predicted performance in undertaking the related tasks. When multiple robots are present, it may be more beneficial to assign the same task set per action to a more capable robot, provided they also minimize time to a similar extent. As a result, the two components of global optimality are highlighted as solving the *conflicts* between unique robot actions and ensuring *coordination* regarding robot abilities. To contextualize, when two actions are conflicting, at most only one of them must be performed, whereas coordination implies that the best allocation of the scheduled actions must be ensured. Ideally, coordination is satisfied through resolving all conflicts optimally. However, the formulation of this thesis supports modeling any amount and type of action-task pairings selected within a prediction horizon, thus forcing the local controllers to select locally optimal but globally suboptimal actions. This sub-optimality can be improved by incorporating the decisions of other robots into the upper controller hierarchy.

In this section, a hierarchical global coordination controller is designed to compare the local optimization outputs of coordinating robot schedules. Thus, at the first stage of global coordination control, the robots that meet a certain criterion for coordination are identified. Then, the scheduled actions based on the previous prediction across individual horizons are combined into a single pool of actions that can be distributed to the set of coordinating robots. Since the proposed control architecture is hierarchical and the global controller behaves as an overseer from the perspective of the SaR mission, the results of the local controllers are used directly instead of repeating the generation of entire feasible action sets at this level. The results from the local stage are assumed to be already optimal, and their utilities can be viewed as approximately individual from each other. With this remark, an extended version of the optimization problem employed in the second stage of the local controllers is introduced to reassign the best tasks through the combined pool of actions at the final stage of the global coordination controller. The entire process is summarized in Figure 3.7, where the summary of notations included in the section is given in Table 3.5.

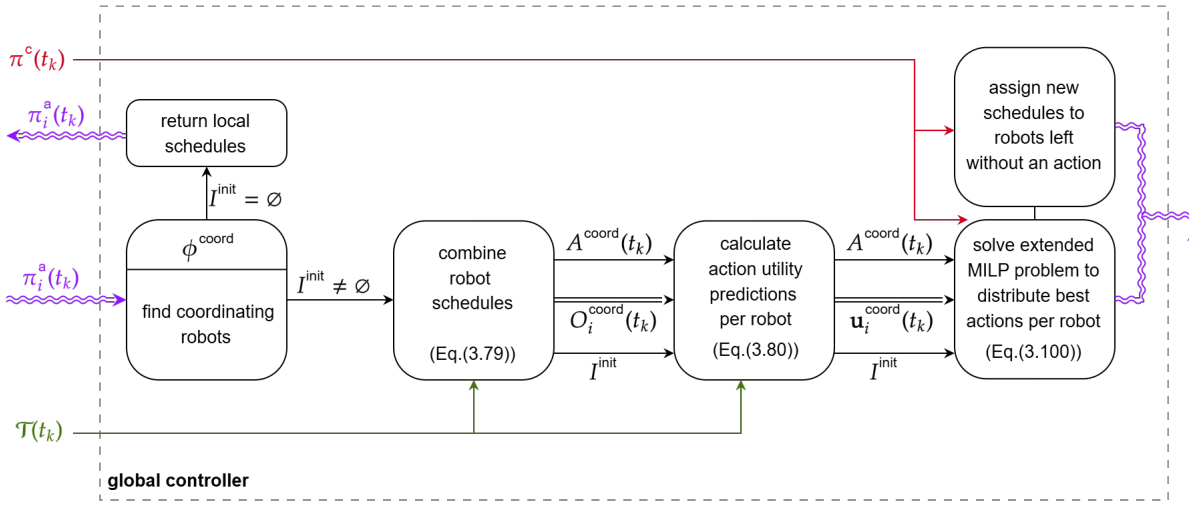


Figure 3.7: Diagram of the proposed global coordination controller.

i	robot iterator	m, n	combined action set iterators	I_i^{init}	index of the initial conditions of the robots on the combined action set
ϕ^{coord}	coordination indicator between two robots	ϕ^{conflict}	conflict indicator between two robots	$I^{\text{coord}}(t_k)$	index set of coordinating robots
$A_i^{\text{coord}}(t_k)$	combined action set of current robot schedules	$O_i^{\text{coord}}(t_k)$	combined observation set corresponding of current robot schedules	$\boldsymbol{u}_i^{\text{coord}}(t_k)$	utility vector for the combined action set
$\Pi^{\text{coord}}(t_k)$	tuple containing the decision variables	$T_{i,m,n}^{\text{coord}}$	transition time of robot i taking action m after action n	$E_{i,m,n}^{\text{coord}}$	energy loss of robot i taking action m after action n
$X_{i,m,n}$	binary decision variable for robot i transitioning from action m to action n	$T_{i,m}$	continuous decision variable for elapsed time of robot i on selecting action m	$E_{i,m}$	continuous decision variable for remaining energy of robot i on selecting action m
$V_{i,m}$	binary decision variable for robot i selecting action m	$U_{i,m}$	continuous decision variable for utility gained by robot i on action m	$U_{i,m}^{\text{select}}$	continuous decision variable for utility gained by robot i selecting action m

Table 3.5: Summary of notations included in the global coordination controller.

Combining Coordinating Robot Schedules

Assume that a function $\phi^{\text{coord}} : \mathcal{S}_i \times \mathcal{S}_j \to \{0, 1\}$ determines if two given robots such that $i \neq j$ and $i, j \in \{1, \ldots, N^{\text{robots}}\}$ need to coordinate in their current state. This coordination indicator may consider the predicted tasks, robot types, or robot distances in practice to determine which robots are evaluated together within the global controller. Let $\boldsymbol{\pi}_i^a(t_k)$ be the most recently optimized schedule on the prediction horizon of the robot i at global time t_k. Then, assume that there exists a function $\phi^{\text{conflict}} : \mathcal{A}_i \times \mathcal{A}_j \to \{0, 1\}$ to determine if two actions from different robots are in conflict. The implementation of this function may be decided upon considering the overlapping observational features of these actions or by setting a distance measure between those that attempt similar tasks. The local controllers are assumed to output actions as independently as possible from each other since the overall utility yield is expected to be higher compared to actions that create redundancy. Furthermore, with an admissible heuristic chosen for conflict detection, removing all conflicts from the robot schedule should increase the independence between different robot actions. Although a combination of scheduled waypoints does not directly alter the action utilities, the timings between them change, considering some actions are essentially skipped if not selected. These remarks naturally lead to an extension of the MILP implementation proposed for the local controllers to generate a global utility with multiple robots. The search space with multiple robots for the same problem would become intractable rather quickly; therefore, the outputs of the local controllers are used as the actions to be allocated to each robot. The combined set of actions of all robot schedules that are selected for coordination is given as follows:

$$A^{\text{coord}}(t_k) = \bigcup_{i \in I^{\text{coord}}(t_k)} (\alpha^{\text{idle}}, v_i^{\text{r}}(t_k)) \cup \boldsymbol{\pi}_i^a(t_k) \tag{3.79}$$

Where $I^{\text{coord}}(t_k)$ is the set of all robot indices that are flagged for coordination. An idle action is added at the beginning of each ordered set that is concatenated to denote the starting point of all robots. Since all actions in $A^{\text{coord}}(t_k)$ are evaluated jointly by the global coordination controller at synchronized global time t_k, each robot i denoted in $I^{\text{coord}}(t_k)$ must independently compute its predicted marginal utility for each action in this shared set. Let $O_i^{\text{coord}}(t_k)$ denote the predicted observation set for the robot i when hypothetically performing each action $a_m \in A^{\text{coord}}(t_k)$, evaluated from its own state $s_i(t_k)$, similar to the first stage of the local controllers. A similar scheme for utility estimation, without time dependency, is employed in this stage, similar to the first stage of the local controllers.

$$\boldsymbol{u}_{i,m}^{\text{coord}}(t_k) = \hat{u}_i(s_i(t_k), a_m), \quad m \in \{1, \ldots, A^{\text{coord}}(t_k)\} \tag{3.80}$$

The marginal utility computations per observation are unnecessary in this case, as the pooled action set is already locally optimal based on the outputs of the local controllers; therefore, correcting for overlapping observations is redundant in terms of approximation quality. If there are significantly overlapping actions in terms of their observations, then it can be said that they are conflicting, therefore their selection can be regarded as mutually exclusive instead of allowing the assignment of both within the robotic team. Finally, to denote the starting index of each robot within the vector $A^{\text{coord}}(t_k)$, the indicator set I^{init} is constructed, which I_i^{init} holds the index of the idle action for the robot i within the action vector.

Design of the Extended Hierarchical MILP Task Allocation

The construction of the optimization model for the global coordination controller closely follows the MILP formulation proposed for the local controllers, where the pool of actions $A^{\text{coord}}(t_k)$ needs to be allocated to each participating robot. Similarly, each robot operates within its transition matrices and selection vectors, with the addition of encoding conflicts and one-to-one assignment of actions to robots. At a higher level, it might not be feasible for all robots to perform all actions in this set. Therefore, additional selection constraints are added to prevent impossible selections. Another key observation is that the robots need to collaboratively track time, maintaining an elapsed time variable for each candidate. This differs from the local controller formulation, in which timings are considered individually for each candidate and normalized with respect to all possible one-step transitions to the corresponding actions. The partial transitional estimation of the MCDM time utility is sufficient for a single robot in terms of its local control, as selecting quicker actions with better utility back-to-back will yield a solution close to the Pareto frontier, denoted by the total utility and elapsed time of all selected actions. For multiple robots, this is insufficient, as while one robot is busy performing an action, the other robot is free to choose any of the remaining actions, and their timings are not additive. To that end, the MCDM time transitions and regular time transitions are combined into a single matrix for all robots, formulated as follows:

$$T_{i,m,n}^{\text{coord}} = \hat{\delta}_i^{\text{time}}(A_m^{\text{coord}}(t_k), A_n^{\text{coord}}(t_k))/\delta t^{\text{max}}(t_k) \tag{3.81}$$

$$E_{i,m,n}^{\text{coord}} = \hat{\delta}_i^{\text{energy}}(A_m^{\text{coord}}(t_k), A_n^{\text{coord}}(t_k)) \tag{3.82}$$

Where $\delta t^{\text{max}}(t_k)$ is selected as at least the maximum difference between the first and last accumulated action execution times on the schedule for all robots. If this value is increased, the robots are allowed to undertake more actions. However, the elapsed time utilities within the MCDM formulation may fail since the difference between time-intensive and efficient selections becomes intractable if the normalization constant is too large. Conversely, if the time normalization is too small, some candidates may not be selected because the elapsed time may exceed the upper bound. Let the combination of all decision variables per robot i be expressed by:

$$\Pi_i^{\text{coord}}(t_k) = (X_i, V_i, T_i, E_i, U_i, U_i^{\text{select}}) \tag{3.83}$$

where X_i is a square matrix denoting the binary decision variables for selecting the transition of the robot i from the candidate m to the candidate n, V_i is the vector of binary decision variables denoting the selection of the candidate m for the robot i. For the state represented by each candidate action, T_i is the vector of continuous decision variables denoting the normalized elapsed time of the robot i, E_i is the vector of continuous decision variables denoting the remaining energy of the robot i, U_i is the vector of continuous decision variables denoting the utility obtained by including the corresponding candidate at its elapsed time of the robot i, and U_i^{select} is the vector of continuous decision variables denoting the utility of the selected actions supplied by the values in V_i of the robot i. Safe return constraints are expressed in a simplified manner for global coordination, with details assigned to their corresponding sections in the MILP optimization constraint presentation.

Traversal Constraints

Similar to the local robot formulations, the feasible traversal within the environment, as actions are sequentially scheduled, needs to be established by the optimization constraints. The difference in the global case is that the action pool is shared by all robots; therefore, the encoding of their selection V_i and transitions X_i is kept separately for each robot. Additionally, the robots are allowed to select any number

of candidates, rather than enforcing a horizon similar to the local controllers. For individual robots, the same rules apply as in the local case. For the entire cooperating robotic team, it is essential to set constraints by specifying which candidate indices correspond to the starting points of each robot, as indicated by the values within the ordered set I^{init}. The traversal constraints for the global coordination controller can be summarized as:

Traversal constraints of the global coordination controller, $\forall i \in I^{\text{coord}}(t_k)$, $\forall m, n = \{1, \ldots, |A^{\text{coord}}(t_k)|\}$

- The total number of transitions must be equal to the number of transitions between selected candidates per robot.

$$\sum_m V_{i,m} - \sum_m \sum_n X_{i,m,n} = 1 \tag{3.84}$$

- Starting candidates for each robot must have no incoming transitions.

$$\sum_i \sum_m X_{i,m,I_i^{\text{init}}} = 0 \tag{3.85}$$

- No self-transitions can occur for any robot.

$$\sum_i X_{i,m,m} = 0 \tag{3.86}$$

- Allow a transition $X_{i,m,n}$ if only both candidates m and n are selected for the corresponding robot.

$$X_{i,m,n} \leq V_{i,m} \ \wedge X_{i,m,n} \leq V_{i,n} \tag{3.87}$$

- The candidate n can not be selected if there is no incoming transition to the corresponding robot.

$$\sum_m X_{i,m,n} = V_{i,n}, \quad \forall n \notin I^{\text{init}} \tag{3.88}$$

- If there is an outgoing transition from candidate m, the candidate must be selected for the corresponding robot.

$$\sum_n X_{i,m,n} \leq V_{i,m} \tag{3.89}$$

Conflict Resolution Constraints

Conflict resolution constraints of the global coordination controller, $\forall i \in I^{\text{coord}}(t_k)$, $\forall m, n = \{1, \ldots, |A^{\text{coord}}(t_k)|\}$

- If the candidates m and n are conflicting, at most only one robot may schedule it across the team of cooperating robots.

$$\phi^{\text{conflict}}\left(A_m^{\text{coord}}(t_k), A_n^{\text{coord}}(t_k)\right) = 1 \Rightarrow \sum_i V_{i,m} + \sum_i V_{i,n} \leq 1 \tag{3.90}$$

- Each candidate can only be selected at most once.

$$\sum_i V_{i,m} \leq 1, \quad \forall m \tag{3.91}$$

As discussed, since a linearization step similar to the first stage of the local controllers are not

employed here, due to the selected action pool being locally optimal for the corresponding robots, to prevent redundancy in robot task allocations, conflicting actions need to be disabled from being selected at the same time at the team level. In this thesis, exceeding a measure of overlapping observations constitutes conflicts; however, additional criteria can be implemented in practice, such as semantical closeness of the task definitions and spatial proximity. Furthermore, it needs to be established that each candidate may only be selected at most once between robots, since the traversal constraints only encode individual feasibility of candidate schedules. The expressions establishing the conflict resolution are provided in Constraints (3.90) and (3.91).

Safe Return Constraints

The continuous decision variables $T_{i,m} \in [0,1]_{\mathbb{R}}$ for each candidate per robot are used for both the MCDM calculations and the tracking of total elapsed time. The energy $E_{i,m} \in \mathbb{R}$ and the elapsed time $T_{i,m}$ variables are constrained similarly to the local controllers. To establish the safe return constraints, at this level of the controller hierarchy, only the last element is checked, rather than considering the entire schedule of the central agent. This eliminates the need for introducing additional decision variables. Not including the intermediary points in the central agent's schedule may result in some actions being infeasible, even though they were selected by the local controller of the same robot. However, if such a robot cannot find any feasible actions due to this simplification, it would still need to recalculate a new schedule without considering the globally allocated tasks or any other actions that may conflict with them. Therefore, it is likely that the discarded actions can be selected again during that phase. With this formulation, the safe return constraints for the global controller are expressed by the following, where $t^{\mathsf{charge}}_{i,H^c(t_k)}(A^{\mathsf{coord}}_m(t_k))$ and $\epsilon^{\mathsf{charge}}_{i,H^c(t_k)}(A^{\mathsf{coord}}_m(t_k))$ denote the robot-specific deadline and minimum energy values for reaching the last element of the central agent schedule when the action m is taken:

Simplified safe return constraints of the global coordination controller, $\forall i \in I^{\mathsf{coord}}(t_k)$, $\forall m, n = \{1, \ldots, |A^{\mathsf{coord}}(t_k)|\}$

- Normalized elapsed time along the set of selected actions for each robot must be sequential.

$$X_{i,m,n} = 1 \Rightarrow T_{i,n} - T_{i,m} = T^{\mathsf{coord}}_{i,m,n} \tag{3.92}$$

- The remaining energy along the set of selected actions for each robot must be sequential.

$$X_{i,m,n} = 1 \Rightarrow E_{i,m} - E_{i,n} = E^{\mathsf{coord}}_{i,m,n} \tag{3.93}$$

- If the candidate m is selected for the robot i, then it must be able to reach the final waypoint of the central agent in time.

$$V_{i,m} = 1 \Rightarrow \delta t^{\mathsf{max}}(t_k) \cdot T_{i,m} \leq t^{\mathsf{charge}}_{i,H^c(t_k)}(A^{\mathsf{coord}}_m(t_k)) \tag{3.94}$$

- If the candidate m is selected for the robot i, then it must be able to reach the final waypoint with sufficient energy.

$$V_{i,m} = 1 \Rightarrow E_{i,m} \geq \epsilon^{\mathsf{charge}}_{i,H^c(t_k)}(A^{\mathsf{coord}}_m(t_k)) \tag{3.95}$$

Utility Constraints

The utility calculation of each selected candidate is the same as that of the local controllers, with the MCDM times replaced with the elapsed times. Each robot keeps track of its continuous utilities per candidate action via the decision variables $U_{i,m}$. Their calculation involves the linearization of the discrete Choquet integral for two variables, similarly to the local controllers, where $\mu_{m,1} = \mu(F^{\mathsf{time}})$, $\mu_{m,2} = \mu(F^{\mathsf{dom}}_m)$, and $\mu_{m,3} = \mu(F^{\mathsf{dom}}_m \cup F^{\mathsf{time}})$ are defined as the weights of the linearized MCDM

formulations within each action corresponding to $\boldsymbol{u}_{i,m}^{\text{coord}}(t_k)$. Thus, the MCDM utilities with respect to elapsed time are generated by the constraints as follow, while maintaining the filtering of the selected utilities for the inclusion in the cost function calculation:

Utility constraints of the global coordination controller, $\forall i \in I^{\text{coord}}(t_k)$, $\forall m = \{1, \ldots, |A^{\text{coord}}(t_k)|\}$

- If the MCDM value related to the elapsed normalized time aggregation is larger than the time-independent utility, linearize the discrete Choquet integral accordingly for robot i.

$$1 - T_{i,m} \geq \boldsymbol{u}_{i,m}^{\text{coord}}(t_k) \Rightarrow U_{i,m} = \mu_{m,1} \cdot \left(1 - T_{i,m} - \boldsymbol{u}_{i,m}^{\text{coord}}(t_k)\right) + \mu_{m,3} \cdot \boldsymbol{u}_{i,m}^{\text{coord}}(t_k) \quad (3.96)$$

- If the MCDM value related to the elapsed normalized time aggregation is smaller than the time-independent utility, linearize the discrete Choquet integral accordingly for robot i.

$$1 - T_{i,m} \leq \boldsymbol{u}_{i,m}^{\text{coord}}(t_k) \Rightarrow U_{i,m} = \mu_{m,2} \cdot \left(\boldsymbol{u}_{i,m}^{\text{coord}}(t_k) - 1 + T_{i,m}\right) + \mu_{m,3} \cdot (1 - T_{i,m}) \quad (3.97)$$

- If the candidate action m is not selected by any robot, then it can not contribute to the total utility.

$$V_{i,m} = 0 \Rightarrow U_{i,m}^{\text{select}} = 0 \quad (3.98)$$

- If the candidate action m is selected by a robot, then it contributes to the total utility.

$$V_{i,m} = 1 \Rightarrow U_{i,m}^{\text{select}} = U_{i,m} \quad (3.99)$$

Optimization Problem

The complete MILP optimization problem for allocating as many candidate actions as possible from the combined pool of locally optimal results while maintaining no conflicts to maximize the sum of MCDM utilities with respect to elapsed time is given by:

Optimization problem defined for the global coordination controller.

$$\underset{\{\Pi_i^{\text{coord}}(t_k)\}_{i \in I^{\text{coord}}(t_k)}}{\text{argmax}} \sum_{i \in I^{\text{coord}}(t_k)} \sum_{m=1}^{|A^{\text{coord}}(t_k)|} U_{i,m}^{\text{select}} \quad (3.100a)$$

s.t.

1. Traversal constraints: (3.84)(3.85)(3.86)(3.87)(3.88)(3.89) (3.100b)
2: Conflict resolution constraints: (3.90)(3.91) (3.100c)
3. Safe return constraints: (3.92)(3.93)(3.94)(3.95) (3.100d)
4. Utility constraints: (3.96)(3.97)(3.98)(3.99) (3.100e)

$\forall m, n \in \{1, \ldots, |A^{\text{coord}}(t_k)|\}, \quad \forall i \in I^{\text{coord}}(t_k)$

The resulting V_i^* and X_i^* from the optimization described in Equation (3.100a) indicates which selection and ordering of actions are allocated per robot schedule. It is indeed possible for some robots to lose all of their respective schedules after this operation; therefore, they need to optimize for their local controllers again with restricted actions to prevent conflicts, until all conflicts are resolved. This is achieved by keeping track of a set of banned actions $A_i^{\text{ban}}(t_k)$ per robot in case a new local optimization becomes necessary, and removing the already allocated tasks from the global problem. Furthermore, with each added robot to the global controller optimization, the state space increases cubically. To combat this, only two robots are optimized at a time until all marked robots for coordination have been

handled. The selection of which robots to optimize first depends on the use case, where it can be hierarchical by prioritizing the same type of robots first, or it can rely on a distance measure of either the robot locations, robot schedules, or both. Since for each iteration, at least one robot will have a nonempty schedule, the overall global optimization framework finishes in at most N^{robot} iterations. To improve the convergence speed to the optimal solution, the local controller outputs are given to the MILP solver as a warm start. Finally, since robots calculate their schedules asynchronously, the proposed algorithm first preprocesses the $A^{\text{coord}}(t_k)$ set to contain only the relevant actions for the current task set $\mathcal{T}(t_k)$. The complete algorithm for the global coordination is thereby given as follows:

Algorithm 2: Global coordination optimization

$I^{\text{coord}}(t_k) \leftarrow$ Get all robots that are marked for coordination

Construct the set of banned actions per robot $A_i^{\text{ban}}(t_k) \leftarrow \emptyset$

while There is at least one pair of robots that needs coordination **do**

  $A^{\text{coord}}(t_k) \leftarrow$ Combine all scheduled actions for the selected robots

  $A^{\text{coord}}(t_k), U_i^{\text{coord}}(t_k) \leftarrow \texttt{preprocess}\big(A^{\text{coord}}(t_k), \mathcal{T}(t_k), I^{\text{coord}}(t_k)\big)$

  Detect conflicts using the Equation (3.90)

  Add the constraint $V_{i,m} = 0$ if the action m is not feasible for the robot i

  $\pi_i^{\text{coord}*}(t_k) \leftarrow$ Perform the optimization described in Equation (3.100a), using the local controller outputs as a warm-start.

  $\pi_i^{\text{a}}(t_k) \leftarrow$ Update schedules by keeping $\pi_{i,m}^{\text{a}}(t_k)$ such that $V_{i,m}^* = 1$ with the order denoted in $X_{i,m}^*$

  if All $|\pi_i^{\text{a}}(t_k)| > 0$ **then**

    continue

  end if

  for Each robot with $|\pi_i^{\text{a}}(t_k)| = 0$ **do**

    Update the banned set of actions $A_i^{\text{ban}}(t_k)$with the actions conflicting with the old schedule

    Perform local optimization without the actions in $A_i^{\text{ban}}(t_k)$

  end for

  Discard one robot from the coordination set I^{coord}

  Update the banned set of actions $A_i^{\text{ban}}(t_k)$ for all remaining robots with the actions fixed for the removed robot.

end while

3.2.3. Central Agent Controller

The central agent is designed to be the coordination and charging hub of the entire robotic system. In practice, it may correspond to a team of humans as mission control, a semi-autonomous structure with teleoperated controls, or another fully autonomous robot similar to the ones formulated thus far, with simpler control requirements. In this thesis, a controller for the last case is presented to maintain the complete autonomy of the entire system and to provide a framework of the related design considerations. As it was briefly introduced previously, the schedule of the central agent along a scheduling horizon $H^{\mathsf{c}}(t_k)$ is denoted by:

$$\pi^{\mathsf{c}}(t_k) = \left\langle (t_j^{\mathsf{c}}, v_j^{\mathsf{c}}) \mid t_j^{\mathsf{c}} \geq t_k,\ t_1^{\mathsf{c}} = t_k \right\rangle_{j \in \{1,\dots,H^{\mathsf{c}}(t_k)\}} \tag{3.101}$$

where t_{j+1}^{c} denotes the time at which the central agent leaves the corresponding vertex v_j^{c}, denoting a time limit for meeting with the robots. The controller needs to ensure that all robots have at least one way of reaching to it without violating constraints along the new central agent schedule. Furthermore, it can be said that if this assumption holds, it must already hold for the current central agent vertex $v^{\mathsf{c}}(t_k)$, therefore the schedule generation can be separated into two phases consisting of finding the vertices first and then allocating times per vertex. If the first phase generates vertices that some robot may not feasibly reach at any time, this formulation guarantees that at least the current vertex is always reachable.

The first phase of the central agent controller starts with extracting the set of valid vertices $V^{\mathsf{c}}(t_k)$ from the current global map. A valid vertex must be reachable by the central agent. Additionally, it must be within $N^{\mathsf{range}} \in \mathbb{R}_{\geq 0}$ range from at least one task in the current task set $\mathcal{T}(t_k)$. Given a spatial distance measure between vertices $d(v_1, v_2) \in \mathbb{R}_{\geq 0}$, the set $V^c(t_k)$ is formulated as:

$$V^{\mathsf{c}}(t_k) = \{v(t_k) \in \mathcal{M}^{\mathsf{c}}(t_k) \mid d(v(t_k), v^{\mathsf{task}}(t_k)) \leq N^{\mathsf{range}},\ \exists\, v^{\mathsf{task}}(t_k) \in \mathcal{T}(t_k)\} \tag{3.102}$$

The selection of a goal vertex for path construction indirectly affects the maximum utility the robots can gain per charging cycle. Since the utilities are coupled to the task locations, the goal vertex is defined as the closest vertex to the weighted center of mass described by the locations of vertices related to the current tasks. The weights $w_l^{\mathsf{c}} \in [w^{\mathsf{max}}, w^{\mathsf{min}}]_{\mathbb{R}}, l \in \{1, \dots, |\mathcal{T}(t_k)|\}$ correspond to the normalized reverse distances between the central agent vertex and task vertices to favor the exploitation of the closer tasks. Then, the center of mass p^{com} is calculated as follows, where $p(v)$ denotes the coordinates of the vertex v:

$$p^{\mathsf{com}} = \left[\frac{\sum_m w_m^{\mathsf{c}}\ p(V_m^{\mathsf{c}}(t_k))}{\sum_m w_m^{\mathsf{c}}} \right], \quad m \in \{1, \dots, |V^{\mathsf{c}}(t_k)|\} \tag{3.103}$$

Then, the initial schedule $\pi^{\mathsf{c}}(t_k)$ is generated with the ordered vertices on the shortest path to the vertex v^{com} of p^{com} from the current vertex $v^{\mathsf{c}}(t_k)$, where $H^{\mathsf{c}}(t_k)$ number of waypoints are generated along it.

