Active Recruitment in Dynamic Teams of Heterogeneous Robots

by

Geoff Nagy

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
October 2016

Thesis advisor

John E. Anderson

Author

Geoff Nagy

Active Recruitment in Dynamic Teams of Heterogeneous Robots

Abstract

Using teams of autonomous, heterogeneous robots to operate in dangerous environments has a number of advantages. Among these are cost-effectiveness and the ability to spread out skills among team members. The nature of operating in dangerous domains means that the risk of loss is higher—teams will often lose members and must acquire new ones. In this work, I explore various recruitment strategies for the purpose of improving an existing framework for team management. My additions allow robots to more actively acquire new teams members and assign tasks among other robots on a team without the intervention of a team leader. I evaluate this framework in simulated post-disaster environments where the risk of robot loss is high and communications are often unreliable. My results show that in many scenarios, active recruitment strategies provide significant performance benefits.

Contents

List of Figures

List of Tables

Acknowledgments

First and foremost, I would like to thank my advisor, Dr. John Anderson, whom I could always count on for guidance and support. Despite your many responsibilities your door was always open to me, and I'm truly grateful for all of your insight and to have been your student. Thank you also to my committee members, all of whom I'm sure were very busy at the beginning of this Fall term.

I would also like to thank my family for originally encouraging me to pursue academia, and thus for understanding that it was their fault in the first place I took part in so few social obligations.

This thesis is dedicated to Suzie. It's not possible to put into words the gratitude I feel for your constant support, particularly for the last year. However, allow me to make the attempt, and say that everything I do becomes easier when you're at my side.

Chapter 1

Introduction

1.1 Introduction

The goal of this research is to develop a framework for managing teams of robots operating in dangerous domains, that allows individuals to actively search for useful robots that support or complement a team's skills, while doing immediately useful work on the task at hand. These two goals must be balanced: time spent actively searching means less work is being performed, and vice-versa. In this research, I explore a range of recruitment strategies in order to maximize the performance across robotic teams in dangerous domains as a whole, and examine the balance between searching for new teammates and performing useful work.

This chapter will begin by defining technical terms that will be used throughout this thesis. Following this, I will present the motivation for my research as well as a brief overview of my approach. I will then describe the problem domain in which I evaluate my implementation. Lastly, I will present my research questions and outline

the organization of this thesis.

1.2 Terminology

The following are some technical terms that will be used throughout my thesis.

- *Heterogeneous robots*—My work is specifically concerned with differences in locomotion, sensing, and ability to affect the environment [Koes et al., 2005; Gunn and Anderson, 2015], insofar as they affect the types of tasks that a robot is able to perform. Robots in my work will also differ in computational capacity, although they will be assumed to be running the same software, albeit in different configurations to account for on-board processing limitations.

- *Task*—For the purposes of this research, a task is a single instance of physical work that a robot can complete. Example tasks include putting out a fire, disarming a bomb, or cleaning up a toxic waste spill. In my work, most tasks are completed only by a single robot [Krieger and Billeter, 2000; Gunn and Anderson, 2015; Kiener and Von Stryk, 2007], rather than requiring the efforts of multiple robots at the same time [Dos Sos and Bazzan, 2011; Matarić et al., 1995]. My approach allows sequential tasks contributing to an overall goal to be completed by any number of robots, but I avoid direct parallelism, with the exception of a special type of task where robots in my work are required to physically exchange useful resources (Section 4.3.2.2).

- *Role*—A role in my work is defined by the tasks that a robot fulfilling that role is expected to be able to complete [Gunn and Anderson, 2015]. For example,

a robot fulfilling a firefighting role should be able to put out fires and navigate through debris while locating victims. As in previous work [Gunn and Anderson, 2015], roles in my work act as a stereotype to provide fast reasoning about a robot's abilities, avoiding the need to inquire about a robot's specific skill set unless absolutely necessary.

- *Team*—For the purposes of my research, I define a team as a group of robots who have (possibly imperfect) knowledge of the team's membership and its abilities, and share a common overall purpose. Teams actions are directed by a single leader (one particular role), who assigns tasks to the other members, among other responsibilities. A robot by itself is considered to be the leader of its own single-robot team, regardless of its suitability for that role. My work features multiple teams in large environments, and they are expected to work together non-competitively in order to complete a common goal [Krieger and Billeter, 2000; Pitonakova et al., 2014]. While to be effective in a dangerous domain a team must consider the value and risk to individual members posed by tasks, team members in my work are non-selfish [Dutta and Sen, 2003]. For example, a team member cannot refuse a task on the grounds that it may result in self-destruction. While multiple real-world agencies providing robots (e.g., for a mining operation or minefield clearing) might be concerned if the risk of destruction was not amortized equally among providers, that is outside the scope of this thesis.

- *Recruitment*—In my work, recruitment is a task that takes the form of a request for assistance (as in previous work [Gage and Murphy, 2004]), either from a

specific robot or from anyone available. This request can be satisfied via an active search for help (at which time little or no immediately useful work is done) or concurrently with another useful task, at the cost of lessening the chances of a successful request. Robots in my work are expected to cooperate as much as possible and will generally accept recruitment requests except when they are prevented from doing so by the environment (e.g., communication issues) or a lack of skills or resources on their part. It is also important to note that recruitment in my work, unlike in many others [Gage and Murphy, 2004; Mathews et al., 2011], may necessitate a physical search since desired robotic agents may not be immediately available or in communications range. Recruitment in my work is a means for a team to acquire new skills, either temporarily or more permanently, so that useful work can be performed that could not be otherwise completed, or that would otherwise be completed less adequately.

1.3 Motivation

Cooperative efforts between agents with common goals often yield better results than independent efforts [Gage and Murphy, 2004; Long et al., 2005]. This is true for both humans and robots, and is largely due to parallelism and specialization. Parallelism allows agents to complete multiple tasks at the same time, and specialization is useful because it means that no agent needs to be a jack-of-all-trades—the team overall will possess all of the skills necessary to complete a set of tasks.

In heterogeneous teams, certain robots will have skills or hardware that their

teammates may not [Koes et al., 2005; Parker, 1998]. The need for specialization arises for many reasons—it may be too costly to provide every member of a team with the same equipment given the value of the work, and equipment that is not always needed may increase design and control complexity of robots [Brooks, 1986]. Both of these factors are exceptionally relevant in dangerous domains such as Urban Search And Rescue (USAR, Section 1.5.1) or ordnance disposal, where damage and destruction is to be expected [Murphy et al., 2000; Carlson and Murphy, 2005]. In such domains, losses can be reduced by spreading risk out over more robots [Howard et al., 2006; Carnegie, 2007], and similarly by associating risk inversely proportionally to the value of particular types of equipment. That is, cheaper robots should be risked more often. This approach is also beneficial because it also results in robots that are parsimonious in design and control, as fewer abilities means greater simplicity.

The disadvantage of this approach is that when losses do occur—either through damage, or robots becoming lost or otherwise separated from the team—the overall performance of the team may be decreased due to the loss of some special skill or ability. To counter these situations, replacement robots must be available—this is necessary in dangerous environments since all robots would eventually become disabled, destroyed, or lost. Such replacement robots might be individuals released into the environment at a later time, previously lost members finding new teams, or even vestigial teams that have too few members to function usefully. Any team must be prepared to integrate replacement members into its existing structure. A team must also recognize when such robots are actually needed. Adding any new member makes the team more of a challenge to coordinate and communicate with, and makes the

team more unwieldy to move through the environment while keeping members close. Gaining a member must be weighed against the overhead that member creates. The effort needed to find and integrate replacements must also be balanced with that of doing useful work in the domain.

The adaptation necessary on the part of a team to accept new member(s) is similarly reflected in the requirements for replacement robots. Robots that are released into the environment individually, or those that become lost from their team, must recognize their situation and act on it. Such action may involve encountering or seeking out other teams to join, or forming a new team that would then grow and adapt by encountering other robots or teams over time [Gunn and Anderson, 2015]. Teams can similarly balance skills by shifting members between one another, or merging. A secondary advantage of joining new teams in this way is an indirect exchange of information, since new members bring knowledge with them [Gunn and Anderson, 2015; Krieger and Billeter, 2000].

The problem of maintaining adequate team members interacts with the problem of adequately distributing tasks among members of a team (this distribution is commonly referred to as *task allocation*). Because of the continual changes in team membership dictated by dangerous domains, robotic teams will not often have an ideal combination of skills [Gunn and Anderson, 2013]. This means that when a task needs to be assigned to some member of a team, it may have to be assigned to a robot not ideally skilled in completing it. Tasks might take longer because of this, or produce lower-quality results. Some tasks might also be completely beyond the capabilities of any current team member. The fact that a task cannot be properly

assigned to a robot because no one is well-suited to perform it, however, can be used as an indicator that new robots need to be acquired.

In these situations, robots can expend varying levels of effort towards locating other robots that have desired skills. More active approaches that involve physical searches may yield faster results at the expense of immediately useful work. Passive approaches, such as those that rely on chance encounters with others as a consequence of performing normal work, will result in more useful work being performed, but may lessen the chances of successfully locating desired robots. These varying levels of recruitment effort form a spectrum with passive strategies on one end and active approaches on the other. Strategies in the middle of this spectrum would contain elements of both extremes, such as simply being aware of potential recruitment opportunities in the distance, rather than actively seeking them out.

There is significant motivation for allowing robots to recruit one another. Much like in human-human interaction, asking someone to lend a quick favour can be beneficial to one party while not overly inconveniencing another. In workplace environments, employees are also posted or transferred more permanently so that their skills can be put to better use more consistently.

Despite the broad range of strategies that can be employed across the recruitment spectrum, existing approaches do not cover much of this breadth. Almost all work on recruitment embraces strategies from the active end of the spectrum ([Dos Sos and Bazzan, 2011; Gage and Murphy, 2004; Mathews et al., 2011]). At the time of writing, there is only one previous work [Gunn and Anderson, 2015] I am aware of that uses passive recruitment as it is defined in my thesis. Although there is work

that explores coalition formation between agents [Dutta and Sen, 2003; Van De Vijsel and Anderson, 2004], these works focus on highly abstract domains that make comparison to more realistic systems, as well as classification in terms of recruitment, difficult. There is also currently little or no research that incorporates elements of both extremes. A separate issue is that much work focusing on team management or recruitment has done so from a theoretical standpoint, with a focus on efficient team management without considering the complications that arise in real-world scenarios such as robot losses, equipment failure, or unreliable communications.

