Active Risk Management in Dynamic Teams of Heterogeneous Robots

by

Soheil Changizi

A thesis submitted to
The Faculty of Graduate Studies of
The University of Manitoba
in partial fulfillment of the requirements
of the degree of

Master of Science

Department of Computer Science
The University of Manitoba
Winnipeg, Manitoba, Canada
August 2024

Thesis advisor Author

John E. Anderson **Soheil Changizi**

Active Risk Management in Dynamic Teams of Heterogeneous Robots

Abstract

In Urban Search and Rescue (USAR), hazards such as structural collapse and fire can significantly endanger robots. To mitigate these risks, it is crucial to plan task allocations that adapt to dynamic environments. Most available strategies begin with evaluating the mission in advance and formulating a static plan, which may be inflexible for unforeseen changes. This project seeks to expand our lab's existing framework by integrating Active Risk Management (ARM) into ongoing missions. The ARM module enhances adaptability by continuously monitoring environmental hazards and initiating risk mitigation tasks.

Additionally, a novel method for detecting and escaping local minima allows robots to adjust their navigation patterns, preventing immobilization. The simulation environment now features realistic fire propagation, introducing a dynamic element that rigorously tests the effectiveness of the robots' risk management strategies. By incorporating these advancements, we aim to increase mission success rates and significantly reduce robot damage in challenging USAR scenarios.

Contents

List of Figures

List of Tables

Acknowledgments

I am deeply grateful to my advisor for their guidance, my committee for their feedback, and my parents for their unwavering support. Thank you to everyone who has helped me along this journey.

This thesis is dedicated to my parents, whose love, support, and encouragement have been my foundation throughout this journey.

Chapter 1

Introduction

1.1 Chapter Overview

This chapter provides some context for my research and introduces concepts and terminology that will be used throughout my thesis. Following that, I discuss my motivation for undertaking this research and state my research questions. Finally, I present an overview of my solution strategy and an outline of the remainder of my thesis.

1.2 Introduction

Real-world disasters cause buildings to collapse, leaving behind an unstructured environment with victims surrounded by hazards. This creates a need for safer Urban Search and Rescue (USAR) to locate the victims and map the area. Many past disasters, such as the 2004 Mid-Niigata earthquakes in Japan, the 2005 Hurricanes

Katrina, Rita, and Wilma in the United States [Murphy et al., 2008], and terrorist attacks on the World Trade Center [Murphy, 2004], have brought attention to using robots in rescue operations.

In an unstructured disaster scene, hazardous materials and blocked paths make it harder for rescue teams to complete their tasks [Horsch, 2012]. USAR response time should be swift enough to locate as many hazard sources and human victims as possible without endangering rescuers and victims [Siciliano et al., 2008; Horsch, 2012]. Due to their capacity to enter potentially hazardous environments, robots are helpful in search missions for victims stranded in disaster zones [Casper and Murphy, 2003; Liu and Nejat, 2013]. During the catastrophic event in Great Eastern Japan in 2011, the caterpillar KOHGA3 robot was utilized to locate missing persons in disaster sites such as a damaged building and an underground mall where there was an increased possibility for collapse [Matsuno et al., 2014].

Robotic teams can assist humans in rescue operations by inspecting disaster zones on their own or under the control of a human operator. One of the advantages of utilizing robots in search and rescue teams is that robots do not become stressed out or fatigued [Burke et al., 2004; Casper and Murphy, 2003]. Robots may be manufactured in vast quantities and can be fixed or replaced if they suffer damage [Casper et al., 2000].

Utilizing multi-robot teams is one method for integrating robots into USAR operations. By expanding the number of robots, rescue operations can be accelerated while these teams provide greater flexibility through redundancy. Furthermore, robots are easily scalable, and damaged robots may be replaced, whereas the loss of a human being is irreparable [Burke et al., 2004]. Furthermore, heterogeneity allows us to distribute the necessary hardware across multiple robots, which is more cost-effective [Wong et al., 2011]. Due to the heterogeneity of the teams, various features can be distributed among team members to reduce the complexity of the robots. There are situations in which a single type of robot cannot operate due to environmental obstacles such as impassable terrain, contaminated water, or collapsing buildings [Surmann et al., 2021; Linder et al., 2010]. After an industrial fire in Berlin in February 2021, ground robots were not usable due to contaminated extinguishing water and collapsing buildings, so UAVs were used to map the area and search for missing people [Surmann et al., 2021]. The same problem arose after strong earthquakes in Italy in 2012 and 2016, so ground, air, and water robots were used to investigate building damage [Kruijff et al., 2014; Kruijff-Korbayová et al., 2016].

For many years, our lab has been conducting continuing research on dynamic teams of heterogeneous robots for USAR environments. This work has included changing roles and adding new members based on losses that occur throughout the operation [Gunn and Anderson, 2015; Nagy and Anderson, 2016]. The most recent modification to the framework permitted agents to refuse to join teams and work independently by evaluating their current skill satisfaction and skill rarity based on location [Fiawoo, 2019].

1.3 Motivation

Not only must rescue robots investigate the area and locate victims, but they must also identify the source of danger, assess risk, develop an informative map, and determine the safest route for first responders [Ashour et al., 2022]. In USAR missions, risk assessment is crucial. Risk assessment improves environmental awareness and prioritizes dangers for intervention. Risk assessments also determine who is at risk and whether more should be done to limit damage and accidents, especially if done early [Ashour et al., 2022].

Uncertain cases that may pose dangers are critical to team formation and task allocation in hazardous environments. Equipment failure is one such case; examples of equipment failure include the inability to traverse and becoming stuck, malfunctioning sensors, and difficulty communicating with other robots. These examples can compromise the mission and degrade the performance of the operation. By considering these circumstances, team leaders can limit risk by not assigning critical tasks to these robots or by eliminating them from the team. However, there is room for recovery, such as when a robot becomes stuck other robots can free it, or when the sensors are not functioning effectively, these robots might be utilized for other tasks, or guided to a base for repair. Also, in the event of communication failure, we can resend messages (if they include critical information) or cease sending messages to reduce the load on communication channels.

Any planning system must recognize that optimal mission performance necessitates some risk-taking. However, to minimize unnecessary losses, team leaders must be able to assess and limit the risk [Undurti and How, 2011]. Changes in the environment's terrain, such as fire propagation or building collapse causing voids, are another example of an unpredictable case. When team leaders are aware of the full extent of the dangers, they are more prepared to decide how to proceed with the operation, such as whether to ignore the risks and press on with locating the victims or dispatch capable robots to the problematic area and neutralize the hazard.

These cases may result in incorrect task or recruitment information. In these circumstances, insisting on performing the tasks could result in the robots being damaged or lost. For example, sending a robot to an area from which no robots have previously returned could jeopardize the remaining resources, but if team leaders have up-to-date data on risks in the area, they could devise a better plan. In dynamic and uncertain rescue scenarios, it is difficult to forecast the extent of risk; hence active risk management is essential.

1.4 Terminology

The following is contextual information regarding technical terms used in this thesis.

- *Frontier*—In the context of robotic exploration and mapping, a frontier is the boundary between the explored and unexplored areas in a map. Robots use frontiers to determine where to move next to efficiently explore and map unknown environments.

- *Hazardous Event*—A hazardous event is defined as a change in the environment detected by some robots that presents a threat. Some robots, such as FireBots, can detect whether or not a location is on fire.

- *Local Minima*—In robotic navigation, this term describes a situation where a robot gets stuck in a repetitive cycle or cannot progress due to constraints in its pathfinding algorithm.

- *Occupancy Grid*—An occupancy grid is a data structure used in robotics to represent the environment. It is a spatial map divided into a grid, where each cell represents the probability of the presence of an obstacle in the corresponding physical location. Robots can use occupancy grids to navigate and understand their surroundings, updating the grid as they explore and gather sensory information.

- *Role*—A role is a position in a team that a robot can fill, depending on the types of tasks that robots can do in that position [Gunn and Anderson, 2015]. For example, a robot taking on a firefighting role should be capable of putting out fires, navigating uneven terrain, and identifying possible victims as it moves [Nagy and Anderson, 2016].

- *Task*—A task is a basic unit of work that a single robot can complete [Gunn and Anderson, 2015]. Putting out a fire, cleaning up debris, freeing other robots that have become stuck, and identifying casualties are some examples of tasks.

- *Team*—A team is a group of robots with partial knowledge of the team's membership and abilities who share a common overall purpose. A single robot (called the team leader) directs team members' actions and assigns tasks to the other members, among other responsibilities. For single-robot teams, we consider the only team member to be the team leader. In this work, multiple teams are involved in a large environment and are working in a non-competitive manner to complete a common goal [Krieger et al., 2000; Pitonakova et al., 2014]. Team members in this work are not selfish [Dutta and Sen, 2003] and are required to do all responsibilities assigned to them provided their task queue is not full, or the task requires a skill the robot does not have. However, robots indirectly have an impact on accepting risky tasks by sending updated information to the team leader. In cases where the robot is incapable of doing a task due to its risks, the team leader will reconsider the assignment based on the updated status of the robot.

1.5 Research Questions

In this section, I present the research questions I will be answering using the approach I describe in Section 1.6

1. **To what degree do active risk management strategies impact the overall mission performance and safety of heterogeneous robot teams in dangerous domains?**

2. **How effectively can we balance active risk management with other mission objectives (maximizing explored areas and finding victims) in teams of heterogeneous robots operating in dangerous domains?**

3. **What are the potential challenges and limitations of sacrificing resources or recovering damaged/lost robots in hazardous domains and how can risk management strategies be adapted accordingly?**

1.6 Solution Approach

This research aims to enhance the existing dynamic team formation framework by incorporating dynamic risk assessment of robot actions. The proposed extension enables robots to identify and avoid potential hazards resulting in damage or loss before continuing their tasks or modifying their team formation for recovery. Like Gunn and Anderson's framework, communication channels between robots are only sometimes reliable and can be sporadic. Additionally, the robots' sensors may provide inaccurate data due to damage, resulting in incorrect information being shared with the team. The research also includes the integration of voids, debris, and fire propagation throughout the mission to generate a more dynamic and challenging environment.

In the context of Urban Search and Rescue (USAR) environments, risk assessment is a necessary process that involves identifying, analyzing, evaluating, and controlling potential hazards [Government of Canada, 2023]. This process is essential in ensuring the safety of rescue workers and maximizing the chances of success in rescue operations.In situations with multiple risks, it can be challenging to determine the most severe and urgent risks that require immediate attention. To overcome this challenge, ranking or prioritizing the hazards can help determine the level of risk severity and the appropriate control measures that should be in place [Government of Canada, 2023].

Because it is not always possible to fully predict the risks associated with a given task, it is not feasible to develop a fixed plan. Sometimes a specific strategy is not practicable due to limited resources and can result in a lower-quality operation. Additionally, changes in team formation and resources may lead to environmental changes, such as unsafe voids created by the collapse of structures. Ignoring these risks can lead to suboptimal performance or catastrophic loss. In active risk management, various factors must be considered to evaluate the risk associated with the environment or task. Majumdar and Pavone [2017] have examined several risk metrics and established mathematical constraints to ensure they are reasonable risk measures.

Once the risk estimation stage is complete, the team leader must reconsider which robots should be dispatched, provided appropriate robot(s) are available. Additionally, other agents must devise a plan for the next task, which requires revising the framework for task assignment and team formation. It is essential to note that it is only possible to understand some unsafe scenarios fully, and there are many ways for a team to deviate from its desired formation. For instance, an unforeseeable event could separate robots from their team, placing them in a hazardous situation that has not been previously encountered. This unpredictability presents an opportunity for a learning framework, but this is beyond the scope of this thesis. Nonetheless, revising the task assignment and team formation framework can help address unforeseen circumstances and enable the team to respond effectively to new risks or challenges. By adapting to the situation, the team can mitigate risks, increase efficiency, and improve overall performance.

Actively assessing and managing risks associated with tasks or the environment can help teams optimize their plans dynamically and avoid costly mistakes. These teams can dynamically optimize their plans by actively evaluating the risks associated with tasks and the environment. We must supplement the current rescue robot framework with active risk management to realize these objectives.

1.7 Thesis Organisation

The remainder of this thesis is organized as follows:

- **Chapter 2: Related Work**—Looks into the existing literature relevant to my work in terms of risk metrics, risk detection, and prevention, as well as tolerance and recovery from hazards.

- **Chapter 3: Methodology**—This section discusses key elements of my active risk management framework: the Risk Manager, Local Minima Escape algorithms, and the Fire Manager module, which changes how fire is simulated in environments.

- **Chapter 4: Implementation**—Offers an in-depth overview of the components (robots, skills, and tasks) implemented for the evaluation of my framework.

- **Chapter 5: Testing and Evaluation**—This section outlines the experiments performed to evaluate my framework and discusses the resulting outcomes.

- **Chapter 6: Conclusion**—Using the findings from Chapter 5, I contextualize the research questions introduced in Section 1.5 within my thesis and explore future work directions.

Chapter 2

Related Work

2.1 Chapter Overview

The primary objective of this thesis is to develop a framework that enables robots to actively manage risks in dynamic teams of heterogeneous robots, particularly in hazardous environments. This chapter provides a comprehensive overview of the related background in risk management strategies and the potential complications that may arise with multiple robots (e.g., a robot encountering unforeseen obstacles or hazards). I have organized previous work into three categories.

The first category focuses on fault tolerance in multi-robot systems, a crucial aspect of active risk management. These studies explore various methods for detecting, handling, and recovering from faults in robot teams, ensuring their continued operation in the face of trouble. The second category encompasses metric-based risk management in multi-robot systems, which examines how robots can quantify and assess potential risks in their environment to make informed decisions, thereby improving the overall safety and efficiency of the team. The final category covers prior research from our lab for multi-robot systems, which have focused on team formation and task allocation in dangerous domains using a dynamic framework, active recruiting strategy, and robot satisfaction.

This chapter will delve into each of these categories, presenting the key concepts, methods, and findings from previous research and highlighting the contributions and limitations of each. By examining these related works, I aim to establish the foundation for the proposed framework and demonstrate the importance of active risk management in dynamic teams of heterogeneous robots.

2.2 Fault Tolerance in Multi-Robot Systems

In multi-robot systems, fault tolerance is necessary, especially in dynamic and unpredictable environments such as Urban Search and Rescue (USAR) missions. The ability of these systems to adapt and respond to unexpected situations, like quenching fires or debris removal, is essential for maintaining their overall performance and reliability while effectively managing and mitigating risks.

Parker's ALLIANCE is a fully distributed, behavior-based architecture designed for cooperative robot teams, utilizing mathematically modeled motivations like impatience and acquiescence for adaptive action selection [Parker, 1998]. The architecture seeks to enhance fault tolerance, adaptivity, and reliability in dynamic, unpredictable environments by enabling individual robots to respond effectively to unexpected changes. However, there are notable limitations of ALLIANCE as an active risk management framework. ALLIANCE does not actively identify or prevent risks, as its primary focus is improving the robotic team's fault tolerance and adaptivity. The architecture's ability to dynamically reallocate tasks in response to robot failures can mitigate the consequences but may not effectively prevent risks from arising. Additionally, ALLIANCE lacks centralized control due to its fully distributed nature, meaning no central agent can monitor the robot environment and make decisions based on the gathered information. There must be some control that is not entirely local to ensure risk coordination and management across the robot team.

Corrêa [2014] introduces an algorithm that generates partitions of agents based on their features to optimize coordination. This approach employs a belief propagation algorithm to achieve near-optimal solutions by utilizing a tree-structured factor graph with agents as variable nodes and groups as factor nodes [Corrêa, 2014]. Although this approach, akin to the ALLIANCE architecture, may passively minimize overall damage, it primarily focuses on improving coordination and does not actively identify or prevent the causes of risks. Although the approach can help detect conflicts or coordination challenges, its effectiveness depends on accurately assessing agent capabilities and task requirements, which may only sometimes capture all relevant factors of risk in a USAR situation.

Cui et al. [2014] proposed ReFrESH, an augmentation of the conventional task assignment framework featuring decider and generator components that facilitate dynamic reconfiguration through systemic self-diagnosis and self-maintenance. The decider gathers performance evaluations of running components, and if any task executor deviates from the intended measures, it initiates the generator. The generator then performs a dependency analysis to generate potential configurations, and the decider estimates the performance of candidates to determine an optimal configuration. Despite focusing on non-functional requirements like resource consumption and performance, ReFrESH lacks explicit consideration of risk factors in its configurations. This limited risk awareness may prove challenging in dynamic and unpredictable USAR environments, where robots must rapidly adapt their roles and interactions to various risk scenarios. Since ReFrESH relies heavily on predefined component dependencies and roles for configuration generation, its flexibility and adaptability in such environments may be limited.

Tihanyi et al. [2021] developed a scalable two-stage framework for multi-robot safe mission planning in uncertain dynamic environments by breaking down the control problem into low-level safe planning for each robot and a distributed auction-based task allocation. Each robot uses Monte-Carlo sampling to plan under dynamic uncertainties, allowing the computation of the multiplicative safety objective for the high-level task allocation [Tihanyi et al., 2021]. While the two-stage framework addresses scalability issues, this approach may not be sufficient for active risk management in real-world situations. Single-robot planning could degrade overall mission performance, as we need to account for the effects of other robots on the environment. Furthermore, the framework relies on heuristics that might not perform well under risk management in practice.