Having obtained a set of vertices that constitute $\pi^{\mathsf{c}}(t_k)$, the second phase assigns feasible times t_j^{c} to each vertex such that all robots can still safely return to the central agent within the new schedule. Vertices at the end of the central agent schedules are assumed to be valid indefinitely by the robot controllers, therefore the agent controller needs to estimate the earliest possible time it can leave its position by considering the robot schedules. Let the integer c_i^{charge} denote the schedule index within the control horizon of each robot that corresponds to a charge action. If there are no charge actions, then equals to the remaining control horizon within the schedule. Then, the time t_2^{c} of which the central agent should leave its current location is defined by:

$$t_2^{\mathsf{c}} = t_k + \mathsf{max} \left\{ \sum_{m=2}^{c_i^{\mathsf{charge}}} \delta^{\mathsf{time}} \left(\pi_{i,m-1}^{\mathsf{a}}(t_k),\ \pi_{i,m}^{\mathsf{a}}(t_k) \right) \mid \forall i \in \{1, \dots, N^{\mathsf{robot}}\} \right\} + \delta^{\mathsf{time}}(v_1^{\mathsf{c}}, v_2^{\mathsf{c}}) \tag{3.104}$$

where $\delta^{\mathsf{c}}(v_1^{\mathsf{c}}, v_2^{\mathsf{c}}) \in \mathbb{R}_{\geq 0}$ is the time it takes to traverse the consecutive vertices in the central agent schedule. Likewise, the remaining times on the planning horizon are calculated with:

$$t_{j+1}^{\mathsf{c}} = t_j^{\mathsf{c}} + \delta^{\mathsf{time}}(v_j^{\mathsf{c}}, v_{j+1}^{\mathsf{c}}), \quad \forall j \in \{2, \dots, |\pi^{\mathsf{c}}(t_k)| - 1\} \tag{3.105}$$

4

Case Study

The proposed controller architectures consist of a number of abstractions and parameters that require tuning for good performance. In this chapter, first, the environmental models are designed to adhere to the problem's mathematical formulation. The simulation environment is set up to reflect a real-life urban city block, with similar proportions and landscapes that would occur after a large-scale disaster. The design and assignment of features of the environment are programmed via a custom code using Python. The specifics of the USaR mission progression, victim modeling, physical properties of the modeled robots, the global task set update function, sensor modeling, and the action conflict definitions are provided in the Section 4.1. In Section 4.2, the methodology of benchmarking the proposed algorithms are given, as well as the tuned control parameters of the robots. The tuning of these parameters are done exhaustively through various simulations. The implementation of the algorithms and analyses in this chapter is done via MATLAB R2024b on a PC with Intel Core i7 Processor with 2.30 Ghz frequency. The GA algorithm is implemented via the native genetic algorithm implementation of MATLAB, using custom fitness, mutation, and crossover implementations. The MILP based controllers are implemented using Gurobi v12.0.1 on the MATLAB environment.

4.1. Simulation Setup

This section details how the specific mathematical modeling of the autonomous robotic team for SaR missions presented in Section 3.1 are implemented for the realistic computer-based simulations involved in the case study of this thesis, by specifying the mission environment, robot parameters, included action types and behaviors, task set evolution and fuzzy system implementation within the context of the proposed mission respectively.

4.1.1. Environment Implementation

The simulation setup is implemented following the same order in the mathematical formulation given in Section 3.1. The global vertex set $\mathcal{V}$ consists of a 100×100 grid equidistantly sampled from a 250000 m^2 USaR area. The search area is assumed to be unknown and unexplored before the start of the USaR mission. The edge set $\mathcal{E}$ is defined by the 8-connectivity of every vertex within its spatial neighborhood; therefore, robots can traverse only along the compass directions with respect to the orientation of the search area. The global feature set $\mathcal{F}$ consists of four categories as given in Figure 4.1. The first feature is the elevation of each vertex called terrain, given in Figure 4.1a, where the edge function ω^{env} models the Euclidean distances between the elevation of connected vertices. The other three features are destruction rate, population density and victims given in Figures 4.1b, 4.1c, and 4.1d respectively. The nonzero areas of destruction rate indicate the different types of structures present in the search area, where each one locally introduces unique features and challenges to the robot team. Similarly, the population density areas are concentrated around the buildings to establish a measurable quality to the prioritization of the destroyed areas. Finally, the victims are placed randomly by considering the population density values as a probability density function. Then, for each victim, the maximum time until that victim is detectable t^{victim} is decided randomly between $t^{\text{victim, max}}$ and $t^{\text{victim, min}}$ regarding a

uniform distribution with respect to the destruction rate. This means that if the destruction is low, the final time of detectability is more likely to be closer to the minimum time value. The measurable feature from vertices that contain victims is the health status, which is modeled by the following function for the victim m at time t_k:

$$\sigma_m^{\text{health}}(t_k) = \max\left\{0,\ 100 + t_k \frac{-100}{t_m^{\text{victim}}}\right\} \tag{4.1}$$

Where the victim can not be detected if the health value reaches zero at a given time.

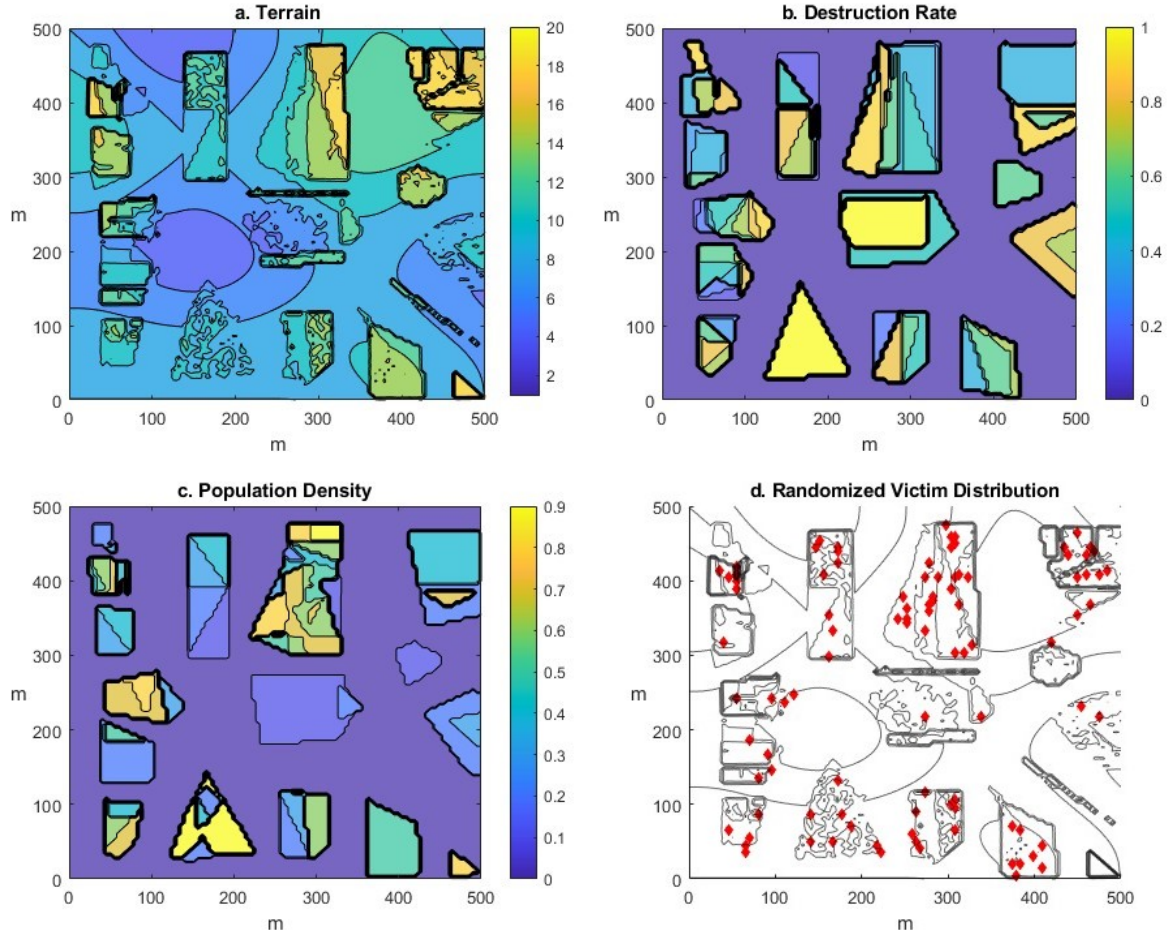


Figure 4.1: Ground truth features of the simulation environment representing the (a) elevation, (b) destruction rate, (c) population density, (d) randomly generated victims.

4.1.2. Action Types & Implementation

The implementation of the robots requires modeling the behavior of the state transition function $f_i^{\mathsf{s}}(s_i(t_k), a)$ and the observation function $f_i^{\mathsf{o}}(s_i(t_k), a)$, for $a = (\alpha, v) \in A_i(t_k)$, $\alpha \in \mathcal{C}_i$, and $v \in V^{\mathcal{M}}(t_k)$. To model the behavior of the robots specifically, some distinct types of actions are provided that aim to address the requirements of a USaR robot. Within this extended formulation, the letter ρ is used to introduce additional notation regarding the physical modeling of the robots, wherever necessary, as a real number, with subscripts denoting their context.

1. `idle` **action:** This action is included to allow the robots to stay idle if there are no other feasible actions present at the time.

2. `move` **action:** This action takes the robot from its current vertex $v_i^{\mathsf{r}}(t_k)$ to a neighboring vertex v. This action is only feasible if the edge connecting between these two vertices is traversable by the robot. It is usually more practical to encode a path via the `move` action directly, since the main objective of the control architecture is the task allocation. Thus, the action $a = (\alpha, v)$ denotes moving to the end vertex v from the current robot vertex using the shortest graph path that exists in the robot map $\mathcal{M}_i(t_k)$ between them. The traversal times and energies are dependent on the distance traveled $\omega^{\mathsf{env}}(v_i^r(t_k), v)$, robot speed ρ_i^{speed}, and energy per meter $d\epsilon_i^{\mathsf{move}}$:

$$\delta_i^{\mathsf{time}}(s_i(t_k), (\texttt{move}, v)) = \omega^{\mathsf{env}}(v_i^{\mathsf{r}}(t_k),\ v)/\rho_i^{\mathsf{speed}} \tag{4.2}$$

$$\delta_i^{\mathsf{energy}}(s_i(t_k^i), (\texttt{move}, v)) = d\epsilon_i^{\mathsf{move}} \cdot \omega^{\mathsf{env}}(v_i^{\mathsf{r}}(t_k), v) \tag{4.3}$$

3. `map` **action**: The mapping action is designed to get measurements from unknown areas of the search area, corresponding to the feature set $F_i^{\mathsf{map}} = \{\texttt{terrain}, \texttt{destruction}, \texttt{population}\}$ corresponding to observation of the terrain, destruction rate, and population values within the range ρ_i^{map}. Since these features are likely to be observed by a camera or a LiDAR in a real scenario, from a vertex v in the area, only the vertices within the direct line of sight can be measured at a given time t_k. The direct line of sight is calculated by the straight line between the target elevation and the current robot elevation, characterized by $v_i^r(t_k) + \rho_i^{\mathsf{height}}$, where the sensor height is a robot-specific parameter. The time and energy expenditure of a mapping action of a robot i is modeled by:

$$\delta_i^{\mathsf{time}}(s_i(t_k), (\texttt{map}, v)) = dt_i^{\mathsf{map}} + \delta_i^{\mathsf{time}}(s_i(t_k), (\texttt{move}, v)) \tag{4.4}$$

$$\delta_i^{\mathsf{energy}}(s_i(t_k), (\texttt{map}, v)) = d\epsilon_i^{\mathsf{map}} + \delta_i^{\mathsf{energy}}(s_i(t_k), (\texttt{move}, v)) \tag{4.5}$$

4. `search` **action:** In real-life scenarios, searching takes more time than mapping and is usually concentrated in the vicinity of areas that have a higher victim likelihood, which depends on the outcome of the `map` actions. Searching for victims are done using microphones, visual cues, heat sensors and various other methods that usually penetrate through walls and obstructions. Thus all vertices within ρ_i^{search} radius can be observed by the robots, where search actions correspond to the observation of the feature set $F_i^{\mathsf{search}} = \{\texttt{victim}\}$. The time and energy expenditure of a victim search action of a robot i is modeled by:

$$\delta_i^{\mathsf{time}}(s_i(t_k), (\texttt{search}, v)) = dt_i^{\mathsf{search}} + \delta_i^{\mathsf{time}}(s_i(t_k), (\texttt{move}, v)) \tag{4.6}$$

$$\delta_i^{\mathsf{energy}}(s_i(t_k), (\texttt{search}, v)) = d\epsilon_i^{\mathsf{search}} + \delta_i^{\mathsf{time}}(s_i(t_k), (\texttt{move}, v)) \tag{4.7}$$

5. `charge` **action:** Robots may return to the central agent for charging at any point, upon which this action can be taken if and only if the robot and central agent vertices are the same at a given time t_k. This action is included to provide a framework of an autonomous system that operates indefinitely and takes the remaining energy into account while planning the next actions. In this case study, it is assumed that the charging takes place instantaneously, considering methods like battery swapping are available in real life scenarios. Thus, the time and energy expenditure of charging is as follows for all robots:

$$\delta_i^{\mathsf{time}}(s_i(t_k), (\texttt{charge}, v^{\mathsf{c}}(t_k))) = 0 \tag{4.8}$$

$$\delta_i^{\mathsf{energy}}(s_i(t_k), (\texttt{charge}, v^{\mathsf{c}}(t_k))) = \epsilon_i(t_k) - 100 \tag{4.9}$$

Note that, similar to the `move` action definition, actions can be compounded to signify multiple actions taking place back-to-back, as given in the expressions for the search and map task resource expenditure. This formulation enables controllers to reduce the dimensionality of the allowed actions at the prediction horizon and create a finer-grained abstraction layer, guaranteeing a relevant observation as a result of considering a compounded task. For the purposes of this thesis, therefore, only `map` and `search` actions are considered for utility generation, while `charge` actions are used to satisfy the safe return constraints. As an extension, and to make full use of the MCDM formulation proposed in this thesis, if both `search` and `map` task are available from the same vertex v for a robot i, then the

compounded action $a = (\{\texttt{search}, \texttt{map}\}, v)$ takes the maximum of their individual times rather than the entire sum, as well as expending reduced energy:

$$\delta_i^{\text{time}}(s_i(t_k), (\{\texttt{search}, \texttt{map}\}, v)) = \max\{dt_i^{\text{search}}, dt_i^{\text{map}}\} + \delta_i^{\text{time}}(s_i(t_k), (\texttt{move}, v)) \tag{4.10}$$

$$\delta_i^{\text{energy}}(s_i(t_k), (\{\texttt{search}, \texttt{map}\}, v)) = (d\epsilon_i^{\text{search}} + d\epsilon_i^{\text{map}})/2 + \delta_i^{\text{energy}}(s_i(t_k), (\texttt{move}, v)) \tag{4.11}$$

This formulation encourages multitasking in the system designed for this case study. By design, the map update function and task set evolution are tied to the specific types of observations performed by robots when they take the relevant actions. At time t_0, the global task set and the robot maps are empty, where the robots are placed at the same vertex as the central agent. Since the mapping actions are required to determine where the search actions should take place, the configuration of the robot maps $\mathcal{M}_i(t_k)$ and the task set $\mathcal{T}(t_k)$ reflects this requirement. The local map vertices only contain the terrain measurements of the mapping attempts done throughout the mission. Each `map` action attempts to measure the vertices in the ground truth $\mathcal{G}$ that fall within the mapping radius of the robot. These vertices are all added to the local map, where each observation vertex that is not in the line of sight, therefore unmeasured, is given a placeholder terrain value of NaN. This approach creates a distinction between the unknown vertices that have not been attempted yet and the ones that have been attempted from a specific vertex but have failed. The observation prediction of the vertices that are not currently in the local map always assumes that they can be measured from all actions taken within the measurement radius. Alternatively, if a mapping action at a given location has produced NaN values, corresponding observations are removed from the prediction of the same action; thus, the robots emergently seek out different lines of sight to map these vertices. Additionally, the local maps only contain edges that are *traversable* by a robot, as decided by the following function:

$$\phi_i^{\text{traversable}}(v_1, v_2) = \text{true} \iff e(v_1, v_2) \in \mathcal{E} \ \wedge \ |v_1^{\text{terrain}} - v_2^{\text{terrain}}| \leq \rho_i^{\text{step}} \tag{4.12}$$

Denoting that the edge needs to exist, and the difference in elevation must be smaller than the maximum step size a robot i can take. After removing infeasible edges, the connected component of the local map that contains the current robot vertex becomes the new local map in the next time step. At the beginning of each path planning step, the local map is updated from the global map; conversely, at the end of each `map` and `search` action, the global map is updated directly. The global map $\mathcal{M}^{\text{c}}(t_k)$ contains all mission information, in addition to the terrain values of the local maps, including the timestamped central agent schedule, individual robot schedules, population density, destruction rate, and victim status. Consequently, the task sets are updated from the global map to enable condensed information to be processed by the local robots without going through all the features of the global mission status.

4.1.3. Task Set Implementation

The global task set $\mathcal{T}(t_k)$ starts empty, and in the context of this thesis, is updated autonomously by the central agent. The update function $\Phi(\mathcal{T}(t_k), a)$ creates tasks that can be attempted by `map` and `search` actions, where they can be simultaneously attached to the same task, from the current state of the global map. Since local maps always contain the vertices where individual robots exist, the first and only task is a mapping task at the starting location. Then, the available mapping tasks are updated by detecting the *frontier* vertices of the global map. Conversely, a mapping task is completed when the corresponding task vertex is no longer a part of the exploration frontier as a result of a robot action. Since the `map` actions can be performed from anywhere in the local robot map, the robots can complete and attempt multiple tasks with a single action. Similarly, search tasks are created on the vertices of the global map that has a search priority index larger than a mission-specific constant, which is set to:

$$\rho_l^{\text{priority}} \geq 0.1, \quad \forall \tau_l(t_k) \in \mathcal{T}^{\text{search}}(t_k) \tag{4.13}$$

Where $\mathcal{T}^{\text{search}}(t_k) \subseteq \mathcal{T}(t_k)$ denotes the set of all search tasks. A search task is completed and is not spawned again if a `search` action takes an observation from the location of the task. Finally, the set of allowed actions $A_i(t_k^i)$ of a robot i at local time t_k contains all vertices within $\rho_i^{\text{map}}/2$ proximity of mapping tasks as `map` actions, and similarly, all vertices within $\rho_i^{\text{search}}/2$ proximity of search tasks as `search` actions on its local map. As a requirement of the controller design, it also contains `charge` and `idle` actions, where all of the elements in $A_i(t_k)$ can be compounded by `move` actions to denote

different paths the robot can take to perform them. As a design choice, the shortest weighted path on the local map to all action-associated vertices is generated using the compounded `move` tasks, constituting the final feasible action set at a given timestep. As a final note, a search task involves tasks that have the feature set $F^{\text{task}} = \{\texttt{victim}\}$, and a mapping task involves tasks that have the feature set $F^{\text{task}} = \{\texttt{terrain}, \texttt{destruction}, \texttt{population}\}$.

The central agent is configured similarly to the robots, of which the only defined actions are `move` and `idle`. The transitional properties are modeled as follows:

$$\delta^{\text{c}}(v^{\text{c}}, v) = \omega^{\text{env}}(v^{\text{c}}, v)/\rho^{\text{speed}}_{\text{central}} \tag{4.14}$$

$$\phi^{\text{traversable}}_{\text{central}}(v_1, v_2) = \text{true} \iff e(v_1, v_2) \in \mathcal{E} \ \wedge \ |v_1^{\text{terrain}} - v_2^{\text{terrain}}| \leq \rho^{\text{step}}_{\text{central}} \tag{4.15}$$

Specifying the coordination and conflict behaviors within the system, the function $\phi^{\text{coord}}(r_1(t_k), r_2(t_k)) = 1$ is set if the average Cartesian distance between scheduled action pairs within the control horizon is at most 200 meters, which covers the footprints of most buildings in the case study. Given two separate actions from robots i and j as $a_i \in \mathcal{A}_i$ and $a_j \in \mathcal{A}_j$, a conflict is defined if both actions are of the same type and the Cartesian distance between the final vertices $d(v_i, v_j)$ upon taking these actions is within half of the average radius of the corresponding observation areas. This is established to create a spatial spread between the robots, enabling more efficient exploration of the USaR area while still allowing simultaneous operation within close proximity, if different types of actions are taken to generate better exploitation of the explored areas. Specifically, in terms of this case study, the conflict detection takes the form of:

$$\phi^{\text{conflict}}((\texttt{map}, v_i)), (\texttt{map}, v_j)) = \text{true} \iff d(v_i, v_j) \leq (\rho_i^{\text{map}} + \rho_j^{\text{map}})/4 \tag{4.16}$$

$$\phi^{\text{conflict}}((\texttt{search}, v_i), (\texttt{search}, v_i)) = \text{true} \iff d(v_i, v_j) \leq (\rho_i^{\text{search}} + \rho_j^{\text{search}})/4 \tag{4.17}$$

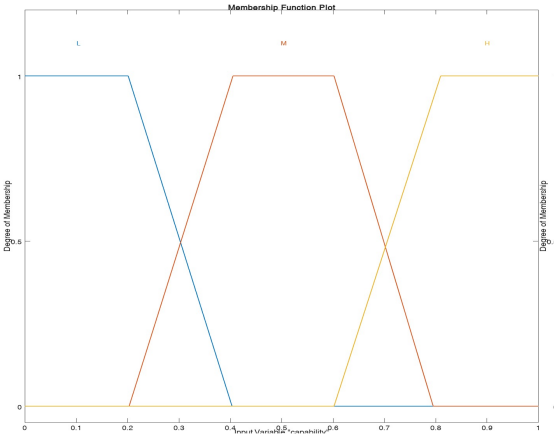
4.1.4. Fuzzy System Implementation

The implementation of the specific membership functions follow the results presented in the work of Hassanzadeh et al. [5] , where a fuzzy inference mechanism for human teams in USaR missions is proposed for effective prioritization of objectives relating to the available environmental information. Specifically, their framework is directly implemented for task evaluation, where the destruction and population estimates are converted to a prioritization value for victim searching. Since there are five membership functions at the output of this model, the fuzzy system for evaluating actions that runs in parallel also has the same number of membership functions for establishing symmetrical processes. The final output of the fuzzy system considers three membership functions for both inputs and outputs for simplicity. Furthermore, trapezoidal membership functions are used when the underlying data is categorical, and Gaussian functions are used when it is continuous and smooth, such as distance and quality measures, which are estimated to be taken parametrically from sensory measurements.

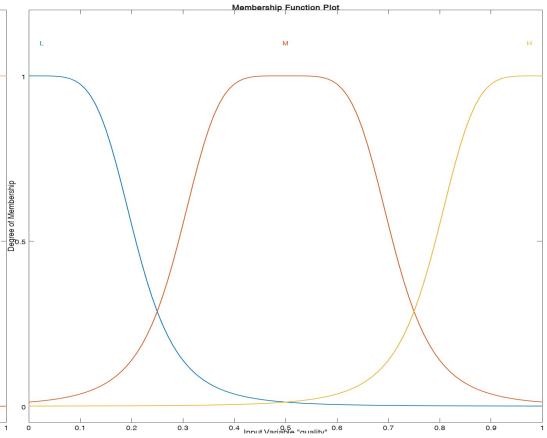
As shown in Figure 3.3, each action linked to a task type is evaluated via a network of fuzzy inference systems. The action evaluation part of this network is designed as a Mamdani Type-1 system where the ruleset is given in Table 4.1. Actions are robot-specific; therefore, their evaluation is designed to be related to the observational capabilities upon taking `search` and `map` actions. The fuzzy input sets $B^{\text{action}}_{m,1}$ $B^{\text{action}}_{m,3}$ include the ”Low”, ”Medium”, and ”High” membership functions to model the robot’s ability to take that action, the quality of observation, and the quantity of observation, respectively. The capability is defined rather vaguely, however, in the context of this case study, it represents the level of reliability of the robot’s action compared to other actions and other robots. The quantity is the normalized number of observations predicted to be measured upon taking the action for the predicted states in the receding horizon control. At the same time, the quality is determined by the average Euclidean distance of the robot’s location during the action to the locations of the predicted observations.

$\mathcal{R}_m^{\text{action}}$	σ^{cap}	σ^{qual}	σ^{quant}	ρ^{action}	$\mathcal{R}_m^{\text{action}}$	σ^{cap}	σ^{qual}	σ^{quant}	ρ^{action}
1	Low	Low	Low	Very Low	15	High	Medium	Medium	High
2	Medium	Low	Low	Very Low	16	Low	High	Medium	Medium
3	High	Low	Low	Low	17	Medium	High	Medium	Medium
4	Low	Medium	Low	Very Low	18	High	High	Medium	High
5	Medium	Medium	Low	Low	19	Low	Low	High	Medium
6	High	Medium	Low	Low	20	Medium	Low	High	High
7	Low	High	Low	Very Low	21	High	Low	High	Medium
8	Medium	High	Low	Low	22	Low	Medium	High	Medium
9	High	High	Low	Medium	23	Medium	Medium	High	High
10	Low	Low	Medium	Low	24	High	Medium	High	Very High
11	Medium	Low	Medium	Medium	25	Low	High	High	High
12	High	Low	Medium	Medium	26	Medium	High	High	Very High
13	Low	Medium	Medium	Low	27	High	High	High	Very High
14	Medium	Medium	Medium	Medium					

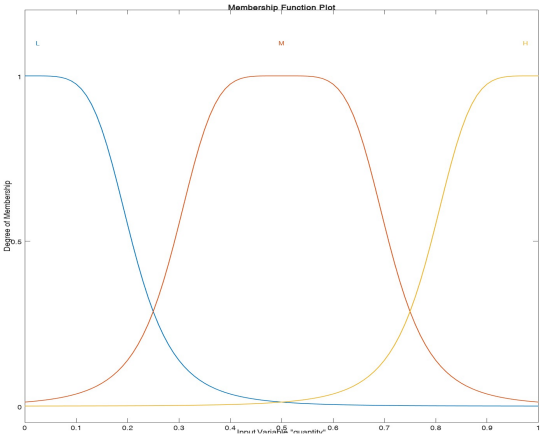
Table 4.1: Rule base of the action evaluation system with inputs σ^{cap} for the action capability of the robot, σ^{qual} for the quality of observations, and σ^{quant} the quantity of observations upon the observation of the action result.



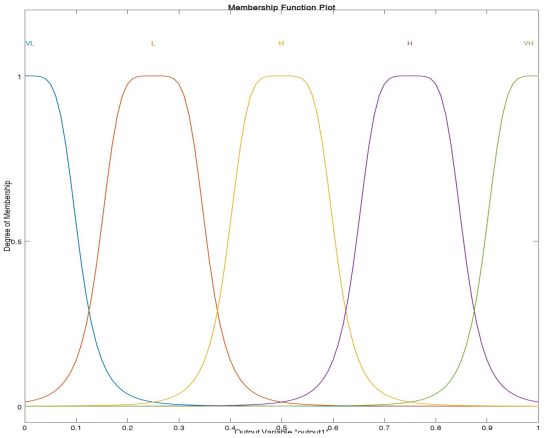
(a) (Low) (Medium) (High) membership of the action capability of a robot.



(b) (Low) (Medium) (High) membership of the action observation quantities.



(c) (Low) (Medium) (High) membership of the action observation quality.

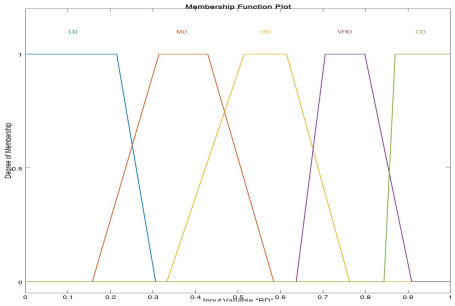


(d) (Very Low) (Low) (Medium) (High) (Very High) membership of the action evaluation output.

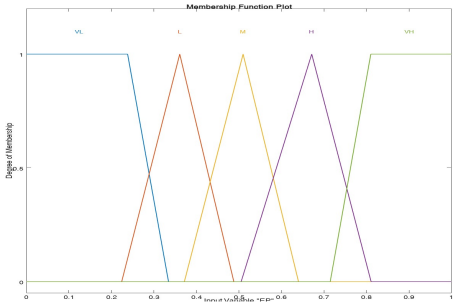
Figure 4.2: Fuzzy membership functions for the inputs and outputs of the Mamdani Type-1 rule bases of the action evaluation system.