Our lab's prior work [Gunn and Anderson, 2015] has relied entirely on passive strategies, by using encounters between teams (including single robots) to continually re-balance membership as opposed to trying to recognize when more active recruitment might be necessary. There are times when the need for this type of recruitment is recognizable: robots may be overwhelmed with too many tasks, robots may be repeatedly tasked with activities to which they are not well suited, or team leaders may find it necessary to search extensively to find the skills necessary to allocate a task.

Once active recruitment is available, balancing this activity with pursuing useful individual work is also important: if a robot is exclusively working on recruitment, other tasks that it could be completing must wait or be done by others. To consider this balance, my work includes a middle ground: communicating recruitment needs while not tasking a robot solely with recruitment. This allows a robot to interleave the performance of useful work with recruitment. The difference between this and merely taking advantage of chance encounters is that in the former approach, robots in

the distance could be actively investigated, rather than only attempting to enhance a team or divide skills between them when they are encountered in close proximity. The likelihood of performing such an investigation is then proportional to the importance of the recruitment opportunity and that of the other tasks the robot has.

1.4 Approach

My work expands on previous research [Gunn and Anderson, 2015], which resulted in a framework for team management and task allocation for dangerous domains. In this framework, teams and individual robots had the ability to recruit others and integrate them, both individually and from existing teams, in order to create more effective teams overall. This was done in the context of team rebalancing and might involve recruiting only one member wholesale swapping of many individuals between teams. However, this approach was entirely reactive—that is, team membership changes could only occur as a consequence of encountering other robots or teams. My thesis research improves this framework by incorporating additional elements of team management to more actively seek out new robots when necessary, while balancing work performed on team management with productive work in the robots' domain.

Like prior work [Gunn and Anderson, 2015], I assume the existence of multiple teams (including lost single robots or replacements), and that encountering these will allow team re-balancing. However, I also take advantage of opportunities for active recruitment by being able to task robots specifically with searching for and recruiting others with particular skills, either temporarily without changing their

team membership, or permanently by inducting them into the team. Similar to this work, I also take into account a number of factors particular to domains such as Urban Search and Rescue (e.g., communication can be sporadic and unreliable due to interference and damaged infrastructure), but my framework is intended to be applicable to a broad range of domains. The fact that my approach is intended to handle degraded communication, for example, does not preclude the possibility of using my framework in domains where communication functions properly.

In addition to supporting a passive team management strategy for comparison (that of [Gunn and Anderson, 2015]), my implementation supports two different modes of recruitment: *concurrent* recruitment, whereby robots perform tasks while also broadcasting recruitment requests as needed (thus still allowing useful work to be completed), and *active* recruitment, whereby robots performing recruitment tasks do so while ignoring other outstanding tasks, at the cost of performing little or no immediately useful work.

All robots in either a concurrent or active recruitment configuration are able to recruit other robots, and can be recruited themselves. A recruitment request takes the form of a wireless broadcast requesting a specific skill. The success of such a request is dictated by factors such as the availability of suitable recruits, success rates of potentially sporadic wireless communications, and other considerations that will be described in detail in Chapters 3 and 4.

Recruitment requests in my implementation take one of two possible forms. Permanent *role-level* recruitment is requested in situations where it is determined that a team needs someone to fill a particular role. This type of recruitment does not involve

a specific task, and the robot that agrees to this request changes teams to join the recruiting one, filling the desired role and avoiding the need for another (possibly less-suited) team member to fill that role instead. The other form of recruitment in my approach is *task-level* recruitment, which is used in situations where a team requires only a single task to be completed, and lacks the ability to complete the task adequately with its current members. By allowing robots to be recruited temporarily for a specific task by another team, that team can temporarily gain the abilities of a robot without the overhead of managing that robot long-term. This is also advantageous in that it prevents other teams from having their skill set permanently diminished. This, of course, assumes that the recruited robot can make its way back to its team, since that team likely continued to move during the robot's absence. Robots might refuse such requests in cases where the task to be carried out is physically far enough away that the robot might not be able to reliably return to its own team. Robots may also refuse requests to avoid being recruited back and forth (i.e., thrashing), since this oscillation may prevent useful work from being completed. These and related issues, and the manner in which my framework addresses them, are discussed in detail in Chapter 3.

The next section provides an outline of my evaluation domain, the simulation environment, and the types of robots and abilities featured in my example implementation.

1.5 Evaluation

The following is a description of the problem area around which I have chosen to focus my implementation, the specific simulation environment developed to evaluate my work, and the robots that operate in this environment.

1.5.1 Urban Search and Rescue

Urban Search and Rescue (USAR) is a domain in which a collapsed structure is explored to locate and extract human casualties. Such environments are extremely dangerous to victims and rescuers alike due to the risk of further collapse, limited breathable atmosphere (e.g., dust or toxic gases), or fire. A well-known example of this domain is the aftermath of the September 11 attacks. In such scenarios, rescue operations may take several days depending on the severity of the disaster [Murphy et al., 2000]. USAR rescuers will also make use of trained dogs to locate casualties, although there are risks that a dog may get caught by the collar when it is carrying sensors or video equipment [Ferworn et al., 2006] that would be useful to survey the area. Additionally, rescue dogs are also not always available and tire easily [Statheropoulos et al., 2015].

Another solution involves the use of robots that are remotely operated by workers outside of the immediate disaster zone [Murphy et al., 2000]. Teleoperated teams of rescue robots have been used in real-world scenarios and such systems are often extremely difficult to use [Casper and Murphy, 2003]. Teleoperation of remote vehicles in general has been studied previously and it is well-known that operators are often plagued by a lack of situational awareness and any delays in the control loop between

the user and robot are highly problematic [McGovern, 1991; Sheridan, 1992]. Casper [2002] notes the particular difficulties with situational awareness in USAR, especially due to the fact that the fine coating of dust makes it difficult to differentiate objects in the environment, requiring additional cognitive effort to process. Potential solutions to these problems involve providing improved control interfaces [Baker et al., 2004; Kadous et al., 2006] or using robots that posssess increased levels of autonomy in order to lighten the user's cognitive load [Crandall and Goodrich, 2002].

In spite of work improving human control of robots, full robot autonomy remains the ultimate goal. There are many good reasons to perform autonomously in this domain rather than relying on human operators: intelligent systems do not suffer from the same situational awareness limitations and do not tire cognitively or physically [Wong et al., 2004]. Using autonomous robots also avoids issues that arise when accommodating human operators. For example, Casper and Murphy [2002] noted that rescue robotics operators were required by contemporary safety guidelines to move far enough away from a disaster zone that wireless communication with robots became difficult. Communication delays in general also limit the applicability of these solutions to other domains where distance is even more of a factor (e.g., applications of this technology in space).

An attractive aspect of autonomous processing is its potential applicability to other domains. If autonomous solutions can be successful for problems of this difficulty (i.e., robotic USAR), then it should ultimately be deployable to a broad range of more conventional problems.

The framework I will describe in Chapter 3 is specifically concerned with dynamic

teams of autonomous robots operating in dangerous environments. As robots in these environments fail as a result of the hazards that are present, remaining robots must be prepared to adapt and dynamically adjust their roles within these teams in order to compensate for these losses, so that useful work can still be performed. The challenging nature of USAR makes it an ideal candidate for evaluation. Because the necessary equipment and robotic hardware (not to mention the challenges of building a realistic post-disaster mock-up) would be prohibitively expensive, I will be conducting my work in simulation. This will allow me to abstract away details that are not relevant to my framework (such as vision processing or sensor fusion), while also providing a level of repeatability and greatly reducing the amount of time it would take to run these experiments in the real world. It is worth mentioning that USAR research involving physical robots also performs many abstractions, such as simplified victims [Kadous et al., 2006; Tunwannarux et al., 2008] or debris [Kitano and Tadokoro, 2001]. Additionally, highly detailed simulations are not strictly necessary for experiments involving dynamic teamwork and would not serve as a direct comparison with prior techniques in the same way that my implementation does. The following two sections outline major features of my implementation environment and the robots I will be using in my study.

1.5.2 Environment

The implementation of my framework (described in Chapter 4) and its evaluation (described in Chapter 5) make use of the Stage simulator Gerkey et al. [2003]. My experiments require the robots to locate as many human victims as possible in a given

time period. Since my work does not focus on visual identification of humans, I use a simulated object perceivable as a human by a robot with the appropriate perceptual ability. There are also objects of human appearance (false victims), in order to confuse robots without adequate perception—this is the same level of abstraction used in [Gunn and Anderson, 2015] and allows comparison with this work.

1.5.3 Robot Types and Abilities

The types of robots in my implementation reflect the need for heterogeneity in dangerous domains. Robots in my approach vary in sensing, locomotion, and computational abilities.

The first three types of robots in my approach form a three-tiered system similar to previous work [Gunn and Anderson, 2015; Carnegie, 2007; Howard et al., 2006]. The least capable of these robots has limited sensing and locomotion abilities, and frequently needs to request assistance from its more-capable teammates. A more advanced robot exists which possesses excellent sensing capabilities and improved computational strength. The third kind of robot in this framework is larger, slower, and computationally more powerful than the previous two types of robots. Due to their increased computing power and memory, their ideal role is as team leaders. Leaders can traverse debris on their own but have limited sensing capabilities.

To further investigate how recruitment facilities affect the performance of heterogeneous robot teams, I have also implemented a highly-specialized robot which is capable of clearing debris. The environments in my evaluation will contain significant debris, making it more difficult for robots to navigate efficiently. There is also the

risk that a robot may become stuck on a piece of debris, rendering it unable to move. The robots in the three-tiered system mentioned above will require the services of the debris-clearing robot in order to improve team performance in situations where debris is problematic to the current tasks at hand.

The next sections describes the research questions this thesis will aim to answer.

1.6 Research Questions

The following are research questions that I will answer using the approach and evaluation I have described in previous sections.

1. **To what degree does the use of various recruitment strategies improve or hamper the overall performance of teams of heterogeneous robots in dangerous domains?** To answer this question, comparisons of team performance (number of victims located, percentage of environment covered) for three different recruitment strategies (*passive*, *concurrent*, and *active*) will be examined in several randomly generated USAR configurations. Chapter 5 describes these environments and how they are generated.

2. **What factors (e.g., availability of replacement robots, wireless reliability, probability of robot failure) determine the recruitment strategy that should be used in a given situation?** Since my evaluation will examine different combinations of these factors, it will be possible to build a list of criteria for determining optimal recruitment strategies in different scenarios where these factors are known.