2.3 Metric-based Risk Management

This section will explore the importance of defining a feasible risk metric to assess risks and make informed decisions in dynamic and unpredictable environments. As the outcomes in these environments are often uncertain, developing a stochastic risk metric that accounts for the inherent variability in agents' actions and environmental changes is essential, enabling more effective risk management and decision making.

Léauté and Faltings present the StochDCOP, which uses random variables with known probability distributions to model uncertainty. It aims to compute an assignment to decision variables that maximize the expected utility based on risk and overall mission performance [Léauté and Faltings, 2009]. Their main contribution is collaborative sampling, a distributed procedure allowing agents to agree on a common sample set for each random variable, leading to new algorithms that solve StochDCOPs. While StochDCOP is helpful for uncertain decision-making optimization, it lacks the comprehensive risk monitoring and mitigation required for active risk management. Furthermore, StochDCOP presumes that probability distributions for uncertain variables are known or can be accurately estimated, which may not always hold in practice. Thus, StochDCOP, though valuable in some instances, should be integrated with other tools and methods for comprehensive active risk management.

Minimizing the underlying distribution's worst-case cost is an alternative to improve risk sensitivity. However, this approach tends to yield overly strict solutions in practical applications. [Majumdar and Pavone, 2017]. Given that the expected cost is indicative of risk neutrality, and the worst-case scenario corresponds to extreme risk aversion, exploring risk metrics between these two extremes is more practical, ultimately achieving a balance between risk tolerance and risk aversion [Majumdar and Pavone, 2017].

Ponda et al. [2012] have followed a distributed chance-constrained task allocation framework that offers a strategy for planning and coordinating multi-agent teams in stochastic and dynamic settings. The framework actively manages uncertainty propagation during task allocation using a global risk threshold by approximating centralized problems into distributable sub-problems for individual agents [Ponda et al., 2012]. However, the approach's reliance on linearity assumptions and specific homogeneous independent and identically distributed distributions restricts its applicability in more general and realistic mission scenarios, where system models and parameters may exhibit greater variability and uncertainty. Furthermore, the risk metric used may not always align with mission objectives or individual agent goals, possibly leading to suboptimal task allocation decisions or team disagreements, ultimately increasing overall mission risk.

Later, Ponda et al. [2013]'s revised framework offered a more structured approach to risk allocation in distributed chance-constrained planning algorithms. The framework enhances risk allocation for both homogeneous and heterogeneous agent teams by utilizing domain knowledge, grouping agents with the same characteristics, and assigning different risk allocations to each group, thus addressing the previous framework's limited applicability [Ponda et al., 2013]. However, the robustness of this approach for highly heterogeneous teams or high reward variance remains to be explored. Despite these improvements, the updated framework still faces potential misalignment issues, where the risk metric may need to align with mission objectives/individual agents consistently. Moreover, the framework's performance is heavily influenced by the choice of stochastic parameters and features, affecting risk allocation and overall mission score/performance.

Conditional-Value-at-Risk (CVaR) [Rockafellar and Uryasev, 2000] is a valuable metric for active risk management by capturing the expected cost in worst-case scenarios based on a specific λ-portion of the tail distribution. When λ is set to one, CVaR corresponds to the mean value of the distribution, reflecting the average cost. On the other hand, as λ approaches zero, CVaR emphasizes the worst-case cost, accounting for the most adverse and unfavorable outcomes in the distribution. By considering various λ values, CVaR enables organizations to manage risk more effectively, evaluating potential costs across various scenarios and striking a balance between risk tolerance and risk aversion.

Nam and Shell [2017] integrate risk preferences into multi-robot task allocation by using a bi-objective cost function, where the coefficient α balances the trade-off between mean costs (μ_{ij}) and risk-adjusted costs represented by Conditional Value-at-Risk (CVaR, $CVaR_{\lambda,ij}$). The parameter λ indicates the confidence level at which the CVaR is calculated, capturing the expected costs in worst-case scenarios beyond a certain probability threshold. The binary decision variable x_{ij} represents whether task j is assigned to robot i (1 if assigned, 0 if not). This approach minimizes expected costs while also considering the risk preferences of the team, as described by equation (2.1). The framework's sensitivity analysis on α allows the team to understand how varying the emphasis on mean cost versus risk impacts the task allocation, leading to consistent optimal assignments [Nam and Shell, 2017].

$$min \sum_{i=1}^{n} \sum_{j=1}^{n} (\alpha \mu_{ij} x_{ij} + (1 - \alpha) CVaR_{\lambda,ij} x_{ij}) \tag{2.1}$$

However, the framework assumes stationary risks and does not account for dynamic changes over time. In numerous real-world situations, risks can evolve rapidly, necessitating an adaptive risk assessment and management process. The challenge of determining suitable α and λ values in dynamic environments arises from rapidly changing risk factors. Accurate values require continuous adaptation based on real-time information, which complicates the approach's effectiveness as an active risk management tool and may impact overall performance in changing conditions.

Zhou and Tokekar [2020]'s approach addresses risk in decision-making under uncertainty by utilizing CVaR in a discrete submodular maximization problem. Using an auxiliary function and parameter τ, they calculate CVaR by sampling the environment and iterating over τ values in $[0, \Gamma]$. This is achieved through the Sequential Greedy Algorithm, which provides an approximation guarantee for the CVaR cost function under a matroid constraint. As for online settings, the Online Triggering Assignment (OTA) algorithm updates assignments with parameter γ for rescheduling frequency [Zhou and Tokekar, 2020].

Zhou and Tokekar's approach faces several challenges that limit its effectiveness as an active risk management framework. The difficulty in identifying ideal τ values and the OTA algorithm's reliance on a user-defined triggering ratio restrict the method's adaptability to various situations and may result in suboptimal solutions. Moreover, the approach assumes specific properties for the utility function, which might not always be true in real-world scenarios, limiting its practical applicability. The potential inaccuracies in approximating CVaR can also contribute to inefficiencies, highlighting the limitations in managing risk actively.

2.4 Existing Framework

2.4.1 Gunn Framework

In this section, I will provide an overview of the key components of the original Gunn and Anderson [2015] framework that are essential to my research.

Attributes, Tasks, and Roles

The framework uses a set of characteristics and capabilities called *attributes* to define a robot's abilities to carry out a specific task. For instance, in a victim identification task, a robot needs a victim identifier sensor to distinguish between an actual victim and a false one (objects in the environment or patterns of debris that might appear humanoid in form, but are not victims). *Tasks* refer to a specific unit of work that a single robot can complete. *Complex tasks* can be broken down into simpler sub-tasks, possibly to be carried out by different robots. For example, the complex task of victim identification can be broken down into identifying a potential victim, confirming that the victim is an actual person, and leaving a fiducial mark next to the victim for the rescue team. These sub-tasks can be assigned to different robots based on their unique abilities and attributes.

Robots need specific attributes to complete the tasks that they are assigned. The *minimum requirements* expressions specify the minimum attributes a robot should have to successfully carry out a specific task. For example, a robot without a victim identifier sensor cannot perform a victim identification task. *Suitability expressions* assign weights to the attributes specified by a task's minimum requirements to evaluate a robot's suitability for a given task. A robot with a higher suitability value is more suitable for the task. Using suitability expressions, teams can efficiently allocate tasks based on each robot's unique abilities, ultimately enabling them to work together more effectively.

Robots can fill positions within a team, known as *roles*, which are characterized by a set of tasks that a robot in that role must perform. However, robots may sometimes occupy a role for which they are not well-suited due to the changing and dynamic nature of the environment or the potential for damage. These situations may arise due to communication difficulties or skill deficiencies within the team, or a damaged robot. Additionally, a robot may be assigned tasks unrelated to its role when no other robot can perform that task.

Each task within a role is associated with a weight that determines its importance within the context of the occupied role. For a robot to be deemed suitable for a role, it must fulfill the minimum suitability requirements of all tasks in that role. Additionally, each role includes an *idle task* that a robot engages in when its task queue is empty, preventing downtime and maintaining overall team efficiency.

Team Management and Maintenance

A *desired team* is critical in USAR missions, where robots face various challenges and obstacles. In the context of the Gunn framework, the desired team as shown in Figure 2.1, serves as a yardstick for determining how to structure or integrate new potential members into an existing team's structure. However, due to the ever-changing nature of USAR missions and the risks posed to robots operating within the environ-

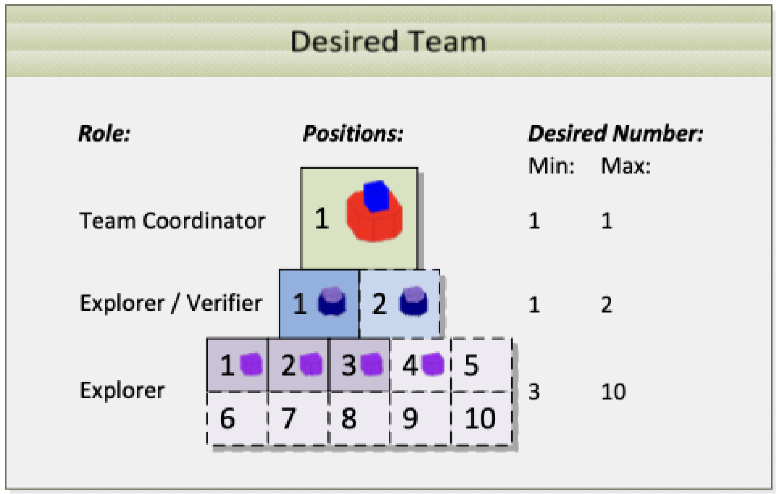

Figure 2.1: A sample desired team definition with three robot roles. Adapted from Gunn and Anderson [2015]

ment, a team may only sometimes perfectly match the desired team requirements at the beginning of a mission. Therefore, it is crucial for teams to regularly assess their structure and roles to ensure they remain effective and capable of handling the tasks they may encounter.

Robots within a team regularly broadcast their location to other team members to keep them updated on their current membership and structure. If a robot is not heard from after a certain period, the team considers it temporarily lost. However, unreliable wireless communication can sometimes cause robots to miss broadcast messages, leading to incorrect assumptions about lost or damaged team members. Such assumptions can result in suboptimal role assignments or redundant roles within the team [Gunn and Anderson, 2015]. Teams can work to rectify this situation as better information becomes available.

In scenarios where robots are lost due to damage or an inability to move in the environment, replacement robots may need to be sent in to fill vacant roles. Rebalancing roles within a team in response to such scenarios can be challenging, but having desired team definition in place can help to guide this process and ensure that the team maintains a balance of skills and roles necessary for success, reducing the risks associated with lost or damaged robots.

The Gunn framework utilizes occasional *role switch checks* to maintain an effective team structure, where team members evaluate their current role and compare it to the current team structure. During a role switch check, a robot creates a list of underfilled roles and ranks them according to their role score value, which is determined by the robot's suitability for that role. If a higher-scoring role exists, the robot will switch to that role and notify its team members. This approach allows for team rebalancing to ensure that team members occupy the roles that best suit their abilities. However, when a team has lost many members and needs to recover lost skills, the framework relies on chance encounters with other teams or stray robots since role-switching alone is insufficient.

Encountering another team provides an opportunity for two teams to rebalance their membership and regain lost skills through a process called *passive recruitment*. Robots exchange information about their team structures at the start of the encounter, and if the more computationally-capable robot is available, it performs the team merges and redistribution process. The selected robot then determines how to distribute robots between the teams to get as close as possible to the desired team definition. In the end, either both teams come closer to meeting the desired team definition, or one team is formed by integrating members of both teams. Once a suitable distribution is determined, the more capable robot communicates this to the other robot, and both robots instruct their teammates to change roles or teams as necessary.

Task Management

The framework employs a task management system in which robots are accountable for identifying/executing tasks based on their skills [Gunn and Anderson, 2015]. The task priority is determined based on the urgency and importance of each task. Each robot maintains a task queue that lists the pending tasks, ordered by their priority and the time they were added. After completing a task, a robot removes it from the queue and selects the next highest-priority task for execution. Robots can discover tasks by themselves or have them assigned by their team leader. The priority queue guarantees that the most critical tasks are prioritized. However, if a robot finds a task with higher priority while executing another, it will stop its current task and switch to the more critical task. If a robot cannot insert a newly discovered task into its queue, it alerts the team leader, who then assigns the task to a suitable robot using either role-based or exhaustive task assignment.

In *role-based task assignment* [Gunn and Anderson, 2015] (Figure 2.2), team leaders use their understanding of the roles involved to identify potential robots to perform the task, minimizing the effort needed to assign tasks. The leader sends a message to all identified team members simultaneously, asking them to complete the task. Upon receiving this request, each robot considers its current workload and the type of task to determine if it is capable of completing the task. If it is, the robot responds to the

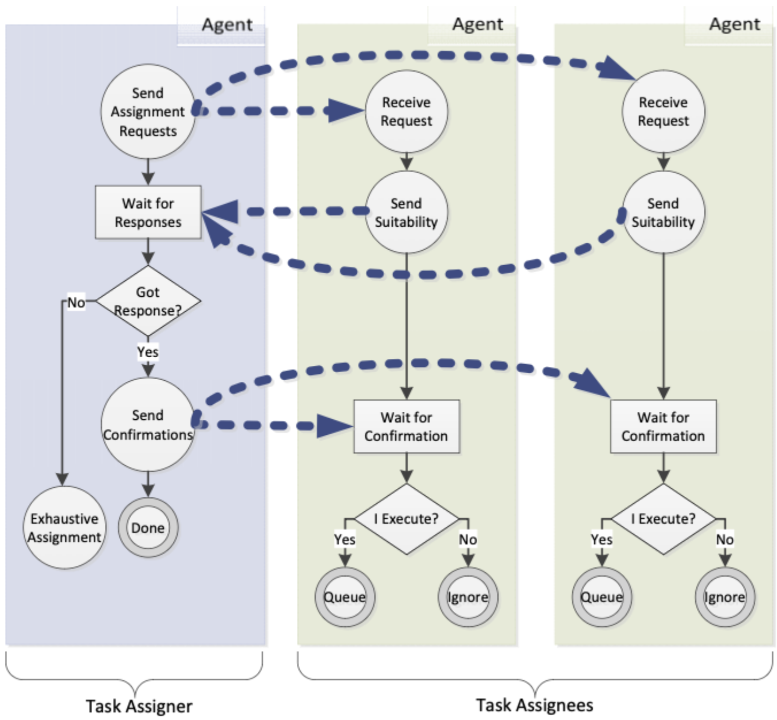

Figure 2.2: The process of role-based task assignment. Adapted from Gunn and Anderson [2015]

leader with information about its suitability for the task and the cost involved. If a robot cannot execute the task due to a full task queue or unsuitability, it sends a rejection response. The leader sets a time frame for receiving responses, and responses received outside this window are ignored. Responses received within the window are evaluated, and the robot with the highest suitability and lowest cost is chosen to perform the task. The chosen robot acknowledges its acceptance of the task and adds it to its queue while the leader deletes it. However, the leader switches to exhaustive task assignment if no robot is found suitable within the given time frame.

Exhaustive task assignment [Gunn and Anderson, 2015] is used as a last resort when role-based task assignment fails, as it requires more computational resources and communication compared to the former method. To use exhaustive task assignment, the team leader sends a wireless message that all team members can receive. Each robot then responds to the leader with its suitability and cost for the task, and the leader assigns the task to the robot with the highest suitability. However, if there are no suitable robots for the task, the leader returns the task to the queue for future reassignment. This process continues until a suitable robot is found or until the task is no longer critical.

Schemas

Gunn framework applies a *schema-based* method [Gunn and Anderson, 2015] to interpret raw data gathered from the environment by robots, determining the appropriate set of actions for them to take. The framework employs perceptual schemas to perceive the surroundings, and the collected data is transferred to motor schemas, which determine the action sequences. Motor schemas enable robots to display intricate behaviors by merging diverse motion vectors. For instance, if an obstacle blocks the path between a robot and a target location, the robot generates a repulsive vector away from the obstacle and an attractive vector towards the goal after receiving data from the perceptual schema. The summation of these two vectors yields a vector that guides the robot around the obstacle towards the goal. I will use the perceptual framework's existing schemas to collect data for my methodology in risk approximation.

2.4.2 Active Recruitment Controller (ARC) Framework

The following section will delve into the ARC framework, an extension of Gunn's previous work. Notably, ARC incorporates a proactive recruitment mechanism that facilitates robot teams' acquisition of new skills, either permanent at the role level or temporary at the task level [Nagy and Anderson, 2016].

Recruitment Spectrum

The requirement spectrum in the ARC framework expands upon the existing passive requirement from Gunn's framework by introducing two additional strategies. It provides a comprehensive view of the different approaches employed in recruitment. These approaches include the *passive* strategy, which was already present in the previous framework and relies on chance encounters for recruitment. Additionally, the ARC framework introduces the *concurrent* strategy [Nagy and Anderson, 2016], where robots recruit while performing other tasks, and the *active* strategy [Nagy and Anderson, 2016], where robots prioritize recruitment over other ongoing tasks. The spectrum offers a nuanced understanding of these three approaches, showcasing the range of effort and priorities robots can employ when forming teams.