$\mathcal{R}^{\text{priority}}_{m,(l)}$	$\sigma^{\text{priority}}_{(l)(\text{dest})}$	$\sigma^{\text{priority}}_{(l)(\text{pop})}$	$\rho^{\text{priority}}_{(l)}$	$\mathcal{R}^{\text{priority}}_{m,(l)}$	$\sigma^{\text{priority}}_{(l)(\text{dest})}$	$\sigma^{\text{priority}}_{(l)(\text{pop})}$	$\rho^{\text{priority}}_{(l)}$
1	Low	Very Low	None	14	Very High	Medium	Medium
2	Medium	Very Low	None	15	Collapsed	Medium	Low
3	High	Very Low	None	16	Low	High	Low
4	Very High	Very Low	Low	17	Medium	High	Medium
5	Collapsed	Very Low	None	18	High	High	High
6	Low	Low	None	19	Very High	High	High
7	Medium	Low	None	20	Collapsed	High	Medium
8	High	Low	Low	21	Low	Very High	Medium
9	Very High	Low	Low	22	Medium	Very High	High
10	Collapsed	Low	None	23	High	Very High	High
11	Low	Medium	Low	24	Very High	Very High	Very High
12	Medium	Medium	Low	25	Collapsed	Very High	High
13	High	Medium	Medium				

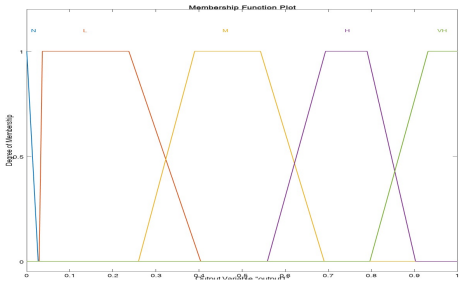
Table 4.2: Rule base of the victim search prioritization of a task $\tau_l(t_k)$ with inputs $\sigma^{\text{priority}}_{(l)(\text{dest})}$ for the destruction rate and $\sigma^{\text{priority}}_{(l)(\text{pop})}$ for the population density estimates in the vicinity of the task.



(a) (Low Damage) (Medium Damage) (High Damage) (Very High Damage) (Collapsed) membership of the measured destruction level.



(b) (Very Low) (Low) (Medium) (High) (Very High) membership of the measured population density estimate.



(c) (None) (Low) (Medium) (High) (Very High) membership of the search prioritization of a task.

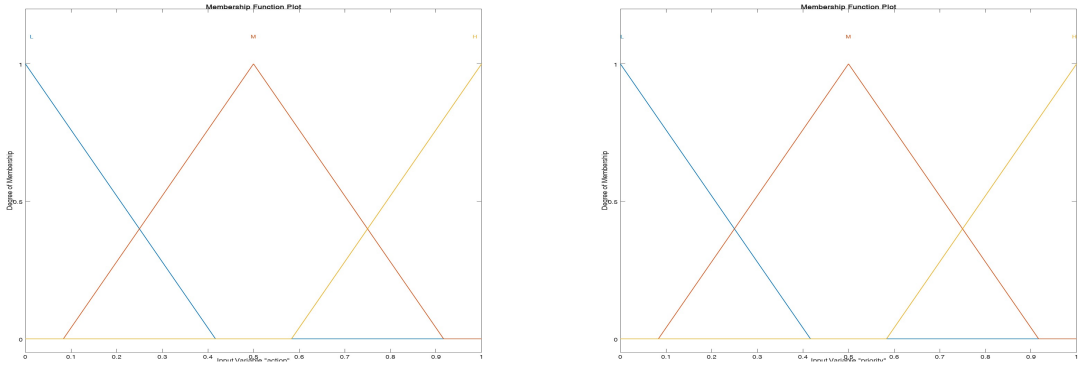
Figure 4.3: Fuzzy membership functions for the inputs and outputs of the Mamdani Type-1 rule bases of the search prioritization system.

The output set C^{action}_m includes ”Very Low”, ”Low”, ”Medium”, ”High”, and ”Very High” membership functions to enable a fine-grained rule set. of which the membership function plots are given in Figure 4.2. Subsequently, the priority of tasks is calculated via another Mamdani Type-1 system to denote the value of attempting a specific task from the perspective of the USaR mission goals independently of the robot’s actions. For a mapping task, the only relevant criteria chosen in this case study is the number of unknown neighbors of the task vertex in the mission map $\mathcal{M}^{\text{c}}(t_k)$. Since there is only one input and one output, a fuzzy system is not specifically modeled for mapping tasks, and their priority is directly set to the normalized value of unexplored vertices adjacent to the task. The fuzzy model of the searching tasks is directly inspired by the real-life USaR decision-making [5] of human teams on the field. The environmental features measured from mapping actions are used as inputs for the task pri-

oritization system. The fuzzy set $B^{\text{priority}}_{m,(search),1}$ models the destruction rate of a vertex with membership functions ”Low”, ”Medium”, ”High”, ”Very High”, and ”Collapsed” and the fuzzy set $B^{\text{priority}}_{m,(search),2}$ models the estimated population density with membership functions ”Very Low”, ”Low”, ”Medium”, ”High”, and ”Very High”. The output set C^{priority}_{m} also includes the categories ”Very Low”, ”Low”, ”Medium”, ”High”, and ”Very High””; where the rule set and the membership functions of the task prioritization system are given in Table 4.2 and Figure 4.3. One important aspect of this rule set is that under some conditions, the search priority is lowered when the destruction rate increases from ”Very High” to ”Collapsed”, since the latter category denotes a very low survival rate.

$\mathcal{R}^{\text{task}}_m$	ρ^{action}	ρ^{priority}	$f^{\text{task}}(\rho^{\text{action}}, \rho^{\text{priority}})$
1	Low	Low	Very Low
2	Medium	Low	Low
3	High	Low	Medium
4	Low	Medium	Low
5	Medium	Medium	Medium
6	High	Medium	High
7	Low	High	Medium
8	Medium	High	High
9	High	High	Very High

Table 4.3: Rule base of the task evaluation system using the action evaluation ρ^{action} and the task prioritization ρ^{priority} values to evaluate the direct utility of attempting a task through an action.



(a) (Low) (Medium) (High) membership of the action evaluation result of the robot. **(b)** (Low) (Medium) (High) membership of the task prioritization result of the robot.

Figure 4.4: Fuzzy membership functions for the inputs and outputs of the Sugeno Type-1 rule bases of the task value evaluation system.

The outputs of the initial fuzzy systems for task prioritization and action evaluation are fed to a Takagi-Sugeno Type-1 fuzzy system for determining the final crisp value $\nu^{\text{task}}_l(\mathcal{T}(t_k), s_i(t_k), a)$ for each affiliated task with the given input. The rule base and the input membership functions of this stage are given in Table 4.3 and Figure 4.4. This stage is chosen to be a Takagi-Sugeno system due to its ability to directly link outputs to functions, which suits the modularity aspect of the controller design. The output categories of the fuzzy system in Table 4.3 correspond to the following constant values in the context of this case study:

$$\nu^{\text{task}}_l(\mathcal{T}(t_k), s_i(t_k), a) = \begin{cases} 0 & f^{\text{task}}_m(\rho^{\text{action}}_i, \rho^{\text{priority}}_l) = \text{Very Low} \\ 0.25 & f^{\text{task}}_m(\rho^{\text{action}}_i, \rho^{\text{priority}}_l) = \text{Low} \\ 0.5 & f^{\text{task}}_m(\rho^{\text{action}}_i, \rho^{\text{priority}}_l) = \text{Medium} \\ 0.75 & f^{\text{task}}_m(\rho^{\text{action}}_i, \rho^{\text{priority}}_l) = \text{High} \\ 1 & f^{\text{task}}_m(\rho^{\text{action}}_i, \rho^{\text{priority}}_l) = \text{Very High} \end{cases} \tag{4.18}$$

4.2. Experiment Setup

This section concludes the construction of the case study by introducing the implemented controllers to be benchmarked against the proposed MILP-based controller, then specifying the metrics of the USaR mission defined for the case study to determine the performance of the included controllers. Finally, the specific mission parameters are detailed to introduce the types of experiments that are carried out and their boundaries.

4.2.1. Implemented Controllers

The hierarchical MILP-based MPC controllers proposed in this thesis perform their optimization considering multiple layers of information available at a given time. These layers are defined by the USaR goals of time minimization and efficient exploitation of the search area exploration, which are conflicting goals. The three main components of the information processing in the proposed controllers are selecting the actions that minimize time, maximizing the robot's fuzzy decision-making system output, and maximizing the actions' MCDM output. The MCDM formulation combines the minimum time and maximum utility of an action into a single value, incorporating a trade-off between the two. To effectively analyze the behavior and performance of the proposed architecture, the following controllers are implemented:

- **GA controller:** The GA controller is the direct implementation of the nonlinear MPC formulation described in Problem (3.35), to establish a baseline for the proposed architecture. This controller considers all possible feasible actions related to the selected task set at a given time and evaluates the MCDM utilities sequentially for a candidate schedule of actions, using observation predictions that contribute new information to the system at each prediction step. For the experiment setup, the maximum iteration count is set to $N^{\text{gen}} = 200$ the number of candidate schedules in a population $n^{\text{pop}} = 20$, the elite count $n^{\text{elite}} = 2$, the crossover ratio $\rho^{\text{co}} = 0.7$, and the mutation rate $\rho^{\text{mu}} = 0.2$. Finally, the maximum stall generations is set to $N^{\text{stall}} = 25$, after considerable tuning of all variables through various trials. The implementation of this controller closely follows the state-of-the-art approach for generalized OP via genetic optimization presented by Kobeaga et al. [76], with the addition of the specific constraints presented in the methodology of this thesis.
- **MILP controller:** The MILP controller is the proposed solution in this thesis, which is implemented as a two-stage algorithm that linearizes the MPC formulation of the robot controllers. The MILP problem does not require any parameters to be set, but the LNS improvement framework described in Algorithm 1 requires a maximum number of iterations and stall iterations, set to $N^{\text{iter}} = 10$ and $N^{\text{stall}} = 5$. The initial partial solution to the incumbent problem is given a maximum of 10 seconds, or until the optimal solution is found before the limit is reached. Similarly, LNS improvement solutions are each given at most 3 seconds. Finally, the number of stochastic best action selection during LNS improvements is set to $n^{\text{best}} = \lceil H_i^{\text{pred}}/2 \rceil$. Finally, the maximum number of overlaps for the first stage constraints $N^{\text{overlap}} = 1$ is set. The reason for not setting this value to zero is to enable a higher fidelity of action selections due to the density of tasks in this case study, at the expense of underestimating action utilities.
- **hybrid controller:** The hybrid controller employs the first stage of the MILP controller for reducing the action space, however, uses the GA controller on the reduced action set. This controller is included to set a bridge between the GA and MILP approaches and make a direct comparison between them possible, since they operate under different assumptions about the action sets in the context of this case study.
- **shortest-time controller:** The shortest-time controller selects the action with the shortest completion time at a given time step from the feasible action set. This controller aims to demonstrate the effect of the mission outputs if the only consideration is minimizing the time, which is one of the highlighted components of the proposed controllers.
- **fuzzy controller:** The fuzzy controller selects the action with the best aggregated fuzzy evaluation as given in Equation (3.29). The fuzzy utility vector $\mathbf{u}_i^{\text{action}}$ contains the utility for both `search` and `map` tasks for the first prediction time step, where the corresponding action that has the maximum value is chosen using this vector. The fuzzy controllers do not consider the timings, and the proposed MPC methods only consider the completion of current tasks rather than the creation

of new ones. A one-step implementation of the fuzzy controller is deemed suitable for comparing with the shortest-time controller and the overall utility exploitation of the MCDM-based utility generation proposed in the methodology.

- **MCDM controller:** The MCDM controller is similar to the fuzzy and shortest-time controllers in the sense that it is a one-step implementation of selecting the action with the most MCDM utility at a given timestep. This controller is included for direct comparison with the other one-step implementations and to cover all three of the optimized variables within the proposed solutions. Furthermore, it enables a better analysis of the effect of the receding-horizon control for the optimization problem laid out in this thesis.
- **random controller:** The random controller selects a random action from the feasible action set at each timestep, and included to establish a baseline for the results in the context of this specific case study environment.

4.2.2. Mission Metrics

The metrics used for quantifying the performance of the robots in terms of both the controller behavior and USaR mission goals are as follows:

- **mapped area:** This metric measures the direct effect of taking mapping actions. Both the total area mapped at the end of the mission duration and the rate at which this value is reached are relevant to the analysis of the USaR performance for this metric.
- **distance covered:** Since the action times are non-uniform through actions, instead of directly measuring the time for certain predetermined milestones, the total covered distance is deemed to be a relevant metric that makes it possible to analyze the efficiency in robot actions. The primary function of the center of mass controller is to minimize the distances that robots need to travel for charging, thereby reducing the time spent idling.
- **detected victims:** The number and health status of the detected victims is the ultimate metric of the USaR goals laid out in this thesis. For all scenarios, there are 100 victims, some of whom may be unreachable or undetectable due to the randomness of their placement. Still, similar to the mapped area metric, the total number at the end of the USaR mission, as well as how fast victims are found and their overall health status, define the success of a controller.
- **task completion time:** Tasks are spawned emergently from the mission state, where multiple can be completed with one action. How quickly good-quality tasks are removed from the pool of available tasks at each control step, inadvertently, should also affect the other metrics, aiming to provide a better insight into the information exploitation that the controllers can offer within the case study. Since tasks are spawned at different times, the duration for which each task stays *alive* after spawning constitutes this metric.
- **utility per control step:** The local controllers aim to maximize the MCDM utility for the entirety of the mission. The direct outputs of mapped area, covered distance, and victim detection cover the USaR mission goals. However, assessing the correlation between the utilities per control step and the USaR outcomes is equally important to conclude if the controller design works as intended. The utilities are calculated for both the fuzzy value of an action, which is the value without time optimization, and for the MCDM value that combines the timing and task completion resulting from the action into a single value.
- **runtime:** The final metric is the runtime of the algorithms. The described optimization problem has a factorial search space by the number of estimated actions and the prediction horizon. While GA offers a generalized solution, the MILP algorithm can efficiently branch and bound this search space within the given time budget. Since there is no time budget enforced on the GA, and a reasonable number of candidate evaluations are provided, the runtime difference between these methods in combination with the other performance metrics gives a complete picture of the performance for the proposed MILP architecture.

4.2.3. Mission Parameters

All controllers in the experiments consider the closest $N^{\text{tasks}} = 200$ tasks to a robot's current position at each timestep to establish locality, where the related feasible action set may vary in size. The selection of the considered task set is set to a high number since the task definitions in the case study are observed to be considerably dense. Although the optimization problem becomes more challenging with the current formulation and may not be the best approach for a real-life implementation, it highlights the fidelity and performance of the proposed solutions as a trade-off. The central agent is considered under two scenarios. The first scenario involves a *static* central agent that remains in its initial location throughout the mission. The second scenario includes a *dynamic* central agent where the center of mass controller is defined in Section 3.2.3. This controller aims to steer the center of the mission toward the midpoint of all available tasks, helping to escape local minima and redundant actions due to over-exploitation of the USaR search area. Since all robots must return to charge at some point, there may be a scenario where the central agent remains in equilibrium and all robots are currently docked; therefore, the task state does not change to alter this equilibrium, effectively halting the entire system. To avoid this situation, N^{range} is set to a suitable value, where the central agent must select a goal point within this range for any attemptable task when a new schedule is made. The size of the central agent schedule at the time of construction is defined by the constant H^{c}.

mission parameters		**MCDM weights**	
t^{end}	10:00 (hh:mm)	μ^{t}	1
N^{victim}	100	μ^{m}	3
$t^{\text{victim,max}}$	10:00 (hh:mm)	μ^{s}	7
$t^{\text{victim,min}}$	05:00 (hh:mm)	μ^{mt}	6
N^{range} (central agent)	50 (m)	μ^{st}	12
$\rho^{\text{speed}}_{\text{central}}$	0.5 (m/s)	μ^{ms}	13
$\rho^{\text{step}}_{\text{central}}$	0.5 (m)	μ^{mst}	13
H^{c}	5		

Table 4.4: Mission parameters of the case study.

The last mission-related parameters to be discussed are the MCDM fuzzy measures μ as introduced in the Equation (3.33). This case study considers two actions, as well as the timing meta-action; therefore, there are seven weights to be set in total, for each value in the power set of these three categories. In practice, the elements within each set of this power set denote a mutual optimization of the involved variables. For example, the fuzzy measure for the set that contains all three action categories quantifies the importance of optimizing all three simultaneously. For the USaR mission defined in this case study, finding victims as fast as possible is the main objective, while mapping the area efficiently is the secondary objective. However, due to the structure of the task generation system, spawning new search tasks depends entirely on good selections of which mapping tasks should be attempted first. By using the initial letters of all categories to denote the corresponding power set, the single category weights are designed to follow $\mu^{\text{t}} < \mu^{\text{m}} < \mu^{\text{s}}$. When multiple goals are possible to be optimized at the same time, the relationship stays the same, however should be substantially higher than their single category counterparts, as $\mu^{\text{m}} < \mu^{\text{mt}}$ and $\mu^{\text{s}} < \mu^{\text{st}}$. If a very good quality action is possible which optimizes for both task types at the expense of time minimization, the candidate is worth exploring, therefore $\mu^{(\cdot)} < \mu^{\text{ms}} \leq \mu^{\text{mst}}$ is established. The final weight μ^{mst} describes the best possible action in the case study, where the action optimizes perfectly for all categories. This is improbable in practice, especially in the framework of this case study. Therefore, its value is set close to the second-best scenario described μ^{ms}. The values for all mission-related parameters and MCDM weights discussed thus far are summarized in Table 4.4. All missions terminate at the 10^{th} hour, and all mission types contain randomly generated 100 victims.

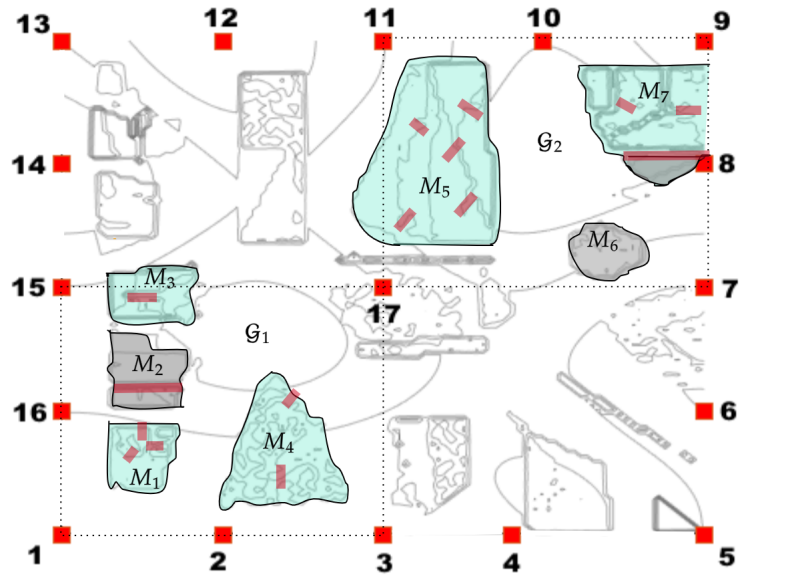


Figure 4.5: Initial conditions and the areas of interest of the experiments.

The controllers proposed in the methodology, when applied simultaneously, operate independently of each other. The local controllers operate with the information on their local maps and the tasks with the closest proximity; however, if there is a conflict in schedules, the locally optimal task allocations are separated into approximately globally optimal schedules. This behavior makes it difficult to assess the direct effects of the local controller design on the mission outputs. Furthermore, the center-of-mass controller defined for the central agent shifts the focal point of the robotic mission closer to the unfinished tasks, therefore affects the local controller decisions in terms of the changed feasibility of different actions at each planning step and makes it difficult to analyze the local exploitation of the gathered environmental information at each charging cycle. For these reasons, the experiments are split into single and multi-robot runs, with two datasets each for static and dynamic central agent conditions. The datasets are gathered with two repetitions for each initial condition given in Figure 4.5, (denoted with the red squares), amounting to 34 runs per controller.

In addition to the quantitative metrics provided for the case study, a qualitative analysis of the behavior of the robot controllers should also be performed. As simplified scenarios, the lower left ($\mathcal{G}_1$) and top right ($\mathcal{G}_2$) quadrants of the search space are separated to run both GA and MILP algorithms, as given in Figure 4.5. These quadrants contain interesting subareas in terms of the traversability, destruction, population density, and the overall challenge to the controllers. The subareas are also marked in Figure 4.5, where blue areas denote buildings that are reachable for a ground robot, and the gray areas denote buildings that are not. Reachability is usually established through step sizes between vertices; however, the pink rectangles in Figure 4.5 denote vertical edge connections to vertex groups with higher elevations that are otherwise inaccessible to a ground robot. The exploration of those areas depends on the exploration of these vertical ramps; therefore, it is a difficult task to achieve consistently. The initial condition for the area $\mathcal{G}_1$ is set to point 1, whereas the initial condition for the area $\mathcal{G}_2$ is set to point 9. These simplified experiments are designed to have a maximum duration of four hours, and 100 victims are placed within them, similar to the global environment. The robot parameters used for the single robot experiments are provided in Table 4.5, of which the traversability analysis is performed based on its selected step size of 2 meters. The prediction horizon is selected as $H^{\text{pred}} = 6$ since the average expectation of the action energy expenditure is 7.5 with the selected parameters, which accumulates to approximately half of the maximum energy of the robot along the prediction horizon. A charge cycle is defined by the actions the robot takes between two charge actions. With the selected value of the prediction horizon, half of a charge cycle is expected to be predicted, which is observed to be sufficiently accurate through trial and error. Finally, the control horizon is selected as one to account for the lack of task spawn predictions in the MPC formulation.

UGV		move actions		map actions		search actions	
N^{tasks}	200	ρ^{step}	2 (m)	ρ^{map}	30 (m)	ρ^{search}	20 (m)
H^{pred}	6	$d\epsilon^{\text{move}}$	0.1 (1/m)	$d\epsilon^{\text{map}}$	5	$d\epsilon^{\text{search}}$	10
ϵ^{crit}	0	ρ^{speed}	1 (m/s)	dt^{map}	00:30 (mm:ss)	dt^{search}	03:00 (mm:ss)
ρ^{height}	1 (m)			$\sigma^{\text{cap}}_{\text{map}}$	1	$\sigma^{\text{cap}}_{\text{search}}$	1

Table 4.5: UGV parameters used in the single robot experiments.

For multi-robot experiments, a similar qualitative analysis is performed on the areas $\mathcal{G}_1$, and $\mathcal{G}_2$, however, this time, two additional robots are modeled to observe the ability of the proposed architecture to allocate tasks to better capable robots, as well as to demonstrate the modularity and heterogeneity claims of this thesis. The first robot is a UAV, whose height and step sizes are large enough to reach any location on the map with unobstructed vision. Aligning with the use of UAVs in real-life USaR scenarios, the mapping capability is given the maximum value while the search capability is reduced since it is not expected to go inside the buildings. In contrast, the second robot is implemented as a crawler, whose step size is given a large enough value that it can reach everywhere in the USaR environment due to its small size and ability to crawl under rubble. This robot still suffers from the same vision blockages as the UGV. While it has the maximum search capability, its mapping capabilities are reduced as a tradeoff. The parameters for these robots are given in Tables 4.6 and 4.7. As the final component of their modeling, the capability is defined as the rate of error in measurements, where the robots will fail to measure random observations in proportion to the loss of capability per action. This differs from the other components of action evaluation, namely quantity and quality, as both are calculated based on the predictions of robot actions rather than actual measurements. The quality degrades with distance for both actions, hence the mapping observations are selected with respect to the probability density function defined by the following parametric formula:

$$f^{\text{map}}(d) = 0.4 + \frac{0.6}{1 + e^{10(d-0.7)}} \tag{4.19}$$

Where the variable d denotes the distance of each predicted observation. For search actions, not only does the probability of detection decrease with distance, but it also changes with the level of destruction in the area. Aligning with the fuzzy evaluation system established for action predictions, the probability of detecting a victim increases with the destruction level until the ”Very Damaged” membership function peak, then it slightly decreases as the destruction level reaches ”Collapsed”. The probability density function is denoted by:

$$c(b) = 1 - \frac{1}{e^{-10(b-0.3)} + e^{b-0.5}} \tag{4.20}$$

$$f^{\text{search}}(b, d) = \frac{0.8}{e^{5c(b)d}} + 0.2 \tag{4.21}$$

Where b denotes the building destruction level for each observation. At each measurement step, the number of observations corresponding to $1 - \sigma^{\text{cap}}$ of the total area within range are selected to be unmeasured, using the probability density functions defined above. With this formulation, the aim is to test the capability of the coordination system to allocate robots that are better suited for certain tasks, how balancing the elapsed time affects these allocations, and what the outcomes of the coordination performance are in terms of the quantitative metrics defined for the single robot case. For completeness, a team with two UGVs is also tested for a meaningful comparison with the results of single-robot experiments.

UAV		move actions		map actions		search actions	
N^{tasks}	200	ρ^{step}	25 (m)	ρ^{map}	30 (m)	ρ^{search}	20 (m)
H^{pred}	6	$d\epsilon^{\text{move}}$	0.1 (1/m)	$d\epsilon^{\text{map}}$	5	$d\epsilon^{\text{search}}$	10
ϵ^{crit}	0	ρ^{speed}	1 (m/s)	dt^{map}	00:30 (mm:ss)	dt^{search}	03:00 (mm:ss)
ρ^{height}	25 (m)			$\sigma^{\text{cap}}_{\text{map}}$	1	$\sigma^{\text{cap}}_{\text{search}}$	0.3

Table 4.6: UAV parameters used in the multi-robot experiments.

Crawler		move actions		map actions		search actions	
N^{tasks}	200	ρ^{step}	25 (m)	ρ^{map}	20 (m)	ρ^{search}	20 (m)
H^{pred}	6	$d\epsilon^{\text{move}}$	0.1 (1/m)	$d\epsilon^{\text{map}}$	5	$d\epsilon^{\text{search}}$	10
ϵ^{crit}	0	ρ^{speed}	0.5 (m/s)	dt^{map}	00:30 (mm:ss)	dt^{search}	03:00 (mm:ss)
ρ^{height}	1 (m)			$\sigma^{\text{cap}}_{\text{map}}$	0.3	$\sigma^{\text{cap}}_{\text{search}}$	1

Table 4.7: Crawler parameters used in the multi-robot experiments.

5

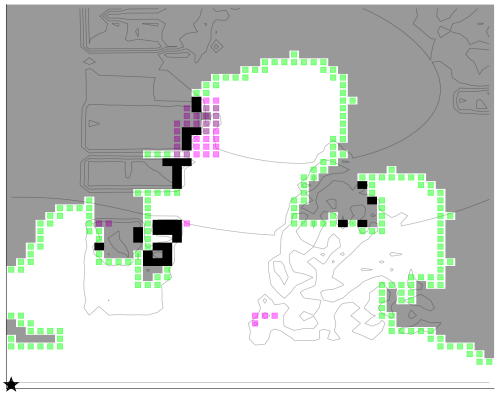
Results & Discussions

This chapter presents and discusses the results of the experimental setup presented in Chapter 4 with the proposed control architecture and environmental modeling in Chapter 3. The experiments are carried out first with only one robot to thoroughly compare the MPC based MCDM methodology proposed in this thesis to other relevant approaches, and then the proposed control architecture is compared with multiple different robotic team comparisons to showcase the performance of the global coordination controller at the top of the hierarchy in the defined system. The layout of the presented results for both types of experiments follows: first, a qualitative analysis and behavioral presentation of how the autonomous robotic team behaves in smaller scenarios, and then, a more detailed quantitative analysis for generalized scenarios across many experiments is provided to conclude the final discussions. At the beginning of this chapter, a representative demonstration of how the task-allocation framework works in a computer-based simulation is presented for better contextualization of the work included in this thesis.