3. **Which recruitment strategies, if any, result in overall best performance of teams of heterogeneous robots if wireless reliability levels are not known?** Robots operating in environments with unknown levels of wireless communication success can adopt an initial recruitment strategy, and then adopt new strategies as time progresses and more is learned about the environment. Where communication with the outside world is possible, replacement robots released later on could be pre-configured with this new knowledge.

1.7 Thesis Organization

The remaining chapters of my thesis are outlined as follows.

- **Chapter 2: Related Work**–Reviews literature relevant to this work, such as various recruitment approaches as well as work which features robots aware of their own abilities and the abilities of others. I also provide a description of a framework which forms a starting point for my work.

- **Chapter 3: Methodology**–This section provides a description of the recruitment spectrum, types of recruitment and relevant strategies, and how my framework implements these concepts.

- **Chapter 4: Implementation**–Discusses how I have implemented my framework in an USAR domain, and provides detailed descriptions of the robots, skills, and tasks that enable me to evaluate my framework.

- **Chapter 5: Evaluation**–Describes the experiments I have used to evaluate my framework, and the results I have obtained by doing so.

- **Chapter 6: Conclusion**–Uses my results to answer the research questions I have outlined in the previous section, and provides directions for future work.

Chapter 2

Related Work

The core focus of my thesis is the development and evaluation of a framework which allows robots to recruit the assistance of other robots in a distributed manner in dangerous environments. As mentioned in Section 1.3, previous work does not cover a wide variety of recruitment strategies, and many of these works do not take into consideration the challenges that may arise with multiple robots even when hardware functions perfectly (e.g., a robot that has fallen out of communications range).

This chapter describes related background work and compares and contrasts it to my own. I divide previous work into a number of categories. The first category deals with *dynamic team formation*, which is the core of my thesis. In these works, robots operate in teams and dynamically adjust membership based on the requirements of the work at hand. (Although there is a growing body of research exploring the formation of human-robot teams [Dias et al., 2006; Johnson et al., 2008] such works are outside the scope of this thesis.) *Foraging* is a well-known multi-robot domain, where robots are responsible for collecting resources and returning them to one or

more home bases, and I discuss approaches that explicitly use recruitment techniques to do so. *Self-interested agents* can also use recruitment strategies, but do so competitively without the inherent assumption of mutual cooperation. Works involving *grouping and assembly* often use recruitment facilities to create formations or form useful groups for completing tasks later on in a coordinated fashion. The final section of this chapter deals with *recruitment among heterogeneous robots* whereby robots will ask other robots for assistance in completing tasks, without the explicit use of teams.

2.1 Dynamic Team Formation

The underlying concept behind my work is team formation and maintenance. This is a broad area of study and there are many strategies for maintaining and forming teams to accomplish goals. In this section I describe previous work where team membership can vary. That is, robots may leave or accept new members at any time according to membership rules (e.g., a robot not heard from after a certain amount of time may be considered to have left the team [Gunn and Anderson, 2015], or teams may be purposefully divided in order to satisfy physical constraints [Dasgupta and Cheng, 2015]).

Kiener and Von Stryk [2007] presented a task allocation framework for teams composed of highly heterogeneous and specialized robots. They demonstrated their approach with a wheeled Pioneer robot and a humanoid robot, whose task was to follow a ball down a hallway and then kick it into a goal. By recognizing and pooling their unique capabilities, the two robots were able to complete their task much more

efficiently than either robot would be able to do alone. This is similar to my own research, but different in significant ways. Most importantly, my work is scalable to more complex applications—the work mentioned here only featured two robots and does not include facilities for managing larger teams or more diverse roles. My approach also involves a higher number of tasks with varying priorities at many different locations, meaning that suitable task allocation or recruitment decisions are not immediately apparent to any agent, unlike the work cited above. Additionally, the tasks that were demonstrated in this work were well-defined at the outset of the robots' mission, while the robots in my work are able to discover tasks on their own without any prior knowledge.

Dasgupta and Cheng [2015] described a framework for dynamically dividing up a team of robots into smaller sub-teams to navigate around arbitrarily-shaped obstacles. It is based on the concept of weighted voting games, where every agent is assigned a *weight* and a course of action is determined by the group of agents whose total weight exceeds a given threshold [Elkind et al., 2007]. Upon encountering an obstacle, the robots divide themselves into a number of sub-teams and each group circumnavigates the obstacle. After the sub-teams clear the obstacle, they merge into a larger team. While the idea of maintaining formation around obstacles is well-explored concept (e.g., [Balch and Arkin, 1998; Fredslund and Mataric, 2002]), the approach given by Dasgupta and Cheng [2015] makes improvements that result in faster robot decision-making and fewer numbers of sub-teams. My own framework supports aspects that are similar to this work in that it allows robots to leave existing teams and form new ones, and also supports the possibility of previously separated

robots becoming integrated into a team (either the same one or a different one) later on. The work done by Dasgupta and Cheng [2015], however, is specifically concerned with the division of teams from a spatial context to satisfy physical constraints, i.e., navigation around obstacles. In contrast, the framework I have developed uses team management strategies for the purpose of acquiring skills to complete specific tasks. I also evaluate my approach in more challenging scenarios (e.g., robot failures and unreliable communication) than those presented in the aforementioned study.

Gunn and Anderson [2015] developed a dynamic team management framework for teams of heterogeneous robots operating in dangerous environments. Their framework was evaluated in a simulated USAR environment, although it is intended to be applicable to a wide range of challenging domains. Robots in their approach were responsible for locating and identifying human casualties and performing team maintenance duties as required. The challenging environmental conditions (debris, communication failures, and robot losses) necessitated the introduction of replacement robots, and existing robots were expected to integrate them (as well as previously lost robots that were rediscovered) into their existing team structures as appropriate. The heterogeneous nature of the robots in this work meant that as robot losses occurred, skills were lost. A major limitation of this work is that robots did not have the ability to explicitly search for robots possessing needed skills: only chance encounters with other robots could be used to bolster a team's abilities. My thesis work uses this framework as a starting point for the development of active recruitment strategies which address this limitation. I also introduce new robot mechanics (such as victim-identification markers and debris-removal equipment) to evaluate my

approach. The framework developed by Gunn and Anderson [2015] forms an integral part of my work and is described in more detail in Chapter 3.

2.2 Recruitment in Robotic Foraging Tasks

Robotic foraging involves the use of multiple robots to collect resources in a distributed fashion and return those resources to a central home base. There is a significant amount of research into this topic [Matarić, 1997; Song and Vaughan, 2013; Lein and Vaughan, 2009; Shell and Mataric, 2006]—it is a popular area of study due to the fact that there are many analogous real-world applications. For example, robots could be used to gather fruit from orchards, harvest wheat from fields, or remove litter from public areas. Foraging algorithms are often inspired by biological systems such as bees [Tereshko and Loengarov, 2005] or ants [Garnier et al., 2007; Song et al., 2012]. The works described below involve robotic foraging tasks where recruitment is used; that is, robots will not only collect resources but will also guide other teammates to locations where resources are known to be plentiful in order to maximize foraging efficiency.

Krieger and Billeter [2000] modeled ant-like food foraging tasks using a team of homogeneous robots. In this work, parameters such as team size, food distribution, and whether or not recruitment was used, were varied to determine how these factors influenced the performance of a robotic team during a food foraging task. The experimenters measured the "energy level" of the robotic nest, which was diminished by robot efforts and incremented with every food item returned to the base. In one experimental configuration, robots could use recruitment by leading an inactive robot

from the home base to the location of a known food item. When this recruitment function was enabled, there was a significant increase in the minimum nest energy levels compared to when recruitment was not used. Additionally, robots who were inclined to be less active otherwise spent more time doing useful work when recruitment was used. In these cases, recruitment was beneficial because it reduced the amount of time that a robot had to spend actively searching on its own for food. The result was that nest energy levels did not drop as low as they would have had recruitment not been used. One of the main differences between this work and my own is that in this study, robots could only recruit each other at the home base. Redirecting or recruiting an actively searching (or even lost) robot was not possible in this study. Such a feature would be useful in more realistic, larger settings where there is a real risk of robots becoming lost or resources are spread more widely and thus harder to find. My work specifically addresses these situations and assumes that resources, tasks, or robots may be significantly spread out, and robots may be required to perform extensive searching in order to locate them.

Additional work [Pitonakova et al., 2014] has endeavoured to discover under what conditions recruitment is beneficial for foraging tasks involving resources of different value. In this study, several different sizes and values of resources were placed in a large simulated environment, while the number of robots and their recruitment approach was varied. Two different types of experiments were conducted: one with communicating robots that actively recruited others to known resource locations, and another with robots that did not communicate or recruit each other. No single strategy (i.e., either recruitment or no recruitment) was successful in all cases. Robots

performing recruitment did better than individualist robots in cases where resources were rare or difficult to locate, and recruitment helped to lessen congestion around popular locations like the home base. The study also found that recruitment was unnecessary (and even detrimental to performance) if resources were abundant, difficult to find in short periods of time, when information regarding resources was unreliable, or when the cost of communication equipment would outweigh the potential gains of any collected resources. While this work explores the use of homogeneous robots for a single task type, my research involves the use of heterogeneous robots for multiple tasks that are inherently different and thus require different hardware or capabilities. My work takes this into account by equipping robots specifically for different tasks, such that no single robot can perform every task (making recruitment more necessary). Interestingly, the experimenters in this study used a toroidal environment (i.e., one that wraps back around on itself) in their simulations, precluding the possibility that robots can become lost by wandering too far away from a central home base. Despite the large simulation area this contributes to a lessened degree of realism; my simulated environments are both finite and bounded, making it much more likely that robots could become lost.

2.3 Self-Interested Agents

Self-interested agents are those that are generally only concerned with meeting their own goals and do not consider the goals of other agents to be as important. In the context of multi-agent systems, this means that robots are designed to take advantage of or not return favours for other robots [Sen and Dutta, 2002; Dutta

and Sen, 2003; Van De Vijsel and Anderson, 2004], or otherwise are motivated by concepts such as *shame* [Gage and Murphy, 2004], as opposed to operating under the assumption that all robots should behave cooperatively as much as possible even if this results in an unfair workload distribution. This section describes previous work where agents use recruitment and also exhibit a measure of self-interest.