Concurrent recruitment in the ARC framework allows robots to actively search for potential recruits while simultaneously carrying out their regular tasks, maximizing efficiency, and capitalizing on opportunities. On the other hand, active recruitment represents a more aggressive approach where robots prioritize recruitment over all other tasks, temporarily suspending additional duties to focus solely on finding and integrating valuable robots. While concurrent recruitment balances recruitment efforts and ongoing tasks, active recruitment demonstrates a dedicated commitment to quickly building a specialized team. The choice between these strategies depends on the specific context and objectives, allowing for tailored approaches that optimize the trade-offs between recruitment efficiency and task execution.

Recruitment Strategies

Task-level recruitment and role-level recruitment are two distinct strategies within the ARC framework that facilitate the acquisition of robots with specific skills and roles, respectively [Nagy and Anderson, 2016]. Task-level recruitment allows for flexible and temporary integration of robots outside the current team, enabling efficient task execution and resource allocation. In contrast, role-level recruitment aims to fill essential roles within the team structure permanently. It involves a more exhaustive search for robots with the required skills and attributes to occupy specific roles identified as under-filled by the team leader. Role-level recruitment ensures the long-term composition and functionality of the team, aligning with the concept of an ideal team.

Within the ARC framework, the team leader is in charge of assigning role-level recruitment tasks. By conducting periodic checks for missing roles, the team leader identifies positions that are either unoccupied or inadequately filled within the team structure. After identifying a critical under-filled role, the team leader designates a team member responsible for carrying out the role-level recruitment task. This team member is entrusted with locating and recruiting a suitable robot possessing the necessary skills and qualifications to fill the identified role. Table 2.1 presents a comprehensive overview of these strategies and their respective applications.

Table 2.1: Recruitment strategies employed in fulfilling role or task requirements. Adapted from Nagy and Anderson [2016]

	Passive	Concurrent	Active
Task-level	wait for robot to join team by chance	while performing regular work, look for a robot to complete a task	halt regular work and search for a robot to complete a task
Role-level	wait for robot to join team by chance	while performing regular work, look for a robot to fill a role	halt regular work and search for a robot to fill a role

Marker Manager

The ARC framework introduced markers as a means for robots to track which potential victims have already been encountered, preventing redundant victim confirmation tasks. When a robot discovers a potential victim, it drops a marker near the victim to indicate that it is being confirmed or has already been confirmed by another robot [Nagy and Anderson, 2016]. However, that framework is not able to differentiate between genuine victims and false positives once they have been confirmed. To manage this marker placement and sharing process, the marker manager is responsible for overseeing the placement of markers by robots near victims and facilitating the exchange of markers among robots without markers.

2.4.3 Fiawoo Framework

In this section, I examine Fiawoo's framework, our latest lab work, which improves team performance in hazardous environments by considering satisfaction, skill rarity, and task locale. The framework enables robots to estimate their utility and make informed decisions for effective team dynamics and task allocation. By evaluating robot satisfaction, this research advances robotics in dangerous domains and achieves team goals [Fiawoo, 2019].

Robot Satisfaction

Robot satisfaction is crucial in evaluating a robot's utility within different team settings. It addresses the need for a robot to assess whether it would be more useful given it's skill set on another team or working independently, as well as the team leader's consideration of a robot's contribution when deciding on coordination efforts [Fiawoo, 2019]. The frameworks prior to this work had limitations in measuring a robot's usefulness and relied on idle tasks that only partially utilized its unique skills. The approach of robot satisfaction encompasses both the individual's and the team leader's satisfaction, using a satisfaction expression to measure their satisfaction with the robot's contribution to the team's goals.

Calculating individual satisfaction involves weighing tasks based on rarity, future satisfaction from upcoming tasks, and considering the number of rejected requests. Satisfaction values are periodically updated, triggered by events like being idle or receiving recruitment requests. Agents with low satisfaction aim to improve it by accepting recruitment to teams in which they might be more satisfied, while those near the base level consider role switches or task-based opportunities. Continually idle agents risk being dropped if their coordination requirements outweigh their temporary usefulness. When expressing a desire to leave, robots are evaluated by team leaders to decide whether to let them go or encourage them to stay [Fiawoo, 2019].

Robots assess their satisfaction and determine willingness states (not willing, somewhat willing, and willing) to guide their actions. Robots in the not willing state may be available for recruitment by other teams if their skills are rare. Those with high satisfaction continue contributing to their current team. In the somewhat willing state, robots make moderate efforts through task-level recruitments or occasional role switches. With satisfaction below the base level, willing robots take a more aggressive approach by accepting role-based recruitments, moving to other teams, or working independently. Team leaders assess robots expressing willingness to leave and decide accordingly [Fiawoo, 2019].

Team leaders not only track individual satisfaction but also closely monitor team members' work completion and satisfaction. While a team member's satisfaction may not align with the leader's contentment regarding the work performed, leaders evaluate members by counting their completed tasks. If a member consistently rejects task assignments, indicating inadequate usefulness, the leader requests their willingness state. In the somewhat willing state, members are instructed to become willing, or may eventually leave the team [Fiawoo, 2019]. However, further action is only taken if a member responds as willing. This process enables leaders to identify under-performing members and manage team dynamics effectively.

Rare Skills in the Environment

In the environment, rare skills are valuable for agents during rescue missions. Recognizing their possession of a currently rare skill, agents respond positively to skill-specific recruitment mechanisms. Tracking the number of recruitment requests from individuals and teams helps estimate skill availability. When agents receive task requests unrelated to their rare skill, they can mark themselves as unwilling to accept those tasks. This way, they can remain available to all teams until their skill is no longer rare (or tasks change so that skill is less needed). Once their skill becomes common or less desirable, they may try to rejoin their original team or explore other opportunities. These dynamics contribute to the adaptability of robot teams in complex missions. There is also the notion of unique skills crucial for specialized tasks requiring specific robot types [Fiawoo, 2019]. Agents determine their unique skills' rarity by counting requests from different robots. Coordinating robots with unique skills enhances overall team performance, emphasizing the importance of specialization and cooperation in achieving successful outcomes.

Robots' Sense of Locality

A robot's sense of locality is crucial for decision-making and mission satisfaction. By tracking tasks and their locations, robots can determine whether working with a team or independently in a specific location will provide greater satisfaction. If an agent with specialized skills experiences low satisfaction, it can accept recruitment requests tied to that location, thereby boosting satisfaction [Fiawoo, 2019]. This location-based approach aligns with the satisfaction mechanism of assessing independence versus team membership. As an agent remains at a location for an extended period, it may separate from its original team when they move on, yet maintain satisfaction by performing independent tasks in the local area.

Suppose the skill associated with those tasks becomes less rare or opportunities diminish. In that case, the agent's satisfaction may decrease, making it more receptive to alternative requests or prompting it to wander and explore other productive areas. Furthermore, if the agent's skill becomes less tied to a specific location, the robot will shift its focus toward other team-based activities [Fiawoo, 2019]. By dividing the environment into task locales and keeping track of location satisfaction, robots can strategically choose regions with the highest satisfaction and revisit areas where they have previously completed useful work. This approach ensures that robots explore unexplored regions and balances productivity and satisfaction.

2.5 Conclusion

This chapter provided an overview of the relevant literature and established the foundation for my work's subsequent discussion. I have identified and incorporated key modules from the existing framework that are integral to my implementation. Building upon this foundation, I will now introduce the novel concepts and additions in Chapter 3.

Chapter 3

Methodology

3.1 Chapter Overview

This chapter outlines the methodologies to advance our lab's existing framework, focusing on integrating active risk management. An overview of the foundational aspects of the framework and the enhancements by Nagy and Fiawoo is provided in section 2.4 of the previous chapter.

3.2 My Framework

My framework adopts a dynamic and collaborative approach to risk management, departing from traditional pre-planned strategies. At its core is the Risk Manager module, which dynamically reallocates tasks based on risks identified by leader and non-leader robots. Non-leader robots can assign risk mitigation tasks upon detecting a risk event, as outlined in Section 2.4.2. However, they also communicate these risks to the team leader for a comprehensive task management response. This dual approach of risk communication and task assignment significantly enhances the adaptability and efficiency of robotic teams, especially in unpredictable and challenging scenarios.

Complementing this is a suite of algorithms designed for escaping local minima, which are crucial in navigating complex terrains and avoiding repetitive motion traps. This aspect of the framework is essential for ensuring that robots maintain operational effectiveness, even under fluctuating conditions, and contributes significantly to the potential of these systems in future disaster response operations.

Additionally, integrating the Fire Manager module into my simulation framework adds realism by simulating the unpredictable spread of fire in USAR scenarios. This module not only tests the robustness of the Risk Manager but also challenges the robots to adapt to rapid and unforeseen environmental changes. Including this module is a critical step in elevating the practical applicability and realism of the simulation, preparing robotic teams for real-world rescue missions where adaptability and quick decision-making are critical.

3.2.1 Risk Manager

A robotic team leader must evaluate environmental risks to optimally allocate tasks for both safety and efficiency in the team's collective operations. This section introduces the Risk Manager module, a component designed to calculate and manage the environmental risks encountered by the teams. The module enhances environmental awareness and optimizes task distribution by systematically quantifying and categorizing risks, such as quenching fire, debris removal, and freeing immobilized robots within defined grids.

For instance, imagine a scenario where a robot within the team encounters debris and detects it using its sensors. The robot can independently decide to address this hazard by assigning a risk mitigation task to itself (if it can mitigate the risk) or other capable robots based on the recruitment strategy. Simultaneously, it informs the team leader of the debris's location. Upon receiving this report, the leader assesses the risk associated with the debris. If the robot has not initiated action, the leader can formulate an appropriate task to address the hazard based on a set priority system. This collaborative and dynamic process of risk assessment and mitigation, involving both the leader and the team members, will be explained in more detail in later sections.

In our lab's previous research, we defined 'satisfaction' as the level of utility that a robot experiences while performing tasks within a given environment [Fiawoo, 2019]. Satisfaction is linked to the completion of tasks, regardless of whether they were assigned or discovered, with different satisfaction values assigned to different tasks. In this thesis, I apply a similar approach to track risk across predefined grids of the environment. However, the methodology mainly focuses on quantifying the environment into grids without directly correlating risk and the satisfaction metrics previously used.

The risk management module marks a significant departure from previous frameworks that relied on static task allocations, often leading to inefficiencies in rapidly changing disaster scenarios. The Risk Manager module ensures that robotic teams can adapt to evolving challenges, mirroring the concept of robot satisfaction by aligning each robot's actions with the collective goal of navigating and succeeding in hazardous environments safely and efficiently. Its dual functionality empowers individual robots to identify and communicate risks effectively and equips team leaders with the necessary information to make strategic decisions.

Risk Features

Risk management within the USAR domain requires a responsive and sophisticated approach to identify, classify, and act upon various risk events. Within my framework, a risk event refers to any situation that demands immediate action, including extinguishing a fire, clearing away debris, or freeing a stuck robot. These events can either be reported by team members to the team leader or logged by the robot, thus leading to the generation of suitable tasks to address the reported risks.

Risk events are captured as data points with comprehensive features to inform the decision-making process. These features, known as risk features, include:

- **RiskType:** Identifies the nature of the risk, which can be fire quenching, debris removal, or freeing a stuck robot a robot. This categorization is pivotal in determining the specific response required.

- **Coordinates (x, y):** The location of the risk event is recorded with precise coordinates, enabling the Risk Manager to pinpoint the exact area of operation.

- **Angle (a):** The angle at which the robot is immobilized provides important information to the robot, allowing it to plan the free operation effectively. With this feature, the robot can determine the direction in which the stuck robot is immobilized, making it easier to free it. This information comes from the stuck robot though the communication channels.

- **Robot ID:** For events requiring freeing a stuck robot, the immobilized robot's identifier is logged, allowing for targeted assistance.
- **Time of Recording:** The timestamp of when the risk was reported is crucial for understanding the urgency and tracking the event's evolution over time.
- **Environmental Features:** These include the number of high obstacles, low obstacles, potential victims, and other robots around the risk, providing a detailed situational context. Recording environmental features is essential to measure the clutteredness of an area, helping the leader decide whether it is feasible to send a robot to that specific location.

These features are vital in risk mitigation prioritization, a process detailed in the next section. The framework employs a system of communication and record-keeping to ensure an efficient and coherent risk management process. Robots and team leaders share risk reports, and acknowledgments are sent to confirm receipt. This exchange prevents the duplication of reports and ensures that all team members understand the current risks.

Risks reported within close temporal and spatial proximity are considered as a single event, to streamline the response. The grid-based layout of the environment aids in clustering these events and assigning tasks strategically. The Risk Manager module oversees this operation, dynamically resetting risk records after a set interval to maintain an up-to-date risk profile. This flexibility allows the framework to adapt to the fluid and evolving nature of USAR scenarios, where risks can change rapidly and unexpectedly.

Algorithm for Mitigation Prioritization

A critical aspect of the Risk Manager module in the USAR domain involves using an algorithmic approach to prioritize which risks should be mitigated first. The framework utilizes a greedy algorithm designed to quickly and efficiently evaluate the aggregated risk features within each grid of the operational environment.

The algorithm analyzes the risk data gathered in each 5m x 5m grid. Critical parameters for ranking risks include the average timestamps of risks present in each grid, which capture the staleness of information, favoring the most recent risks. The search for risks to mitigate is based on a threshold distance from the robot assigned to the task, omitting risks that are farther away. However, if no risk is within a 6m radius, the algorithm defaults to selecting the closest and most recent risk. Notably, the quantity and types of obstacles are not factored into this version of the algorithm due to the complexity of establishing meaningful thresholds without a learning algorithm.

One significant advantage of the greedy algorithm is its deterministic nature, which allows for scalability and implementation in robots with minimal computational capabilities. It's also more interpretable compared to a learning algorithm. However, its efficacy is limited to known features such as distance and time, and while these features are effectively utilized, their optimization could be further enhanced with a learning algorithm.

Despite its efficacy, the algorithm's heuristic nature implies that it operates on predefined rules and does not learn from past experiences. Future enhancements of the framework could explore the integration of machine learning techniques. Such advancements would enable the system to adapt and optimize its decision-making process over time based on historical data and evolving operational dynamics. The greedy algorithm is a robust and reliable tool within the Risk Manager module, effectively balancing simplicity, speed, and functionality for emergency scenarios.

3.2.2 Escaping Local Minima

Navigational schemas such as goal seeking, random wandering, and obstacle avoidance are fundamental for directing robot behavior in autonomous robotics. However, these schemas often lead to scenarios where robots get stuck in loops or continually encounter the same obstacle, a phenomenon known as falling into 'local minima.' I have developed two algorithmic approaches for recognizing and escaping local minima to address this. The first approach is frequency-based, designed to detect looping behaviors, while the second employs an Exponential Moving Average (EMA) to identify when a robot is not making further progress. Implementing these strategies benefits the overall mission objectives, such as putting out fires, removing debris, confirming potential victims, and exploring the area more effectively.

EMA-Based Collision Detection

A challenge in autonomous robotic navigation, particularly when combining random wandering with obstacle avoidance, is the potential for robots to become trapped in repetitive cycles. This issue arises when a robot, directed away from an obstacle by its avoidance system, is inadvertently steered back to the same spot by the wandering algorithm, creating a loop that impedes effective navigation.

To address this, I have developed the Exponential Moving Average (EMA)-Based Collision Detection method, which builds on existing EMA techniques. This method, as shown in Equation 3.1, employs EMA focusing on recent position data (x_t) of the robot:

$$\mathrm{EMA}_t = \lambda \cdot x_t + (1 - \lambda) \cdot \mathrm{EMA}_{t-1} \tag{3.1}$$

Here, x_t represents the robot's location at a time 't.' This method quickly identifies repetitive movement patterns, such as continuously colliding with and backing away from the same obstacle. Looping behavior is detected using the position variation measure, defined in Equation 3.2:

$$\sigma_t = |\mathrm{EMA}_t - x_t|, \text{if } \sigma_t < \delta \text{ then loop is detected} \tag{3.2}$$

When the position variation (σ_t) drops below a predefined threshold (δ), it indicates that the robot is likely in a loop, triggering an automated maneuver to navigate out of it.

This EMA-based approach is characterized by its rapid response and adaptability, with adjustable parameters that can be fine-tuned for different tasks and environmental conditions. Calibration of these parameters is crucial to ensure accurate detection of actual local minima scenarios and to avoid false positives, typically achieved through extensive simulation testing.

One key advantage of the EMA-based approach is its rapid response to environmental changes. The system is designed to be adaptable, with adjustable EMA parameters that can be tuned to suit different tasks and environmental conditions. Calibration of these parameters is a critical process accomplished through extensive testing in simulated environments. The goal is to fine-tune the system to accurately detect actual local minima scenarios while avoiding false positives that could lead to unnecessary course corrections.

The EMA-based system does face certain challenges. There is a risk of over-correction for minor positional variations. Additionally, this mechanism would likely be most effective in exploration tasks, where repetitive collisions with the same obstacle are common. While it may react more slowly to gradually emerging obstacles, its focus on recent position data might take time to capture slow-developing changes in the robot's trajectory. The following section will address local minima encounters in goal-based tasks.

Count-Based Approach Loop Detection

The Count-Based Approach for Loop Detection methodically identifies and rectifies looping behaviors during goal-oriented tasks. A loop is detected when the robot encounters the same position more than twice in a short time window. Upon this detection, the system activates a corrective measure by adding a random steer vector to the robot's obstacle repulsion vector. This vector is directed outward from the loop, disrupting the repetitive pattern. After implementing this corrective action, the loop detection counter is reset, enabling the robot to continue its intended path without the recurring loop.