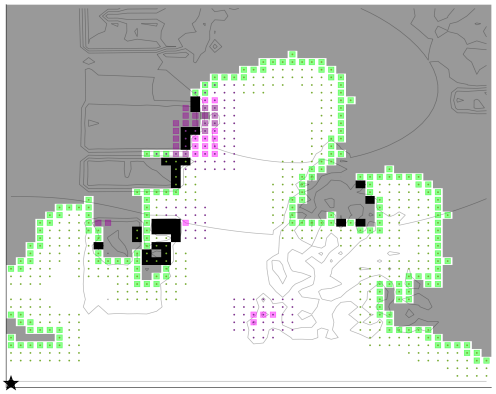
5.1. Demonstration of the Task Allocation Behavior

Starting with the demonstrative example of how task allocation occurs at a timestep where the robot starts from the static central agent after a charging action, the steps of the MILP controller operation are shown in Figure 5.1. The vertices that are not explored, i.e., not on the local map, are marked gray, the vertices that are on the local map but are not accessible are marked black, and others are marked white. Furthermore, the central agent is represented by a pentagram, and the robot's location is marked with a black circle, both of which are positioned in the bottom left corner, as the robot had just finished charging at this specific timestep. The optimization procedure proposed in this thesis begins by distilling the set of considered tasks, as shown in Figure 5.1a. Here, the green squares represent a mapping task established by the global map's exploration frontier. The purple squares denote search tasks spawned by the observations of map actions that exceed the search priority threshold. The set of feasible actions related to these tasks is given in Figure 5.1b, where dots represent an action with the corresponding color to their related tasks. The feasible set of actions must consist of actions that the robot can move to, perform, and return to the charger without violating the safe return constraints. Therefore, selecting at least one action from this set is always possible. The MILP controller assumes that the utilities of sequential actions are independent and can be calculated from the start. To maintain this assumption, the first stage of the MILP controller selects actions that maximize the task utility while adhering to a maximum of one overlap constraint for each unique observation of the selected actions. The selected actions and the area of observation for each of them are shown in Figure 5.1c. Initially, in this example (performed separately from the structure explained for the case study), there are 225 map tasks and 43 search tasks, totaling 596 map actions and 118 search actions. After the first stage of the MILP controller, the total number of map actions and search actions reduces to 19 and 6, respectively, corresponding to a 96.4% reduction in the search space. The second stage of the MILP controller then finds the best order of H^{pred} actions from the reduced action set, and calculates the

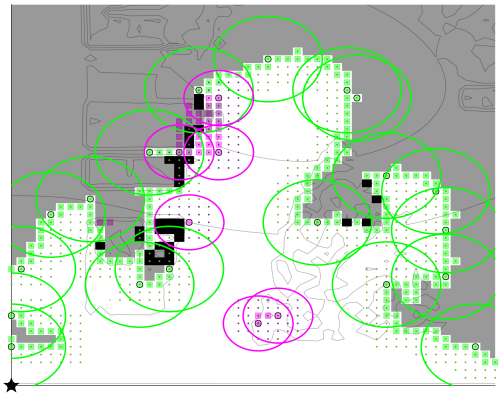
shortest path between the scheduled waypoints, as shown in Figure 5.1d. The resulting path maximizes coverage while minimizing the time and eventually plans to make its way to the area with the most search tasks. This specific selection of waypoints can be attributed to the postprocessing of action values per observation after the first stage of MILP, where observations within overlapping areas have reduced utility contributions. If this were not in place, the bottom two search tasks would be selected, resulting in a redundancy of observations. The trade-off for this approach is the underestimation of singular actions; however, since the overlap is set to one in this case study, the adverse effect of this trade-off generally weighs less. Furthermore, the MCDM fuzzy measures assign higher scores to actions that share a vertex, as both type categories can be performed simultaneously from there. However, in such cases, the search action is almost always prioritized over map actions due to the MCDM weights selected for this case study. As a comparison, the GA output is given in Figure 5.2, where the selection of waypoints is directly performed on the initial feasible action set. Although the general paths and the portion of the map that the robot is steered towards are the same, it is immediately observable that the GA output is much smoother and able to exploit the vertices that contain both good utility search and map actions related to them. The ability to choose any action leads to selecting a smoother final path with fewer redundancies at the expense of increased computational burden and a much higher-dimensional search space.



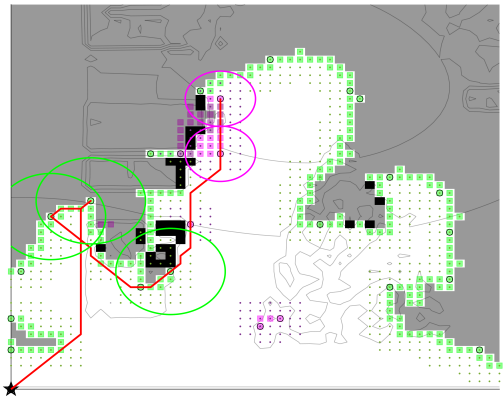
(a) The set of tasks available at a given time (green squares for map, purple squares for search).



(b) The set of feasible actions related to the current tasks, shown by the dots with the corresponding colors for related actions.



(c) Actions and their area of observation selected by the first stage of the MILP controller.



(d) MILP controller output schedule, red line denoting the selected path.

Figure 5.1: Operation steps of the proposed MILP controller, at the beginning of a charging cycle.

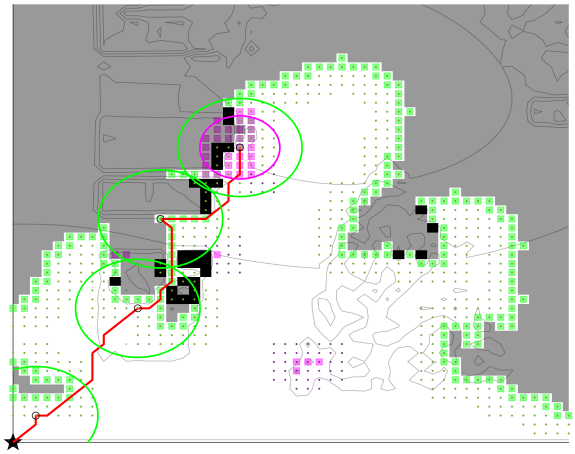


Figure 5.2: Selected optimal schedule of the GA controller for the same problem given in MILP operation.

5.2. Single Robot Experiments

The single robot experiments are performed with a UGV configuration where the two types of mission goal-related actions, `map` and `search` can be performed by the robot. The mapping actions are modeled based on a camera and LiDAR setup; therefore, the line of sight from the height of the robot affects what features and vertices are observable from a given state. On the contrary, the searching tasks have a shorter range; however, they can penetrate through walls and obstacles. Furthermore, measurement errors are not included in this section; however, the modeling of the sensor data reflects the capabilities assigned to the fuzzy inference system of the model to reflect real-life behavior as closely as possible. The UGV description also has a maximum step size that the robot is allowed to take, thus making the traversal through the terrain and the corresponding optimal paths non-trivial in the experiments.

5.2.1. Behavior of the Control Architecture

The behavioral analysis of the control architecture for a single UGV mission is presented first, with a qualitative analysis comparison for GA and MILP-based controllers over the small areas represented by $\mathcal{G}_1$ and $\mathcal{G}_2$. In total, nine charging cycles are presented for each environment per controller, allowing for a meaningful comparison of behaviors at specific intervals across varying environmental factors in the selected areas. Then, quantitative results over multiple runs within the small environments with different controllers are presented to conclude the behavioral analysis. The results of the subarea $\mathcal{G}_1$ for the MILP and GA controllers are given in Figures 5.3 and 5.4 respectively. Likewise, the results of the subarea $\mathcal{G}_2$ is provided in Figures 5.5 and 5.6 with the same order. For all of them, the same legend format is used, where the blue line shows the actual path the robot took at each control step, and the dashed red line shows the path it will take to return to the charger in the next timestep, denoting a single charge cycle per image. The red diamonds denote undetected victims, while the green diamonds denote detected ones.

Qualitative Analysis

Starting with the MILP controller on subarea $\mathcal{G}_1$, at the first charge cycle the robot mostly performs mapping actions around M_1 and discovers the edges of M_4, where it immediately starts searching, bypassing the closer but less prioritized search tasks in the vicinity of M_1. By the second cycle, due to its proximity, the robot is steered towards M_1, where both search and mapping actions are performed effectively to reach and explore the inside of the building. The third cycle marks the final exploitation of the information gathered from the previous cycle, by consecutively performing well-separated search

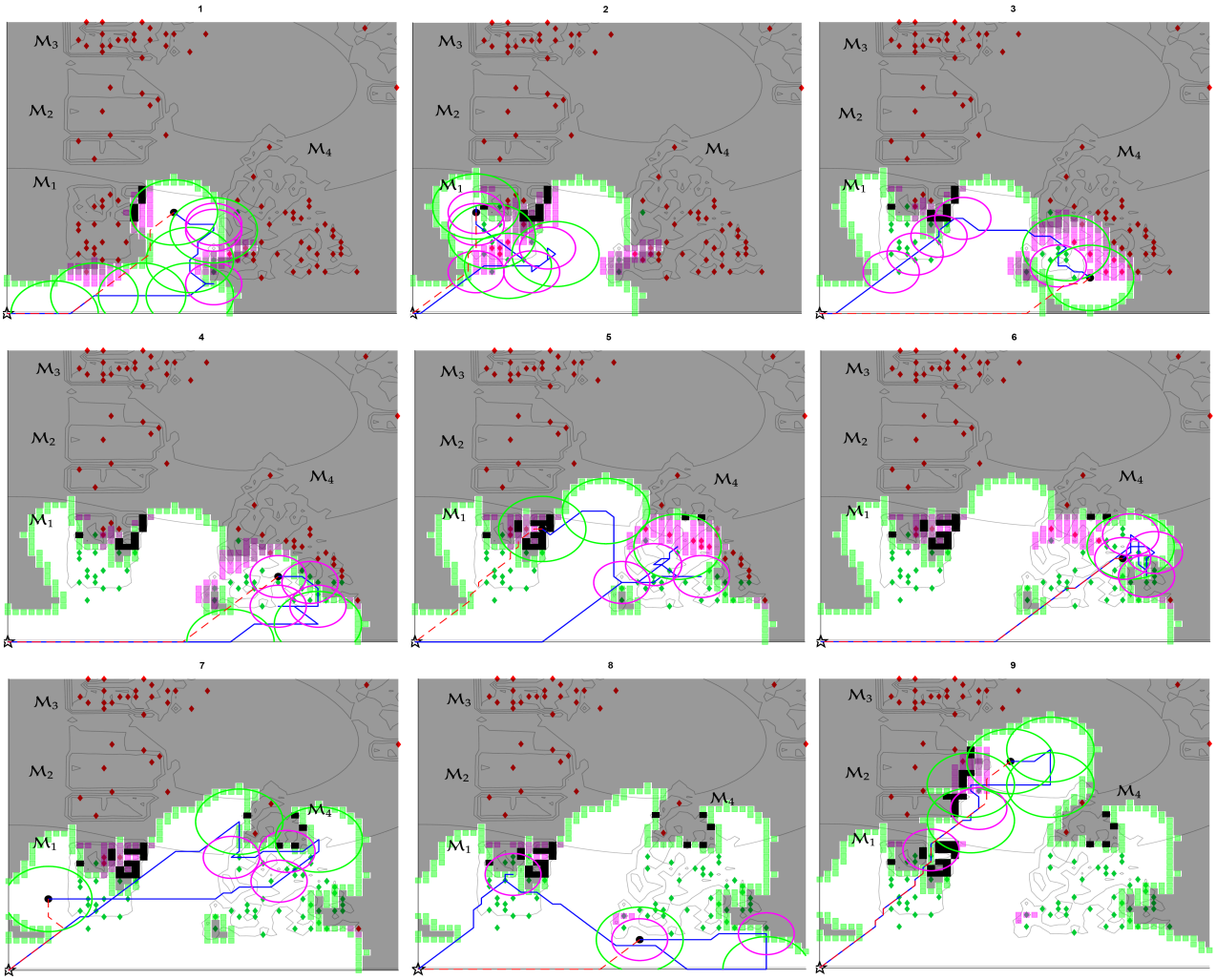


Figure 5.3: Progression per charge cycle of the single robot mission in the $\mathcal{G}_1$ environment with static central agent and MILP controller.

actions to complete the exploration of M_1, and making its way to map M_4. The reason that map actions are taken here instead of better-utility yielding search actions is that the robot has already expended a significant amount of energy on its way to that location, and the only feasible actions remaining are the map actions due to their lower energy cost. The robot approaches M_4 from the shortest path at the fourth cycle due to the higher utility concentration there, and performs the search actions it could not do in the previous cycle. This behavior consists in the fifth, sixth, and seventh cycles where most of M_4 is explored by the end of it. In the fifth and seventh cycles, the effects of the shrinking feasible action set are even more pronounced, as map actions are taken on the way back to the charger toward the end of each cycle. In the eighth cycle, the effect of local minima starts to be observed due to the static placement of the central agent. However, since most of the available actions around M_1 and M_4 are exhausted, the robot finally reaches M_2 and maps a big portion of the unexplored area by the end of the ninth cycle. In comparison, the GA controller starts by mapping a big portion of M_1 and performs multiple search actions without steering away towards M_4, highlighting the ability of the GA controller to exploit local information better than the MILP implementation. Within the first cycle, it is already able to detect the vertical ramps and utilize them quite well. This utilization continues until the end of the fourth cycle, where M_1 is almost fully explored and the gap between M_1 and M_4 is mapped, in contrast to the MILP controller that never discovered that portion of the map. While completing the tasks in the explored area faster than the MILP controller, the GA controller did not enter M_4 until the sixth cycle, by which the MILP controller had already finished the tasks in the lower half of M_4. However, the GA controller takes the cycles six, seven, and eight to fully map and search the lower half M_4,

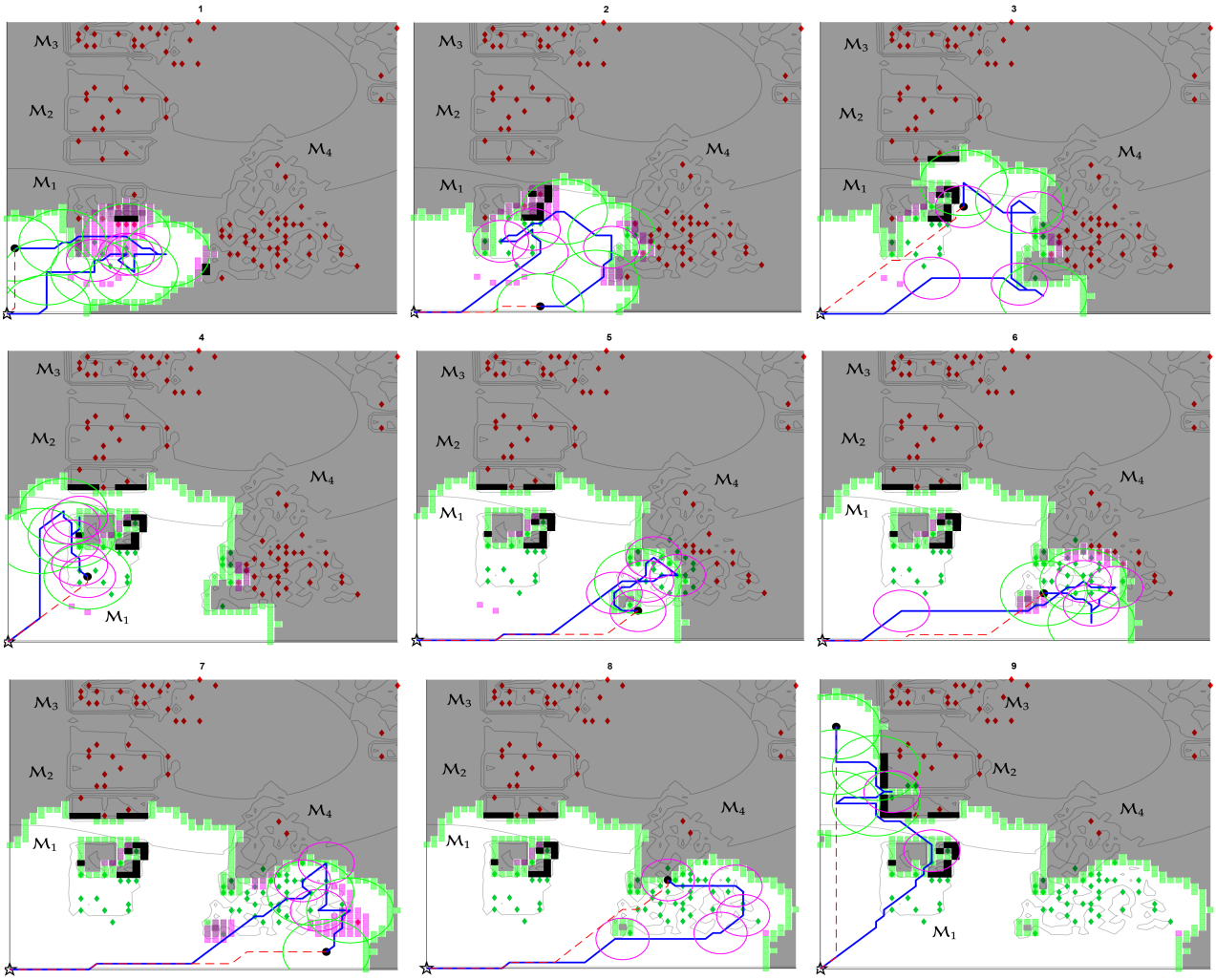


Figure 5.4: Progression per charge cycle of the single robot mission in the $\mathcal{G}_1$ environment with static central agent and GA controller.

where it detects approximately the same number of victims within that area by the end of the eighth cycle, compared to the MILP controller. By then, the mapped area is also similar in size, however, due to the exploration of more hard-to-find corridors earlier on, the GA controller chooses to map the entirety of the left side of the search area instead of steering towards the middle by the end of cycle nine. The main takeaway from the comparisons within the subarea $\mathcal{G}_1$ is that, while the behaviors are quite similar, the GA algorithm appears to be able to explore regions of interest more thoroughly than the MILP controller, before exploring new parts of the map.

The subarea $\mathcal{G}_2$ offers more challenges in terms of the complexity of local areas due to the structure of M_7. This building contains multiple pathways, and the interior sections are only explorable if the specific paths and ramps are quickly and efficiently mapped. Starting with the MILP controller, it is quick to find the closest entrance of M_7 to the central agent and makes its way into the right-hand side section of the building through the vertical ramp. The robot continues to complete this section of M_7 at the end of the second cycle. By the third cycle, a significant portion of the inside of the building is completed; therefore, the MILP controller continues exploring down the passage inside the building to reach and search around the corridor south of M_7, finding another inner passage, and returns to the charger using the same way. Since most of the immediately available tasks are completed within M_7, the robot spends the fourth cycle mapping a large portion of the $\mathcal{G}_1$ environment and searching for victims within M_4 remotely from the outside. At the fifth cycle, the robot continues to perform the same behavior; however, this time it discovers the second pathway within M_7, upon which it spends some

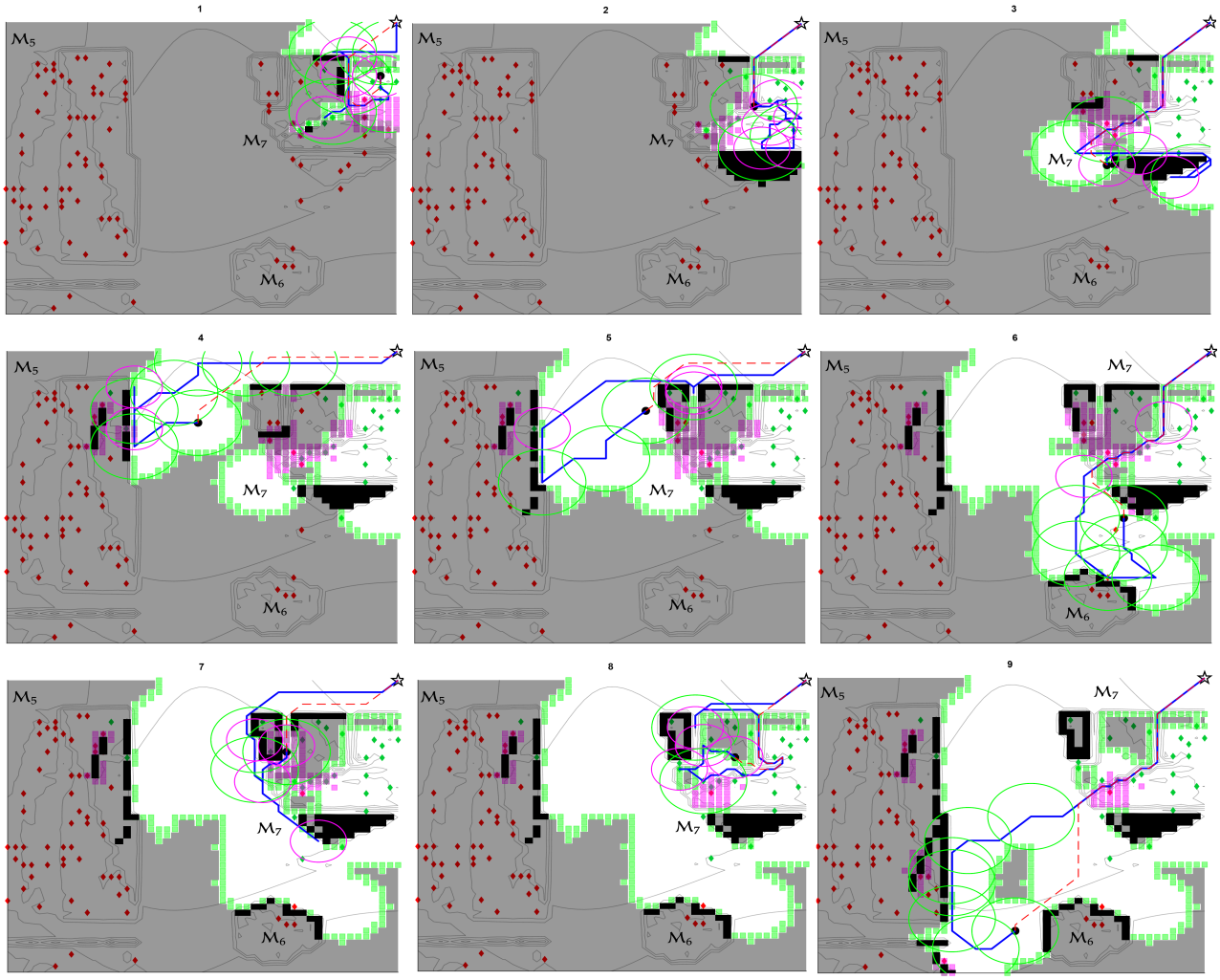


Figure 5.5: Progression per charge cycle of the single robot mission in the $\mathcal{G}_2$ environment with static central agent and MILP controller.

redundant search actions, since the discovery of a new pathway spawns additional search tasks in the same area. In the sixth cycle, the MILP controller steers the robot to the first passage and efficiently maps the bottom left of the $\mathcal{G}_2$ area, reaching M_6 already. Although it was discovered in the fifth cycle, the robot opts not to continue down the second passage it had already found and instead chooses a path that leads to a larger area, yielding a greater sum of utility. Cycles seven and eight are spent on mapping and searching the left-hand side of M_7 through the recently discovered pathway. Finally, the robot completes most of M_7 at the end of the ninth cycle and spends the remaining time mapping the rest of $\mathcal{G}_2$.

Another observation from this experiment is that once a continuous visual obstruction is reached, similar to the case in the final cycle, the controllers take smaller steps along the wall to potentially find a line of sight that would enable the measurement of the map portion on the other side. Ultimately, this behavior facilitates the efficient discovery of various pathways throughout the experiments. In comparison, the GA controller starts its first cycle similarly to the MILP controller. At the end of the second cycle, the robot has already completed the right-hand side of M_7 and has found the entrance to the second passage. However, the exploration of multiple entry points early on makes the controller spend extra time going back and forth between them, as observed on cycles three and four, leading to inefficient behavior compared to the MILP controller. Although it is established that the GA controller takes advantage of the local information better and explores the area more thoroughly, this behavior creates an adverse effect in this case due to the complexity of M_7 and the layout of the inner passages.

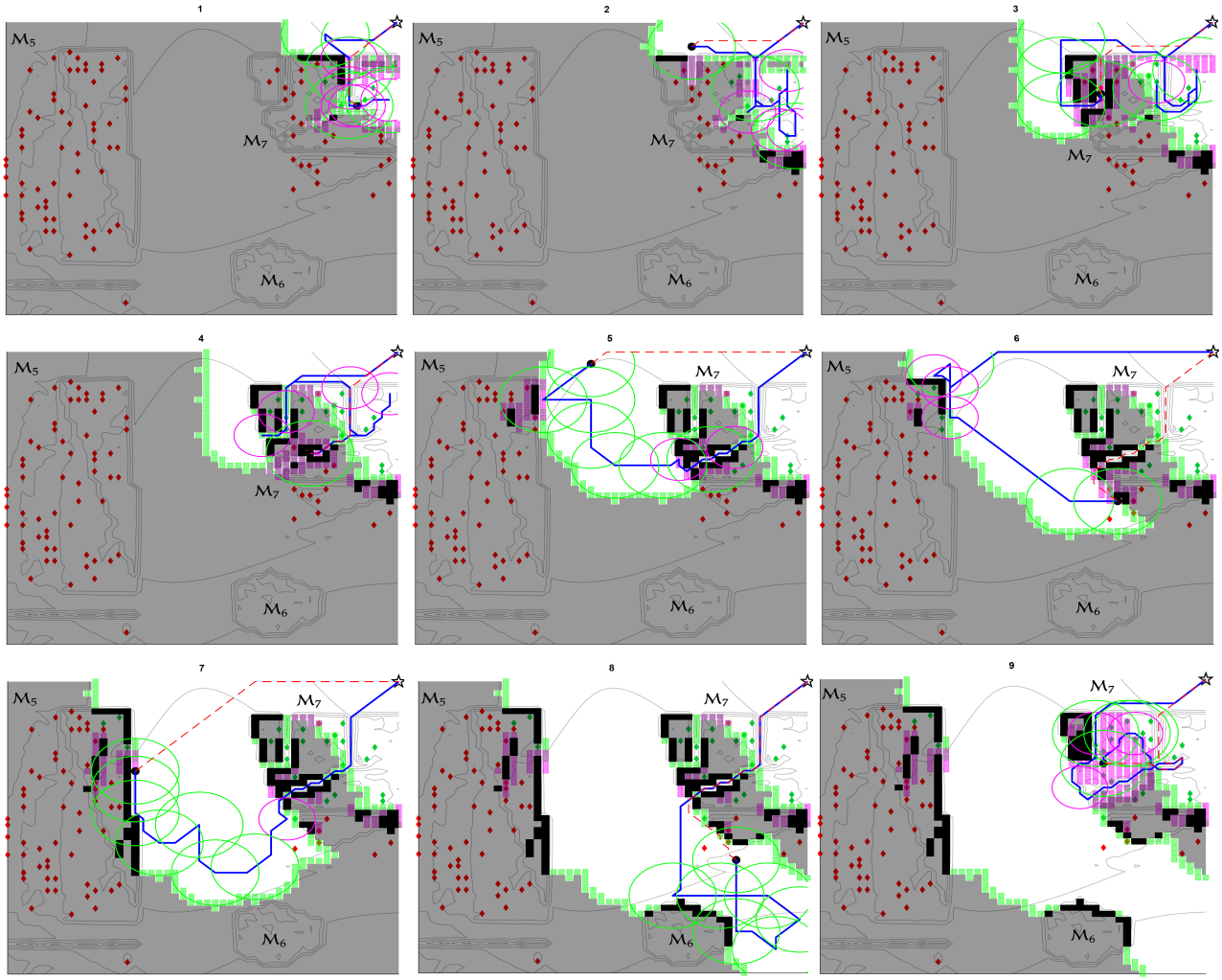


Figure 5.6: Progression per charge cycle of the single robot mission in the $\mathcal{G}_2$ environment with static central agent and GA controller.

This behavior enables the GA controller to find the victims within M_7, more specifically, the ones that are close to the left-hand side of the building faster, at the expense of mapping considerably less area than the MILP controller by the end of cycle six. Still, having completed exploring most of M_7, the GA controller is quick to compensate for the reduced mapping of the area at cycle seven, by systematically exploring all of the middle part of $\mathcal{G}_2$. At the eighth cycle, both controllers had achieved similar results in terms of victim discovery and mapped area, albeit at different time rates. During the ninth cycle, the GA controller steers the robot back to the inside of M_7, and performs redundant actions on the areas that had already been searched due to the local minima induced by the static central agent. Comparing the GA and MILP controller behaviors in $\mathcal{G}_2$ area, the established results from the previous analysis are reinforced. However, the local strength of the GA controller led to a more detrimental behavior than the MILP controller, which, in this case, can be said to have performed better, albeit with a small margin.

Quantitative Analysis

The results for all controllers in terms of mapped area and detected victim count percentages for the smaller experiments done in $\mathcal{G}_1$ and $\mathcal{G}_2$ are summarized in Table 5.1. Note that although the first nine charge cycles are analyzed qualitatively, these results are obtained at the end of four simulation hours, by which time most controllers had charged more than nine times. As observed from the behavioral analysis, the GA and MILP controllers yield similar results, with the MILP controller achieving a higher victim count percentage in the $\mathcal{G}_1$ environment. This is likely due to the exploration towards the middle

part of the map, rather than the left side, at the end of the final discussed charge cycle. For the $\mathcal{G}_2$ area, the results are the same. Compared to the other methods, the MCDM controller achieves the closest performance to the MPC-based controllers. This is an expected result, as GA and MILP controllers employ a receding horizon approach to make more informed decisions, rather than greedily selecting the best utility in a one-shot manner. However, the stark difference between the performances supports the notion that this specific problem formulation can not be solved optimally by always taking the next best action.