Dutta and Sen [2003] conducted active recruitment simulations among individual self-interested, heterogeneous agents in a highly abstract domain. In this work, a number of generic "tasks" exist, and each agent has exactly one task type at which it is more proficient than the others. Any agent may ask for help from another agent, but only *reciprocal* agents will potentially agree to these requests, if it is determined (either by asking other trustworthy agents or through the potential recruit's own experience) that the requesting agent is likely to return the favour eventually. The second type, *selfish* agents, do not lend help under any circumstances. Dutta and Sen [2003] showed that as agents interact and learn about one another's behaviours, selfish agents effectively become shunned and reciprocal agents perform better—due to their cooperative behaviours—than they would individually. van de Vijsel and Anderson [2005] improved upon this work by adding additional levels of realism, such as a 2D grid environment and randomized agent properties such as speed, trust level towards other agents, and honesty regarding an agent's own ability to complete a task. Additionally, agents did not possess perfect memory and were made to randomly forget task details. Despite these challenges, the agent model proposed by Van de Vijsel and Anderson outperformed the Dutta/Sen model [Dutta and Sen, 2003] considerably. My own work assumes that agents are non-competitive and share a

common overall goal; therefore, robots will not refuse requests for help unless a lack of resources (including proper equipment) prevents them from assisting. Additionally, these works assume that all agents can complete all task types (albeit with varying proficiency) while in reality certain agents may be completely incapable of performing certain tasks. This is a key element in my work: robots may find themselves tasked with jobs that they cannot complete on their own and will have no choice but to request assistance from other robots.

Gage and Murphy [2004] demonstrated three recruitment models whereby unmanned aerial vehicle (UAV) robots would attempt to recruit other ground-based robots to perform investigations of certain areas of a simulated minefield. A *greedy* approach (whereby the UAV would recruit the nearest idle robot), and a *random* approach (whereby the UAV would recruit a random robot) were used as baselines to compare against a third strategy, *affective* recruitment. Using this strategy, ground-based robots had the ability to reject recruitment requests, at the cost of building up *shame*. Each robot had a pre-defined *shame threshold*, and sufficient shame levels would cause the robot to accept a recruitment request. Experiments showed that the affective recruitment model resulted in the least UAV waiting times in situations where communications were sporadic, and that affective recruitment resulted in the least amount of wireless communications when many agents were available. This study involved the use of a heterogeneous robot team with little task variety: all ground-based robots were identical and were only recruited to perform a single type of task. Thus, the closest available robot for a task was always the best recruitment choice. In my own work, an ideal recruitment choice is not always apparent or avail-

able. For example, searching for a distant robot with better skills may be a better choice than settling for a moderately-equipped robot that is nearby. Like my work, this system is tested against sporadic communication failures. However, Gage and Murphy's system does not take into account the possibility that some useful agents may be out of wireless communication range. In these cases, searching mechanisms much like the ones employed in my work would be useful when desired agents are not immediately accessible. Additionally, recruitment in Gage and Murphy [2004] was purely one-way; aerial robots were the only robots performing recruitment, and only did so with ground-based robots. My work allows richer forms of interaction by permitting any robot to potentially recruit another as needed.

Comi et al. [2014] proposed a framework for managing inter-agent trust in domains containing self-interested agents. They prove that their agent model positively influences the network of relationships between agents and also rewards agents which exhibit positive behaviours (e.g., honesty and competency), as opposed to those that behave aggressively or non-virtuously. This is accomplished by tracking a series of metrics formed by agents' (a) trust in receiving good work from another agent (*reliability*), (b) feedback obtained regarding the performance of an agent relative to how said agent promised it would perform (*honesty*), and (c), the competency of an agent assessed by other agents, weighed by their own honesty values (*reputation*). This model is intended to be used in more abstract scenarios (an example given was cloud services [Comi et al., 2014]) and may not be applicable to physical settings involving real robots in more challenging environments. Communication problems can lead to outdated and/or limited knowledge regarding other robots, and although it is worth

noting that this can still occur with abstract networked agents, it becomes much easier for a system of real robots to suffer from communication problems due to limited communication range or hardware failures.

2.4 Recruitment-Based Grouping and Assembly

Grouping and assembly is an important aspect of robotic control because it has a large number of real-world applications such as formation flying [Vásárhelyi et al., 2014; Mahmood and Kim, 2014] or coordinated object pushing [Matarić et al., 1995; Hu et al., 2011]. There is a large body of work on these topics, but few of them deal with the use of recruitment to satisfy these goals. The works described below feature robots or agents that use recruitment strategies to realize their formation or group.

Mathews et al. [2011] demonstrated a system for homogeneous robotic formation control, whereby robots would guide each other to specific positions to form chains, stars, or any arbitrary shape by physically connecting with one another. Robots communicated using a combination of radio broadcasts and infrared messaging, allowing the robots to communicate messages that included range and bearing information as well. Using these communication methods, robots attempting to extend their current formation (including formations consisting of only themselves) would recruit other robots that were detected to be at convenient positions and orientations relative to the recruiting robot. This system was validated using simulated and real robots. An important difference between this work and my own is that this study focuses on the use of recruitment solely to satisfy abstract formation tasks, and the framework demonstrated is not extendable to other types of work. My work also operates under

the assumption that desired robots may not be immediately available and supports the use of physical searching in order to locate them.

Pinciroli et al. [2009] experimented with recruitment among heterogeneous agents. In their approach, ground-based robots ("foot-bots") were recruited and collected underneath overhead flying "eye-bots" for the purpose of completing tasks on the ground as a group. Foot-bots could be added or removed at any time, and they reorganized themselves to ensure that all recruiting eye-bots controlled the required number of foot-bots. As in Gage and Murphy [2004], recruitment was one-way; only the eye-bots could recruit the foot-bots. While the recruitment approach used was demonstrably effective in distributing foot-bots appropriately among the eye-bots, it is difficult to evaluate how effective this approach would be in a more realistic setting where robots (i.e., the foot-bots) would also be expected to complete tasks for the eye-bots and therefore might not always be available for recruitment. My work addresses the possibility that robots may be unable to perform requested tasks, necessitating that a recruiting robot look elsewhere for help (and that there may be a significant cost associated with searching).

2.5 Recruitment Among Heterogeneous Robots

While many of the approaches listed in previous sections deal with heterogeneous robots, this section is specifically concerned with related work where robots do not form explicit teams and differ enough in their skill sets that they cannot complete all task types. For this reason, robots will frequently need to recruit other available robots in order to complete those tasks.

Dos Sos and Bazzan [2011] explored task allocation strategies for agents in simulated domains and presented an agent model called *eXtreme-Ants*. It is based on a model called Low Communication Distributed Constraint Optimization (LA-DCOP) [Scerri et al., 2005] which allocates tasks to agents in such a way that agents maximize their usefulness given the resources they possess (i.e., the classic knapsack problem). Key differences between eXtreme-Ants and LA-DCOP include (a) the former relies on a probabilistic algorithm to decide whether or not to execute a task, greatly easing computational requirements, and (b) eXtreme-Ants incorporates recruitment mechanisms inspired by those used by ants to recruit other ants to transport large food sources as a group [Robson and Traniello, 1998; Hölldobler et al., 1978]. Experiments showed that in abstract domains where the number and types of tasks remain static, LA-DCOP out-performs eXtreme-Ants in terms of the number of tasks that are completed in a fixed time. However, in more dynamic domains where the number of tasks can change (e.g., in a USAR domain a fire can spread from one building to several) eXtreme-Ants demonstrated superior performance in terms of the number of tasks that could be completed in a fixed time. My work features a similar recruitment approach as used by Dos Sos and Bazzan [2011], but with the property that every task can be completed by a single properly-equipped agent, i.e., no task requires the attention of multiple robots. Thus, recruitment in my approach takes place when a robot decides it cannot complete a task, and attempts to find another robot to do it instead. This is different from how recruitment is used in Dos Sos and Bazzan [2011] where agents recruit others to complete single tasks as a group. Additionally, agent perception in this work was highly abstracted: by simply occupying the appro-

priate space, agents automatically received knowledge of nearby tasks (e.g., victims and their health status) by use of an abstract *sensing* facility. In my work I simulate robot sensors that have comparatively limited sensing power, and the robots themselves differ in terms of how they sense the environment. Sensing itself in my work is an allocatable (and recruitable) task.

Costa et al. [2012] demonstrate the use of recruitment in cooperative multi-robot box pushing tasks. In this study, physical robots (either a *pusher* or a *grasper*) explored a small environment and were tasked with pushing boxes of various sizes and weights to certain locations. *Pusher* robots only had the ability to push obstacles, and *graspers* could both push objects and pick them up with an arm attachment. When encountering an obstacle, robots attempted to move it either by pushing or grasping, depending on their capabilities. Robots failing to move the obstacle using either approach initiated a recruitment request based on the type of object. The authors used an auction-based process whereby the nearest robot was most likely to be recruited for the box-moving task. After a successful recruitment, the newly-tasked robot and its recruiter would coordinate among themselves to move the box together. Unfortunately, the authors make no mention of how their system handles cases where two robots are insufficient to move a particularly large or heavy obstacle. Additionally, there is no explanation provided for how the authors' framework deals with situations where a robot might be desired for recruitment by two different robots at the same time. The fact that the authors assume robots have access to perfectly reliable communication means that the system likely does not scale well to more complex scenarios. My own framework is designed to handle conflicting recruit-

ment requests—robots are able to accept multiple requests and execute the associated tasks based on their importance. Furthermore, my framework is explicitly designed to handle situations where communications are unreliable, and recovers gracefully from situations in which robots suddenly fail to communicate or become disabled altogether. Lastly, I assume that robots are often not in communications range of each other at all. This means that in many cases, recruitment requests often cannot be satisfied simply by broadcasting a message: robots will often need to perform physical searches in order to find desired skills.

2.6 Autonomous Control and Navigation

In order to convey several aspects of the robotic control software used in the example implementation of my work (Chapter 4), it is necessary to provide a brief overview of some concepts used in autononmous navigation and control.

2.6.1 Schemas

A well-known approach for autonomous robotic control is the use of *schemas* [Arkin, 1987]: sensing and motion behaviours which interpret the environment and provide real-time responses in a continual perception-action cycle.

Perceptual schemas are responsible for perceiving the environment and providing data to *motor schemas* that control individual robot behaviours. For example, a rescue robot's *human casualty detector* perceptual schema would be used to detect life signs while a separate *obstacle detector* schema locates nearby walls or debris. The resulting sensor readings would be passed to *move to casualty* and *avoid obstacles*

motor schemas.