An example of this approach in action is when a robot encounters a configuration of obstacles that either causes it to remain stationary (due to canceling repulsion vectors) or leads it into a cycle of being repulsed from one obstacle to another and back again. The Count-Based Approach detects and mitigates these scenarios by introducing a random element in direction, effectively breaking the cycle and allowing the robot to continue its path.

Possible enhancements to this method include finding the optimal exit point from the loop considering both obstacles and the goal. However, this strategy might be less effective in environments with highly dynamic or densely clustered obstacles. Another potential enhancement is integrating learning methods, enabling the robot to remember and avoid previously encountered looping scenarios.

Compared to the other loop detection methods, the Count-Based Approach is distinct in its simplicity and rapid response. While other approaches can find complex loops by analyzing movement patterns over time, it may require more computational resources. Conversely, the Count-Based Approach excels in environments where immediate action and computational efficiency are crucial, making it an ideal choice for time-sensitive and resource-constrained situations.

3.2.3 Fire Manager

The Fire Manager introduces dynamically activated ”propagatable fire” objects, each tagged with a specific timestamp that determines its appearance in the environment. Unlike static elements, these fires only become active and visible when their corresponding timestamps are reached, adding a layer of realism to the fire scenarios.

This mechanism accurately simulates the escalating nature of real-world fire emergencies, where fires can spontaneously appear and spread based on adjacent active fires.

Moreover, fire in the simulation now brings real consequences for the robotic teams. Each fire has a probability of causing operational failure or slowing down a robot for a specific duration, which is determined based on a normal distribution. This addition not only enhances the realism of the simulation but also significantly impacts the robotic teams' risk management strategies.

The module incorporates a randomization process to manage the complexity and unpredictability of fire spread. Each fire can potentially spread to any of its eight neighboring cells, with the occurrence and timing of this spread being determined randomly. This forces robotic teams to adapt their risk management strategies to evolving fire hazards continuously. However, improvements can still be made, such as enabling vertical fire propagation to different levels.

Efficiently integrating these propagatable fires into the existing simulation map, which includes elements like walls and debris, is achieved through a quadtree data structure. This spatial partitioning approach ensures that new fires are placed logically and realistically, respecting the environment's layout and avoiding overlaps with established structures. Notably, adding propagatable fire does not hinder the simulation's speed since all fires are pre-loaded into the environment from the start.

3.3 Conclusion

In this chapter, I have detailed my methodology and the foundational aspects necessary for the development of my framework. The subsequent chapter will offer an in-depth look at my implementation, including an explanation of the existing components that my work relies upon.

Chapter 4

Implementation

4.1 Chapter Overview

This chapter details the essential components of my framework's implementation, which is crucial for conducting practical evaluations and experiments in a designated domain. My research, an extension of earlier work [Gunn and Anderson, 2015; Nagy and Anderson, 2016; Fiawoo, 2019], necessitates a brief discussion of certain foundational aspects of these frameworks. These elements form the groundwork for my implementation, which focuses mainly on risk management strategies and the modifications introduced by the Fire Manager module to enhance simulation realism.

4.2 USAR: Concepts and Objectives

Urban Search and Rescue (USAR) operations, which focus on locating and extracting individuals from collapsed structures, are crucial to my thesis. Characterized by their urgent and high-risk nature, these operations necessitate rapid and effective response strategies to mitigate dangers to both victims and rescuers.

The core of my thesis within a simulated USAR environment aims to achieve two main objectives: 1) identifying and locating as many human casualties as possible, and 2) thoroughly exploring the operational environment. Integral to this is the balance between accomplishing these mission objectives and minimizing risks, ensuring that the actions of robotic teams do not complicate the tasks of actual rescue teams or endanger human victims.. Therefore, active risk management is essential as it equips robotic teams to handle primary objectives efficiently while adeptly responding to evolving risks, such as sudden fire outbreaks or structural instabilities.

Opting for a simulated environment was a strategic decision aligned with my research focus on multi-robot team management and coordination. While real-world implementation with physical robots offers valuable insights, simulation allows for controlled, extensive experimentation without the prohibitive costs and risks of real-world testing. Following the precedents set by Gunn and Anderson [2015] and Nagy and Anderson [2016], this approach facilitates the extensive evaluation required, considering the time and practical challenges of physical experiments.

The inclusion of the Fire Manager module in the simulation notably enhances realism. Simulating the unpredictable nature of fire spread challenges robots to adapt their strategies in real time, mirroring the complexities of actual USAR missions. This feature underscores the importance of robots responding quickly and effectively in changing conditions, aligning with mission success and risk reduction goals.

4.3 The Simulated Enviornment

In my thesis, I have employed the Stage multi-robot simulator [Gerkey et al., 2003], a platform renowned for accurately emulating real-world robotic behaviors. Stage's capability for repeatable, faster-than-real-time experimentation, as highlighted by Vaughan [2008], is invaluable for the extensive testing required in my research. This approach allows me to concentrate on the fundamental aspects of my study without the complications of physical robot maintenance.

Stage offers comprehensive tools for creating virtual environments and robots and a robust programming interface for defining their behaviors and interactions. Like the Fiawoo [2019] framework, I used Stage 4.1 and only had to make a few modifications due to updates to the Postgres version to rectify any issues caused by the changes. Figure 4.1 shows a randomly generated USAR environment. Figure 4.2 provides a closer view of the objects used in my work.

4.3.1 Simulation Concepts

In the simulation, various frameworks are employed to achieve the main goals. These include:

- **Victim Identification**: The main goal of the robots in the simulation is to find and identify human victims using Stage's fiducial sensor model. Two detection sensors are utilized: a basic sensor with a range of 4.0 meters and a field of view of 180 degrees that identifies potential victims and an advanced sensor with a range of 6.0 meters, which can differentiate between genuine and false victims within 4.0 meters.

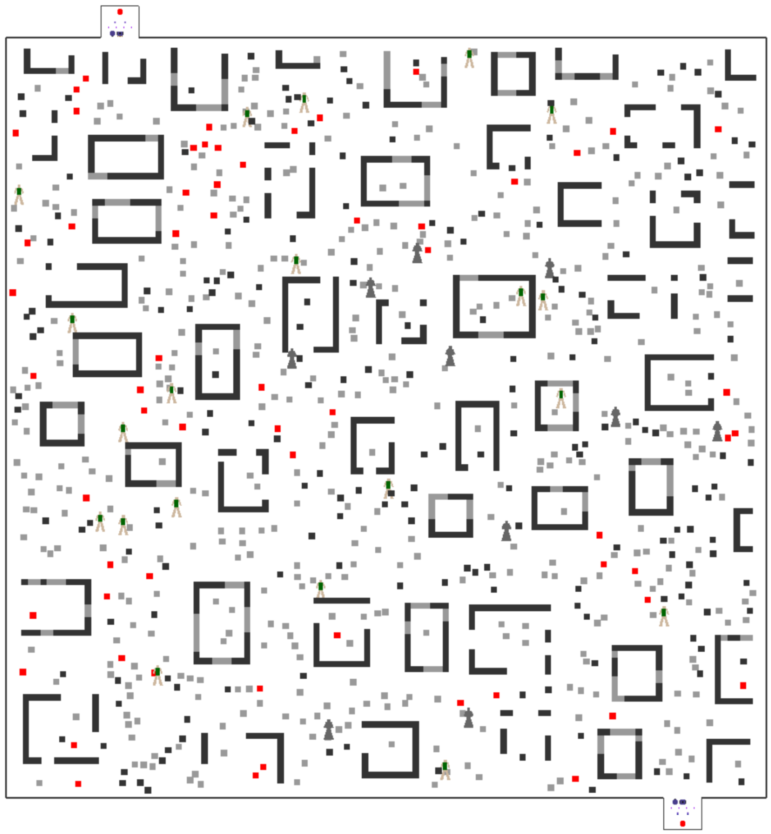

Figure 4.1: An example of a randomly generated 60 by 60 m world within my framework, adapted from Fiawoo [2019].

- **Robot Detection**: Certain robots in the simulation are equipped with robot detectors, using the fiducial sensor model provided by Stage, to simulate visual identification of other robots within a 6.0-meter range and a 180-degree field of view.

- **Area Exploration**: Robots explore the environment guided by a team leader. They maintain an occupancy grid of the environment and identify frontiers, which are the boundaries between explored and unexplored space. Robots are assigned to explore these frontiers to collect data and report to the team leader.

- **Managing Domain Knowledge**: Each robot in the simulation is responsible for maintaining operational domain knowledge, including the locations of victims, the state of the robot's occupancy grid, and information about other robots. This knowledge is acquired either through direct discovery or by communication with other team members.

- **Unreliable Communication**: Robots are equipped with short-range wireless communication modules, simulating real-world limitations like environmental obstructions affecting communication range. The simulation includes random communication failures, with a success parameter determining the likelihood of successful message transmission.

- **Robot Failure**: The simulation incorporates the possibility of random robot failures, either temporary or permanent. Each robot type has a defined failure probability, with temporary failures lasting between 3 and 4 minutes and permanent failures lasting the mission's duration. When a robot fails the role it occupied is no longer filled and another robot must replace it. Thus the robot assigning a task may not ultimately be the same one the results are reported back to, though it will be an individual occupying the same role.

- **Fiducial Victim Markers**: Robots are tasked with placing fiducial victim markers next to detected victims. These markers, limited in number per robot, assist human rescuers in locating casualties and can contain medical supplies or food.

- **Active Risk Management**: Strategies employed by robots to manage and

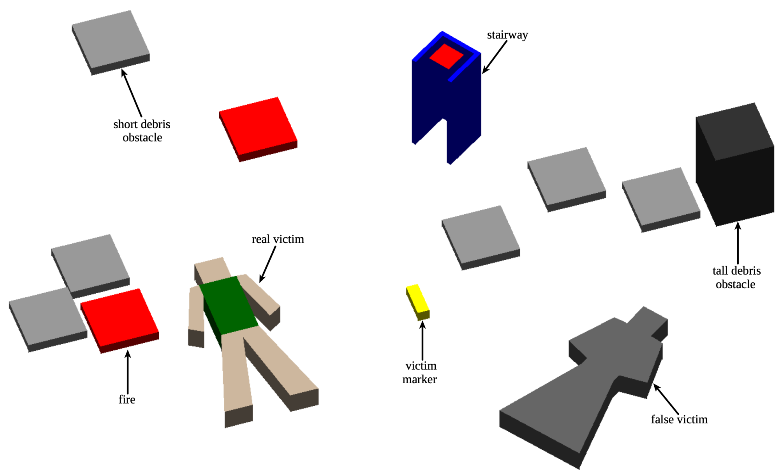

Figure 4.2: Objects within an environment as used in my research. Adapted from Fiawoo [2019]

mitigate risks encountered during missions. These include how robots assess and respond to dynamic risk factors in the environment, ensuring safety and efficiency in operations.

- **Escaping Local Minima**: Techniques used by robots to avoid getting trapped in repetitive patterns or obstacles. These include methods like algorithmic adjustments and sensory data analysis to recognize and navigate out of such situations effectively.

4.3.2 Environment Objects

In the simulation, various environmental objects are included, such as:

- **Fire:** In the simulated environment, red objects, as shown in Figure 4.2, represent fire hazards. These fires are propagatable and can only be extinguished by robots equipped with fire-extinguishing gear. Additionally, traversing through fire may result in temporary or permanent robot failure and can subject robots to random slowdowns, reflecting the challenges of navigating through fire in real-world scenarios.

- **Removable Debris:** Illustrated as light grey objects in Figure 4.2, low-lying debris pose a navigational challenge. Robots with tracked drive systems can traverse these obstacles, while wheeled robots must navigate around them, risking immobilization. Traversing debris also tends to slow down robots. Robots equipped with debris-clearing tools can remove these obstacles, facilitating passage for all robots and providing assistance to any immobilized robots.

- **Fixed Obstacle:** Tall, dark grey objects represent impassable obstacles in the environment. All robot types must navigate around these obstacles as they are unable to move through them.

- **Victim:** Specific objects, randomly placed within the environment, symbolize human casualties needing assistance. Two types of victim objects are present: real victims and false victims, the latter being debris configurations that resemble real victims. Advanced sensor capabilities are required to differentiate between these two types.

- **Marker:** Robots are equipped with markers to be deployed near potential victims if no other markers are detected in the vicinity. Each robot carries a maximum of two markers and can request additional markers from other robots if its supply is depleted, ensuring continuous marking capability for victim identification.

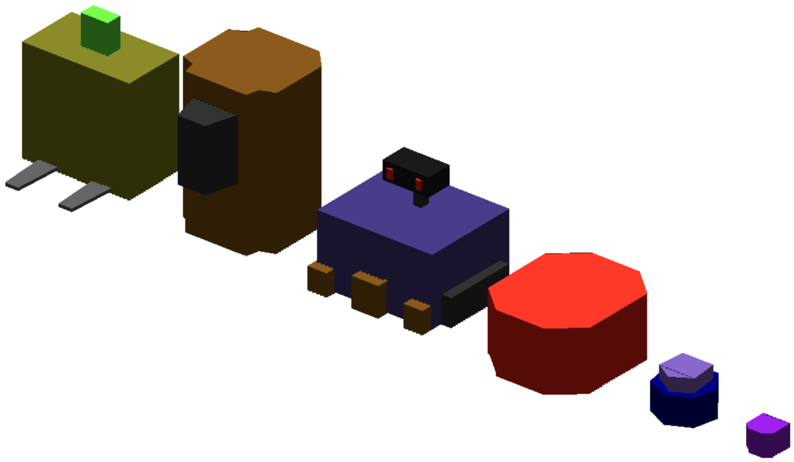

Figure 4.3: The different robot types used in my framework. From left to right, UnstuckBot, FireBot, DebrisBot, MaxBot, MidBot, MinBot. Adapted from Fiawoo [2019]

4.4 Robot Types

In my implementation, I have used six different types of robots. The MaxBot, MidBot, and Minbot were initially developed by Gunn and Anderson [2015], and the DebrisBot was added to the framework by Nagy and Anderson [2016]. Fiawoo [2019] has contributed two more robot types to the framework: the FireBot and StairBot. However, in my implementation, I have only repurposed the StairBot for the unstuck robot task and renamed it to UnstuckBot. Figure 4.3 visually represents the different robot types used in my framework.

- **MinBots**: These are small, expendable robots with basic victim-sensing abilities, primarily used for potential victim discovery and exploration. MinBots navigate using sonar sensors but lack robot detection capabilities and advanced computational resources, making them unsuitable for leadership roles. They can perform recruitment tasks but cannot provide path information, increasing the risk of other robots getting stuck or lost.

- **MidBots**: Larger than MinBots, MidBots possess improved computational power and memory, enabling them to assume leadership roles if more capable robots are unavailable.. They have advanced victim-sensing abilities to distinguish real from false victims within 4.0 meters and can detect frontiers and assign tasks, although they lack path-planning capabilities.

- **MaxBots**: These robots have the highest memory and computational capabilities, suitable for leadership roles. They have basic victim-sensing abilities and rely on MidBots for accurate victim identification. MaxBots are equipped with a tracked drive system for traversing low-lying debris.

- **DebrisBots**: Specialized in debris removal, DebrisBots are equipped with equipment to clear low-lying debris. They have sonar arrays for distinguishing between removable debris and taller obstacles. They also have basic victim and robot detection capabilities. However, they are not fit for leadership roles due to their specialized function.

- **FireBots**: FireBots are specialized robots equipped with fire extinguishers to put out fires within a 4.0-meter range. They have fire, victim, and robot sensing capabilities and a tracked drive system for moving through debris. These robots are not suitable for leadership roles due to their specialized nature.

- **UnstuckBot**: Previously known as StairBot for its ability to transport robots between different floor levels, this robot has been repurposed to aid in disengaging stuck robots from debris or obstacles. While retaining its stair-lift capability, its primary function is now to navigate to and assist immobilized robots. It is equipped with a tracked drive system and basic victim and robot detectors. However, due to its specialized functions, it is not suitable for leadership roles.

4.5 Tasks

4.5.1 Gunn Framework Tasks

These tasks are derived from the original framework by Gunn and Anderson [2015] and are incorporated into my framework.

- **Explore**: Robots default to this task when idle, roaming the environment randomly, except for the DebrisBot.
- **Explore Frontier**: Robots with the capability to find frontiers carry out this task, which involves moving to a target area, conducting a brief exploration, and relaying discoveries to the team leader.
- **Find Team**: New robots use this task to integrate into the environment, moving inward from their entry point for five minutes or until they join an existing team or form a new one.
- **Find Victim**: This implicit task assesses a robot's ability to locate victims without being queued as a formal task.
- **Confirm Victim**: Robots with advanced victim detection sensors perform this task to verify potential victims at reported locations.

- **Manage Team**: This task encompasses the functions required for a robot to direct a team, including task assignments.

- **Encounter**: When two teams meet, this task directs the merging and redistribution of team members (See 2.4.1 for merge and redistribution details).

4.5.2 ARC Framework Tasks

The tasks from the prior framework by Nagy and Anderson [2016] have been adopted in my work.

- **Begin Role Recruitment**: Triggers the role recruitment process when activated.

- **Find Robot**: Seeks to locate other robots by conducting a random walk, utilized during active recruitment.

- **Unguided Debris Removal**: The DebrisBot's standard operation, performing a random walk to locate and eliminate any reachable low-lying debris.