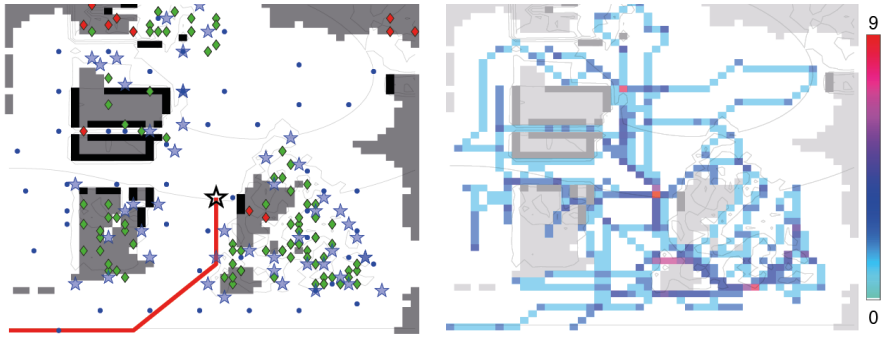
		GA	MILP	hybrid	MCDM	shortest-time	fuzzy	random
$\mathcal{G}_1$	victim count (%)	73	81	79	51	12	44	38
	mapped area (%)	66.8	65.8	66.1	49.8	26.3	87.8	44.8
	victim health (%)	68.2	67.7	67.5	72.3	55.8	41.2	45.6
	distance (km)	6.3	5.7	5.8	4.9	1.6	11.2	8.1
$\mathcal{G}_2$	victim count (%)	38	38	38	27	21	24	18
	mapped area (%)	69.9	70.0	69.8	54.7	16.7	73.4	44.6
	victim health (%)	74.5	75.1	75.0	74.1	68.4	56.2	48.2
	distance (km)	6.6	5.9	5.7	5.3	1.9	13.0	8.3

Table 5.1: Key results for each main mission metric related to $\mathcal{G}_1$ and $\mathcal{G}_2$ experiments at the end of four simulation hours.

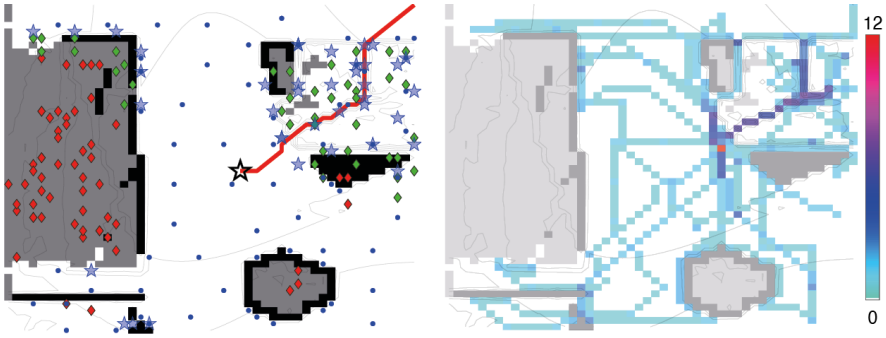
Judging by the results presented in Table 5.1, the proposed MILP method consistently outperforms all baselines by achieving the best balance between victim count and mapped area across both scenarios, although for individual metrics, other controllers may perform better. It reaches the highest victim count (81% and 38%) and maintains high mapping performance (65.8% and 70.0%) while keeping travel distance relatively low and victim health at strong levels. Compared to alternatives, it improves victim count by up to 113% and mapped area by over 46% in critical cases. GA and hybrid methods follow closely but offer slightly lower results, sacrificing optimality for general balance. MCDM achieves decent mapping and health levels but lags behind in victim discovery due to its focus on coverage. Within the $\mathcal{G}_1$ environment, the MCDM method achieved a higher victim health percentage; however, with a lower number of victims, this information suggests that only prioritizing the more immediate search to the initial state is preferred by this controller. Shortest-time minimizes distance drastically but performs poorly in all other metrics. Overall, MILP achieves the most favorable trade-off between discovering victims and mapping effectively, while controlling distance and maintaining acceptable victim health for the small area comparisons.

To contextualize this observation, additional plots are provided in Figures 5.8 and 5.9 for all controllers. The first plot for each environment in Figure 5.8 marks where the search and map actions had taken place, where search is denoted by a star and map is denoted by a dot. The second plot for each environment shows which vertices are visited and how many times they are visited throughout the mission. When compared to GA and MILP, the MCDM controller appears to be stuck in local minima and has taken multiple search actions in very close proximity redundantly. This, in turn, leads to poor time optimization in general, as search tasks are time- and energy-intensive. The second set of plots in Figure 5.9 shows the fuzzy system evaluation of the actions taken at each control step in the first column, and the MCDM evaluation of the actions in the second column. The variance in the MCDM evaluation of the MCDM controller is lower than that of GA and MILP controllers, while the fuzzy evaluations are nearly identical. This further supports the idea that the MCDM controller consistently gets stuck in local minima around the buildings. Compared to the other controllers, the shortest-time approach significantly performs worse across the board. Investigating the heatmap patterns in Figure 5.8c , it behaves as an exhaustive controller where the sensors are always on whenever new information can be gathered. This substantiates the approach of modeling the time and energy expenditure of actions in a general sense, as most works in the literature do not consider this aspect. Due to the dense nature

of the tasks in this case study, the detrimental effect of this approach is even more pronounced. The action evaluation plots in Figure 5.9c consistently show low values, validating the correct implementation of the deteriorating quality and quantity of observations when the robot repeatedly performs similar actions in close proximity. In contrast, the fuzzy controller achieves the best mapping percentage for both cases across all controllers, while simultaneously detecting the least amount of victims. This is a direct result of how the fuzzy evaluation system combines the task priority information with the action evaluation. Since search tasks can only spawn from good-quality mapping actions, the prioritization of map actions substantially increases compared to search actions without any MCDM weights. Additionally, while task prioritization approaches one when very specific and rare circumstances occur (i.e., the building is severely damaged and the population density is extremely high), the mapping task prioritization is only tied to the number of unexplored vertices in close proximity to a frontier location. For these reasons, the fuzzy controller almost always selects the next best mapping action. The MPC-based controllers outperform all other methods convincingly, as laid out in the Table 5.1.



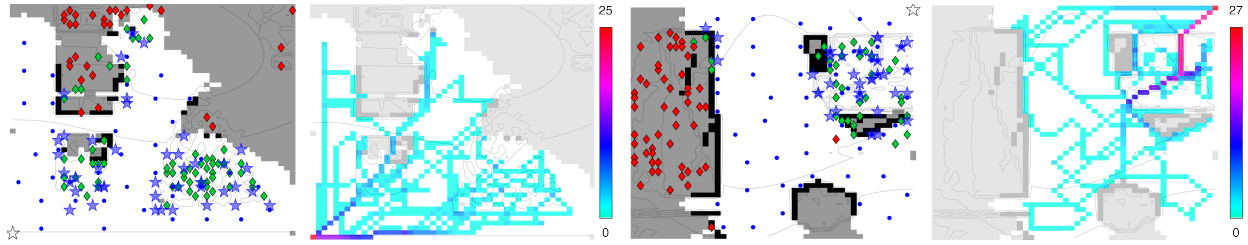
(a) MILP controller with dynamic central agent in $\mathcal{G}_1$ environment. (left: actions, right: traversal)



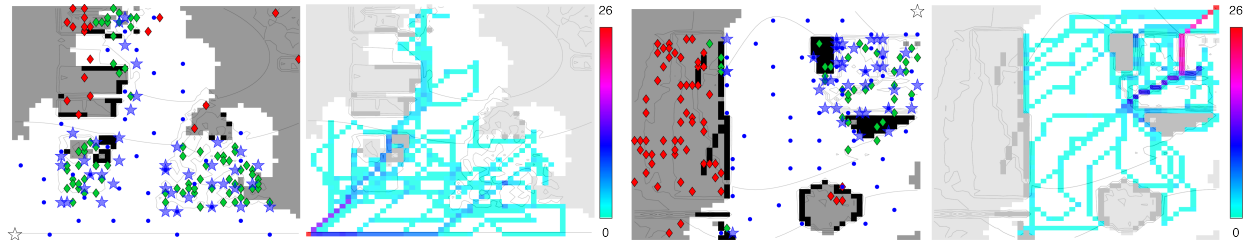
(b) MILP controller with dynamic central agent in $\mathcal{G}_2$ environment. (left: actions, right: traversal)

Figure 5.7: Action and traversal heatmaps of the MILP controller with dynamic central agent.

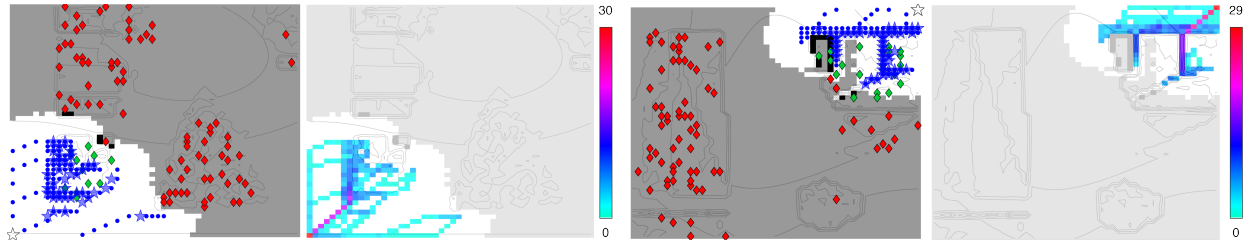
Concluding the small area analysis, Figure 5.7 show the final state of the USaR mission with the central agent controller activated at the end of four simulation hours. The red line marks the path the central agent took through the mission, where it slowly converges to the center of both areas as time progresses, to stay close to all unattempted available tasks that remain. The movement heatmaps demonstrate a much more balanced distribution of the robot in the USaR area, where the number of maximum visits to a vertex is reduced by three times compared to the static cases. Furthermore, the total mapped area percentage for $\mathcal{G}_1$ and $\mathcal{G}_2$ becomes 85.2% and 79.8% respectively for the MILP controller when the dynamic central agent is enabled. Likewise, the percentage of detected victims are 88% and 41% respectively. Compared with the static case, the number of victims is close enough, which may indicate that the saturation time has been reached for these small environments to detect the victims within areas M_1, M_4 and M_7 that are closest to the initial starting point. The close to 15% increase in the mapped area percentage can be attributed to the central agent moving closer to the unexplored areas over time, enabling the robots to map them without being constrained by the energy and time accumulation present in the static case.



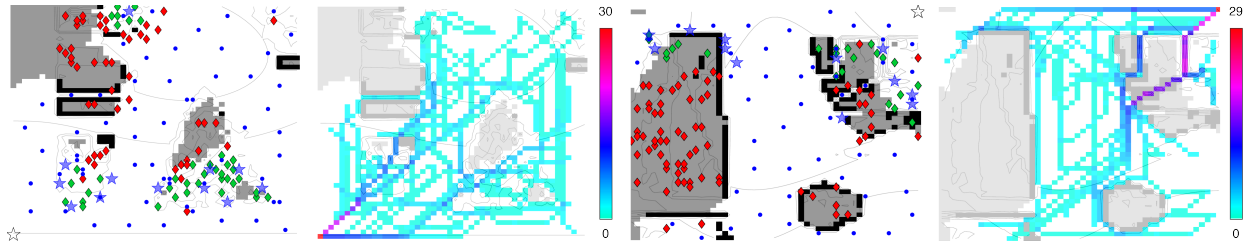
(a) GA controller (left to right: $\mathcal{G}_1$ actions, $\mathcal{G}_1$ heatmap, $\mathcal{G}_2$ actions, $\mathcal{G}_2$ heatmap)



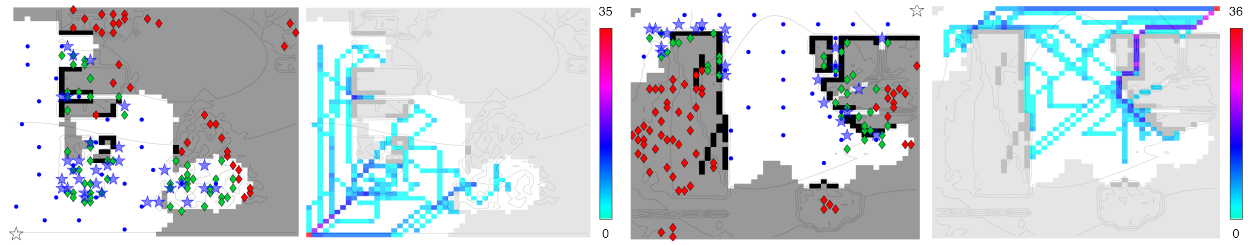
(b) MILP controller (left to right: $\mathcal{G}_1$ actions, $\mathcal{G}_1$ heatmap, $\mathcal{G}_2$ actions, $\mathcal{G}_2$ heatmap)



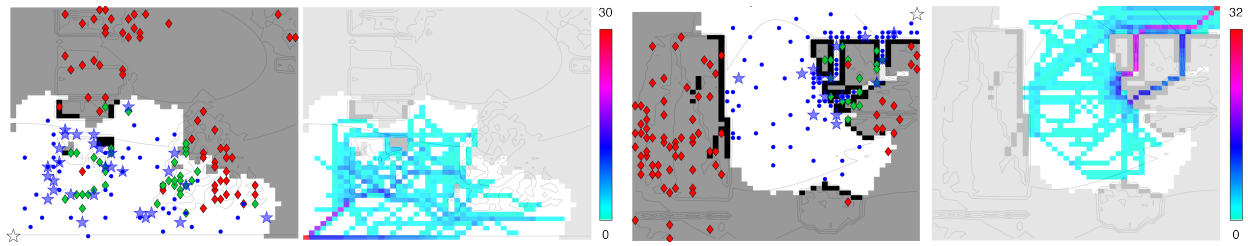
(c) shortest-time controller (left to right: $\mathcal{G}_1$ actions, $\mathcal{G}_1$ heatmap, $\mathcal{G}_2$ actions, $\mathcal{G}_2$ heatmap)



(d) fuzzy controller (left to right: $\mathcal{G}_1$ actions, $\mathcal{G}_1$ heatmap, $\mathcal{G}_2$ actions, $\mathcal{G}_2$ heatmap)

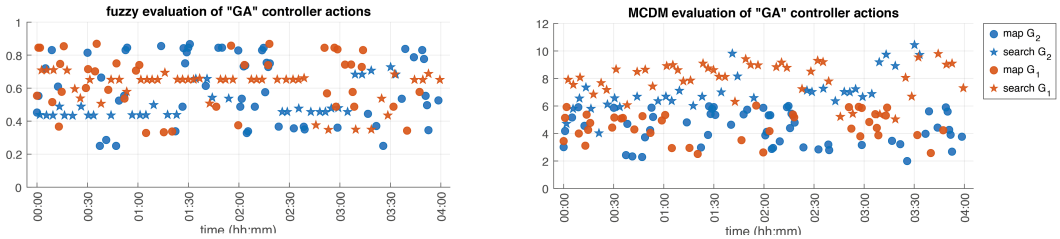


(e) MCDM controller (left to right: $\mathcal{G}_1$ actions, $\mathcal{G}_1$ heatmap, $\mathcal{G}_2$ actions, $\mathcal{G}_2$ heatmap)

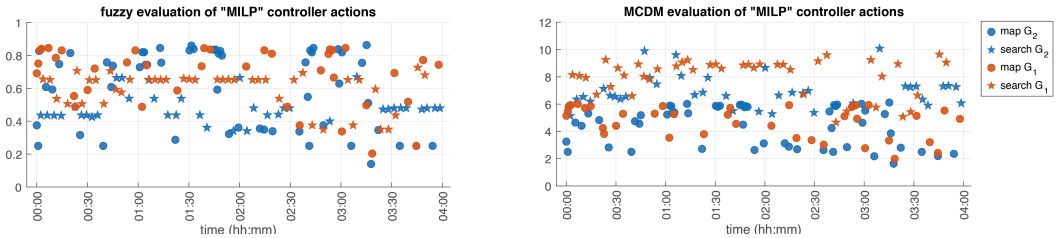


(f) random controller (left to right: $\mathcal{G}_1$ actions, $\mathcal{G}_1$ heatmap, $\mathcal{G}_2$ actions, $\mathcal{G}_2$ heatmap)

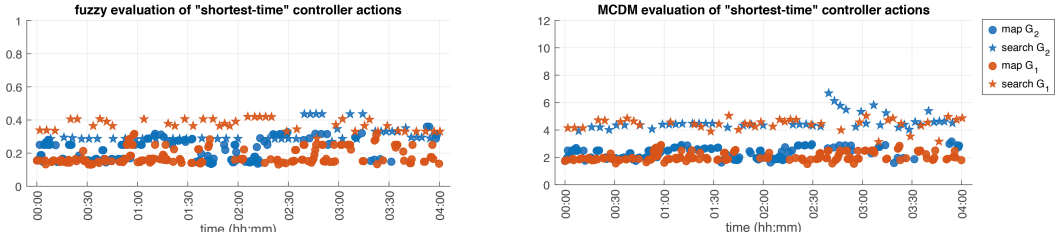
Figure 5.8: Action and movement heatmaps for the experiments in $\mathcal{G}_1$ and $\mathcal{G}_2$ at the end of four simulation hours.



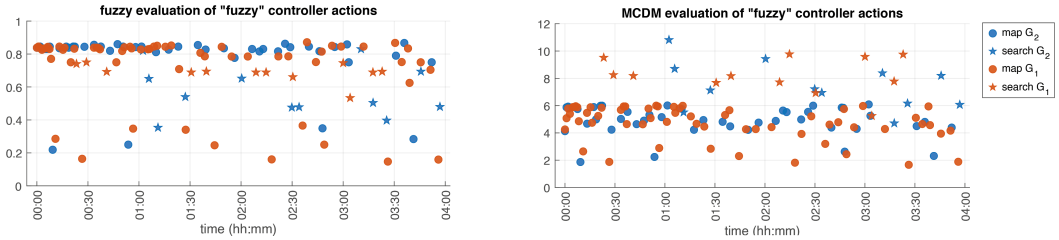
(a) Control action utilities of the GA controller.



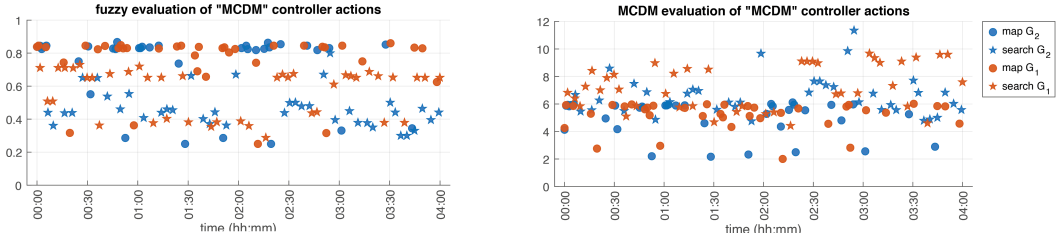
(b) Control action utilities of the MILP controller.



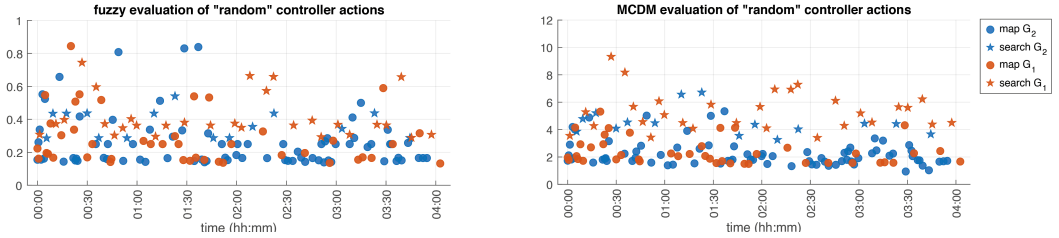
(c) Control action utilities of the shortest-time controller.



(d) Control action utilities of the fuzzy controller.



(e) Control action utilities of the MCDM controller.



(f) Control action utilities of the random controller.

Figure 5.9: Control action utilities through $\mathcal{G}_1$ and $\mathcal{G}_2$ environment experiments.

5.2.2. Analysis of the Control Architecture in Structured Experiments

After presenting the detailed behavioral analysis of the control architecture within smaller and distinct environments, to draw more meaningful and generalizable results, longer experiments were conducted over a larger area with 17 different initial points and randomized victim placement, with multiple repetitions. The results of these experiments are presented in this section.

		GA	MILP	hybrid	MCDM	shortest-time	fuzzy	random
static	mapped area (%)	44.5	41.7	41.6	26.9	6.3	48.0	17.2
	victim count (%)	45.4	41.1	41.2	24.4	3.5	14.5	11.4
	victim health (%)	71.1	73.4	71.5	51.6	48.5	47.7	47.2
	total distance (km)	18.1	16.5	16.9	12.9	4.6	28.0	22.3
dynamic	mapped area (%)	62.3	60.2	57.0	35.4	6.5	55.7	16.9
	victim count (%)	57.3	59.3	56.0	35.0	6.7	23.9	19.9
	victim health (%)	71.2	76.0	74.1	57.2	44.1	41.6	46.5
	total distance (km)	12.1	10.4	10.7	8.9	3.5	19.3	15.8

Table 5.2: Outputs of USaR goals for single robot experiments in $\mathcal{G}$ at the end of ten simulation hours.

In Table 5.2, the results of the USaR mission metrics at the end of ten simulation hours are summarized. Starting with the mapped area percentage, the overall progression of the averaged area size through the USaR mission is given in Figure 5.10. Similar to the local experiment results, the fuzzy controller maintains the largest mapped area percentage for the static central agent case, ending at 48% of the USaR area. This value is likely to be the maximum that can be reached without moving the central agent with the selected robot parameters. It is closely followed by GA, MILP and hybrid controllers ranging from 44% to 41%, where they approximately double the amount of MCDM controller coverage. Only the shortest-time controller falls below the random controller, reaching a maximum of 6.3% area coverage. When the dynamic central agent is introduced; GA, MILP, hybrid, and MCDM controllers almost double their mapped area percentages. Interestingly, moving away from the local minima induces all three of the proposed controllers in this thesis to catch up the mapping performance of the fuzzy controller around halfway through the mission time, and surpass it by the end of the simulations. This is likely due to the efficient exploration of these controllers when introduced to a new environment and the effect of the dynamic central agent reducing the amount of redundant search actions in general. Continuing with the averaged number of detected victims given in Figure 5.11, there is a clear separation between the GA, MILP and hybrid controllers and the rest. The closest result following them comes from the MCDM controller, where similar to the mapped area percentages, is almost doubled by the proposed solutions. Including the dynamic central agent, the number of detected victims is approximately increased by 25% for all controllers. Once again, the shortest-time controller performs the worst for this metric, while the fuzzy controller closely follows the random controller. This further establishes that the fuzzy controller can not employ a meaningful victim search strategy the way the inference system is designed in this thesis. However, when combined with the MCDM method for effective prioritization of mission goals with respect to elapsed time, the results dramatically increase, supporting the optimality claims of the proposed architecture. The covered distance metric given in Figure 5.12 indicates the efficiency of taken actions, where it is expected to be lower if the controllers achieve better local exploitation. Indeed, after the fuzzy controller which is able to travel further due to mostly choosing map actions, therefore spending less time per action in general, the random controller results in the most distance covered out of all remaining approaches. In terms of these USaR mission metrics, it can be said that the controllers with overall higher area coverage and victim detection with lower covered distances are the best suited for a USaR task. This is indeed the case for GA, MILP and hybrid controllers.

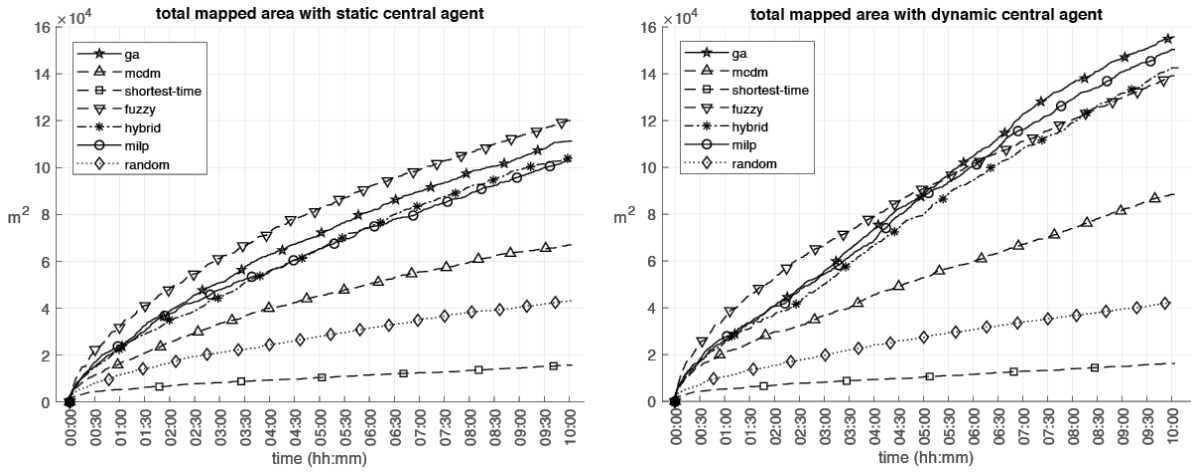


Figure 5.10: Total mapped area progression of single robot experiments.

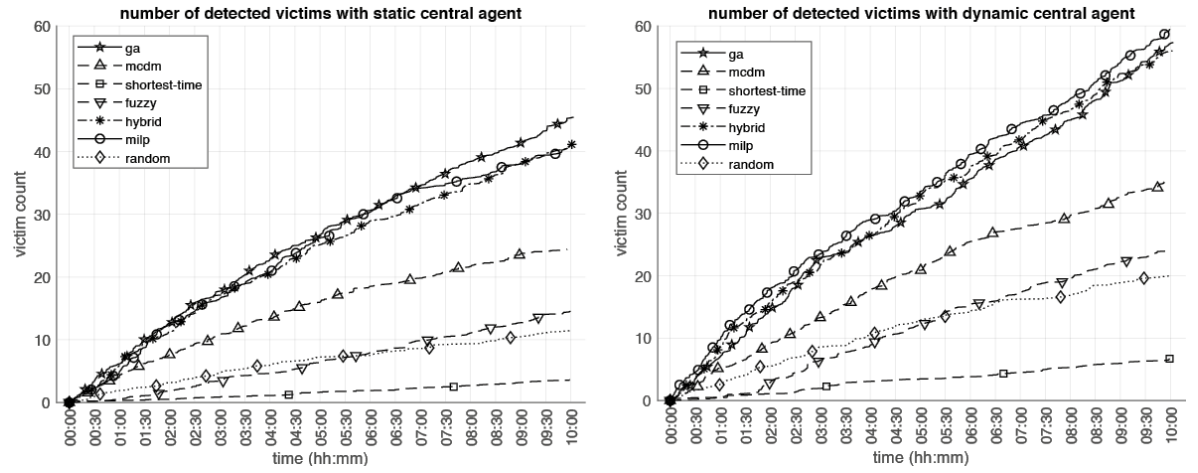


Figure 5.11: Detected victim count progression of single robot experiments.

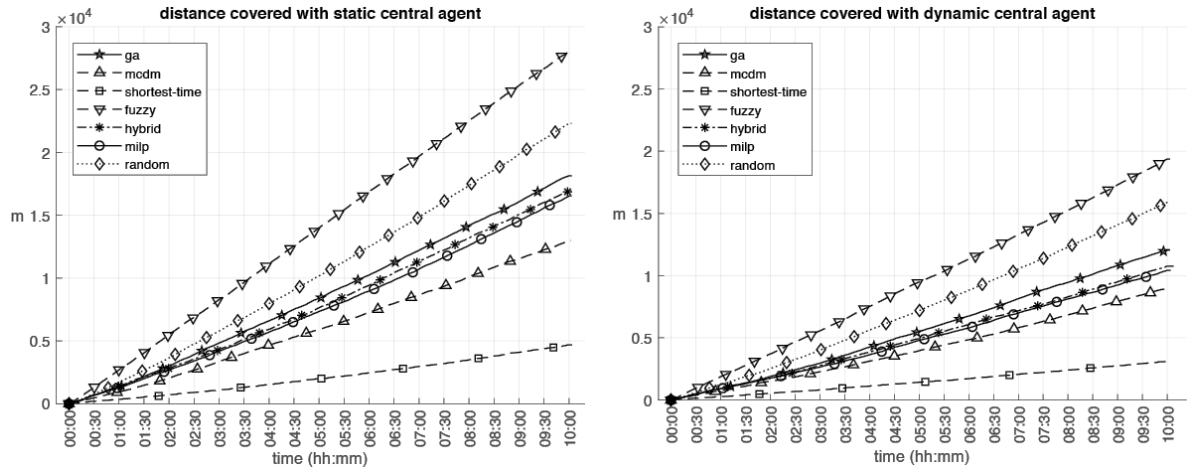


Figure 5.12: Covered distance progression of single robot experiments.