Motor schemas are individual behaviours which adjust motor responses based on the output of perceptual schemas. A major advantage of using motor schemas is that their resulting motion vectors can be combined to yield more complex behaviours. Using the above example, a robot's *move to casualty* motor schema would generate an attraction vector to a nearby casualty while its *avoid obstacles* motor schema would generate a repulsive force away from nearby obstacles. The summation of these action vectors would result in motion guiding the robot around obstacles towards a victim. This process is depicted in Figure 2.1.

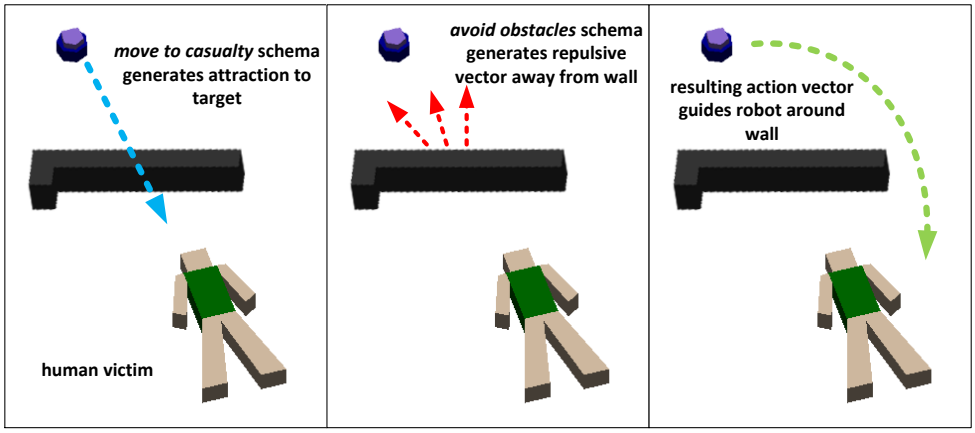

Figure 2.1: Two active motor schemas guiding a robot around an obstacle to a human casualty. Adapted from Gunn [2011].

The framework I use as a starting point for my work [Gunn, 2011] makes heavy use of perceptual and motor schemas to guide robot actions. I have also implemented additional schemas to support my work. The schemas I use are described in Section 4.4.

2.6.2 Occupancy Grids and Frontiers

Robots in my work are responsible for maintaining a map of the environment. To accomplish this, I rely on already-existing facilities implemented by Gunn [2011] which make use of *occupancy grid maps* [Elfes, 1989]. An occupancy grid map is a representation of an area where each grid cell represents a fixed square area of space. In the existing implementation, each grid cell is assigned a value indicating the confidence that the area represented by the cell is occupied, e.g., by debris or other obstacle(s) [Gunn, 2011]. A simplified example of an occupany grid map is shown in Figure 2.2.

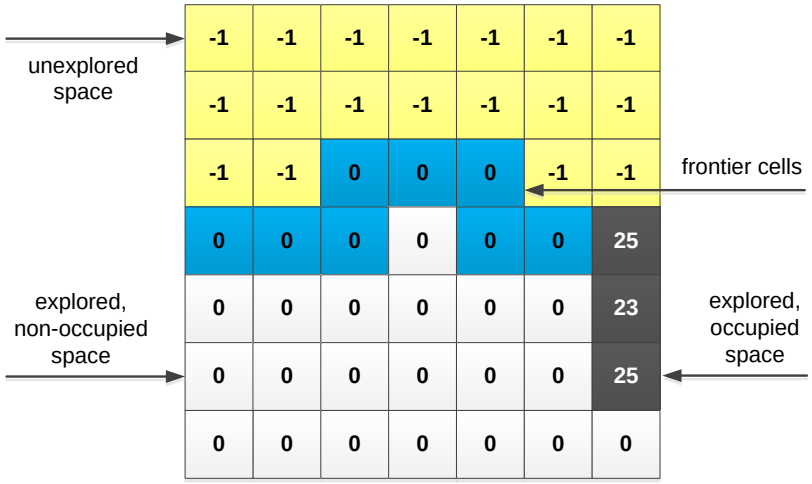

Figure 2.2: A simplified example of an occupancy grid. Higher numbers indicate greater confidence of the presence of obstacles. Adapted from Gunn [2011].

Occupancy grids can be used to identify *frontiers* [Yamauchi, 1997], which are boundaries between explored and unexplored areas (Figure 2.2). Therefore, directing robot exploration efforts towards frontiers helps to augment knowledge of the envi-

ronment more efficiently than random exploration. These facilities have already been implemented in Gunn [2011], and appropriately-equipped robots are able to detect frontiers and direct robot exploration efforts towards these locations (Section 4.3.1.2).

Chapter 3

Methodology

In this chapter, I describe the core methodology of my thesis. The main contribution of my thesis is a framework implementing active recruitment facilities for managing teams of heterogeneous robots in dangerous domains. This framework provides strategies which help compensate for robot losses and communication failures. It does this by allowing robots to perform searches of varying degrees in order to acquire new team members or delegate tasks to useful robots in a more distributed manner without the direct involvement of a team leader. These search mechanisms have trade-offs associated with them. More active strategies involving aggressive searches should increase the likelihood of useful robots being found more quickly, but at the cost of less immediately useful work being performed. Less active strategies will allow robots to complete more work, but will lower the chances of encountering other useful robots. I have implemented these strategies (issues and details related specifically to the implementation of my methodology appear in Chapter 4) as additions to an already-existing framework from our lab [Gunn, 2011], which I will henceforth refer

to as the *Gunn framework*. This is a very large framework, and in order to adequately describe my contributions, an explanation of the Gunn framework is required. I begin this chapter by describing it in detail and the facilities it provides for handling task allocation and team management in dangerous environments. Implementation-specific details are discussed in Chapter 4.

3.1 The Gunn Framework

Before going into framework details, it is important to describe the major themes embodied in the Gunn framework [Gunn, 2011] that are also central principles in my own work, as these lead to particular design and implementation decisions. The main concept underlying these works is that teams of heterogeneous robots working in a dangerous environment may not always possess an ideal combination of skills for the tasks at hand. To ensure that useful work can still be completed, robots may be assigned tasks for which they are not well-suited. Robots may even be entirely incapable of performing some tasks they are assigned. In the extreme case, a robot that has strayed and become lost would be on its own and at least momentarily responsible for any task that arose.

As robot losses occur due to the hazards of the domain, existing team members must be prepared to fill in for missing roles. Robots must recognize which roles they are best suited to fill and should rebalance themselves accordingly to ensure that the most mission-critical roles within the team are occupied. Teams should also be prepared to integrate newly discovered robots that are encountered, as well as decide how to exchange useful members between teams when opportunities for such

interactions arise.

These decisions become difficult due to the imperfect knowledge of team structure. As robots become physically spread out (and potentially lost or damaged) and new robots join the team elsewhere, no single robot will have perfect knowledge of the team and its members. Unreliable wireless communication can further degrade team knowledge as updates on team membership are lost or delayed. A robot may even join a team and become lost again before information about the first membership change reaches most members. Even accurate information regarding team structure quickly becomes stale, and models of team structure will differ between robots.

The spread of robots across a distance coupled with sometimes rapid team changes and stale information means that no single robot, no matter how able, can be counted upon consistently to make accurate decisions. Decisions must be made in a centralized manner across a team, using the best local information available combined with unreliable global information. Not all robots will have the same decision-making capabilities, however. The concept of leadership must exist in order to provide as much useful global information as possible, and to serve as a source of decision-making to be relied upon when information is lacking locally. Thus, teams in this framework have leaders at all times—even a robot lost by itself serves as its own leader, even if it is poorly suited to do so. This leadership role is ceded to the most able known member available in the same way that other roles are shifted over time as team membership changes.

Having described the basic themes involved, the following sections provide more specifics on the components of the Gunn framework [Gunn, 2011] that are integral to

my work.

3.2 Attributes, Tasks, and Roles

Robots in this framework are heterogeneous; due to differing abilities, one robot might not be able to complete some task as effectively as another robot. Some robots may even be entirely unsuited for certain types of work. Every robot used in this framework is described using a series of properties to represent its physical or computational characteristics or abilities. These are formally referred to as *attributes* and they enable robots to have an estimation of their own abilities, as well as the abilities of others. Example attributes might include a robot's size or whether a robot is equipped with sufficient computational capabilities to perform complex task allocation (actual attributes used in my implementation are described in Section 4.3.1). With this knowledge, robots can determine to what degree a particular task lies within their capabilities (or the capabilities of another robot), or whether or not a robot with known attributes can perform a task at all. These attributes can also be overridden as robots lose resources or become damaged: for example, a firefighting robot would track the estimated *volume of chemical* it carried, and this attribute would decrease as fires are extinguished.

For the purpose of my thesis, a task is the smallest unit of work that a robot can complete. Most tasks in this framework are meant to be completed by a single robot, although more than one robot may be aware of a particular task at one time. In my framework, multiple robots may individually contribute to a larger goal that is divided into tasks (e.g., exploration). I also support the division of a task into

multiple subtasks to a limited degree: one task in my framework (Section 4.3.2.2) requires the physical exchange of resources between two robots, and this is divided into two subtasks. In this situation, subtask division is fixed between only two robots and does not change during execution.

Every robot maintains a priority queue (Section 3.6.1) of the tasks assigned to (or discovered by) them, ordered by task importance. Intuitively, some tasks will be more important than others. For example, a task for administering medical aid to a critically injured human victim is more important than exploring the environment. Robots might also be aware of outstanding tasks that are beyond their capabilities, and robots must be able to recognize these situations. The actual tasks used in the implementation of my framework are described in Section 4.3.2.

To deal with potential limitations in robot skills, every task in this framework specifies a *suitability* expression and a *minimum requirements* expression [Gunn, 2011]: these are expressed in terms of the required attributes that are necessary to complete a task. A minimum requirements expression describes the bare minimum attributes that a robots must possess in order to perform a particular task. If a robot fails to meet these requirements for a task, the robot is understood to be completely unsuitable for it (although the lack of a sufficiently capable robot nearby may result in the unsuitable robot being assigned the task regardless). A suitability expression evaluates to a numeric value that is used to determine which, among several robots who successfully meet the minimum requirements, is the most suitable robot for a particular task. This provides an efficient method for deciding how to allocate tasks among team members. Higher values indicate greater suitability, and a robot that

fails to meet the minimum requirements for a task is understood to have a suitability value of zero for that task. Although task suitabilities provide a means for a robot to communicate how capable it is for a particular task, this framework provides *roles* as a heuristic to simplify this process when assigning tasks (task allocation is described further in Section 3.6.3).