- **Guided Debris Removal**: Directs a DebrisBot to a specific location to clear debris, which may involve aiding a trapped robot.

- **Donate Marker**: Oversees the transfer of a victim marker from one robot to another. (More on Marker Manager in 2.4.2).

- **Wait For Marker**: Instructs a robot to await a marker donation. If the marker is not received promptly, the robot will discontinue the wait and may seek a marker from another robot.

4.5.3 Fiawoo Framework Tasks

In my framework, only the Extinguish Fire task has been used out of the three tasks presented in Fiawoo [2019]'s work.

- **Extinguish Fire**: This task is exclusive to the FireBot, which is equipped with extinguishing gear. The robot is required to locate and extinguish fires at a specific location, using Fiawoo's fire detection schema for identification and suppression.

- **Manage Stairway**: Not applicable in my framework, which focuses on single-level environments. Originally, this task was for StairBots with teleporters, who were to stay within a 2-meter radius of stairways for assisting with floor transitions.

- **Stair Lift**: Also excluded from my single-level focused framework. In the previous framework, this task involved non-teleporter robots requesting StairBots' assistance for moving between floors within a 30-second timeframe.

4.5.4 My Framework

In addition to the tasks outlined thus far, my framework adds one additional task.

Unstuck Robot

UnstuckRobot task is introduced in my framework to assist robots that become immobilized. When a robot detects that it is stuck, it will create a task to signal either the team leader or nearby robots to dispatch an UnstuckBot for help, using any of the available recruitment methods.

After accepting the task, the UnstuckBot approaches the immobilized robot, maintaining a distance of approximately 5 meters. It calculates the optimal position for dislodging the immobilized robot, considering the positions and orientations of both itself and the other robot. This placement allows the UnstuckBot to apply force in a direction that maximizes the chance of freeing the stuck robot.

If the first attempt is unsuccessful, the UnstuckBot will retry up to two additional times, totaling three attempts. If these attempts also fail, the task is temporarily abandoned, leaving a window of two minutes before it can be retried. During this period, the stuck robot might free itself by random movement or by other robots (such as a debrisbot) altering the environment. If the robot remains stuck after this window, another UnstuckRobot task is initiated for a fresh attempt at dislodgment.

This approach ensures efficient utilization of UnstuckBot's capabilities and addresses a significant issue in mission operations. A stuck robot is a liability to the team's overall function, as it cannot contribute to exploring the area, identifying human casualties, or flagging potential victims. By quickly addressing such instances of immobility, the UnstuckRobot task plays a crucial role in maintaining the operational efficiency of the entire robotic team, ensuring that each robot can effectively fulfill its mission objectives without being hindered by immobilization.

4.6 Attributes

As outlined in Section 2.4.1, every robot within the framework is characterized by a specific set of attributes, derived from previous work [Gunn and Anderson, 2015; Nagy and Anderson, 2016; Fiawoo, 2019]. These attributes, which allow robots to evaluate their own abilities and those of others, are divided into physical, computational, and sensory categories.

Table 4.1: Physical properties and attributes of robots used

	MinBot	**MidBot**	**MaxBot**	**DebrisBot**	**FireBot**	**UnstuckBot**
locomotion	wheeled	wheeled	tracked	tracked	tracked	tracked
length x width (cm)	10 x 10	20 x 20	44 x 38	40 x 50	45 x 35	40 x 40
expendability	1.0	0.25	0.05	0.1	0.1	0.1
debris remover	no	no	no	yes	no	no
fire extinguisher	no	no	no	no	yes	no
lifter	no	no	no	no	no	yes
marker count	2	2	2	2	2	2

Table 4.2: Capabilities of robots used

	MinBot	**MidBot**	**MaxBot**	**DebrisBot**	**FireBot**	**UnstuckBot**
victim tracker	yes	yes	yes	yes	yes	yes
frontier finder	no	yes	yes	no	no	no
maintain team map	no	yes	yes	no	no	no
assign tasks	no	yes	yes	no	no	no
planner	no	no	yes	no	no	no

- **Physical Attributes**: These attributes define a robot's physical characteristics, such as speed, size, navigational abilities, and their pieces of equipment. An addition to my framework is the 'lifter' attribute, which ensures that a robot, particularly an UnstuckBot, can free a stuck robot. More details on each robot type's physical attributes are available in Table 4.1.

- **Computational Attributes**: These attributes represent heuristic capabilities within a robot's control software. They cover functions such as maintaining an environmental map and path planning. These ensure the robot has the necessary computational and memory resources for task execution.

- **Sensory Attributes**: These include the types of sensors equipped on each robot model, vital for their functional operations. In addition to the existing sensors from previous frameworks, I have implemented the 'In Fire' and 'In Debris' sensors across all robots. These sensors are purposefully designed to identify when a robot is in fire or surrounded by debris. While they do not provide predictive warnings about imminent hazards, their primary function is to enable robots to report these potential dangers to their team leader or to call upon other specialized robots equipped to handle such situations. Comprehensive information about the sensors equipped in each robot type is provided in Table 4.3.

	MinBot	**MidBot**	**MaxBot**	**DebrisBot**	**FireBot**	**UnstuckBot**
victim sensor	basic	advanced	none	basic	basic	basic
robot sensor	no	yes	yes	yes	yes	yes
fire detector	none	basic	basic	basic	advanced	basic
in fire	yes	yes	yes	yes	yes	yes
in debris	yes	yes	yes	yes	yes	yes
sonar sensor (#units)	5	10	3	14	8	12
sonar range	4m	6m	2m	6m	6m	6m
laser rangefinder	none	none	yes	none	none	none
victim marker detector	yes	yes	yes	yes	yes	yes

Table 4.3: Physical properties and attributes of robots used

4.7 Role Types

In this section, I refine the roles integral to my research, acknowledging the existing fire extinguisher role and introducing the new robot lifter role. These roles, designed to leverage each robot's attributes, facilitate task assignments through a system of suitability values, calculated based on the robots' capabilities and task requirements, as compared in Table 4.4.

- **Team Leader**: Ideally embodied by the MaxBot, the team leader commands the team by maintaining a grid-based risk list and issuing risk mitigation tasks. With its planning capabilities and the location-specific nature of these tasks, the MaxBot is well-equipped to assign them effectively.

- **Coordinator/Explorer**: Charged with victim verification and exploration,

Table 4.4: The possible robot-role combinations with their suitabilities

	MinBot	**MidBot**	**MaxBot**	**DebrisBot**	**FireBot**	**UnstuckBot**
Leader	11	83	106	26	26	23
Coordinator/Explorer	24	136	52	32	32	27
Explorer	124	56	52	52	52	17
Debris Remover	0	10	10	110	10	10
Fire Extinguisher	5	15	15	15	115	15
Robot Lifter	5	15	15	15	15	105

the MidBot excels in this role and can manage a risk list as a temporary leader. Lacking a planning module, it primarily focuses on recruiting other robots for risk mitigation.

- **Explorer**: Entrusted with scouting uncharted areas for potential victims, the MinBot is the choice robot due to its expandability. Other types are normally reserved for roles that match their higher capabilities.

- **Debris Remover**: This role is specific to the DebrisBot, equipped to clear obstructive debris and facilitate the team's movement.

- **Fire Extinguisher**: Occupied by the FireBot, this role involves addressing fire hazards with its advanced detection and extinguishing gear.

- **Robot Lifter**: A new role I have introduced, the robot lifter is responsible for rescuing immobilized robots, enhancing the team's resilience and operational continuity.

These roles are strategically defined to ensure that robots effectively contribute to the mission's goals and actively manage risks. By optimizing role assignments, the framework prevents inefficiencies and ensures that every robot's capabilities are utilized to their fullest potential within the simulated USAR environment.

4.8 Autonomous Control

This section details the updated low-level control mechanisms for my robotic implementation. While perceptual schemas remain as in prior frameworks [Gunn and Anderson, 2015; Nagy and Anderson, 2016; Fiawoo, 2019], there are critical enhancements in Localization and hazard detection to improve risk management. On the motor side, I have refined the Avoid Obstacle schema and introduced the EMA-Based Random Movement and Unstuck Robot, equipping robots with improved autonomous navigation and problem-solving capabilities.

4.8.1 Perceptual Schemas

Robots in my framework utilize perceptual schemas to process sensory data, providing them with essential environmental insights, underpinning their navigational decisions and task execution to ensure mission objectives are efficiently met. These schemas include:

- **Localization**: Monitors a robot's position and orientation using the team's shared coordinate system. When it detects a robot is stuck, unlike the previous framework where a debris removal task was initiated, the immobile robot now informs others, prompting the initiation of an UnstuckRobot task.

- **Process Range Data**: Gathers data on nearby obstacles from rangefinder sensors, which is helpful for the UnstuckRobot task.
- **Detect Debris**: Works alongside localization to log the presence of debris, updating the risk list with the locations of such hazards to facilitate future avoidance and support the UnstuckRobot and DebrisRemoval tasks.
- **Detect Obstacles**: Compiles an obstacle list from range data, contributing to detecting impediments that may require an UnstuckRobot's intervention.
- **Detect Robots**: Identifies other robots using the robot detector sensor, assisting in tasks such as freeing immobilized team members.
- **Detect Victims**: Utilizes victim detector sensors to maintain a list of observed victims.
- **Detect Lost**: Measures the distance a robot travels to a target location. If the robot exceeds five times the required distance without success, it drops or re-queues the task.
- **Detect Markers**: Perceives victim markers within the environment using marker detector sensors.
- **Detect Fire**: Implemented to identify fires, updating the risk list with their locations via the fire detector sensor, thus aiding in the QuenchFire task.

4.8.2 Motor Schemas

Motor schemas outline the sequence of actions for robots, influenced by data from perceptual schemas. These motor schemas provide robots with the autonomy needed for sophisticated actions, enhancing their ability to navigate and perform tasks within their operational environment:

- **Avoid Obstacles**: Generates a repulsive force to steer the robot away from obstacles. In my framework, if the robot encounters local minima, as discussed in Section 3.2.2, it will receive an additional random steer vector to help escape the trap and move away from the obstacles.

- **Move To Location**: Enables a robot to navigate to a designated point in the environment, utilizing path data if available or a reactive strategy if not.

- **Turn In Place**: Instructs the robot to rotate clockwise on the spot without moving linearly.

- **Random**: Prevents the robot from getting stuck by generating a random motion vector for redirection.

- **Recover Stuck**: Collaborates with the localization perceptual schema to recognize and respond when a robot is stuck by initiating a backward movement to disengage.

- **Seek Debris**: Attracts the robot to short, navigable obstacles and repels it from tall, non-navigable obstacles or non-debris entities, using input from the detect obstacles schema.

- **Move To Robot**: Creates a vector to guide a robot toward another, based on information from the Detect Robots schema.
- **EMA-Based Random Movement**: As noted in Section 3.2.2, this schema intervenes when a robot is stuck in a stationary position by applying a random force to aid its escape.
- **Unstuck Robot**: Activated to move a robot stuck in debris, allowing it to resume its mission.

4.9 Framework-Specific Modules

This section summarizes the key functional modules integral to my implementation. All modules, except for the newly added fire manager, have been adopted from the existing framework [Gunn, 2011; Nagy, 2016; Fiawoo, 2019]. Together, these modules synergistically improve the robots' ability to navigate, make decisions, and execute their designated tasks within my research framework's complex, simulated environment:

- **Encounter Manager**: Directs the merging and redistribution of robotic teams when two or more robots meet within the environment. "It ensures that resources are allocated in the best way possible, reducing redundancy during such encounters."
- **Knowledge Manager**: Acts as the storage and reference point for all information acquired by a robot. It includes robot and other robots' attributes and the current team structure, which is essential for strategic decision-making.

- **Communication Manager**: The hub for handling all inter-robot wireless communications. It ensures messages are accurately received, interpreted, and responded to, facilitating seamless team collaboration.

- **Role Manager**: Administers the assignment of roles and the dynamic restructuring of team roles. It is responsible for assessing when a role change is needed and managing recruitment processes to align with team objectives.

- **Task Manager**: Manages the individual and team tasks, maintains the task queue, and oversees communication and procedures for task allocation. It ensures tasks are prioritized and executed in an order that aligns with team goals.

- **Location and Rare Skill Manager**: Manages specialized tasks tied to specific locales and identifies robots with rare skills critical to mission success. This module guides robots to areas where their unique skills are most needed, enhancing their contribution and satisfaction.

4.9.1 Fire Manager

The Fire Manager is a newly integrated module that introduces the unpredictable nature of fire in real-world Urban Search and Rescue (USAR) scenarios. This module is specifically designed to manage the beginning and propagation of fire within the simulated environment. Before the simulation, the locations and times at which fires will spawn are determined randomly and recorded within the map file. This pre-planning is necessary due to the requirements of the Stage simulation environment, which requires the pre-specification of all environmental objects and their dynamics.

Upon initiation of a simulation, the Fire Manager reads these predetermined fire event details from the map file and faithfully recreates them within the virtual environment at the scheduled spawn times. It oversees the initial ignition of these fires and continually assesses their presence in the environment. The module also simulates fire spread when propagation can only occur between adjacent active fires. This ensures that the spread of fire is both realistic and computationally manageable within the constraints of the simulation, allowing for the accurate study of robotic behavior and effectiveness in response to such hazardous and evolving conditions. The Fire Manager's role in simulating this aspect of environmental change is crucial, as it directly influences the strategic planning and adaptability required of the robotic teams engaged in the simulation.

4.10 Mission-Specific Modules

This section outlines the modules instrumental in supporting robots to achieve high-level goals in the USAR domain. Collectively, these modules enhance the robotic team's capability to perform complex tasks, navigate efficiently, and manage resources effectively in the challenging USAR environment.

- **Environment Mapper**: Manages a shared map of the environment, integrating data from team exploration activities.
- **Frontier Finder**: Generates tasks to explore new frontiers using an occupancy grid-based approach to analyze the robot's environmental map.
- **Victim Tracker**: Keeps a running list of identified victims, updating their

statuses (potential or confirmed) as new information is acquired.

- **Planner**: Calculates suitable paths to designated locations, like victim sites, ensuring assigned robots navigate efficiently and avoid getting stuck.
- **Marker Manager**: Oversees the process of releasing markers when victims are detected, aiding in the systematic identification and tracking of victims.
- **Satisfaction Manager**: Tracks and utilizes a robot's satisfaction, derived from task suitability and completion time, to guide its decision-making. It periodically assesses satisfaction levels and sets a 'willingness flag' accordingly, influencing the robot's readiness for task and role changes to improve operational satisfaction.

4.10.1 Risk Manager

Risk Detection

Different robots within the framework detect hazards in the environment through various means. When a robot either passes by fire or debris, becomes stuck, or receives communication from a stuck robot, it records this information as a risk. These detected risks are then communicated to the team leader, to keep the leader up to date updated on the known hazards within the operational environment.

Prioritization of Mitigation Tasks

The framework utilizes a greedy algorithm to select and prioritize risk mitigation tasks, efficiently processing risk data within each 5m x 5m grid of the operational environment. The algorithm first evaluates risks within a 6m radius of the assigned robot, prioritizing them based on two key factors:

1. **Average Timestamps**: Risks are prioritized by considering the average timestamps in each grid, focusing on the most recent information.

2. **Proximity-Based Selection**: Risks within a 6m radius are prioritized based on their proximity to the robot. If no risks are found within this radius, the algorithm then considers risks outside this range, returning the risk with the highest score if the prioritized list is not empty. If the list is empty, it checks for any risk in nearby grids and returns the first one found. This algorithm is outlined in Algorithm 1

This method ensures that the most immediate and relevant risks are addressed first, focusing on the recency of information and proximity to the robot, enhancing the efficiency of risk management in dynamic USAR scenarios.

Risk Response Selection

Robots tasked with risk mitigation also use a greedy algorithm for task prioritization, favoring nearby risks. This approach ensures that response times are minimized, and that the most accessible hazards are addressed first, contributing to more efficient and effective risk management.

Algorithm 1 Proximity-Based Risk Prioritization

Require: Grids with risk data, proximity threshold (6m)

Ensure: Prioritized list of risks

Initialize an empty list for prioritized risks

for each grid G in the environment sorted by distance **do**

  if G is within 6m proximity **then**

    Calculate d_i for all risks in G: distance of risk r_i from the robot

    Calculate t_i for all risks in G: time elapsed since r_i was detected

    Compute Risk $\text{Score}_G = \sum_{i=1}^{n} \left(\alpha \cdot \frac{1}{1+t_i} + \beta \cdot e^{-d_i} \right)$

    Add risk to the prioritized list based on its score

  else if G is outside 6m proximity and prioritized list not empty **then**

    return Risk with the highest score

  else if no risks found within proximity threshold **then**

    if Risk R found in G **then**

      return R

    end if

  end if

end for

4.11 Conclusion

This chapter has outlined my implementation in the context of a USAR environment, setting the stage for evaluating my framework. The following chapter will detail the experiments conducted to evaluate the effectiveness of my work.

Chapter 5

Testing and Evaluation

5.1 Introduction

In this chapter, I evaluate my proposed solution by benchmarking it against the most recent version of our lab's framework [Fiawoo, 2019]. This assessment was carried out using a simulated USAR environment, aligning closely with the methodologies employed in these earlier studies. The evaluation focused on several critical factors also comparable to previous work, including the reliability of wireless communications, the recruitment strategy implemented, and temporary, permanent, or no failures experienced by robots during missions.