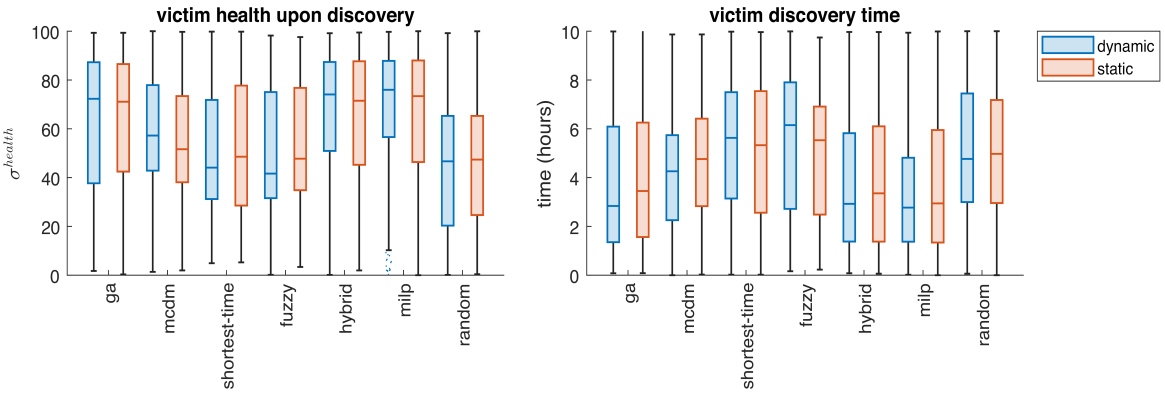


Figure 5.13: Distribution of victim status for single robot experiments at the end of ten simulation hours.

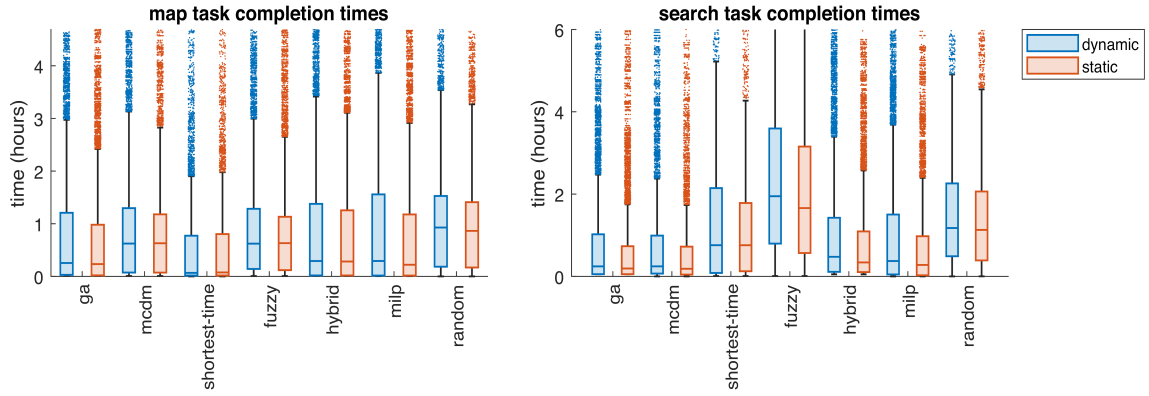


Figure 5.14: Time of task completion after spawn for single robot experiments at the end of ten simulation hours.

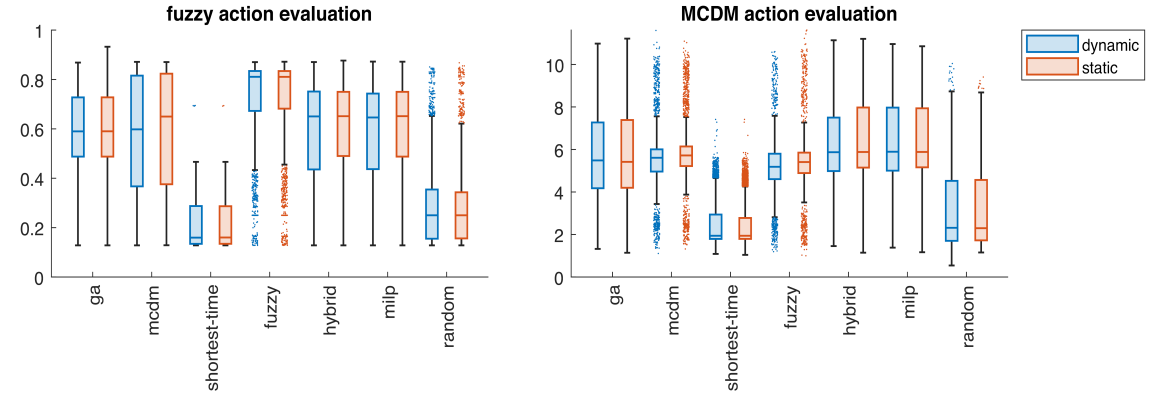


Figure 5.15: Distribution of control action utility for single robot experiments at the end of ten simulation hours.

The boxplots in Figures 5.13, 5.14, and 5.15 provide detailed insight into the distribution of various key metrics for both dynamic and static environments. The Table 5.3 further summarizes these findings by combining the dynamic and static central agent scenarios in terms of the median placement and the inter-quartile range, wherever medians do not sufficiently represent the differences between the data. From this table, it is clear that when the best-performing methods per metric are investigated, the MILP method achieves the best results for the direct mission goals. While the fuzzy controller achieves the highest fuzzy utility, the MILP controller achieves the highest MCDM utility with an inter-quartile range 2.8 times larger than that of the pure-MCDM controller. Still, the fact that the MILP controller achieves better victim health times and counts highlights the strengths of the proposed architecture in balancing trade-offs while optimizing behavior with respect to mission goals, as provided as preferences between decision criteria.

		GA	MILP	hybrid	MCDM	shortest-time	fuzzy	random
victim health (%)	median	71.2	74.7	72.8	54.4	46.2	44.6	46.9
victim discovery time (h)	median	3.1	2.8	3.1	4.5	5.4	5.8	4.8
map task completion time (h)	median	0.24	0.26	0.29	0.63	0.08	0.63	0.89
search task completion time (h)	median	0.25	0.31	0.37	0.21	0.76	1.80	1.15
fuzzy evaluation of performed actions	median	0.60	0.65	0.65	0.62	0.16	0.81	0.25
	IQR	0.22	0.29	0.34	0.45	0.15	0.16	0.19
MCDM evaluation of performed actions	median	5.5	5.9	5.9	5.7	1.9	5.3	2.3
	IQR	3.2	2.8	2.8	1.0	1.1	2.9	2.8
runtime (s)	median	28.8	12.8	20.3	-	-	-	-
	IQR	27.2	12.7	17.9	-	-	-	-

Table 5.3: Summary of key results from the boxplots provided for the control architecture evaluation (IQR = Inter-Quartile Range).

For the victim health upon discovery, the Figure 5.13 shows the distribution of the victim health percentages at the end of the simulation times for each controller. As expected, these values closely mirror the victim discovery time, which are inversely proportional to each other. Overall, the GA, MILP, and hybrid controllers achieve over 70% health rate upon victim discovery. The performance of the other controllers in this metric is also relatively high, between the 51.6% achieved with the MCDM controller and 47.2% of the random controller. However, these controllers also find significantly fewer victims; therefore, the performance of the proposed controllers becomes even more highlighted when it is considered that they discover twice as many victims as the best-performing other methodology. The task completion time distribution for both types of tasks is given in Figure 5.14. This metric is interesting in that it provides insight into how effectively the new information is utilized by the controllers. In general, most search tasks should be undertaken as fast as possible, while minimizing the map task completion equates to faster exploration of the important areas, as well as possibly generating new search tasks that may lead to more victim discovery. The GA and hybrid controllers excel in this metric, with a median completion time of 15 minutes for both map and search task completion. The MILP controller results in a slightly higher median by 5 minutes for map tasks and 8 minutes for search tasks. As observed previously, the shortest time controller executes map tasks almost as soon as they spawn, while the fuzzy controller mostly ignores the search tasks until the local task set starts consisting mostly of search

tasks, leading to the dramatic increase observed in their median. Interestingly, while all distributions experience a slight increase with the dynamic central controller, the fuzzy and shortest-time controllers experience a slightly worse performance.

The action evaluation distributions in terms of the fuzzy system output and MCDM output per control action are given in Figure 5.15. This metric is primarily included to determine if the controllers function as intended. While the GA, MCDM, hybrid and MILP controllers output approximately similar medians for both evaluation types, the MCDM controller results in a higher variance for fuzzy action evaluation and a much smaller variance for the MCDM action evaluation, which further establishes its optimality for the local minima but not for the global problem in general. Likewise, the fuzzy controller maximizes the fuzzy action evaluation but results in a slightly lower MCDM evaluation, with a considerably small variation. These observations lead to the conclusion that inclusion of both these approaches into the proposed solutions lead to the exploration and time minimization trade-off necessary for a successful USaR mission.

Finally, the runtime distribution is given in Figure 5.16. Since all three of the controllers behave similarly across all metrics, the runtime with respect to the scale of the problem at hand is the final arbitrator of which controller should be selected in practice. The GA controller does not have a time limit like the MILP controller, only a sufficiently large amount of maximum and stall generations set for achieving solutions that are close enough to the global optimum. The median for the GA optimization runtime is 28 seconds, while it falls down to 20 seconds for the hybrid controller. This improvement can be attributed to the task selection implementation of the first-stage MILP controller, which significantly reduces the dimension of the search space. Furthermore, the hybrid controller directly uses GA optimization after the task set selection, using the same hyper-parameters. The runtime of the hybrid may be improved with further tuning; however, for smaller prediction horizons similar to this case study, the MILP controller is significantly more advantageous. The median for the MILP controller is at around 12 seconds, meaning most of the time the incumbent partial solution found in the first 10 seconds is efficiently improved by the LNS heuristic afterwards. Additionally, all three controllers find the optimal solution faster when the robot has reduced energy, since the safe return constraints consistently disable the consideration of high-cost tasks along the prediction horizon. Having said that, the GA based controllers do not have a direct way of limiting the infeasible sequential selections and can only check for them after evaluating the entire population of each generation. Conversely, the MILP controller has built-in constraints to eliminate the branches in the search space that lead to such infeasibilities, and with sufficiently tight bounds on the decision variables, the branch-and-cut approach of the linear solvers converges to a good solution fast enough within the given time budget.

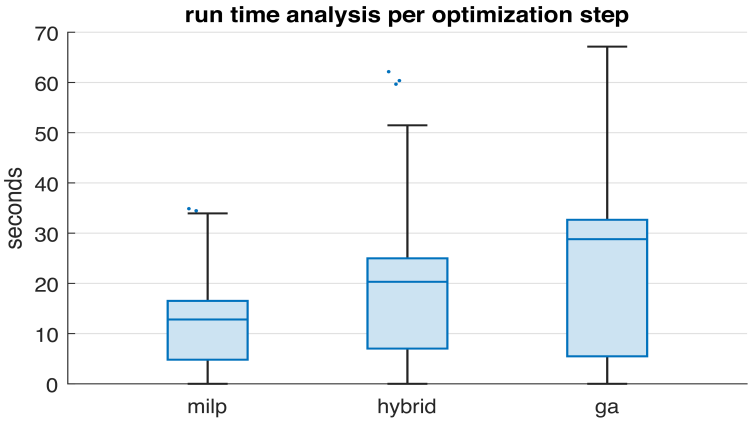


Figure 5.16: Distribution of runtime per optimization step.

5.3. Multi-Robot Experiments

The multi-robot experiments are conducted with two different teams of robots, where the first team consists of a heterogeneous composition comprising a UAV and a crawler-type robot, whereas the second team consists of a homogeneous composition with two UGVs with identical parameters. Furthermore, errors in action observations are introduced to the robotic systems for this part, to establish a clearer distinction between robot actions and decision-making prioritization with respect to the robot's capabilities for undertaking different task allocations. Only the proposed MILP controller is presented in this section, as it has already been established to be the best-performing controller. Therefore, the main focus of the multi-robot experiments is to benchmark the cooperation behavior maintained by the global coordination controller. Similar to the single-robot results, a behavioral analysis of robot coordination is presented, comparing it to task allocations made without any coordination within the small subsets of the defined simulation environment. Then, a thorough analysis of the main metrics established by the coordination is presented, based on the results of generalized experiments.

5.3.1. Behavior of the Global Coordination Controller

The behavioral analysis of the global coordination controller is performed within the areas $\mathcal{G}_1$ and $\mathcal{G}_2$ using a heterogeneous team consisting of a UAV and a crawler-type robot. The Figures 5.17 and 5.18 contain various cooperation optimization results throughout missions within $\mathcal{G}_1$ and $\mathcal{G}_2$ using the UAV-crawler team, respectively. While the legend for task, victim, and terrain visualization is the same as the single robot plots, here, the UAV is represented by the blue lines and the crawler is represented by the red lines. For each decision on their respective paths, the dashed circles represent mapping actions, and the continuous circles represent search actions.

Qualitative Analysis

Starting with the mission in $\mathcal{G}_1$, the Figure 5.17a shows that both robots picked the exact same paths where all actions are conflicting due to the limited size of the task set. The coordination strategy addresses this by allocating the tasks with better utility mapping to the UAV actions, while steering the crawler toward the less important tasks. Furthermore, since the UAV is twice as fast as the crawler, it selects more back-to-back map actions than the less capable crawler to be executed simultaneously. Figure 5.17b shows the mission state where the M_4 area is being explored, and the UAV has just finished charging at the central agent location. The M_4 area is densely populated with both mapping and search tasks. Therefore, the local controllers for both robots planned their routes to achieve the most efficient run, mapping and searching the area simultaneously. Since the UAV is more capable for mapping and the crawler is for searching, the coordination optimization allocated the search actions that cover more of the search tasks to the crawler, while the UAV retained the remaining non-conflicting actions. While both robots chose the mapping actions on their first waypoint due to their proximity and shorter execution time, the overall paths they take for the globally optimized allocation are also shortened, showing the capability of the global controller approach to distribute the robots in the USaR area better than the direct local controller outputs put together. Aligning with this, the UAV still attempts search tasks immediately after mapping, as it is better for the USAR goals that the robots do not always wait for each other to finish their immediate tasks. As shown in Figure 5.17c, all search tasks are allocated to the crawler, while all mapping tasks are allocated to the UAV, resulting from global coordination. Simultaneously, the shortest execution-time paths for the new allocation of local controller outputs are solved. Finally, the Figure 5.17d demonstrates a new scenario, where almost all of the tasks within the $\mathcal{G}_1$ environment had already been finished, thus both local controllers attempt to solve the remaining tasks at the same time. Since the UAV is better suited for mapping tasks, all mapping actions are allocated to it. Since the crawler can not be given any other action without violating the conflict constraints, it plans a new path by discarding all possible conflicting actions with the globally allocated UAV actions. The only remaining option is the single search task left in the environment, to which the crawler is directly steered at the end of its new local planning. Note that while this search task does not achieve anything and should ideally be removed from the task set, it still provides a good example of the global coordination optimization behavior.

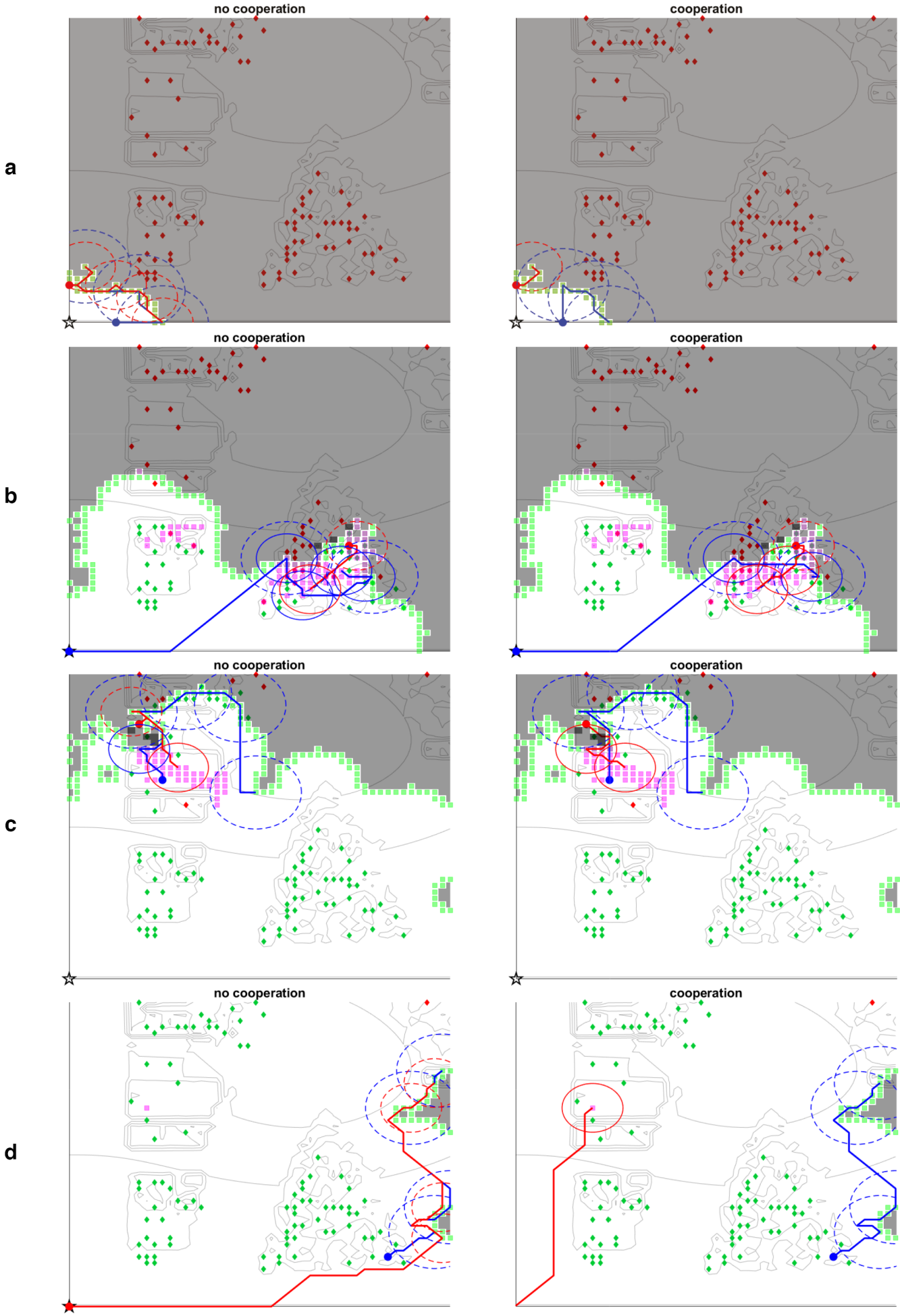


Figure 5.17: Comparison of global cooperation controller outputs at different stages of $\mathcal{G}_1$ exploration with the UAV and crawler team.

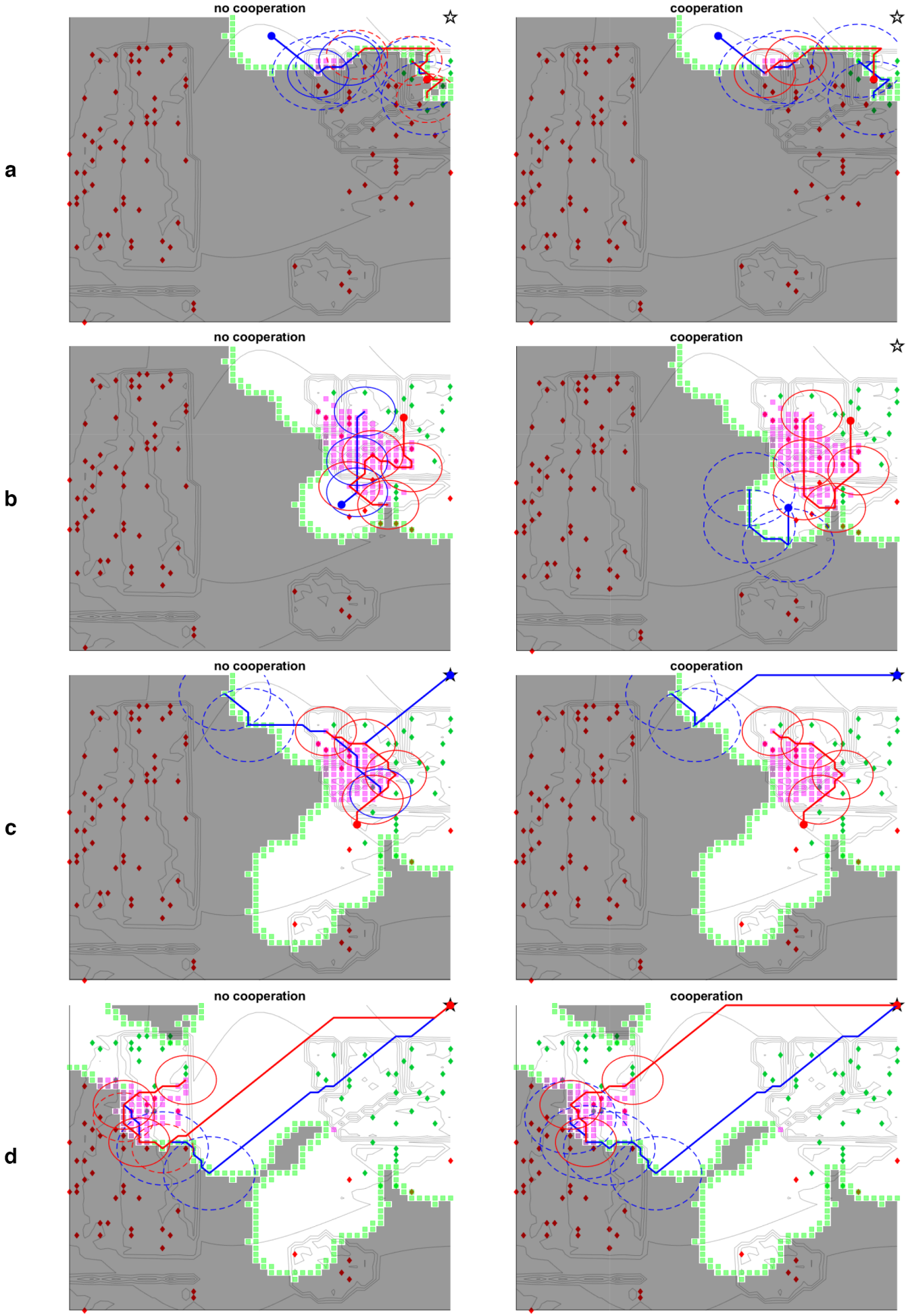
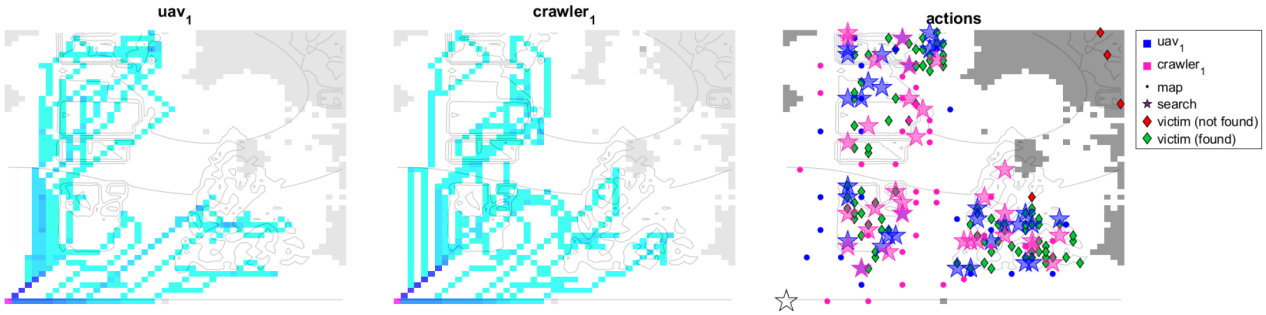


Figure 5.18: Comparison of global coordination controller outputs at different stages of $\mathcal{G}_2$ exploration with the UAV and crawler team.

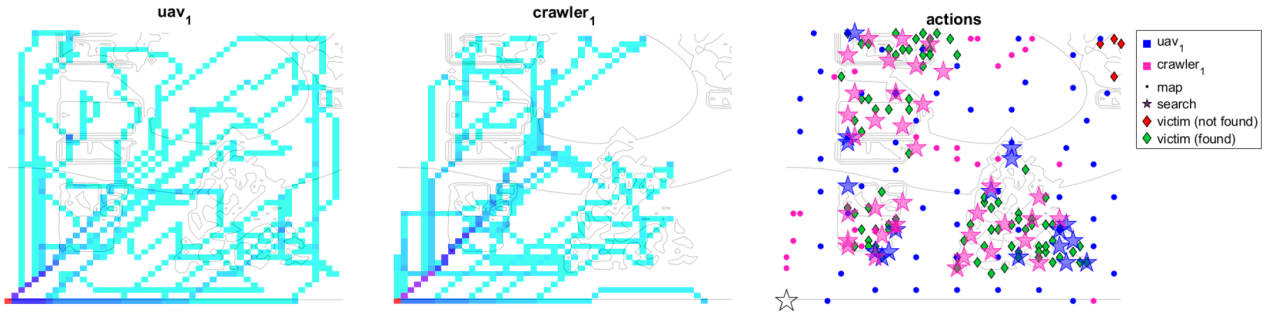
Continuing the qualitative analysis of the global coordination controller behavior within the $\mathcal{G}_2$ environment, the Figure 5.18a shows the mission state where a portion of M_7 is already discovered as soon as the mission starts, and the crawler planning for its new schedule after performing initial searches on the discovered part. Both robots plan their schedule at the local level to exploit the discovery of M_7 , picking a conflicting combination of both search and map actions along the same routes approaching from different ends. The global coordination controller's most optimal allocation is to assign all search tasks to the crawler and all mapping tasks to the UAV, operating simultaneously on the same path. This example shows that the coordination controller not only aims to spread the robots within the USaR area, but also helps them operate simultaneously on parallel tasks if a highly exploitable local task set is present. The second mission instance, as shown in Figure 5.18b, demonstrates that most of the M_7 area is already mapped, and the search tasks within that area yield the best local utility for both robots, as can be seen from the controller outputs before cooperation. As a result, most scheduled actions across robot paths are conflicting, as they are all allocated to the crawler due to its higher capability of performing them. Similar to the previously explained scenarios in $\mathcal{G}_1$, the UAV plans a new local path disregarding all the actions that conflict with the global coordination output, therefore choosing to map new areas instead. Note that the crawler allocation order of the chosen search tasks is not the most optimal path, which may be caused by either the time normalization of the elapsed times within the MILP formulation of the global controller being too large, reducing the effect of travel times between actions, or the bias caused by feeding the local controller outputs to the global controller as a warm start. Regardless, the tasks are allocated correctly, and the inefficient behavior of the current schedule is expected to be corrected in the next planning step for the crawler, as more search tasks are removed from the global task set. This hypothesis is further supported by Figure 5.18c, where the mission state is a few path planning steps after the previous example. Here, the UAV had just finished charging and planned a path that first aims to search the high-priority tasks within the M_7 area and then moves towards mapping a new portion of the map. In the meantime, the crawler steers towards the remaining search tasks that it failed to optimally order in the previous example, before returning to charging. As a result of the global coordination controller, the crawler retains its local path, while the UAV is allocated a direct path for the only non-conflicting, yet best-suited, mapping tasks. The final example in Figure 5.18d jumps ahead in $\mathcal{G}_2$ where the M_7 area is completed and the robots have just started exploring M_5. In this case, both robots had finished charging simultaneously and are now in the same location. Since both robots have full energy and can plan for the entirety of their prediction horizon, they initially choose the faster-to-execute map tasks locally and steer towards the concentrated search tasks, for which they select the exact same locations for search actions. Consequently, the global coordination controller assigns all search tasks to the crawler and all mapping tasks to the UAV. This is due to the distinct difference in perceived capability of the robots for performing different actions, where if the robots were similar enough, the allocation result would be more balanced for all examples provided in Figure 5.18. In this case study, however, the seemingly trivial prioritization of the crawler for search tasks and the UAV for mapping tasks highlights the effectiveness of the global coordination controller, provided that the fuzzy models on which the robots rely for decision-making are well-tuned.