Robot *roles* are defined by a collection of task types that a robot occupying that role is expected to be able to complete [Gunn, 2011]. This is a useful shortcut for determining if a robot can perform a particular task. If the role a robot occupies contains a certain task type, it is assumed that the robot can perform that task and possesses attributes that are appropriate for completing it. This may not be true if a team suffers from skill deficits or communication problems have resulted in the robot occupying a non-ideal role. A robot can also become damaged or run out of resources (e.g., a medical robot running out of supplies, which would be reflected by a change in that robot's attributes). A robot occupying a certain role is still able to complete other tasks that are not listed in its role description if required. Every task type in a role has an associated *weight* describing how important that task is considered in the context of that role. For example, a *firefighter* role will place a high weighting on an *extinguish fire* task but a lower weight on a task for *administering medical aid*. A robot must meet the minimum suitability requirements of all tasks contained in a role in order to be considered suitable for that role. All roles contain a default *idle task* that specifies an activity the robot should perform if the robot has run out of work (Section 3.6.1). Additionally, all roles have a domain-dependent *importance* value which is used to simplify the team maintenance process (Section 3.4) and the task

allocation process (Section 3.6.3). The roles used in my implementation are described in Section 4.3.3.

3.3 Imperfect Team Knowledge

Robots will periodically broadcast their presence to other team members to help ensure that everyone has up-to-date knowledge regarding the current structure of the team. Robots that have not been heard from after a given length of time are considered to have left the team. The presence of unreliable wireless communication poses difficulties for ensuring that robots have accurate knowledge regarding their teammates, since communication failures can cause robots to miss membership updates. This can lead robots to incorrectly assume that members have been lost or have left the team. Subsequent rebalancing of roles can therefore cause robots to occupy non-optimal roles, or ones that are already unknowingly occupied. For example, a team leader suffering from communication problems might be considered lost, and a poorly-suited robot may decide to take its place. This incorrect balancing will be remedied as better information becomes available.

3.4 Team Maintenance

As previously mentioned, this framework is intended to be used in environments that pose significant risks to the robots operating in them. Teams are expected to lose members, either due to damage or because some robots become stuck or lost. As mentioned in Chapter 1, it becomes necessary to release replacement robots into the

environment periodically, and existing teams should be prepared to integrate these robots (and previously lost robots that were rediscovered) into their team if required.

Central to team maintenance in this framework is the concept of an *ideal team*. The overall balance of skills required for a team to function well in a given environment should be defined at the outset (e.g., by humans), and this balance serves as a goal for team maintenance. Due to the dangers of the domain it is expected that a team will typically not match this definition except at the outset of a mission. The ideal team definition is used as a guide when determining how to best restructure a team or how potential new members should be integrated into the existing structure. More formally, the ideal team definition is formulated as a list of desired robot roles, the minimum and maximum number of robots that should occupy each role, and the relative importance of the roles within the context of the entire team. An example ideal team definition is shown in Figure 3.1: a single leader role with a high importance, one or two robots that can verify victims, and several small robots that can explore the world.

3.5 Role Changes

To ensure that a team of robots approximates the definition of an ideal team as closely as possible given current membership, existing members of a team will periodically perform a *role switch check* [Gunn, 2011]: after a domain-dependent amount of time (with a random offset to prevent multiple members from changing into the same role), each robot will evaluate their current role against their knowledge of the current team structure. As previously mentioned, every role has an associated

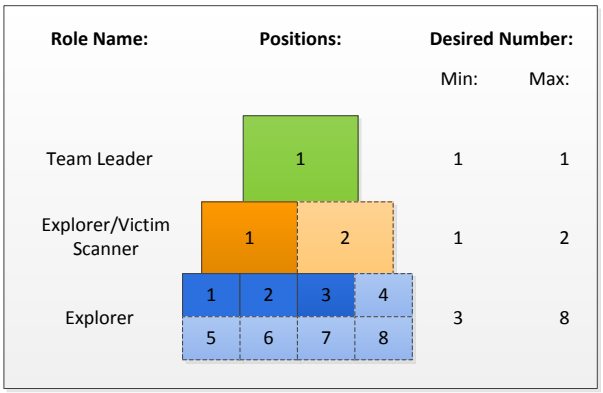

Figure 3.1: An example of an ideal team definition containing three robot types. Adapted from Gunn [2011].

importance value describing how critical the role is to the ideal team definition. When determining which role (if any) a robot should switch to, the robot iterates through all roles in the ideal team definition that are currently underfilled, and computes a *role score* value, which is the sum of the robot's *suitability* for that role and the *importance* of the role to the ideal team definition. The robot switches to the role with the highest *role score* and notifies its teammates of this change. If the highest-evaluated role is the same as the robot's current role, no change is made and no notification takes place. An example description of a failure and the subsequent role change is depicted in Figure 3.2.

It is important to note that a robot uses its own limited knowledge to evaluate whether or not a role is currently underfilled in its team, and this knowledge may be stale due to communication failures or rapid changes in team membership. In such cases, role switch checks may lead to robots occupying sub-optimal roles. For

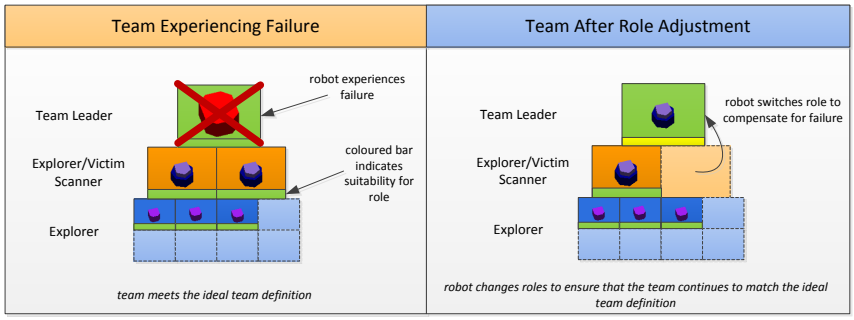

Figure 3.2: An example of a team before and after a robot failure and subsequent role change. Adapted from Gunn [2011].

example, a less-capable robot experiencing communication failures may determine that its teammates are no longer available, and will switch to a team leader role to which it is poorly-suited. In the case of severe communication failures, this can lead to all (or most) robots occupying non-optimal roles. Role change announcements might also be missed by some robots for the same reason; as a result, robots may incorrectly be seen as occupying different roles by other members of the team. The danger and communication disruption inherent in this domain dictate that such inconsistencies will always be present. New information, subsequent role checks, and other elements of this framework, however, help to ensure that knowledge is consistent enough among team members to be useful.

The Gunn framework uses a specific equation for determining the best role that a robot should switch to:

$$W = \begin{cases} \frac{D_{min}-N}{D_{min}} I_r & \text{if } N < D_{min} \\ \frac{D_{max}-N}{2D_{max}} I_r & \text{if } N \geq D_{min} \wedge N < D_{max} \\ I_r & \text{if } N \geq D_{max} \text{ and robot is better suited.} \end{cases} \tag{3.1}$$

When performing a role switch check, the suitability of a robot to switch into a role is multiplied by W, a weighting that encourages robots to fill under-filled roles. The importance of role r is denoted by I_r, and the number of robots in that role is defined by N. The minimum and maximum number of desired robots in that role (Section 3.4) are defined by D_{min} and D_{max}, respectively.

It was noted in [Gunn and Anderson, 2015] that for certain values, Equation 3.1 can influence role switches away from desired roles. A new formula was proposed by Gunn and Anderson [2015] (Equation 3.2), and this formula is used in my framework.

$$W = \begin{cases} \frac{D_{min}-N}{D_{min}} I_r & \text{if } N < D_{min} \\ \frac{D_{max}-N-D_{min}}{D_{max}-D_{min}} \frac{1}{2} I_r & \text{if } N \geq D_{min} \wedge N < D_{max} \\ I_r & \text{if } N \geq D_{max} \text{ and robot is better suited.} \end{cases} \tag{3.2}$$

3.5.1 Losing and Gaining Team Members

The Gunn framework relies on wireless communication between robots to communicate instructions, results of tasks, or details about the environment [Gunn, 2011]. Communication occurs frequently to ensure that other robots and team leaders have the most up-to-date knowledge possible. In an ideal case, robots on the same team will always be in communication range of each other. The limitations of wireless mes-

saging and interference from the environment [Carlson and Murphy, 2005; Murphy et al., 2000] may not always allow this, however. It is assumed that if a given robot has not been heard from after a period of time, the robot is no longer available and is considered to be lost from the team. Therefore, there is a direct link between communications reliability and the accuracy of a given robot's view of its team structure. Where communications are not working reliably, a robot is likely to have an incorrect view of the current team structure, and in the case of severe failures may even assume that it has become lost from its team entirely. Physical distance between team members is also a factor since members beyond wireless communication range may not directly receive all membership updates. However, such messages can be relayed between robots to ensure the information spreads.

When two teams encounter each other as a result of normal work in the domain, it provides the opportunity to rebalance team membership and recoup losses that may have occurred in terms of lost robots or skills, via the *rendezvous system*. It is important to note that this only occurs as a result of chance encounters—robots in the Gunn framework do not actively search for useful new teammates, regardless of the perceived state of their team. This is a major limitation that my thesis work addresses (Section 3.7). A team *encounter* [Gunn, 2011] begins with two robots A and B from different teams directly observing one another. At this point, the robots exchange information regarding the structure of their teams. The most computationally-capable robot (A) is selected to perform a team merge and redistribution process. This process does not take place if neither robot is computationally suitable for the task. Robot A then performs an internal process to determine how the robots on the two different

teams should be distributed such that each team can best approximate the definition of an ideal team. The resulting team distribution(s) will either be (*1*) two teams possibly meeting the ideal team definition more closely, or (*2*) one team formed by absorbing all members of both teams (this includes the case of encountering a single stray robot). Once robot A determines a suitable distribution it communicates this result to robot B. Both representatives are then tasked with instructing their teammates to change teams or roles as appropriate. Once the membership changes have taken place the resulting team(s) continue with their regular tasks as assigned by the team leader.

The distributed nature of team rebalancing actions (as well as other facilities such as role switch checks, Section 3.5) means that there is potential for interference and inconsistency. Unreliable communication also presents further challenges. For example, it is possible that multiple pairs of robots encountering each other in disparate locations might initiate separate team rebalancing negotiations, causing team membership conflicts. This can cause the complete breakdown of both teams. However, the same facilities would also be used to reconcile these inconsistencies: physically close robots would quickly join together and form new teams.