Following this introduction, I will outline the design and methodology of my experiments, present the results obtained, and engage in a detailed discussion to interpret these findings and their broader implications.

5.2 Evaluation Metrics

To assess the efficacy of my framework, I monitor five key metrics throughout each experiment: the percentage of victims identified, the percentage of explored areas, areas revisited, the number of fires extinguished, and the number of piles of debris cleared.

In assessing victim identification, I consider both positively and negatively identified victims as reported to the team leader, diverging from the work of Fiawoo [2019], which only accounted for positively identified victims. Including negatively identified victims provides a broader measure of the framework's efficacy in task management, focusing on the system's ability to dispatch robots capable of identifying potential victims and reporting back. This shift emphasizes the importance of efficient task assignment over the precision of victim identification, which depends on the sophistication of the victim detection algorithm rather than the framework's operational effectiveness.

For exploring and revisiting areas, I compile data on the percentage of the environment explored and the extent of areas revisited, introducing a new metric to measure the reduction in repetitive exploration. This approach addresses the challenge of robots getting stuck in local minima by measuring the decrease in redundant traversal of the same areas. Unlike previous frameworks, which did not explicitly track revisited areas, this metric offers insight into the framework's navigational efficacy and its ability to encourage thorough exploration without unnecessary repetition.

Concerning active risk management, the metrics extend to the number of fires extinguished and the number of piles of debris cleared. These indicators reflect the framework's responsiveness to dynamic environmental challenges and its strategic prioritization of tasks critical to mission success. This nuanced approach to evaluating performance underscores the framework's adaptability and the strategic allocation of resources towards managing hazards in the operational domain.

5.3 Environment Generation and Setup

For my implementation's evaluation, I selected five generated world files from previous work [Nagy and Anderson, 2016; Fiawoo, 2019], each tested with ten different random seeds to ensure variability in aspects like robot movement, chance of failure, and communication success. This approach allows each test to have distinct random factors, providing a robust assessment of the framework under various conditions.

In the world files used for the experiments, all elements, including replacement robots and fire propagation patterns, were predefined as per the requirements of the Stage framework. This framework necessitates that every object, such as robots and environmental features, be explicitly defined before the simulation begins. Consequently, each of the five world files was set up with distinct fire propagation sequences, introducing unique challenges and enhancing the variability of each test run.

The independent variables for my experiments, detailed below, were chosen to evaluate various aspects of the framework comprehensively:

- **Communication Reliability**: This variable determines the likelihood of successful message delivery among robots. Possible values include 100% (perfect communication), 60%, and 20%, representing varying communication challenges.

- **Recruitment Configuration**: Defines the recruitment strategy employed during the experiment. Options include active, concurrent, and passive recruitment [Nagy and Anderson, 2016], dictating how robots assign tasks or recruit team members.

- **Replacement Robots**: Indicates whether additional robots are used in a trial. This includes 10 MinBots, 2 MidBots, 1 MaxBot, 1 DebrisBot, and 1 FireBot, released after a specific time into the simulation.

- **Robot Failure**: Controls the probability of each robot type experiencing temporary or permanent failure during trials, with settings for no failures, minimal failures, and major failures as shown in Table 5.1.

- **Active Risk Configuration**: Controls the activation of active risk management strategies within the framework, assessing their impact on overall mission success.

To ensure effective and affordable computing resources, I utilized the AWS platform. Based on the available resources, I chose c7a.8xlarge instance where the letter *c* indicates that it is part of the Compute Optimized family, designed for compute-intensive workloads. The number *7* denotes that it belongs to the 7th generation of Compute Optimized instances, and the letter *a* specifies that this instance uses AMD processors. The size designation 8xlarge means that this instance has 32 virtual CPUs and 64 GiB of memory, providing substantial computational power and memory for demanding tasks. This setup was chosen to ensure efficient processing and the capability to run multiple tests concurrently. Given the array of independent

Table 5.1: Robot types and the different configurations of failure probability [Fiawoo, 2019]

level	**model**	**prob. permanent failure**	**prob. temporary failure**	**avg. % total time failed**
none	MinBot	0.000	0.000	**0.000%**
	MidBot	0.000	0.000	**0.000%**
	MaxBot	0.000	0.000	**0.000%**
	Other types	0.000	0.000	**0.000%**
minimal	MinBot	0.000	0.008	14.9%
	MidBot	0.000	0.006	11.9%
	MaxBot	0.000	0.004	9.0%
	Other types	0.000	0.008	14.9%
moderate	MinBot	0.002	0.014	25.1%
	MidBot	0.002	0.012	20.9%
	MaxBot	0.002	0.010	18.5%
	Other types	0.002	0.014	25.1%

variables and the scope of the testing plan, a total of 5400 tests were conducted. This comprehensive approach allowed for simultaneous execution across different configurations, maximizing the use of available computational resources and providing a robust framework evaluation.

5.4 Experiment Results and Analysis

In Fiawoo [2019]'s work , the baseline used was derived from the work of Nagy and Anderson [2016]. Instead of using this earlier baseline, I make comparisons based on satisfaction-tracked performance, which corresponds to Fiawoo [2019]'s final contributions. This new baseline is now the standard for my evaluation. I collected data after each trial in my experimental setup, utilizing the metrics introduced in Section 5.2. The focal point of these results is to examine the influence of Active Risk Management (ARM) on various aspects, including Area Explored, Area Revisited, Victims Confirmed, and risk mitigation strategies like Debris Removal and Fire Quenching. ARM, which I will refer to as my framework moving forward, is evaluated against this new baseline to highlight its impact.

5.4.1 Victims Successfully Confirmed to Leaders

Figure 5.1 shows the percentage of victims that were identified by team leaders using different recruitment modes, and with varying communication success rates. The study also considered the impact of active risk management being turned on or off, as well as the probabilities of robot failure, and the use of replacement or non-replacement robots.

By looking at at a communication success rate of 20%, achieving the 10% confirmed victim threshold proved challenging for all setups except for no-fail active and no-fail passive ones. Even with these setups, only a slight exceeding of the threshold was observed. The results suggest that regardless of the framework used or the presence of replacement robots, achieving the task at a 20% communication success level

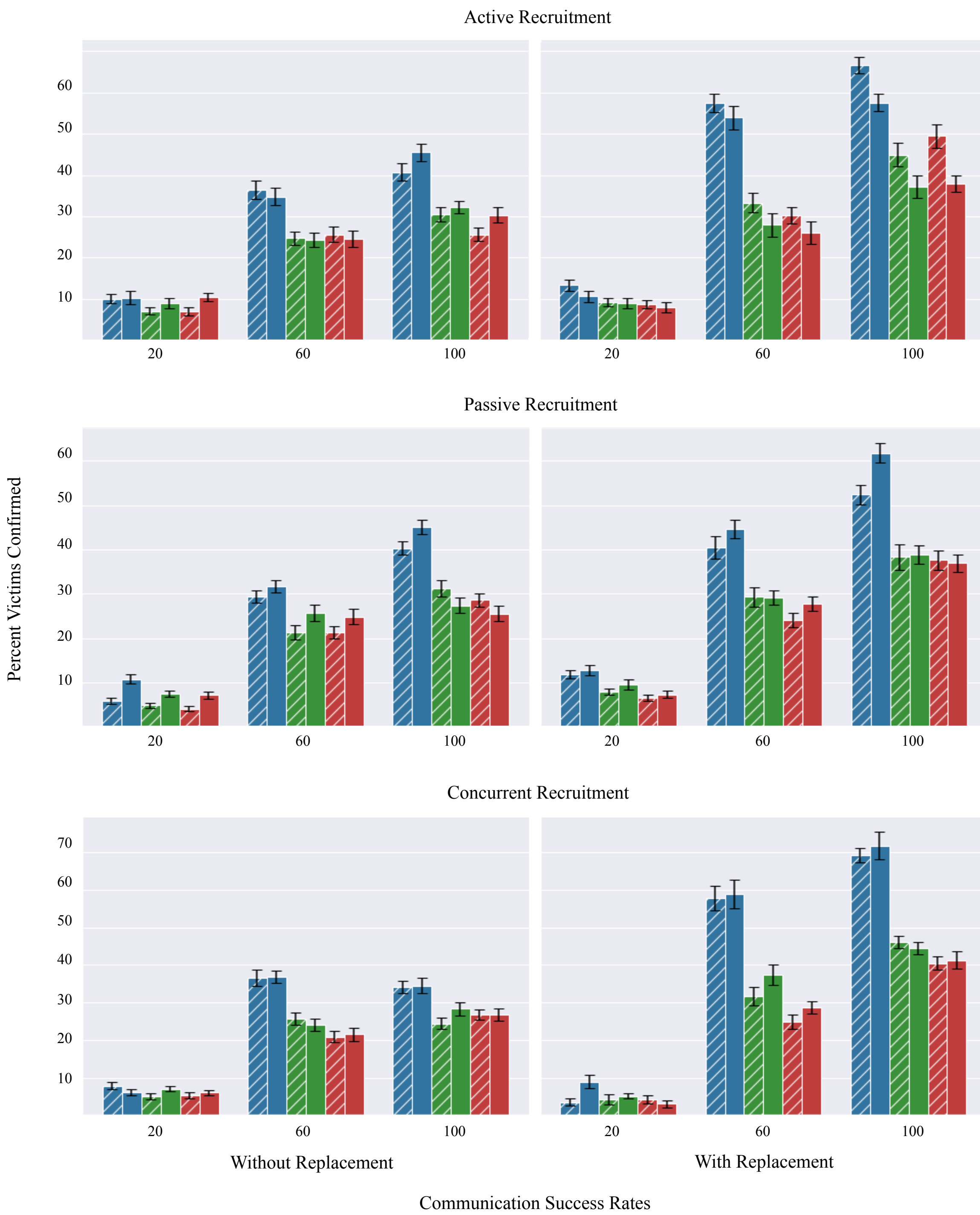

Figure 5.1: Percentage of victims identified using various configuration

is unattainable.

Under passive recruitment, with a 100% communication success rate and the inclusion of replacement robots, the ARM framework reached a victim confirmation rate of 61.75%, closely mirroring the top performance observed in concurrent recruitment, which peaked at 71.78% under similar conditions. This contrasted with the active recruitment strategy, where even with replacements, the victim confirmation rate hovered around 57.58% at the same communication success rate, underscoring a noticeable performance gap.

The resilience of the passive and concurrent strategies was further highlighted at lower communication success rates (60% and 20%), where they consistently outperformed active recruitment. For instance, at a 60% communication rate with replacements, Concurrent recruitment maintained a victim confirmation rate of 58.93%, significantly higher than Active recruitment's 53.92%. This trend persisted across failure configurations, with Passive recruitment just as far behind, illustrating the strategic advantages of these recruitment modes in maintaining higher performance levels despite communication challenges.

After evaluating the ARM framework's performance in various configurations, findings indicate that the Passive and Concurrent recruitment strategies outperformed the Active approach in victim confirmation tasks, particularly when communication success rates reached 100%.

The selection of a recruitment strategy significantly influences the performance of the ARM framework in search and rescue operations, particularly concerning victim confirmation tasks. Active recruitment, with its intensive focus on team assembly and environmental risk management, may weaken its effectiveness in victim confirmation. The demands of coordinating teams and managing risks could potentially divert the robots' attention and resources from victim identification and confirmation.

Conversely, Passive and Concurrent recruitment strategies offer an approach that could potentially yield higher success rates in victim confirmation. By reducing the immediacy of recruitment demands, these strategies allow robots to direct a more concentrated effort toward victim confirmation. This more streamlined focus can increase success rates in identifying and confirming victims, primarily when the environment supports high communication success rates and the operational complexity is managed adeptly.

In conclusion, the analysis highlights the crucial role of recruitment strategy selection in improving the ARM framework's performance in search and rescue operations. Passive and Concurrent recruitment strategies stand out for their operational efficiencies, adaptability, and reduced communication demands, which offer a more effective pathway to maximizing victim confirmation rates. These strategies are particularly beneficial in environments with high communication success rates and where operational complexity can be managed with skillful execution.

5.4.2 Area Coverage

ARM's passive strategy shows a higher area coverage compared to the active and concurrent strategies because of the recruitment mechanisms employed. The passive strategy recruits opportunistically through chance encounters, allowing robots to continue their assigned tasks with minimal interruption, which leads to more efficient and extensive area coverage. In contrast, the active and concurrent strategies have a higher probability of assigning risk mitigation tasks of discovered risks to Firebots and Debrisbots, sending them back to already explored areas. Moreover, other robots actively search for or respond to messages to find the capable fire or debris bots to exchange information, which diverts them from their primary exploration tasks and reduces the time and focus they can dedicate to exploration. In the active strategy, the disparity in area coverage between ARM and the baseline, in contrast to the passive and concurrent strategies, arises because ARM's framework requires intermittent shifts in robot focus from exploration to risk management. This dynamic results in a significant difference in coverage outcomes.

In scenarios characterized by optimal conditions of passive recruitment, 100% communication reliability, no failures, and the presence of replacement robots, ARM achieved a coverage of 79.05%. The difference in performance (to the baseline) was more pronounced in active recruitment. Transitioning to the concurrent recruitment strategy, both Fiawoo and ARM exhibited somewhat reduced performance levels compared to the active and passive strategies, peaking at 74.45%. However, the competitive dynamics between the two frameworks were quite intriguing in this strategy. ARM showcased a superior performance in most instances within the concurrent recruitment framework, or the two were closely matched in their coverage outcomes. This suggests that ARM's strategy may have certain advantages or adaptabilities that come to the forefront in concurrent recruitment scenarios.

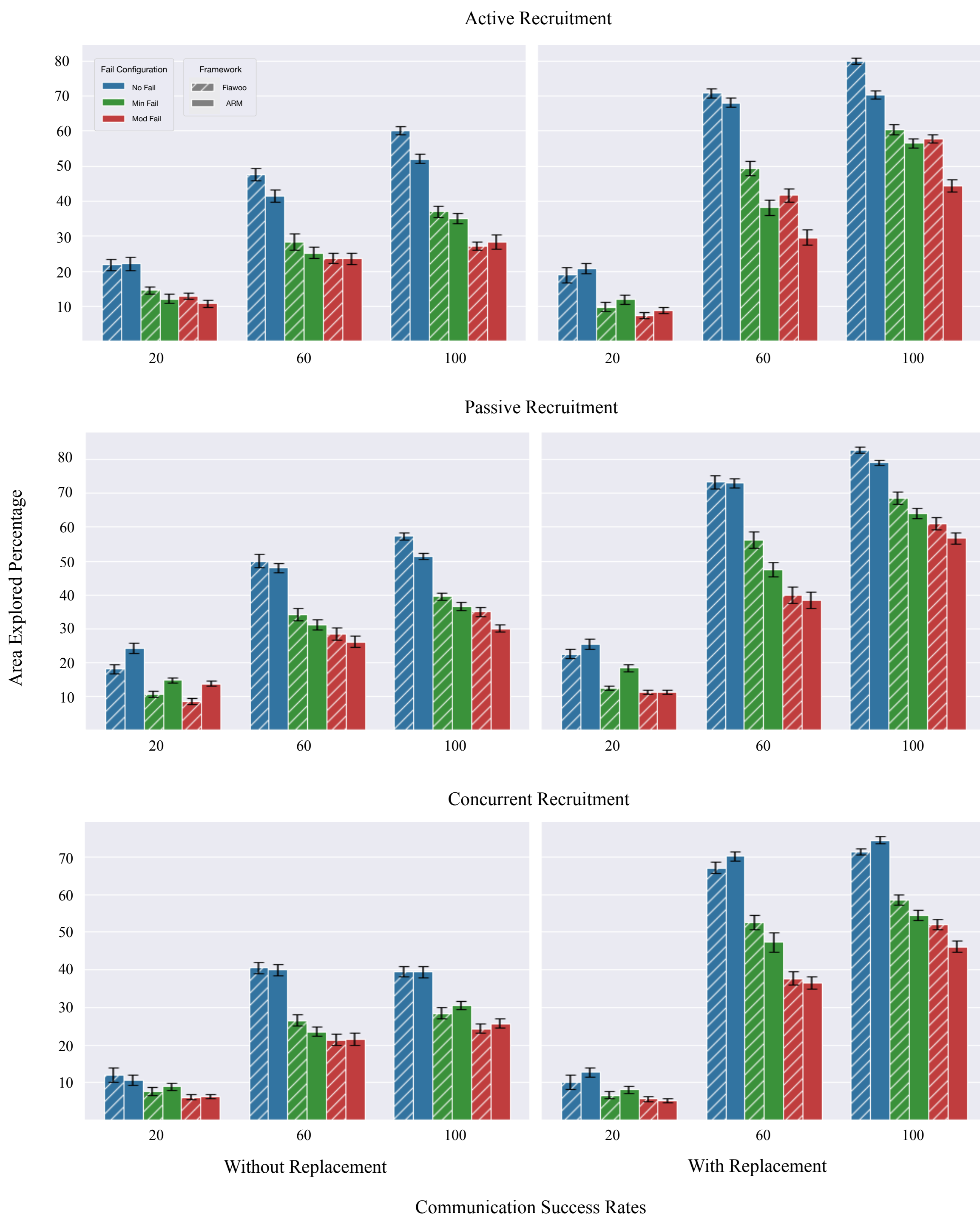

Figure 5.2: Percentage of area coverage using various configuration

5.4.3 Area Revisited

One of the pivotal indicators of a robot's exploration efficacy lies in its ability to explore areas without repeatedly visiting already explored sections. This efficacy is crucial in robotic operations where minimizing redundancy and maximizing coverage is essential. The ARM framework consistently demonstrates this efficacy across all configurations, revisiting areas significantly less than its Fiawoo counterpart, as seen in Figure 5.3. This difference can be attributed to ARM's advanced algorithms prioritizing escaping local minima and actively avoiding getting stuck, thereby ensuring a more streamlined exploration process.