Quantitative Analysis

The Figures 5.19a and 5.19b show the heatmap of the locations each robot visited and the action distribution for each robot for the environments $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively. Note that for $\mathcal{G}_1$, when the UAV and the crawler cooperate, all tasks within the USaR are is completed in just three hours, therefore for the no cooperation case only the results of the first three hours are shown instead of the entirety of the four hour mission. The results when the robots do not cooperate are given in Figure 5.19a, where it can be seen that although most of the prioritized areas M_1, M_2, M_3 and M_4 are completed, both robots selfishly try to concentrate on the same areas and the action distribution is evenly distributed. In contrast, the cooperation results given in Figure 5.19b show that the majority of the chosen mapping actions are performed by the UAV while the majority of the search actions are performed by the crawler, and the entirety of the mission area is explored before the simulation ends. Furthermore, the heatmap of robot movements is less jittery in the sense that bigger loops are taken from the central agent around the USaR area and can cover more of the $\mathcal{G}_1$ environment compared to the no-coordination experiment.

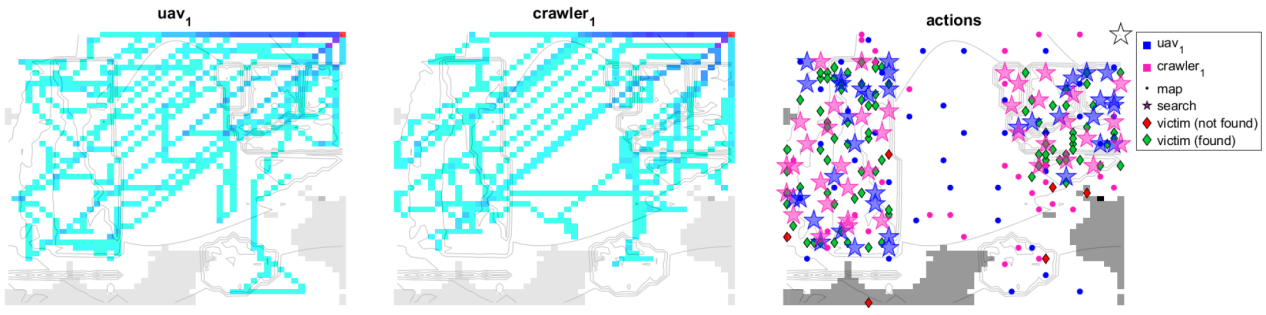


(a) Experiment with no coordination. (left to right: UAV heatmap, crawler heatmap, taken actions)

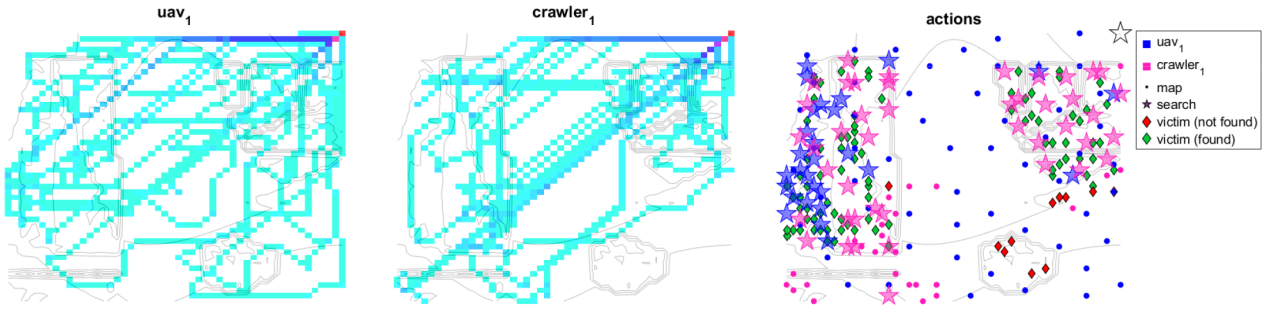


(b) Experiment with coordination. (left to right: UAV heatmap, crawler heatmap, taken actions)

Figure 5.19: UAV and crawler mission in $\mathcal{G}_1$ at the end of three simulation hours.

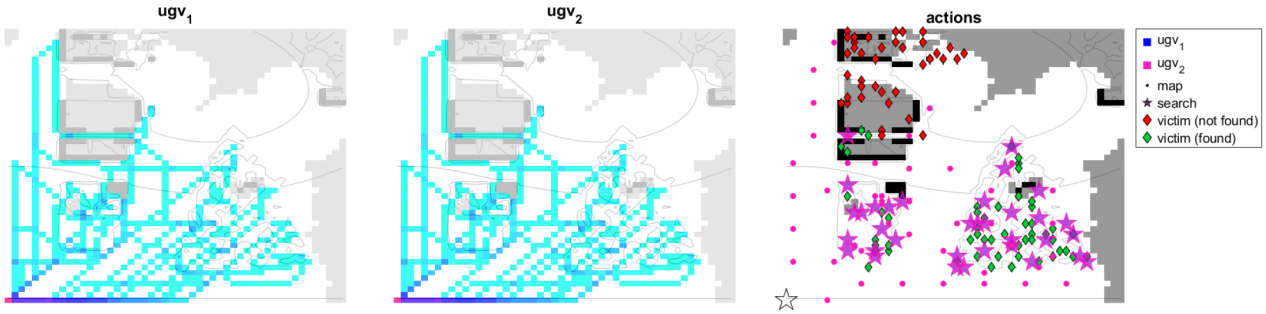


(a) Experiment with no coordination. (left to right: UAV heatmap, crawler heatmap, taken actions)

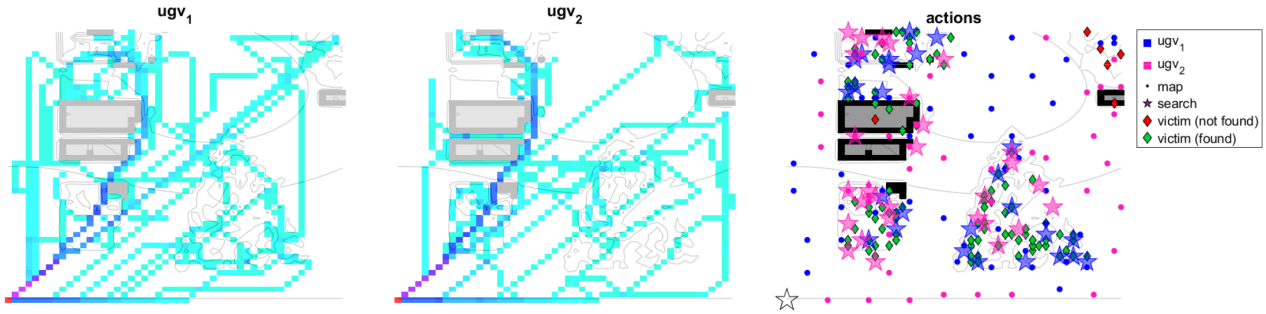


(b) Experiment with coordination. (left to right: UAV heatmap, crawler heatmap, taken actions)

Figure 5.20: UAV and crawler mission in $\mathcal{G}_2$ at the end of four simulation hours.

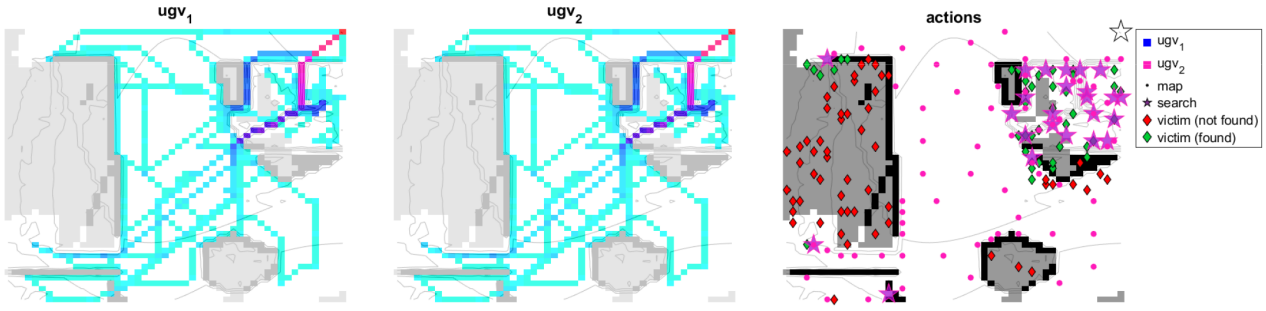


(a) Experiment with no coordination. (left to right: UGV_1 heatmap, UGV_2 heatmap, taken actions)

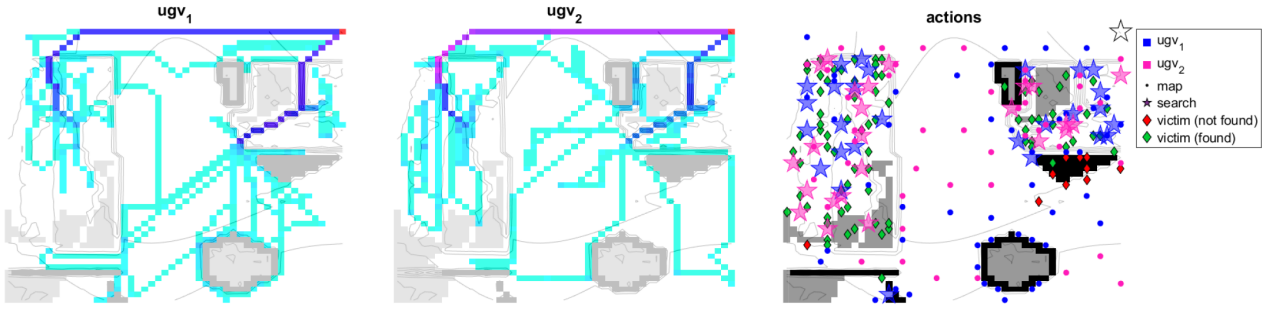


(b) Experiment with coordination. (left to right: UGV_1 heatmap, UGV_2 heatmap, taken actions)

Figure 5.21: Two UGV mission in $\mathcal{G}_1$ at the end of three and a half simulation hours.



(a) Experiment with no coordination. (left to right: UGV_1 heatmap, UGV_2 heatmap, taken actions)



(b) Experiment with coordination. (left to right: UGV_1 heatmap, UGV_2 heatmap, taken actions)

Figure 5.22: Two UGV mission in $\mathcal{G}_2$ at the end of four simulation hours.

heterogeneous team		mapped area (%)			detected victims			mission end time
		UAV	crawler	**total**	UAV	crawler	**total**	**(hh:mm)**
$\mathcal{G}_1$	**coordination**	93	7	100	20	76	96	03:00
	no coordination	65	14	79	29	61	90	
$\mathcal{G}_2$	**coordination**	91	9	100	22	71	93	04:00
	no coordination	66	19	85	26	58	84	

Table 5.4: Summary of key results from the qualitative analysis of the heterogeneous team coordination experiments within small missions.

homogeneous team		mapped area (%)			detected victims			mission end time
		UGV_1	UGV_2	**total**	UGV_1	UGV_2	**total**	**(hh:mm)**
$\mathcal{G}_1$	**coordination**	50	46	96	45	49	94	03:30
	no coordination	66	5	71	52	6	58	
$\mathcal{G}_2$	**coordination**	39	43	82	41	40	81	04:00
	no coordination	61	4	65	38	5	43	

Table 5.5: Summary of key results from the qualitative analysis of the homogeneous team coordination experiments within small missions.

The action distribution behavior is even more prevalent in the results within $\mathcal{G}_2$ environment presented on Figure 5.20. Since this environment introduces more challenges than the $\mathcal{G}_2$ environment, the experiment where the robots cooperate take at least four hours to complete all possible tasks. The Figure 5.20a shows the results when the robots do not coordinate, and is similar to the results of the $\mathcal{G}_1$ environment. However, on Figure 5.20b the M_7 area is almost exclusively searched by the crawler, while it is almost exclusively mapped by the UAV. In contrast, the M_5 area is searched by both of them equally, noting that the corresponding building within contains the most victims in the entirety of the USaR environment $\mathcal{G}$ with higher destruction levels, therefore usually prioritized first upon exploration. For that reason, both robots are steered to search this area simultaneously. The effect of the speed difference between the robots is visible here, observing that the UAV searches are mostly concentrated on the farthest side of M_5 to the central agent. Comparing the heatmaps for both robots, the crawler typically performs large loops around the search area to maximize the efficiency of its task allocation, while doing so, it spends a considerable amount of time traversing compared to the UAV. To maximize the simultaneous operation utilities, the global coordination controller assigns the UAV the search tasks where it can reach faster than the crawler. Since the UAV search capability is low, the corresponding measurements contain errors, accounting for the higher concentration of the search actions taken by this robot on the left-hand side of M_5. The mission ends before any robots prioritize searching M_6, which is an acceptable tradeoff since most of the important areas are efficiently completed by that time. The results are summarized in the Table 5.4, where it is clearly visible that the complete area coverage in $\mathcal{G}_1$ is achieved with coordination, while the non-coordinated team covers only 79% of the map. Detected victims increase by 6.7%, and the mission finishes a simulation hour earlier, indicating both improved effectiveness and operational efficiency. In the more complex $\mathcal{G}_2$ area, coordination still boosts area coverage by 8% and raises victim detection by nearly 12%, again with a full hour saved in mission time.

For completeness, a team with two UGVs is also tested, as shown in Figures 5.21 and 5.22. It can be seen that the robots take identical actions and paths when the global coordination controller is not used. In contrast, the coordinated experiment in $\mathcal{G}_1$ given in Figure 5.21b is not only completely finished in three and a half simulation hours, but also the robots are distributed evenly among the tasks they undertake. A clear partitioning is visible, especially when compared to their respective heatmaps,

indicating that the global controller achieves spatial dispersion of the robots, in contrast to the concentrated behavior of the UAV-crawler team. This behavior is more advantageous for homogeneous teams, which is an emergent property of the designed control architecture presented in this thesis. Remarkably, in the experiments within $\mathcal{G}_2$ given in Figure 5.22b, the robots can find their way into the M_5 area, which was not achieved in other corresponding UGV experiments. The majority of the USaR area, in this case, is completely explored as well, highlighting the effectiveness of the coordination strategy. Since both robots have completely the same properties, and errors within time estimations and action executions are not modeled alongside the observation error implementation, the robots always take the same actions when presented with the same information. This behavior is also visible from the summarized results provided in Table 5.5, where, without coordination, most action executions are performed by the first robot that is initialized in the simulations. The smaller number of detections achieved by the other robot can be attributed to the corrections made for the erroneous measurements by the other robot at the same time. In $\mathcal{G}_1$, coordination leads to a 62% increase in detected victims and a 35% gain in area coverage, while reducing mission time by 30 minutes. The effect is even more pronounced in $\mathcal{G}_2$, where the number of detected victims rises by 88% and area coverage improves by 26%, again with a time reduction of half an hour. These numbers highlight a key insight: coordination has a disproportionately positive effect on homogeneous teams, compensating for their lack of functional diversity through more effective spatial distribution and improved task management.

5.3.2. Analysis of the Global Coordination Controller in Structured Experiments

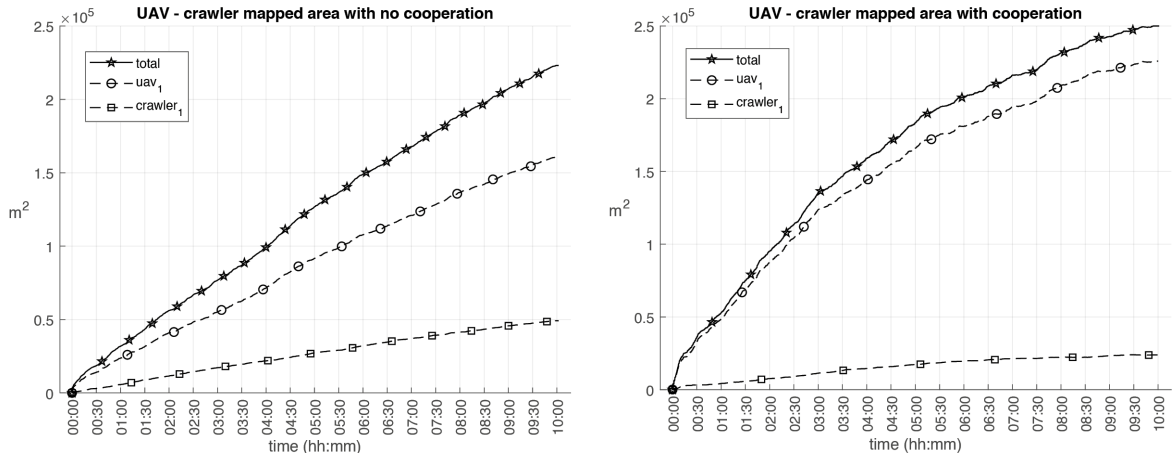


Figure 5.23: Total mapped area progression of UAV-crawler team experiments with dynamic central agent.

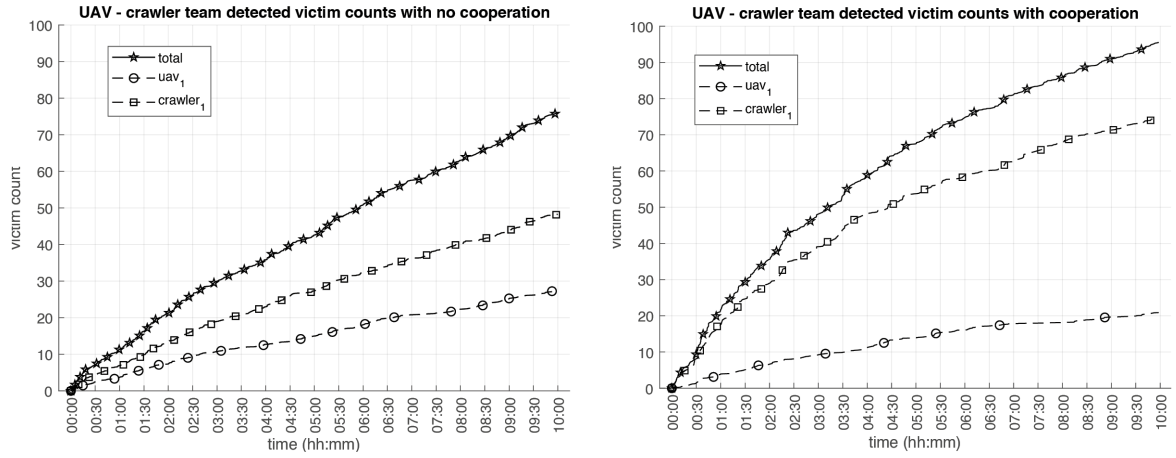


Figure 5.24: Detected victim count progression of UAV-crawler team experiments with dynamic central agent.

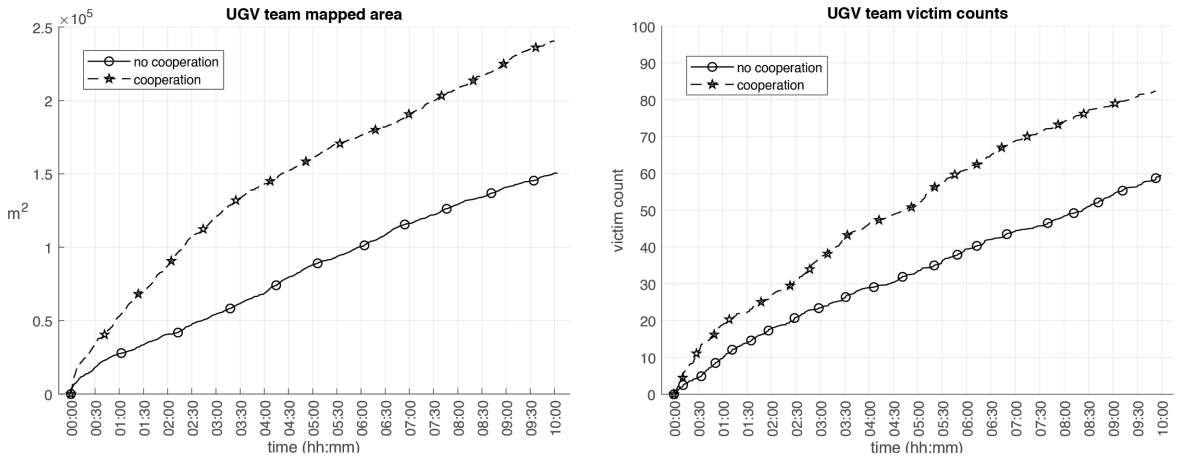


Figure 5.25: Total mapped area and detected victim count comparisons of the UGV team with dynamic central agent.

	mapped area (%)			**detected victims (%)**			**victim health (%)**		
	UAV	crawler	**total**	UAV	crawler	**total**	UAV	crawler	**total**
coordination	90.3	9.6	100.0	20.9	74.5	95.5	53.3	71.2	67.2
no coordination	65.5	19.8	85.3	27.5	48.3	75.9	62.4	62.1	62.2
	Nr. of search actions			**Nr. of map actions**					
	UAV	crawler	**total**	UAV	crawler	**total**			
coordination	70.9	116.3	187.2	167.2	88.7	255.9			
no coordination	83.5	87.9	171.4	131.0	118.9	249.9			

Table 5.6: Outputs of the USaR mission in $\mathcal{G}$ for the UAV-crawler team with dynamic central agent at the end of ten simulation hours.

Since the behavioral preferences are highlighted more obviously for the heterogeneous robotic team, on Figures 5.23 and 5.24, the total mapped area and victim detection progressions are given for the UAV-crawler team with a dynamic agent within the general USaR area $\mathcal{G}$, respectively. The dynamic agent is selected for these experiments because, from the single-robot results, the effect of an autonomously moving central agent on enabling the system to converge to a globally optimal state is consistently observed, and the effect of enabling coordination between robots is generally independent of the central agent's schedules. Thus, the proposed architecture of complete autonomy is used for the final set of multi-robot experiments. On Figure 5.23, without coordination, the system can map 85.3% of the USaR area at the end of the mission, with UAV actions accounting for 65.5% and the crawler actions for 19.8% of this result. In contrast, for all 34 experiments performed from 17 different initial conditions, the robot team was able to map 100% of the USaR area with coordination. Furthermore, 90.3% of the $\mathcal{G}$ environment is mapped by the UAV, and the remaining is mapped by the crawler. This result further reinforces the observation that the global coordination controller can assign tasks more suitable to each individual robot, thereby increasing the system's time efficiency. Since the total areas are quite close to each other, another interesting observation from the Figure 5.23 can be made regarding the rise time of the averaged mapped area curve. Without coordination, the 25% total area is mapped at around three simulation hours, whereas this duration drops to around one and a half simulation hours for the coordinated robots. For 50% the total area, the robots spend approximately five hours and two hours forty-five minutes without and with coordination, respectively. Thus, cooperation generally enabled the UAV-crawler team to map the USaR area twice as fast at the earlier stages of the mission, which in turn generated more search tasks to be evaluated sooner.

A similar trend is observed for the averaged victim detection results provided in Figure 5.24, where the team with coordination detects 95.5% of the victims and the team with no coordination detects 75.9% of all victims. The correlation to the faster rise times achieved by the coordination controller is also observable for the victim detection. These outcomes together lead to the conclusion that without coordination, the robots spend more time on actions that are ineffective due to the reduced capabilities defined in the robot models, even though the capability values of robots are included in the local controller predictions. This makes sense from the perspective of local controllers, since even with a reduced capability, the robots still need to perform searching and mapping as efficiently as possible. In Table 5.6, further data is provided regarding the average health status of victims for both cases, as well as how many times on average the robots performed search and mapping actions. From the victim health percentage results, it can be seen that without coordination, the robots measure whoever they can selfishly, resulting in similar health percentages at around 60%. In contrast, with coordination, the crawler can detect substantially more victims with average health 71.2%, meaning the proposed architecture not only assigns more searching tasks to the crawler but can also prioritize more important areas much more effectively during the USaR mission. In return, the UAV detects victims that are not as urgent as others, while this results in a lower health percentage, it means that these victims are discovered much later during the mission from less destroyed areas, which is a desirable behavior to save as many victims as possible with better health conditions. From the averaged number of performed actions for both cases, it can be seen that coordination results in more actions taken due to the better rise times of exploration, enabling more options faster than the no-coordination case. Finally, the comparison of total mapped areas and detected victims for a team with two UGVs using a dynamic central agent is provided in Figure 5.25. Here, the effectiveness of the proposed system on homogeneous teams is easily observed, with 60.2% of the USaR area mapped without coordination compared to the 96.2% reached with coordination. Similarly, without coordination, the homogeneous team finds 59.3% of the victims while with coordination, this value reaches 82.3%. For both of these metrics, the global coordination controller achieves approximately 50% increase within the allocated mission time, which is consistent with the expectation of doubling the total number of robots operating at the same time.

6

Conclusions & Topics for Future Research

Addressing the autonomy needs of a heterogeneous team of robots in the search and rescue (SaR) domain is a multifaceted challenge that requires both rapid exploration of unknown environments and a reliable mechanism for determining which tasks are most relevant at any given time. Effective performance depends not only on identifying and prioritizing tasks among competing objectives but also on understanding the alternative ways to achieve similar goals while minimizing overall mission time. In response to these demands, this thesis introduced a unified architecture for autonomous multi-robot teams operating in urban search and rescue (USaR) missions, aimed at achieving adaptive and energy-aware coordination under dynamically evolving task conditions. The design and implementation of the proposed framework were grounded in key challenges identified in the SaR literature. To this end, the work introduced a generalized system model that decouples robot identity from fixed roles, instead characterizing each robot through its available heterogeneous actions and the resulting environmental observations. This abstraction enabled an observation-driven notion of task formation, allowing robots to dynamically interpret and evaluate their own roles within a shared, mission-level objective space.

The first research question posed in this thesis concerned the dynamic prioritization and selection of tasks in a USaR environment using a combination of fuzzy reasoning and multi-criteria utility modeling, and how this approach improves flexibility and decision quality over static or heuristic allocations. In response, the research introduced a two-layer decision framework that first utilizes a fuzzy inference system to evaluate imprecise or emergent task characteristics and then applies a multi-criteria decision-making (MCDM) utility model to rank candidate robot actions across mission-relevant factors, such as urgency and information gain. This dynamic, observation-driven prioritization enables robots to continually reassess task importance as conditions evolve, achieving a level of adaptability that fixed heuristics cannot match. In contrast to static rule-based methods, which often fail when confronted with unexpected tasks or changing mission priorities, the fuzzy–MCDM approach offers greater flexibility by encoding human-like reasoning about task value and adjusting to new information on the fly. This two-stage mapping effectively converts complex, uncertain inputs into a unified priority metric at each planning step, enabling continuous adaptation, while the dynamicity of the task allocations as the missions progresses is maintained by the robust predictive framework the MPC modeling provides. As a result, the designed framework consistently improves the decision quality. Experiments demonstrated that the proposed task allocation strategy identified victims more quickly and achieved higher overall utility scores than heuristic baselines in the majority of 17 simulated single-robot scenarios. Specifically, the methods using the proposed decision-making prioritization architecture found two to three times more victims compared to other methods, while balancing high area mapping percentages, which is a competing objective in the formulated case study. Robots employing this adaptive prioritization reached critical targets sooner and maintained higher victim health levels, demonstrating more effective focus on urgent, high-value tasks without relying on mission-specific hand-tuned heuristics. Thus, for the first research question, it is concluded that the fuzzy–MCDM strategy provides a more flexible and robust task allocation approach, markedly improving decision quality in dynamic task expressions within

USaR environments.