Similarly, role switches (Section 3.5) can occur at the same time as team rebalancing without the knowledge of the team representative. Role switches might also take place before an encounter without the knowledge of the future representative, or might be missed entirely when communication fails. All of these situations can lead to conflicts in role occupations that will ultimately have to be corrected in the future by subsequent team encounters and role switch checks.

3.6 Task Management

Robots in the Gunn framework are responsible for identifying and completing tasks to the degree that their sensors or hardware allow them. Only robots with the appropriate sensing abilities can discover certain types of tasks, and a robot's ability to complete a task is dictated by how well-equipped the robot is for that task. Task completion is carred out according to task priority (Section 3.6.1) to ensure that the most important tasks are completed first. To facilitate this, each robot maintains a priority queue of outstanding tasks.

3.6.1 Task Queue

A robot's task queue is an ordered backlog of tasks that remain for the robot to complete [Gunn, 2011]. Tasks are inserted into the task queue by order of task priority and time, i.e., the oldest, highest-priority task will be selected for completion first. Upon completing a task the robot will remove the task from the queue and begin execution on the next oldest, highest-priority task. Tasks can be inserted into the queue in either of two ways: by the robot discovering a task on its own (Section 3.6.2), or assignment by a team leader (Section 3.6.3).

It is possible that as a result of completing its assigned tasks a robot will have an empty task queue. Recall from Section 3.2 that every role defines an *idle task*; upon completion of all outstanding tasks a robot will perform work on its idle task (for example, a debris-clearing robot, in the absence of specific debris-clearing duties, will perform a random walk and remove any debris that it encounters). The idle task has the lowest possible priority and work on it will cease when other tasks are discovered

or assigned by a team leader (Section 3.6.2 and Section 3.6.3).

The use of a queue to store outstanding tasks ensures that robots will complete work on the most important tasks first. However, it is possible that while executing a task a robot may become aware of another task that has a higher priority. The Gunn framework supports *task preemption*: newly inserted tasks will take precedence over currently-executing tasks that have strictly lesser priority. Robots perform regular checks of their task queues to make sure there is not a more important task available for completion. If a more important task is discovered the robot will immediately halt work on the current task and begin execution of the higher-priority task. The preempted task is re-inserted into the queue for execution later on. This allows, for example, robots to halt exploration duties upon discovering a victim in need of assistance. The process of task discovery and task allocation are described in the next two sections.

3.6.2 Task Discovery

Upon discovering an outstanding task (e.g., a victim in need of assistance) a robot will evaluate its suitability for that task. If the robot meets the minimum suitability requirements for the task and if the robot's task queue is not already above some domain-dependent threshold (the Gunn framework uses a maximum of 5 tasks), the robot will insert the task into its queue. If either of these conditions is not met (i.e., the robot has a full task queue or is not suitable for the task), the robot will attempt to inform the team leader of the existing task. Since only team leaders are able to assign tasks, it is necessary for the robot to notify the team leader so that the task

can be delegated to a suitable robot (my framework implements extensions allowing non-leaders to assign tasks—see Section 3.9.1). It is important to note that such messages may not actually reach a team leader (e.g., due to communication failures, robot failure, or limited wireless range), in which case the corresponding task will ultimately need to be rediscovered later on. The task assignment process is described in the next section.

3.6.3 Task Allocation

It is the responsibility of the team leader to distribute assignments among team members. This can be done in two ways [Gunn, 2011]. Initial attempts to assign tasks use role heuristics to minimize the level of effort and communication (*role-based task assignment*, Section 3.6.4). If this process fails, the team leader uses a more expensive *exhaustive task assignment* process (Section 3.6.5). To avoid overloading the wireless medium, leaders only attempt to assign a maximum of five tasks of each type at one time.

The two different task assignment strategies are described in the following sections—my own additions to these processes are described in Section 3.9.1.

3.6.4 Role-Based Task Assignment

The use of roles reduces the level of effort that is required for a team leader when assigning tasks. Rather than attempting to exhaustively match assignee attributes with those necessary for a specific task, leaders can use knowledge of the roles associated with a task and the robots currently filling them to identify likely assignees

[Gunn, 2011] (recall that roles are defined by a list of task types, Section 3.2). Since robots may be occupying non-optimal roles, some of these potential assignees may not, in fact, be able to complete the task. To assign a task the leader will form a list of known robots on the team who occupy an appropriate role and simultaneously send a message to each of these robots, requesting that they complete the specified task at the appropriate location.

Robots receiving a task completion request consider the task type and their current workload. If a receiving robot determines that the task is within their capabilities and the robot has a sufficiently small workload (the Gunn framework uses a maximum of five tasks), the robot will respond to the leader by indicating its suitability for the task as well as the *cost* to complete the task. Cost is computed as a result of a robot's physical distance to the task location, since the Gunn framework assumes that all tasks to be completed are physical ones (e.g., examine a victim, explore a particular location, etc.). If either the robot has a full task queue or is not suitable for the task, the robot will send a rejection response indicating it cannot complete the task.

The leader waits two seconds to collect responses to the task completion request. Any responses received outside of this window are ignored. Positive responses (those containing task suitability levels and an associated cost) are processed and the leader will send a confirmation message to the robot which indicated the highest suitability for the task. In situations where multiple robots individually indicate the highest suitability, the confirmation message is sent to the robot that indicates the lowest task cost. Any robot who does not receive a confirmation message will not attempt

to complete the task. The robot who receives the confirmation will add the task to their queue and send an acknowledgement to the team leader, indicating that the task has been successfully assigned. Upon receiving this acknowledgement, the leader will consider the task assigned and will remove the task from its queue.

If the leader does not receive any suitability responses within the two second window (this includes responses that are all rejections, no responses at all, or a mix of both), it will move to *exhaustive task assignment* (Section 3.6.5). This will also occur if a leader sends an assignment confirmation to an assignee but does not receive an acknowledgement indicating that the robot is now aware of the new task. This can occur if the assignee falls out of wireless range or if wireless communications temporarily break down. For example, an assignee may receive a confirmation message and add a particular task to its queue as a result of a normal task assignment, but it is possible that the team leader will not receive the assignee's acknowledgement, causing the team leader to believe that the task was not successfully assigned. Another robot may subsequently be assigned the same task, resulting in duplication of effort. While this is mostly undesirable, there are potential benefits: two robots taking two different paths to the same task location, for example, may result in additional useful work being discovered.

Communication problems may also prevent a leader from receiving all transmitted responses to a task completion request. Although this may result in the leader ultimately making less-than-optimal task assignments, the leader can only make task assignment decisions based on the perceived availability of other team members and will attempt to make the best assignments possible based on its own knowledge. A

leader may also receive no responses whatsoever, either due to communication problems or because no other robots are actually available. In either case, failure to assign a task using this method results in using exhaustive task assignment as a fallback. This process is illustrated in Figure 3.3.

3.6.5 Exhaustive Task Assignment

Because of the extra computational and communication effort required, exhaustive task assignment is used only when role-based assignment fails [Gunn, 2011]. A team leader does not use role heuristics when assigning tasks with this strategy—it simply broadcasts a wireless message that can be heard by any member of the team. This broadcast follows the same format as the task completion request in the previous section, and responses and communication flow are identical. Thus, any robot from the same team can respond to this message, and the leader will attempt to find an ideal task assignment as outlined previously. If exhaustive task assignment for a task fails, the team leader re-inserts the task into its queue for reassignment at a later time.

I have now described all of the existing components necessary for an understanding of my work. These mechanisms are highly distributed, and the difficulties of operating in dangerous domains such as USAR dictate this since no single robot can be counted on consistently. The remainder of this chapter focuses on describing the methodology and various components of my own framework.

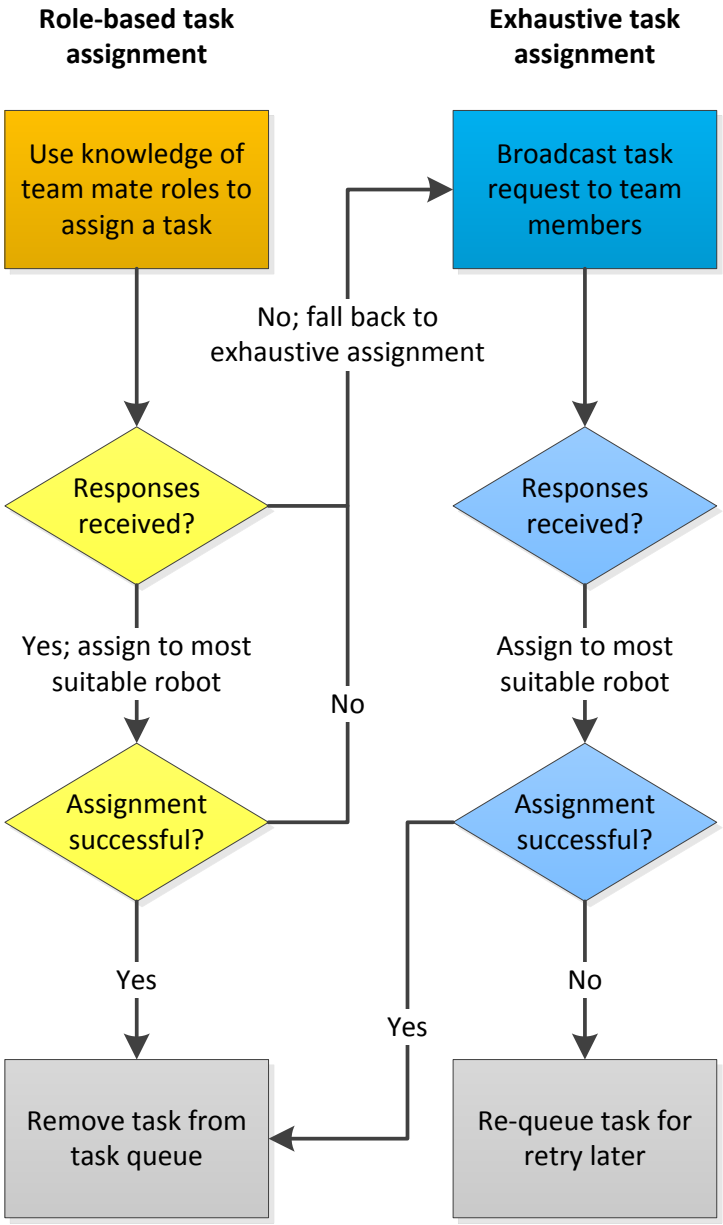

Figure 3.3: Describes the transition from role-based task assignment to exhaustive task assignment.