Looking at the performance across different recruitment strategies reveals interesting patterns in both ARM frameworks and the baseline. In the Active Recruitment strategy, the gap in exploration efficacy between the two becomes more pronounced. ARM achieves a peak re-exploration coverage of 13,505.86 m^2 under optimal 100% communication rate conditions, with replacement robots and no failures. In contrast, the baseline covers a higher area of 16,985.17 m^2 under the same configuration. This difference underscores the intricate balance between task-specific exploration and overall coverage, with ARM's algorithmic approach showcasing its efficacy.

Conversely, the exploration patterns in the Concurrent Recruitment strategy exhibit a closer alignment between ARM and the baseline. Both frameworks demonstrate a peak coverage around the 16,000 m^2 mark, with ARM slightly edging out the baseline. On the other hand, the Passive Recruitment strategy paints a less favorable picture for both frameworks. In this approach, Fiawoo achieves a peak re-exploration coverage of 22,265.66 m^2, whereas ARM's coverage reaches 18,775.31 m^2.

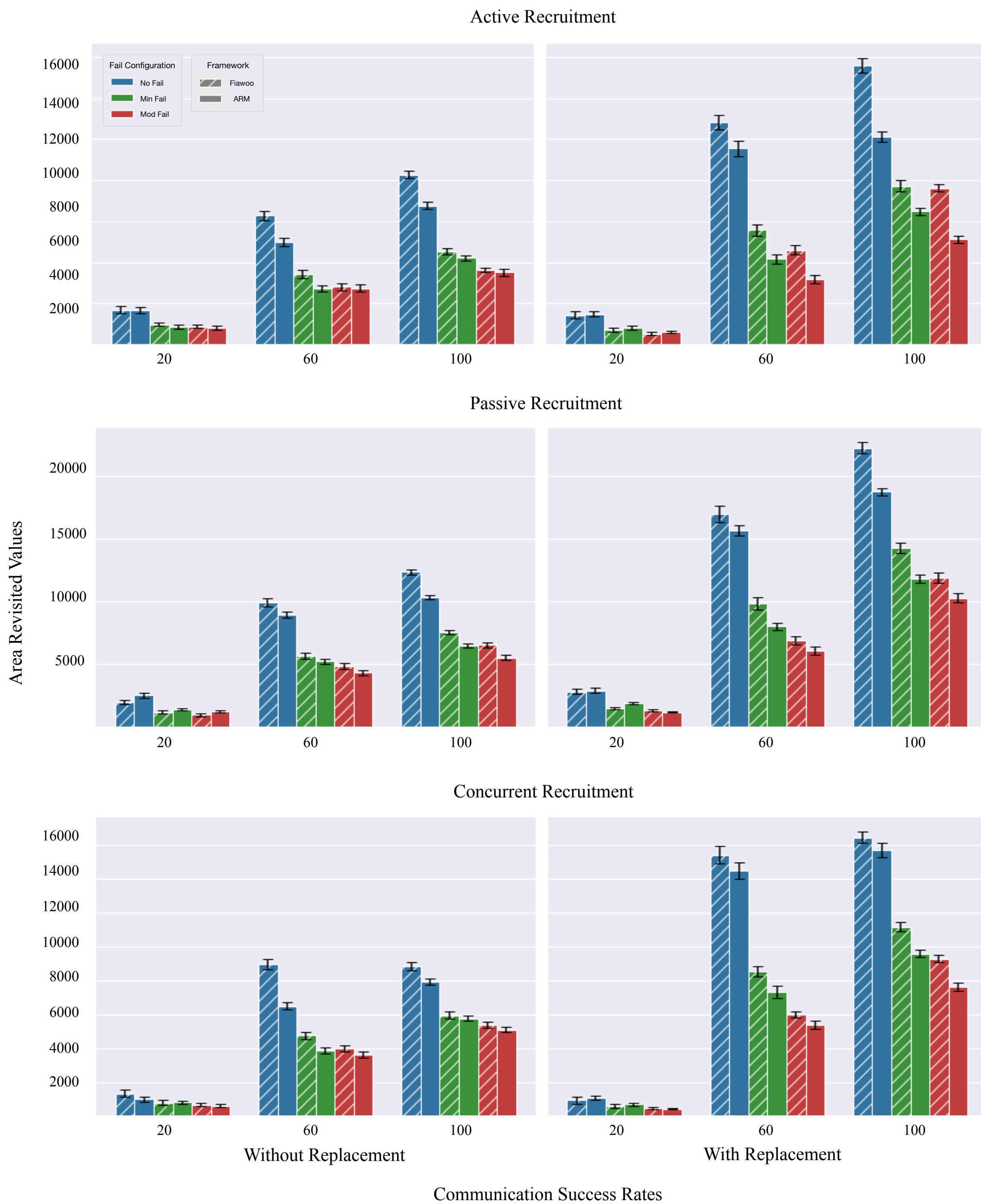

Figure 5.3: Percentage of revisited area coverage using various configuration

The disparity in performance across recruitment strategies can be attributed to the inherent differences in their operational dynamics. In the Active strategy, robots are equipped with a dual-purpose task queue encompassing recruitment and individual tasks. This focused approach minimizes random wandering, leading to more structured and purpose-driven exploration. Conversely, the Concurrent strategy capitalizes on the higher chances of robot encounters through recruitment broadcasts, facilitating task-oriented exploration and reducing aimless wandering.

Contrarily, the Passive Recruitment strategy reveals a suboptimal performance trend for both frameworks. Relying only on chance encounters for recruitment results in repetitive exploration and increased revisited area coverage. The Passive strategy's dependency on random encounters underscores its inefficiency compared to the more structured Active and Concurrent strategies, highlighting potential areas for optimization in future iterations of robotic exploration frameworks.

5.4.4 Fire Quenched

Based on the data presented in Figure 5.4, we can see that Fiawoo's and ARM's Active recruitment strategies showed no change in the number of fire quenches despite variations in communication success rate when no replacement robots were involved. Fiawoo extinguished around 13 fires, while ARM extinguished 40. Regarding replacement robots, ARM slightly improves performance with better communication rates, while Fiawoo's performance declines when communication improves. When no failure occurs and replacement robots are deployed, Fiawoo demonstrated the highest fire quenching values, with up to 45.05 fires quenched at a 20% communication success rate. On the other hand, ARM achieved a significantly higher fire quenching value of 70.03 fires quenched under identical conditions but with a 100% communication success rate.

In the passive recruitment strategy, Fiawoo's performance remains consistent across different communication rates when no replacement robots are used, showing no variation in the number of fires quenched, which averages around 22. On the other hand, ARM demonstrates an increasing trend in the number of fires quenched only in the no-fail scenario, peaking at around 45, with the performance being consistent across both minimal and moderate fail cases. When replacement robots are employed, both Fiawoo and ARM show consistent performance across all communication rates, albeit with minor fluctuations. The highest for Fiawoo is 43.15 fires under the no-fail with replacement configuration at a 100% communication success rate. At the same time, ARM is 61.875 fires under the same configuration.

In the concurrent recruitment strategy, the performance of Fiawoo remains consistent across different communication rates when no replacement robots are used. The number of fires quenched by Fiawoo with no replacement varies minimally, with the peak performance seen in the no-fail configuration, which quenched around 25. On the other hand, ARM exhibits fluctuations in performance without replacement, peaking at the 20% communication rate, where it managed to quench 39.05 fires. For both Fiawoo and ARM with replacement robots, there are variations in the number of fires quenched across different communication rates. Fiawoo's peak performance with replacement is observed at the 20% communication rate, where it quenched 43.2 fires, while ARM reached its peak at the same communication rate, quenching 53.625

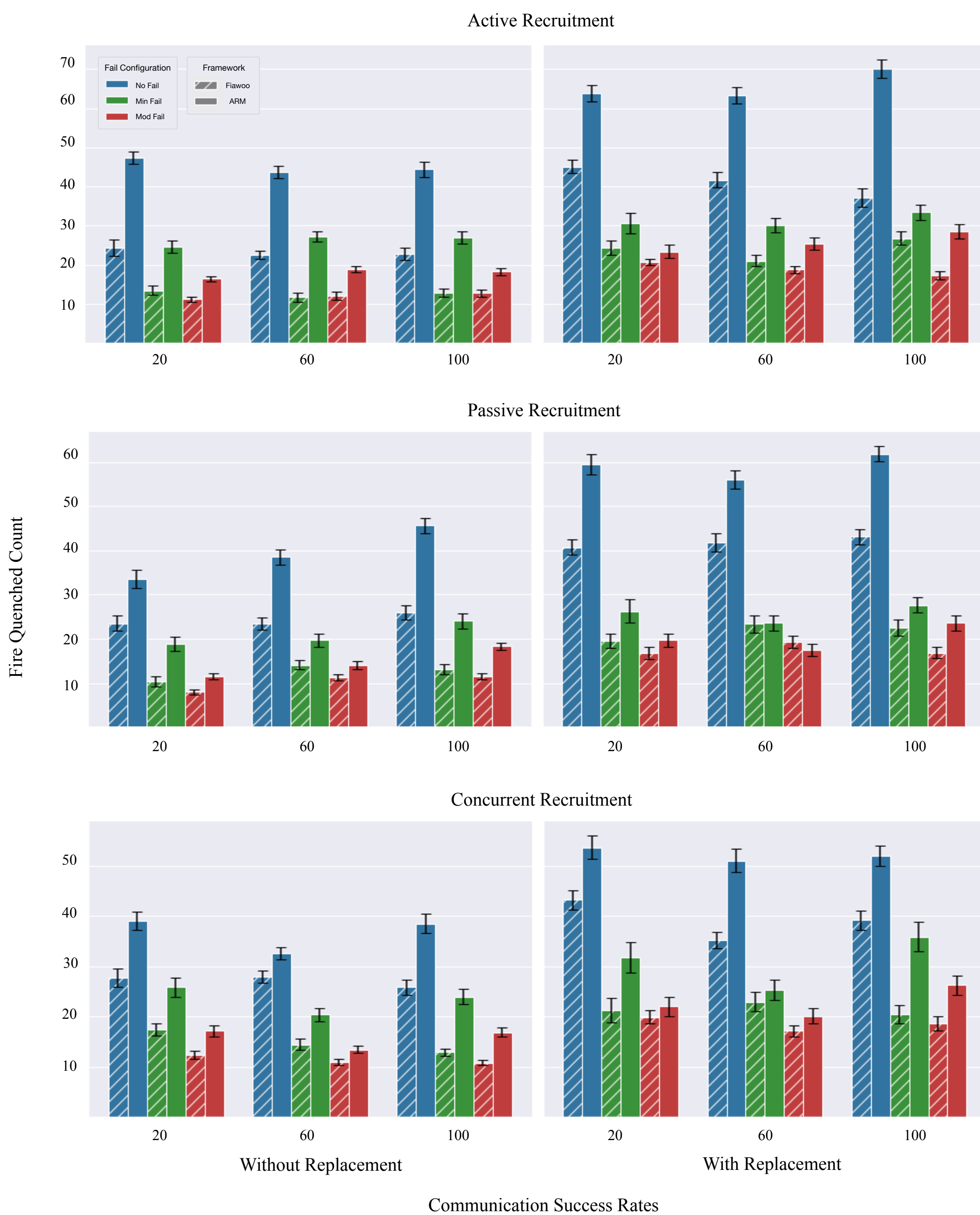

Figure 5.4: Number of fire quenched using various configuration

fires.

Across the Active, Passive, and Concurrent recruitment strategies, the communication rate does not significantly impact the number of fires quenched due to the unique role of Firebots in this task. Since Firebots are the sole units equipped with fire sensors and primarily responsible for fire detection, they invariably create and complete their own tasks. There are limited instances where multiple Firebots may be present on a team, potentially leading to a detected fire being assigned to another Firebot. In the case of ARM, the slight increase in the number of fires quenched at higher communication rates is attributed to other robots upon entering a fire-affected area, prompting their slow movement to be reported as a risk mitigation task. However, this increment is relatively minor as the Firebot already detects the fires before the risk report.

For Fiawoo, the choice of recruitment strategy does not yield any discernible difference in the number of fires quenched. The reason behind this uniformity is the exclusive role of Firebots in fire detection and task completion. Except for rare cases involving multiple Fireboats on the same team, the Fireboats autonomously detect and address the fires. The slight discrepancies observed in the 'with replacement' cases are due to the potential presence of multiple Firebots on a team. In contrast, the recruitment and risk mitigation processes utilize distinct communication channels for ARM, leading to variations in the number of fires quenched across different recruitment strategies.

The analysis reveals that the active recruitment strategy generally outperforms the passive and concurrent strategies regarding the number of fires quenched. The active strategy focuses intensively on team assembly and environmental risk management, potentially diluting its efficacy in victim confirmation tasks, as discussed in 5.4.1. In contrast, the passive and concurrent strategies reduce the immediacy of recruitment demands, allowing for a more concentrated effort on victim confirmation.

5.4.5 Debris Removed

As shown in Figure 5.5, for both Fiawoo and ARM, the active recruitment strategy consistently demonstrates the highest performance in terms of debris removal. In the case of ARM, without replacement, the performance peaks at a no-fail configuration with approximately 55.125 debris items removed. When replacement is considered, ARM's peak performance jumps to around 72.03 debris items. Fiawoo, on the other hand, achieves lower debris removal numbers, peaking at 38.23 debris items without replacement, and with replacement, it significantly increases to 65.51 debris items.

The gap between Fiawoo and ARM's performance is more pronounced in the passive recruitment strategy. Unlike the active strategy, where robots abandon their tasks to actively search for a specific robot to recruit, in the passive strategy, robots rely on chance encounters for recruitment. This means Debrisbot mostly clears debris independently or upon these chance recruitments. The ARM framework has a more efficient way of managing risk mitigation tasks. It operates independently of the recruitment strategy, informing Debrisbot of debris locations through communication, which enhances the overall efficacy of the debris removal process. Specifically, ARM achieves a peak debris removal of around 57.825 without replacement and 68.875 with replacement, whereas Fiawoo's passive strategy yields a peak performance of

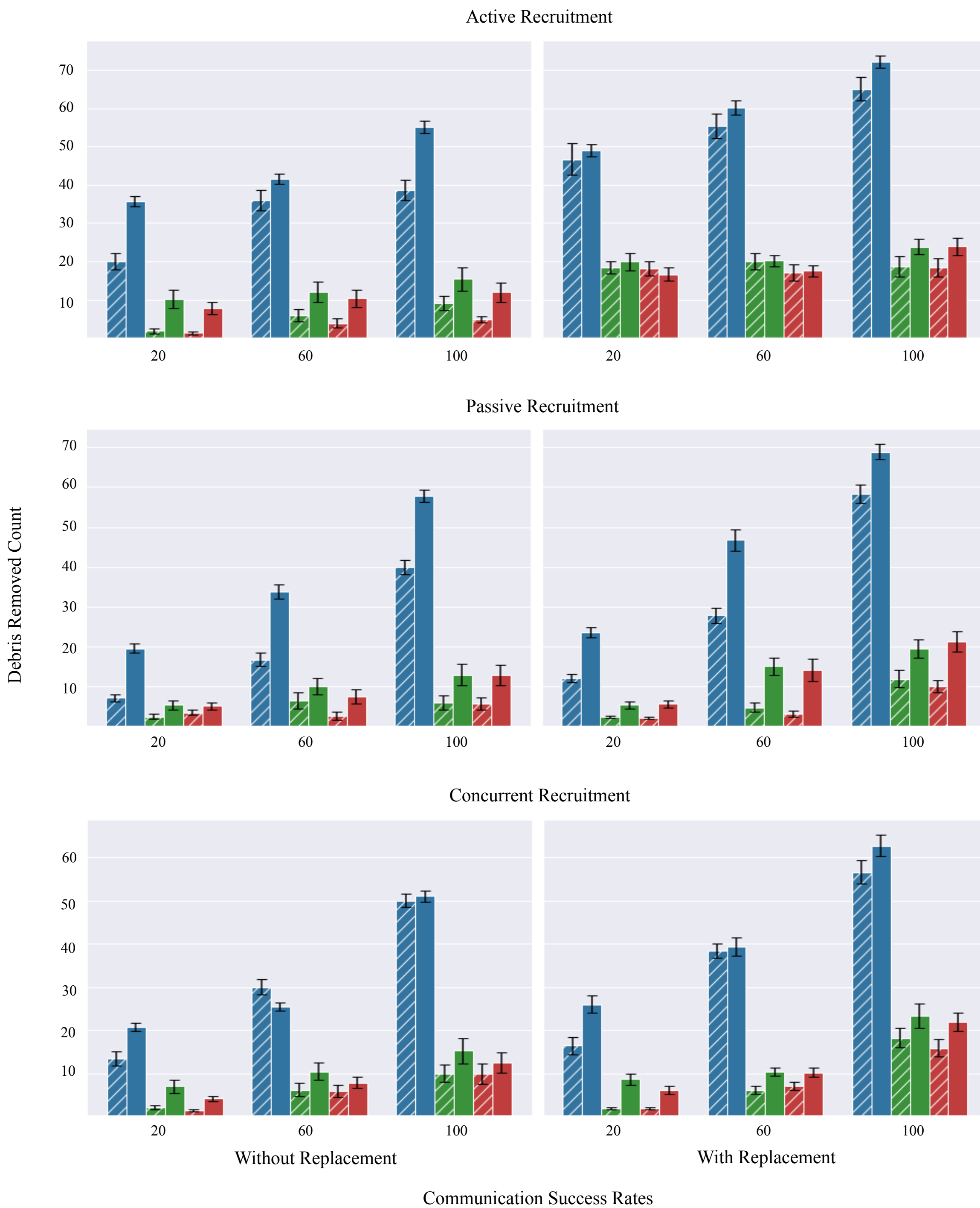

Figure 5.5: Number of debris removed using various configuration

40.675 debris items without replacement and 58.1 with replacement. Regarding the concurrent strategy, ARM peaked at around 51.0 debris removal without replacement and 62.7 with replacement. Fiawoo performs slightly better than the passive strategy, reaching 50.675 debris items without replacement and 56.925 with replacement.