The second research question focused on how to reduce the computational complexity of predictive decision-making, arising from the generalized system model, to enable real-time planning, and how the resulting controller performs in a fully autonomous single-robot setting with charging constraints and a mobile central agent. To address this, the thesis formulated a model predictive control (MPC) approach for task planning and introduced a novel two-stage strategy to make it tractable in real time. In the first stage, each robot's local controller prunes the decision space by filtering for a subset of high-value, nearly independent candidate actions using the fuzzy utility outputs and by removing overlaps in their predicted observations. In the second stage, the controller solves a simplified scheduling problem over this reduced action set with a mixed-integer linear programming (MILP) implementation, essentially an orienteering-type routing optimization that incorporates travel times, charging opportunities, and safe-return energy constraints. This decomposition significantly reduces the combinatorial complexity of planning while preserving the multi-criteria structure of the decision model, effectively enabling real-time, online re-planning. Most MILP solvers in contemporary use employ efficient branch-and-cut structures that are well-suited to the sequential ordering maintained by the proposed constraints of the optimization problem for both local and global controllers. This effectively narrows down the search space to the most relevant action-task pairings as more resources are allocated to the optimization process. Furthermore, the linear encoding of the model predictions and action ordering enables direct implementations on hardware that can facilitate parallel processing, making the proposed method of this thesis a strong architecture for real-life embedded implementation. In a fully autonomous single-robot scenario, where the robot periodically returns to a mobile central charging agent, the controller demonstrated performance comparable to that of a state-of-the-art nonlinear MPC solution, but with significantly greater efficiency. While the comparisons to non-predictive control approaches highlighted the importance of a look-ahead structure within an evolving mission, the proposed MPC linearization achieved similar mission outcomes to a genetic algorithm baseline that directly solved the nonlinear problem. Yet, the proposed methodology produced high-quality solutions significantly faster, meeting real-time requirements that the metaheuristic struggled to meet. The controller proved capable of continuous energy-aware operation, proactively scheduling charge-return actions and navigating with safe-return constraints always in mind. In practice, the robot planned ahead to loop back toward the mobile charger before battery depletion and optimized its route for both mission utility and power management. Additionally, the local controller maintained scalability with respect to increasing task density and spatial complexity while remaining responsive to the evolving mission-level information flow. Thus, for the second research question, it is established that the two-stage MILP linearization of the generalized MPC framework, applicable to all types of robotic missions that can be represented by the mathematical model provided in this thesis, meets real-time requirements and enables continuous, energy-aware operation without sacrificing planning quality for small to medium prediction horizons.

The third research question investigated the impact of integrating a global hierarchical extension on top of the local controllers on the coordination, scalability, and overall performance of a multi-robot team with abstract heterogeneity, particularly in terms of conflict resolution and task consistency under increasing mission complexity. The thesis addressed this by introducing a global coordination controller that acts as a higher-level supervisor to the individual robot planners. This global layer takes the tentative plans from each local controller and merges them into a consistent team-wide strategy whenever a coordination criterion is met, explicitly resolving conflicts such as duplicate task assignments and overlapping action regions. Since the results from the individual robots are already locally optimal and feasible in terms of safe return, a global redistribution of the combined pool of coordinating robot actions is sufficient for approximating a synchronous and mission-level optimal behavior of the robotic team. The addition of this hierarchical coordinator significantly improved multi-robot collaboration. Tasks were allocated to the most suitable robots based on capabilities and workload, minimizing redundant efforts and idle time, while maintaining a unified mission focus even as new tasks emerged. This led to improved scalability and overall performance. In heterogeneous two-robot trials, for example, involving an unmanned aerial vehicle (UAV) and a crawler-type unmanned ground vehicle (UGV), the coordinated system achieved near-complete area coverage and victim detection rates above 90%, compared to substantially lower performance without coordination. The globally coordinated team also demonstrated faster exploration in early mission phases, as evidenced by approximately twice the rise times of mapped areas and detected victim counts in the presence of the coordination controller. In

homogeneous-team scenarios, introducing the global coordination layer similarly enhanced effectiveness, with coordinated teams completing more mission objectives in less time. These results confirm that the hierarchical extension enhances task consistency, resource distribution, and mission throughput. The coordination mechanism, designed to handle abstract heterogeneity through capability-based rather than role-based modeling, scaled smoothly to various team compositions and mission scales without requiring hard-coded behavioral logic. Thus, for the third research question, it is observed that the global coordination mechanism significantly enhances the operational effectiveness of both heterogeneous and homogeneous team compositions, while eliminating redundant task allocations by establishing a mission-level preference for ambiguous robot capabilities.

In summary, the findings across all three questions demonstrate a cohesive advancement in autonomous SaR task allocation. This thesis established a unified hierarchical control architecture that supports adaptive task prioritization, efficient real-time planning under operational constraints, and coordinated multi-robot behavior. The full-stack approach yielded consistent gains in mission performance, including faster victim recovery, improved utility accumulation, and more complete area exploration, even under energy and communication limitations. By tightly integrating fuzzy-MCDM-based utility modeling with scalable predictive control and coordination layers, the system responded effectively to evolving task demands without relying on static policies or handcrafted roles. The architecture's capacity for autonomous operation, proactive energy scheduling, and dynamic conflict resolution was validated across extensive scenario-based evaluations. Looking forward, several extensions could significantly enhance the capabilities and applicability of the proposed architecture. First, allowing dynamic adjustment of the MCDM utility weights throughout the mission would enable the system to shift its priorities, for example, emphasizing exploration early on and focusing on victim retrieval as information accumulates, mirroring real-world strategic adaptations. Second, introducing temporal logic constraints, such as requiring certain actions to follow others (e.g., transporting a payload after picking it up), would support more complex and goal-oriented behaviors beyond the current atomic task model. In parallel, integrating learning-based models could enable robots to refine their utility estimations and task prioritization based on environmental feedback and past performance, thereby improving robustness in novel or uncertain conditions. Another promising direction is integrating a framework for handling planned tasks that are expected to become available in a known future time. By enabling robots to anticipate and schedule tasks in advance, the system can support more proactive and temporally coordinated behaviors, thereby improving overall mission efficiency and responsiveness to time-sensitive opportunities. Beyond these improvements to decision-making, future work could also explore scalability under larger teams and partial observability, integration with human operators for semi-autonomous coordination, and real-world deployment to assess performance under physical constraints such as noisy sensing, energy degradation, or communication failures. Together, these directions represent a promising path toward more adaptive, interpretable, and resilient autonomous SaR systems.

References

[1] CRED. *EM-DAT - the International Disaster Database*. UCLouvain, 2023. URL: www.emdat.be.

[2] Sean Grogan, Robert Pellerin, and Michel Gamache. “The Use of Unmanned Aerial Vehicles and Drones in Search and Rescue Operations - a Survey”. In: *Proceedings of the PROLOG*. Hull, UK, 2018.

[3] R.R. Murphy et al. “Mobility and sensing demands in USAR”. In: vol. 1. IEEE Computer Society, 2000, pp. 138–142. DOI: 10.1109/IECON.2000.973139.

[4] Alaa Khamis, Ahmed Hussein, and Ahmed Elmogy. “Multi-robot task allocation: A review of the state-of-the-art”. In: *Studies in Computational Intelligence* 604 (2015), pp. 31–51. ISSN: 1860949X. DOI: 10.1007/978-3-319-18299-5_2.

[5] Reza Hassanzadeh and Zorica Nedovic-Budic. “Where to Go First: Prioritization of Damaged Areas for Allocation of Urban Search and Rescue (USAR) Operations (PI-USAR Model)”. In: *Geomatics, Natural Hazards and Risk* 7.4 (2016), pp. 1337–1366. DOI: 10.1080/19475705.2015.1058861.

[6] Christopher de Koning and Anahita Jamshidnejad. “Hierarchical Integration of Model Predictive and Fuzzy Logic Control for Combined Coverage and Target-Oriented Search-and-Rescue via Robots with Imperfect Sensors”. In: *Journal of Intelligent and Robotic Systems: Theory and Applications* 107 (3 Mar. 2023). ISSN: 15730409. DOI: 10.1007/s10846-023-01833-2.

[7] Nicola Basilico and Francesco Amigoni. “Exploration strategies based on multi-criteria decision making for searching environments in rescue operations”. In: *Autonomous Robots* 31 (4 Nov. 2011), pp. 401–417. ISSN: 09295593. DOI: 10.1007/s10514-011-9249-9.

[8] Milt Statheropoulos et al. “Factors That Affect Rescue Time in Urban Search and Rescue (USAR) Operations”. In: *Nat Hazards* 75 (2015), pp. 57–69.

[9] Robin R Murphy. *Disaster Robotics*. USA: MIT Press, Sept. 2017. ISBN: 978-953-51-3376-6.

[10] Anna Rom and Ilan Kelman. “Search Without Rescue? Evaluating the International Search and Rescue Response to Earthquake Disasters”. In: *BMJ Global Health* 5.12 (2020).

[11] Eduardo Reinoso, Miguel A. Jaimes, and Luis Esteva. “Estimation of Life Vulnerability Inside Buildings During Earthquakes”. In: *Structure and Infrastructure Engineering* 14.8 (2018), pp. 1140–1152. DOI: 10.1080/15732479.2017.1401097.

[12] Navid Hooshangi et al. “Urban Search and Rescue (USAR) Simulation System: Spatial Strategies for Agent Task Allocation Under Uncertain Conditions”. In: *Natural Hazards and Earth System Sciences* 21.11 (2021), pp. 3449–3463. DOI: 10.5194/nhess-21-3449-2021.

[13] J.L. Burke and R.R. Murphy. “Human-Robot Interaction in USAR Technical Search: Two Heads are Better Than One”. In: *RO-MAN 2004. 13th IEEE International Workshop on Robot and Human Interactive Communication*. 2004, pp. 307–312. DOI: 10.1109/ROMAN.2004.1374778.

[14] Styliani Verykokou et al. “3D Reconstruction of Disaster Scenes for Urban Search and Rescue”. In: *Multimedia Tools and Applications* 77 (8 Apr. 2018). sensory requirements of UAVs, pp. 9691–9717. ISSN: 15737721. DOI: 10.1007/s11042-017-5450-y.

[15] J.L. Casper and R.R. Murphy. “Workflow Study on Human-Robot Interaction in USAR”. In: *Proceedings 2002 IEEE International Conference on Robotics and Automation*. Vol. 2. 2002, 1997–2003 vol.2. DOI: 10.1109/ROBOT.2002.1014834.

[16] Shashank Govindaraj et al. “Command and Control Systems for Search and Rescue Robots”. In: *Search and Rescue Robotics*. Rijeka: IntechOpen, 2017. Chap. 8. DOI: 10.5772/intechopen.69495. URL: https://doi.org/10.5772/intechopen.69495.

[17] Elena Messina and Adam Jacoff. “Performance Standards for Urban Search and Rescue Robots”. In: *Unmanned Systems Technology VIII*. Vol. 6230. International Society for Optics and Photonics. SPIE, 2006, p. 62301V. DOI: 10.1117/12.663320.

[18] Binoy Shah and Howie Choset. “Survey on Urban Search and Rescue Robots”. In: *Journal of the Robotics Society of Japan* 22.5 (2004), pp. 582–586. DOI: 10.7210/jrsj.22.582.

[19] Jinguo Liu et al. “Current research, key performances and future development of search and rescue robots”. In: *Frontiers of Mechanical Engineering in China* 2 (4 Oct. 2007), pp. 404–416. ISSN: 16733479. DOI: 10.1007/s11465-007-0070-2.

[20] Markus Eich et al. “A versatile stair-climbing robot for search and rescue applications”. In: 2008, pp. 35–40. ISBN: 9781424420322. DOI: 10.1109/SSRR.2008.4745874.

[21] L. Bruzzone and G. Quaglia. “Review article: locomotion systems for ground mobile robots in unstructured environments”. In: *Mechanical Sciences* 3.2 (2012), pp. 49–62. DOI: 10.5194/ms-3-49-2012. URL: https://ms.copernicus.org/articles/3/49/2012/.

[22] Karsten Berns et al. “Unmanned Ground Robots for Rescue Tasks”. In: *Search and Rescue Robotics*. Rijeka: IntechOpen, 2017. Chap. 4. DOI: 10.5772/intechopen.69491. URL: https://doi.org/10.5772/intechopen.69491.

[23] Yugang Liu and Goldie Nejat. “Robotic urban search and rescue: A survey from the control perspective”. In: *Journal of Intelligent and Robotic Systems: Theory and Applications* 72 (2 Nov. 2013), pp. 147–165. ISSN: 15730409. DOI: 10.1007/s10846-013-9822-x.

[24] Aksel Andreas Transeth, Kristin Ytterstad Pettersen, and Pal Liljeback. “A survey on snake robot modeling and locomotion”. In: *Robotica* 27.7 (2009), pp. 999–1015. DOI: 10.1017/S0263574709005414.

[25] Houxiang Zhang et al. *A Novel Modular Mobile Robot Prototype for Urban Search and Rescue*. 2008. DOI: 10.5772/6055. URL: www.intechopen.com.

[26] Sangchul Han et al. “Snake Robot Gripper Module for Search and Rescue in Narrow Spaces”. In: *IEEE Robotics and Automation Letters* 7.2 (2022), pp. 1667–1673. DOI: 10.1109/LRA.2022.3140812.

[27] Alexander Ferworn et al. “Dog and Snake Marsupial Cooperation for Urban Search and Rescue Deployment”. In: *2012 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. 2012, pp. 1–5. DOI: 10.1109/SSRR.2012.6523887.

[28] Tetsushi Kamegawa et al. “Development of a separable search-and-rescue robot composed of a mobile robot and a snake robot”. In: *Advanced Robotics* 34.2 (2020), pp. 132–139. DOI: 10.1080/01691864.2019.1691941.

[29] Justin Huff, Stephen Conyers, and Richard Voyles. “MOTHERSHIP - A serpentine tread/limb hybrid marsupial robot for USAR”. In: 2012. ISBN: 9781479901654. DOI: 10.1109/SSRR.2012.6523893.

[30] Ariel Braverman. “Unmanned aerial systems (UAS) in urban search and rescue-methodology, capacity development, and integration”. In: *Journal of Emergency Management* (1 Jan. 2021), pp. 33–38. ISSN: 33735433. DOI: 10.5055/jem.0496.

[31] Rudin Konrad, Daniel Serrano, and Pascal Strupler. “Unmanned Aerial Systems”. In: *Search and Rescue Robotics*. Rijeka: IntechOpen, 2017. Chap. 3. DOI: 10.5772/intechopen.69490. URL: https://doi.org/10.5772/intechopen.69490.

[32] Scott Morton and Nikolaos Papanikolopoulos. *A Small Hybrid Ground-Air Vehicle Concept*. 2017. ISBN: 9781538626825.

[33] Chinthaka Premachandra et al. “A study on development of a hybrid aerial terrestrial robot system for avoiding ground obstacles by flight”. In: *IEEE/CAA Journal of Automatica Sinica* 6 (1 Jan. 2019), pp. 327–336. ISSN: 23299274. DOI: 10.1109/JAS.2018.7511258.

[34] Di Zhang et al. “Development of a Hybrid Locomotion Robot for Earthquake Search and Rescue in Partially Collapsed Building”. In: *2019 IEEE International Conference on Mechatronics and Automation (ICMA)*. 2019, pp. 2559–2564. DOI: 10.1109/ICMA.2019.8816327.

[35] Francisca Rosique et al. *A systematic review of perception system and simulators for autonomous vehicles research*. Feb. 2019. DOI: 10.3390/s19030648.

[36] Steve Burion. “Diploma Work Human Detection for Robotic Urban Search and Rescue Human Detection for Robotic Urban Search and Rescue”. PhD thesis. École Polytechnique Fédérale de Lausanne, 2004.

[37] Andreas T. Güntner et al. “Sniffing Entrapped Humans with Sensor Arrays”. In: *Analytical Chemistry* 90 (8 Apr. 2018), pp. 4940–4945. ISSN: 15206882. DOI: 10.1021/acs.analchem.8b00237.

[38] Nathan Michael et al. “Collaborative mapping of an earthquake-damaged building via ground and aerial robots”. In: *Journal of Field Robotics* 29 (5 Sept. 2012), pp. 832–841. ISSN: 15564959. DOI: 10.1002/rob.21436.

[39] Jorge Pena Queralta et al. “Collaborative multi-robot search and rescue: Planning, coordination, perception, and active vision”. In: *IEEE Access* 8 (2020), pp. 191617–191643. ISSN: 21693536. DOI: 10.1109/ACCESS.2020.3030190.

[40] A. Mäyrä et al. “Optical sensors and algorithms for life-sign detection in USaR-operations”. In: *AIP Conference Proceedings*. Vol. 1537. 2013, pp. 41–46. ISBN: 9780735411616. DOI: 10.1063/1.4809690.

[41] Fahed Awad and Rufaida Shamroukh. “Human Detection by Robotic Urban Search and Rescue Using Image Processing and Neural Networks”. In: *International Journal of Intelligence Science* 04 (02 2014), pp. 39–53. ISSN: 2163-0283. DOI: 10.4236/ijis.2014.42006.

[42] Paulo Borges et al. “A Survey on Terrain Traversability Analysis for Autonomous Ground Vehicles: Methods, Sensors, and Challenges”. In: *Field Robotics* 2 (1 Mar. 2022), pp. 1567–1627. DOI: 10.55417/fr.2022049.

[43] Maurizio Pollino et al. “Assessing earthquake-induced urban rubble by means of multiplatform remotely sensed data”. In: *ISPRS International Journal of Geo-Information* 9 (4 Apr. 2020). ISSN: 22209964. DOI: 10.3390/ijgi9040262.

[44] Albert Y. Chen et al. “Supporting Urban Search and Rescue with digital assessments of structures and requests of response resources”. In: *Advanced Engineering Informatics* 26 (4 Oct. 2012), pp. 833–845. ISSN: 14740346. DOI: 10.1016/j.aei.2012.06.004.

[45] Quentin Picard et al. *A survey on real-time 3D scene reconstruction with SLAM methods in embedded systems*. 2023. arXiv: 2309.05349 [cs.RO]. URL: https://arxiv.org/abs/2309.05349.

[46] Xieyuanli Chen et al. “Robust SLAM system based on monocular vision and LiDAR for robotic urban search and rescue”. In: *2017 IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. 2017, pp. 41–47. DOI: 10.1109/SSRR.2017.8088138.

[47] F Yamazaki et al. *Construction of 3D models of buildings damaged by earthquakes using UAV aerial images*. 2015. URL: https://www.researchgate.net/publication/283506944.

[48] Shiyong Zhang et al. “Fast Active Aerial Exploration for Traversable Path Finding of Ground Robots in Unknown Environments”. In: *IEEE Transactions on Instrumentation and Measurement* 71 (2022). ISSN: 15579662. DOI: 10.1109/TIM.2022.3158425.

[49] Dimitrios Chatziparaschis, Michail G. Lagoudakis, and Panagiotis Partsinevelos. “Aerial and ground robot collaboration for autonomous mapping in search and rescue missions”. In: *Drones* 4 (4 Dec. 2020), pp. 1–24. ISSN: 2504446X. DOI: 10.3390/DRONES4040079.

[50] Héctor Azpúrua, Mario F.M. Campos, and Douglas G. Macharet. “Three-dimensional Terrain Aware Autonomous Exploration for Subterranean and Confined Spaces”. In: *Proceedings - IEEE International Conference on Robotics and Automation*. Vol. 2021-May. Institute of Electrical and Electronics Engineers Inc., 2021, pp. 2443–2449. ISBN: 9781728190778. DOI: 10.1109/ICRA48506.2021.9561099.

[51] Bernardo Esteves Henriques, Mirko Baglioni, and Anahita Jamshidnejad. “Camera-based Mapping in Search-and-Rescue via Flying and Ground Robot Teams”. In: *Machine Vision and Applications* 35.5 (Aug. 2024), p. 117. ISSN: 1432-1769. DOI: 10.1007/s00138-024-01594-4. URL: https://doi.org/10.1007/s00138-024-01594-4.

[52] Daniel S. Drew. “Multi-Agent Systems for Search and Rescue Applications”. In: *Current Robotics Reports* (2 2021), pp. 189–200. DOI: 10.1007/s43154-021-00048-3/Published.

[53] Cyril Robin and Simon Lacroix. “Multi-robot target detection and tracking: taxonomy and survey”. In: *Autonomous Robots* 40 (4 Apr. 2016), pp. 729–760. ISSN: 15737527. DOI: 10.1007/s10514-015-9491-7.

[54] J. Ricardo Sánchez-Ibáñez, Carlos J. Pérez-Del-pulgar, and Alfonso García-Cerezo. *Path planning for autonomous mobile robots: A review*. Dec. 2021. DOI: 10.3390/s21237898.

[55] David González et al. “A Review of Motion Planning Techniques for Automated Vehicles”. In: *IEEE Transactions on Intelligent Transportation Systems* 17 (4 Apr. 2016), pp. 1135–1145. ISSN: 15249050. DOI: 10.1109/TITS.2015.2498841.

[56] Alberto Valero-Gomez et al. “Fast Marching Methods in Path Planning”. In: *IEEE Robotics Automation Magazine* 20 (Dec. 2013), pp. 111–120.

[57] Mohd Nayab Zafar and J. C. Mohanta. “Methodology for Path Planning and Optimization of Mobile Robots: A Review”. In: *Procedia Computer Science*. Vol. 133. Elsevier B.V., 2018, pp. 141–152. DOI: 10.1016/j.procs.2018.07.018.

[58] Jianqiang Li et al. “A Hybrid Path Planning Method in Unmanned Air/Ground Vehicle (UAV/UGV) Cooperative Systems”. In: *IEEE Transactions on Vehicular Technology* 65 (12 Dec. 2016), pp. 9585–9596. ISSN: 00189545. DOI: 10.1109/TVT.2016.2623666.

[59] Mohammad R. Alenezi and Abdullah M. Almeshal. “Optimal Path Planning for a Remote Sensing Unmanned Ground Vehicle in a Hazardous Indoor Environment”. In: *Intelligent Control and Automation* 09 (04 2018), pp. 147–157. ISSN: 2153-0653. DOI: 10.4236/ica.2018.94011.

[60] Micael S. Couceiro, Rui P. Rocha, and Nuno M.F. Ferreira. “A novel multi-robot exploration approach based on Particle Swarm Optimization algorithms”. In: 2011, pp. 327–332. ISBN: 9781612847696. DOI: 10.1109/SSRR.2011.6106751.

[61] Junyan Hu et al. “Voronoi-Based Multi-Robot Autonomous Exploration in Unknown Environments via Deep Reinforcement Learning”. In: *IEEE Transactions on Vehicular Technology* 69 (12 Dec. 2020), pp. 14413–14423. ISSN: 19399359. DOI: 10.1109/TVT.2020.3034800.

[62] Jongho Shin, Dongjun Kwak, and Kiho Kwak. “Model Predictive Path Planning for an Autonomous Ground Vehicle in Rough Terrain”. In: *International Journal of Control, Automation and Systems* 19 (6 June 2021), pp. 2224–2237. ISSN: 20054092. DOI: 10.1007/s12555-020-0267-2.

[63] Patrick Scheffe et al. *Receding Horizon Control Using Graph Search for Multi-Agent Trajectory Planning*. Nov. 2022. DOI: 10.36227/techrxiv.16621963.v2.

[64] Jong Wook Park et al. “Advanced Fuzzy Potential Field Method for Mobile Robot Obstacle Avoidance”. In: *Computational Intelligence and Neuroscience* 2016 (2016). ISSN: 16875273. DOI: 10.1155/2016/6047906.

[65] K.N. McGuire, G.C.H.E. de Croon, and K. Tuyls. “A comparative study of bug algorithms for robot navigation”. In: *Robotics and Autonomous Systems* 121 (2019), p. 103261. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2019.103261. URL: https://www.sciencedirect.com/science/article/pii/S0921889018306687.

[66] Hailong Qin et al. “Autonomous Exploration and Mapping System Using Heterogeneous UAVs and UGVs in GPS-Denied Environments”. In: *IEEE Transactions on Vehicular Technology* 68 (2 Feb. 2019), pp. 1339–1350. ISSN: 00189545. DOI: 10.1109/TVT.2018.2890416.

[67] Farzad Niroui et al. “Deep Reinforcement Learning Robot for Search and Rescue Applications: Exploration in Unknown Cluttered Environments”. In: *IEEE Robotics and Automation Letters* 4 (2 Apr. 2019), pp. 610–617. ISSN: 23773766. DOI: 10.1109/LRA.2019.2891991.

[68] Alejandro Sarmiento, Rafael Murrieta-Cid, and Seth Hutchinson. “An efficient motion strategy to compute expected-time locally optimal continuous search paths in known environments”. In: *Advanced Robotics* 23 (12-13 Sept. 2009). path cost on a graph with dynamic weights, pareto optimal heuristic, similar to de Koning, pp. 1533–1560. ISSN: 01691864. DOI: 10.1163/016918609X12496339799170.

[69] Ryan J. Meuth et al. “Adaptive task allocation for search area coverage”. In: *2009 IEEE International Conference on Technologies for Practical Robot Applications, TePRA 2009*. 2009, pp. 67–74. ISBN: 9781424449927. DOI: 10.1109/TEPRA.2009.5339643.

[70] James R Riehl, Gaemus E Collins, and João P Hespanha. *Cooperative Graph-Based Model Predictive Search*. 2007.

[71] Geoffrey A. Hollinger and Sanjiv Singh. “Multirobot Coordination With Periodic Connectivity: Theory and Experiments”. In: *IEEE Transactions on Robotics* 28.4 (2012), pp. 967–973. DOI: 10.1109/TRO.2012.2190178.

[72] Bingxi Li et al. “Planning Large-Scale Search and Rescue using Team of UAVs and Charging Stations”. In: *2018 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2018*. Institute of Electrical and Electronics Engineers Inc., Sept. 2018. ISBN: 9781538655726. DOI: 10.1109/SSRR.2018.8468631.

[73] Samira Hayat et al. “Multi-objective UAV path planning for search and rescue”. In: *Proceedings - IEEE International Conference on Robotics and Automation*. Institute of Electrical and Electronics Engineers Inc., July 2017, pp. 5569–5574. ISBN: 9781509046331. DOI: 10.1109/ICRA.2017.7989656.

[74] Julian De Hoog, Stephen Cameron, and Arnoud Visser. “Dynamic team hierarchies in communication-limited multi-robot exploration”. In: *8th IEEE International Workshop on Safety, Security, and Rescue Robotics, SSRR-2010*. 2010. ISBN: 9781424488995. DOI: 10.1109/SSRR.2010.5981573.

[75] Ravi Kant and Abhishek Mishra. “The Orienteering Problem: A Review of Variants and Solution Approaches”. In: *Proceedings of the 26th World Multi-Conference on Systemics, Cybernetics and Informatics (WMSCI 2022)*. Orlando, FL, USA: International Institute of Informatics and Systemics (IIIS), 2022, pp. 41–46. ISBN: 978-1-950492-64-0. DOI: 10.54808/WMSCI2022.01.41.

[76] Gorka Kobeaga, María Merino, and Jose A. Lozano. “An efficient evolutionary algorithm for the orienteering problem”. In: *Computers and Operations Research* 90 (2018), pp. 42–59. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2017.09.003. URL: https://www.sciencedirect.com/science/article/pii/S0305054817302241.

[77] Félix Quinton, Christophe Grand, and Charles Lesire. “Market Approaches to the Multi-Robot Task Allocation Problem: a Survey”. In: *Journal of Intelligent & Robotic Systems* 107 (Feb. 2023). DOI: 10.1007/s10846-022-01803-0.

[78] Mohamed Badreldin, Ahmed Hussein, and Alaa Khamis. “A Comparative Study between Optimization and Market-Based Approaches to Multi-Robot Task Allocation”. In: *Advances in Artificial Intelligence* 2013 (Nov. 2013), pp. 1–11. ISSN: 1687-7470. DOI: 10.1155/2013/256524.

[79] Athanasios Tsalatsanis, Ali Yalcin, and Kimon. P. Valavanis. “Dynamic task allocation in cooperative robot teams”. In: *Robotica* 30.5 (2012), pp. 721–730. DOI: 10.1017/S0263574711000920.

[80] Ruofei Bai et al. “Hierarchical Multi-robot Strategies Synthesis and Optimization under Individual and Collaborative Temporal Logic Specifications”. In: (Oct. 2021). DOI: 10.1016/j.robot.2022.104085. URL: http://arxiv.org/abs/2110.11162%20http://dx.doi.org/10.1016/j.robot.2022.104085.

[81] Meng Guo et al. “Hierarchical Motion Planning under Probabilistic Temporal Tasks and Safe-Return Constraints”. In: (Feb. 2023). URL: http://arxiv.org/abs/2302.05242.