Interestingly, when there is a chance of failure, whether minimal or moderate, the Debrisbot struggles to perform the debris removal task adequately. This results in fewer debris removals as the Debrisbot primarily focuses on clearing debris near the entrance and aimlessly wanders in the environment afterward. Unlike fires, which tend to cluster together and are easily detectable by Firebots, debris is more scattered and not propagatable, making the chances of a Debrisbot encountering debris much lower. This likely explains the consistently low performance of debris removal across minimal and moderate failure configurations.

5.5 Chapter Overview

In this chapter, I have presented and discussed the performance of the methodology I developed. I compared my approach with Fiawoo's work, discussed the metrics I used to evaluate my framework and presented my results.

Chapter 6

Conclusion

6.1 Chapter Overview

In this chapter, I will start by revisiting my research questions and providing a summary of how the results discussed in Chapter 5 address them. The main contributions of my thesis will be presented in Section 6.3, followed by a discussion of future research directions in Section 6.4.

6.2 Answers to Research Questions

1. **To what degree do active risk management strategies impact the overall mission performance and safety of heterogeneous robot teams in dangerous domains?**

   Active risk management strategies significantly influence the overall mission performance and safety of heterogeneous robot teams in dangerous domains, particularly in exploration and victim confirmation tasks. The choice of recruitment strategy is pivotal, with the Passive and Concurrent approaches outperforming the Active strategy in victim confirmation tasks. A higher victim confirmation rate correlates with efficient exploration, highlighting the contribution of passive and concurrent strategies to better exploration and mission performance. The ARM framework's consistent, superior performance across most configurations underscores its effectiveness in managing risk and enhancing mission performance, suggesting potential for optimization in balancing risk management with exploration and victim confirmation tasks.

2. **How effectively can we balance active risk management with other mission objectives (maximizing explored areas and finding victims) in teams of heterogeneous robots operating in dangerous domains?**

   Balancing active risk management with other mission objectives, such as maximizing explored areas and finding victims, is a critical challenge for heterogeneous robot teams in hazardous environments. The choice of recruitment strategy significantly affects the ARM framework's efficacy in victim confirmation tasks. The Active recruitment approach, focusing on team assembly and environmental risk management, may compromise its effectiveness in victim confirmation tasks and exploration efficiency. In contrast, the Passive and Concurrent strategies enable a more concentrated effort on victim confirmation by reducing the immediacy of recruitment demands. This focused approach contributes to higher victim identification and confirmation success rates, enhancing exploration efficiency and mission performance.

Since risk management and team management utilize the same communication channels, focusing on allocating resources to these two systems can result in a more balanced active risk management approach. This approach can avoid undermining the mission's main objective and benefit it by optimizing resource allocation. For example, an overemphasis on risk mitigation, particularly in active and passive strategies, can divert robots from their primary exploration and victim confirmation tasks, negatively impacting mission performance.

3. **What are the potential challenges and limitations of sacrificing resources or recovering damaged/lost robots in hazardous domains and how can risk management strategies be adapted accordingly?**

   In active risk management for robotic missions, balancing mitigating risks and achieving the main objectives (e.g., exploration and victim confirmation) is key. Overemphasis on risk resolution can derail these objectives, while ignoring risks may lead to mission failure due to operational disruptions. Finding a middle ground that keeps the mission both effective and efficient is crucial. When freeing stuck robots, decision-making must carefully weigh the risks and time involved. The current method, which primarily considers proximity for determining which robot to free first, might overlook the importance of a robot's role in the mission, suggesting a need for a more nuanced approach.

   Adapting to challenges and limitations within this context can benefit from a learning-based approach, where machine learning algorithms enable robots to learn from past experiences and environmental interactions. This would allow for better risk assessment, task prioritization, and decision-making based on a comprehensive analysis of past incidents. Such an adaptive learning strategy promises to refine risk management in real-time, improving the safety and efficiency of missions in hazardous environments by ensuring that the robots' actions align with mission goals and the dynamic conditions they encounter.

6.3 Contributions

This thesis has made a number of significant contributions that enhance the capabilities of the existing framework:

1. A framework that equips robots with the ability to actively address environmental risks, including fires, debris, and situations where robots become stuck, enhancing the robot team's overall risk management capability.

2. Incorporating a communication-based risk information relay system that allows capable robots to receive and act upon risk reports, facilitating efficient and targeted risk mitigation tasks within the environment.

3. Improving navigation efficiency by introducing count-based loop detection and an EMA-based collision detection mechanism that enables robots to recognize when they are stuck in these local minima and take them out of those situations, improving navigation efficiency.

4. A fire propagation feature that allows fires to spread dynamically, making the environment more unpredictable and challenging, thereby increasing the simulation's realism and complexity.

6.4 Future work

While significant progress has been achieved through this research, it has also highlighted new questions and challenges that need to be addressed. The following areas for future work aim to build on the current accomplishments, focusing on enhancing communication systems and improving risk mitigation strategies. These advancements are essential to further optimize the performance and reliability of the heterogeneous robot teams in complex and dynamic environments.

6.4.1 Communication Enhancements

1. **Balancing the Messages in Communication Channels:** Investigate methods to optimize and balance the messages transmitted across the communication channels to avoid overloading the robots with conflicting tasks. By implementing efficient message scheduling and prioritization algorithms (discussed in 6.4.2), the system can ensure that the robots receive clear and concise instructions regarding risk mitigation and mission objectives. This will help prevent the robots from being overwhelmed and confused by conflicting tasks, thereby improving their performance and the overall mission performance and safety of the heterogeneous robot teams.

2. **Implementing Protocols Robust to Communication Failures:** Implement redundancy by establishing backup communication links and developing robust communication protocols resilient to communication failures. The system can ensure continuous and reliable data transmission, even during commu-

nication disruptions and failures, by creating redundant communication paths and incorporating error-checking and correction mechanisms into the communication protocols. This will enhance the system's resilience to communication failures, improve the overall reliability and robustness of the communication infrastructure, and ensure the uninterrupted operation and coordination of the heterogeneous robot teams in dangerous and challenging environments.

6.4.2 Risk Mitigation Prioritization

Develop and implement a more sophisticated machine learning algorithm for risk mitigation prioritization. This could involve integrating advanced machine learning techniques, such as reinforcement learning or deep learning, to enable robots to learn and adapt their risk mitigation strategies dynamically based on real-time conditions and mission requirements. Additionally, incorporating context-awareness and environmental factors into the algorithm can help in more accurate risk assessment and prioritization. Right now, we are prioritizing based on proximity; incorporating other features for prioritization could lead to more efficient and effective risk mitigation strategies.

Furthermore, not all hazards present in the environment are disruptive to the mission, especially in areas that have already been explored and identified. In such cases, managing resources in terms of risk mitigation becomes crucial to ensure efficient use of robot capabilities and to avoid unnecessary efforts that do not contribute to the mission's objectives.

6.5 Conclusion

Significant contributions have been made to the field of heterogeneous robot teams, enhancing their capabilities to manage environmental risks such as fires, debris, and navigation challenges. The developed framework includes a communication-based risk information relay system and improved navigation efficiency through advanced detection mechanisms, significantly improving risk management and operational efficacy. Furthermore, the dynamic fire propagation feature has added complexity and realism to simulations, pushing the boundaries of current frameworks.

Despite these advancements, the research has identified new challenges, providing directions for future work. Key areas for further exploration include optimizing communication systems to prevent overload, developing robust protocols for reliable data transmission, and implementing advanced machine learning algorithms for sophisticated risk mitigation prioritization.

Bibliography

R. Ashour, M. Abdelkader, J. Dias, N. I. Almoosa, and T. Taha. Semantic hazard labelling and risk assessment mapping during robot exploration. *IEEE Access*, 10: 16337–16349, 2022.

J. L. Burke, R. R. Murphy, M. D. Coovert, and D. L. Riddle. Moonlight in miami: A field study of human-robot interaction in the context of an urban search and rescue disaster response training exercise. *Hum.-Comput. Interact.*, 19(1):85–116, June 2004. ISSN 0737-0024. doi: 10.1207/s15327051hci1901\%262_5.

J. Casper and R. R. Murphy. Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 33(3):367–385, 2003.

J. L. Casper, M. Micire, and R. R. Murphy. Issues in intelligent robots for search and rescue. In G. R. Gerhart, R. W. Gunderson, and C. M. Shoemaker, editors, *Unmanned Ground Vehicle Technology II*, volume 4024, pages 292 – 302. International Society for Optics and Photonics, SPIE, 2000. doi: 10.1117/12.391640. URL `https://doi.org/10.1117/12.391640`.

A. Corrêa. Distributed team formation in urban disaster environments. In *2014 IEEE*

Symposium on Intelligent Agents (IA), pages 57–64, 2014. doi: 10.1109/IA.2014.7009459.

Y. Cui, J. Lane, R. Voyles, and A. Krishnamoorthy. A new fault tolerance method for field robotics through a self-adaptation architecture. In *2014 IEEE International Symposium on Safety, Security, and Rescue Robotics (2014)*, pages 1–6, 2014. doi: 10.1109/SSRR.2014.7017646.

P. S. Dutta and S. Sen. Forming stable partnerships. *Cognitive Systems Research*, 4 (3):211–221, 2003.

S. Fiawoo. Independent activity and local opportunity for dynamic robot team management in dangerous domains. Master's thesis, Department of Computer Science, University of Manitoba, Winnipeg, Canada, July 2019.

B. Gerkey, R. T. Vaughan, and A. Howard. The Player/Stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics*, volume 1, pages 317–323, 2003.

G. Government of Canada. Hazard and risk - risk assessment: Osh answers, Feb 2023. URL `https://www.ccohs.ca/oshanswers/hsprograms/risk_assessment.html`.

T. Gunn and J. Anderson. Dynamic heterogeneous team formation for robotic urban search and rescue. *Journal of Computer and System Sciences*, 81(3):553–567, 2015. ISSN 0022-0000. doi: https://doi.org/10.1016/j.jcss.2014.11.009. URL `https://www.sciencedirect.com/science/article/pii/S0022000014001500`. Special

Issue on selected papers from the 4th International Conference on Ambient Systems, Networks and Technologies (ANT 2013).

C. H. Horsch. *Urban search and rescue robots, the influence of team membership of robots on team performance*. PhD thesis, Master's thesis, TU Eindhoven, 2012.

M. J. B. Krieger, J.-B. Billeter, and L. Keller. Ant-like task allocation and recruitment in cooperative robots. *Nature*, 406(6799):992–995, Aug 2000. ISSN 1476-4687. doi: 10.1038/35023164. URL `https://doi.org/10.1038/35023164`.

G.-J. M. Kruijff, I. Kruijff-Korbayová, S. Keshavdas, B. Larochelle, M. Janíček, F. Colas, M. Liu, F. Pomerleau, R. Siegwart, M. A. Neerincx, et al. Designing, developing, and deploying systems to support human–robot teams in disaster response. *Advanced Robotics*, 28(23):1547–1570, 2014.

I. Kruijff-Korbayová, L. Freda, M. Gianni, V. Ntouskos, V. Hlaváč, V. Kubelka, E. Zimmermann, H. Surmann, K. Dulic, W. Rottner, et al. Deployment of ground and aerial robots in earthquake-struck amatrice in italy (brief report). In *2016 IEEE international symposium on safety, security, and rescue robotics (SSRR)*, pages 278–279. IEEE, 2016.

T. Linder, V. Tretyakov, S. Blumenthal, P. Molitor, D. Holz, R. Murphy, S. Tadokoro, and H. Surmann. Rescue robots at the collapse of the municipal archive of cologne city: A field report. In *2010 ieee safety security and rescue robotics*, pages 1–6. IEEE, 2010.

Y. Liu and G. Nejat. Robotic urban search and rescue: A survey from the control perspective. *Journal of Intelligent & Robotic Systems*, 72:147–165, 2013.

T. Léauté and B. Faltings. E[dpop]: Distributed constraint optimization under stochastic uncertainty using collaborative sampling. *Proceedings of the IJCAI'09 Distributed Constraint Reasoning Workshop (DCR'09)*, pages 87–101, 2009. URL `http://infoscience.epfl.ch/record/146587`.

A. Majumdar and M. Pavone. How should a robot assess risk? towards an axiomatic theory of risk in robotics. *CoRR*, abs/1710.11040, 2017. URL `http://arxiv.org/abs/1710.11040`.

F. Matsuno, N. Sato, K. Kon, H. Igarashi, T. Kimura, and R. Murphy. *Utilization of Robot Systems in Disaster Sites of the Great Eastern Japan Earthquake*, pages 1–17. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014. doi: 10.1007/978-3-642-40686-7_1. URL `https://doi.org/10.1007/978-3-642-40686-7_1`.

R. R. Murphy. Trial by fire [rescue robots]. *IEEE Robotics Automation Magazine*, 11(3):50–61, 2004. doi: 10.1109/MRA.2004.1337826.

R. R. Murphy, S. Tadokoro, D. Nardi, A. Jacoff, P. Fiorini, H. Choset, and A. M. Erkmen. *Search and Rescue Robotics*, pages 1151–1173. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. ISBN 978-3-540-30301-5. doi: 10.1007/978-3-540-30301-5_51. URL `https://doi.org/10.1007/978-3-540-30301-5_51`.

G. Nagy and J. Anderson. Active recruitment mechanisms for heterogeneous robot teams in dangerous environments. page 276–281, Berlin, Heidelberg, 2016.

Springer-Verlag. ISBN 9783319341101. doi: 10.1007/978-3-319-34111-8_34. URL https://doi.org/10.1007/978-3-319-34111-8_34.

C. Nam and D. A. Shell. Analyzing the sensitivity of the optimal assignment in probabilistic multi-robot task allocation. *IEEE Robotics and Automation Letters*, 2(1):193–200, 2017. doi: 10.1109/LRA.2016.2588138.

L. E. Parker. Alliance: An architecture for fault tolerant multirobot cooperation. *IEEE transactions on robotics and automation*, 14(2):220–240, 1998.

L. Pitonakova, R. Crowder, and S. Bullock. Understanding the role of recruitment in collective robot foraging. pages 264–271, 07 2014. ISBN 9780262326216. doi: 10.7551/978-0-262-32621-6-ch043.

S. S. Ponda, L. B. Johnson, and J. P. How. Distributed chance-constrained task allocation for autonomous multi-agent teams. In *2012 American Control Conference (ACC)*, pages 4528–4533. IEEE, 2012.

S. S. Ponda, L. B. Johnson, and J. P. How. Risk allocation strategies for distributed chance-constrained task allocation. In *2013 American Control Conference*, pages 3230–3236. IEEE, 2013.

R. T. Rockafellar and S. Uryasev. Optimization of conditional value-at-risk. *The Journal of Risk*, 2(3):21–41, 2000. ISSN 1465-1211. doi: 10.21314/jor.2000.038. URL http://dx.doi.org/10.21314/JOR.2000.038.

B. Siciliano, O. Khatib, and T. Kröger. *Springer handbook of robotics*, volume 200. Springer, 2008.

H. Surmann, D. Slomma, S. Grobelny, and R. Grafe. Deployment of aerial robots after a major fire of an industrial hall with hazardous substances, a report. In *2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, pages 40–47. IEEE, 2021.

D. Tihanyi, Y. Lu, O. Karaca, and M. Kamgarpour. Multi-robot task allocation for safe planning under dynamic uncertainties. *CoRR*, abs/2103.01840, 2021. URL `https://arxiv.org/abs/2103.01840`.

A. Undurti and J. P. How. Decentralized risk sharing in teams of unmanned vehicles. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, pages 290–295. IEEE, 2011.

R. Vaughan. Massively multi-robot simulation in stage. *Swarm Intelligence*, 2(2): 189–208, 2008. ISSN 1935-3820. doi: 10.1007/s11721-008-0014-4. URL `https://doi.org/10.1007/s11721-008-0014-4`.

C. Wong, G. Seet, and S. Sim. Multiple-robot systems for usar: Key design attributes and deployment issues. *International Journal of Advanced Robotic Systems*, 8, 03 2011. doi: 10.5772/10532.

L. Zhou and P. Tokekar. Risk-aware submodular optimization for multi-robot coordination, 2020.