3.7 The Active Recruitment Controller Framework

A major limitation of the Gunn framework is that it does not provide facilities for actively acquiring robots with skills that a team may have lost due to the dangers of the domain. The only existing mechanism is passive and is based on individual robots from teams encountering one another. The purpose of my thesis is to develop and evaluate more active strategies for recouping robot losses and acquiring skills when needed, either temporarily or permanently. I have used the Gunn framework as a starting point to develop my own framework, which I will henceforth refer to as the *Active Recruitment Controller* (ARC) framework.

The ARC framework provides additions that are based on the concept of recruitment as defined in Chapter 1. These additions provide facilities enabling active search for useful robots in the domain and are manifested in two different types of requests: *role-level* and *task-level*. Role-level recruitment (Section 3.9.2) is used to bring in new skills to the team by inducting new members. In situations where robots are too busy or otherwise incapable of performing a particular task, or a permanent member is not desired, task-level recruitment (Section 3.9.1) is used to assign a task to another robot (possibly from another team) without bringing that robot into the team. Before describing these facilities, it is necessary to explain the range of effort that robots can commit to their recruitment duties. I have discretized this range into three *recruitment strategies* and describe them in the following section.

3.8 The Recruitment Spectrum

Robots may expend varying degrees of effort toward recruiting other robots. Aggressive strategies such as physically exploring an area for a desired robot are more likely to yield beneficial results. The downside of such approaches is that while searching in this manner, robots will not be able to perform other useful work (such as assisting a human victim). Less active approaches, e.g., operating on tasks as normal but broadcasting wirelessly for help, will result in more immediately useful domain-related work being performed but will lessen the chances that a desired robot will be found. Strategies that rely on random encounters with other robots represent the minimum level of recruitment effort but maximize the amount of immediately useful work that can be completed. Thus, there are trade-offs associated with different recruitment efforts and these strategies form a spectrum of effort with passive approaches on one end and active ones on the other (Figure 3.4).

The ARC framework defines three different recruitment strategies along this spectrum: *passive*, *concurrent*, and *active* recruitment. Passive recruitment is the strategy already used by the Gunn framework, since robots focus on performing immediately useful work and rely only on chance encounters with other robots to rebalance roles and acquire new skills (this process was described in Section 3.1). The following sections describe the two mechanisms I have added on top of the existing passive recruitment strategies to address limitations of the Gunn framework: concurrent and active recruitment.

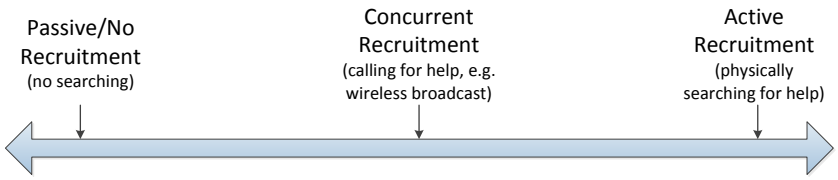

Figure 3.4: The recruitment spectrum.

3.8.1 Concurrent Recruitment

The concurrent recruitment strategy aims to increase the likelihood of encountering useful robots in the domain while still maintaining a focus on completing useful tasks. While the exact behaviours of robots in my implementation using this strategy differ based on whether a robot is recruiting for a role or a task (Section 3.9), the general process is the same: robots performing recruitment duties execute their tasks as normal, with the exception of performing additional broadcasts to contact and recruit other robots. Concurrent recruitment tasks are created either by a team leader deciding that a missing role needs to be filled (Section 3.9.2) or by a robot deciding on its own that it should recruit for a particular task (Section 3.9.1). In either of these cases the robot ultimately tasked with the recruitment duty enables the *recruitment broadcast system* (Section 4.5), which is responsible for handling communications with other robots for the purposes of finding a suitable robot. The nature of these broadcasts and the resulting communication with other robots varies depending on whether the robot is recruiting at a task or a role level. The sections on task-level recruitment (Section 3.9.1) and role-level recruitment (Section 3.9.2) describe these processes in detail.

3.8.2 Active Recruitment

The goal of the active recruitment strategy is to maximize the likelihood of encountering useful robots, and it accomplishes this by placing an emphasis on searching as opposed to the completion of outstanding tasks. Unlike concurrent recruitment, an active recruitment search is a task itself (Section 3.2) that can be assigned from one robot to another. Since robots can only perform one task at a time, robots performing active recruitment cannot complete any other work. However, the aggressive nature of this strategy greatly increases the chances that a useful robot will be encountered.

As with concurrent recruitment, active recruitment tasks are created either by a team leader deciding that a missing role needs to be filled (Section 3.9.2) or a robot requiring assistance with a particular task (Section 3.9.1). The resulting active recruitment task is inserted into the robot's task queue in order of task priority. Recruitment task priorities are determined by the type of the recruitment task (role- or task-level). Role-level recruitment tasks are considered the highest priority task due to the importance of maintaining an adequate set of skills on the team. Thus, the insertion of role-level active recruitment tasks in a robot's task queue will cause the robot to preemptively halt work on their current task and immediately begin executing the active recruitment task. Priorities for task-level recruitment requests are determined by the priority of the task being recruited for, ensuring that the importance of active recruitment tasks are weighed against those of the other tasks in the robot's task queue.

An active recruitment task involves a physical search for another robot. Upon starting up an active recruitment task, the robot will begin a random exploration

task with the purpose of finding a suitable robot. Upon beginning this active search, the rendezvous system is temporarily disabled (Section 3.5.1) for a short period of time (two minutes in my implementation) to allow the robot to search for a recruit without immediately being inducted into the first new team it encounters (after this length of time passes, the rendezvous system is reactivated again). The recruiter will also enable the *recruitment broadcast system* (Section 4.5) to begin recruitment broadcasts and handle communications with other robots. As with concurrent recruitment, the active recruitment communication process will vary depending on the level (task or role) of the recruitment task. Sections 3.9.1 and 3.9.2 describe these processes in detail.

3.8.2.1 Preemption of Active Recruitment Tasks

Because active recruitment is considered a normal task, there is the possibilty that it can be preempted in favour of a higher-priority task (Section 3.6.1). A robot currently executing an active recruitment task to locate a debris-clearing robot, for example, may encounter a human victim in need of medical aid. In these situations, the robot will shut down the active recruitment task and disable the recruitment broadcast system. This will halt the physical search and the robot will stop sending recruitment broadcasts. It is possible that a robot will halt an active recruitment task while it is in the process of negotiating with a potential recruit—to ensure that recruitment communications are not cut off prematurely, any negotiations already started are allowed to finish concurrently while the robot starts up execution of the higher-priority task. Sections 3.9.1 and 3.9.2 describe the recruitment negotiation

processes.

Recall from Section 3.6.2 that robots are able to discover tasks on their own (e.g., a victim requiring assistance) without the involvement of a team leader. If a robot discovering another task is already too busy to either take on the task itself or perform active recruitment, existing facilities in the Gunn framework are used to attempt to pass on the task to a team leader (Section 3.6.2). This is unnecessary in concurrent recruitment settings since configuration recruitment duties are executed alongside normal work.

3.9 Task- and Role-Level Recruitment

As previously mentioned, there is a cost associated with managing members of a team (Section 3.6.3), and this cost increases as the size of the team grows. Situations may arise where a team only needs to temporarily acquire a skill without incurring the cost of managing an additional team member. To support the idea that either permanent or temporary skills might be desired, the ARC framework supports two levels of recruitment: task-level (temporary) and role-level (permanent) recruitment. Task-level recruitment involves searching for a robot with the desired skills and requesting that it complete a specific task (Section 3.9.1). Role-level recruitment involves a robot being tasked by a team leader to locate another robot to occupy a role which the leader has judged to be underfilled. For either of these recruitment levels, the recruiting robot executes a search (either a *concurrent* or *active* search, Section 3.8) in an attempt to find a suitable robot. If such a robot is found, the two robots will negotiate the recruitment request. An outline of these strategies and how

they are used is shown in Table 3.1.

	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

Table 3.1: Strategies used to satisfy either task or role requirements.

Although task-level and role-level recruitment are separate mechanisms (Table 3.1), both of these techniques are always employed together when concurrent or active recruitment is used (i.e., task-level and role-level recruitment are always enabled together). Neither of them is enabled in passive settings.

3.9.1 Task-Level Recruitment

The ARC framework supports the concept of enlisting the help of other robots to complete a specific task. Unlike regular task assignment (Section 3.6.3), robots performing task-level recruitment are not restricted to communicating with team members only, meaning that recruiting robots can also assign tasks to members of other teams as well. Importantly, this process does not require the intervention of a team leader: robots occupying non-leadership roles can perform task-level recruitment on their own and recruited robots do not need to ask permission from their leader to execute a task for which they have been recruited. It is also important to note

that robots in my work are not selfish (such as those in many other applications, e.g., Sen and Dutta [2002]; Dutta and Sen [2003]; Van De Vijsel and Anderson [2004]; Comi et al. [2014]) and will accept task-level recruitment requests when they are able. Robots may decline task recruitment requests if they are too low on resources or have too many pending recruitment duties already. Additionally, robots will not recruit others as a means of avoiding work: robots will only perform task-level reruitment when it is strictly necessary.

Recall that robots can either discover tasks on their own (Section 3.6.2) or have tasks assigned to them by a team leader (Section 3.6.3). If a robot has a full task queue during either of these situations, or if the task made aware to the robot is beyond its capabilities, this is indicative that the robot is not an ideal choice for that task. In either of these cases, the robot will add the task to a list of tasks requiring recruitment. The next two sections describe the process by which these tasks are given to other robots.

3.9.1.1 Communication and Task-Assignment Process

The ARC framework relies on the already-existing facilities in the Gunn framework to perform task-level recruitment. Specifically, robots use the *exhaustive task assignment* system (Section 3.6.5) to assign tasks to other robots—a robot examines the tasks it is aware of that are marked for recruitment, and attempts to assign them. The difference between the Gunn framework's approach and my own is that in my framework, non-leader robots are now able to assign tasks using these facilities to satisfy task-level recruitment. While the Gunn framework uses role heuristics during

initial task assignment attempts (and then falls back to exhaustive task assignment if this fails), this is not an option for task-level recruitment since ideal recruitment candidates are not assumed to be on the current team, and recruiters are unlikely to have up-to-date knowledge regarding the capabilities of any nearby robots (or even which ones are nearby). Communication problems or robot failures are also handled using the same facilities as described in Section 3.6.5.

3.9.1.2 Initiating Task-Level Recruitment

Normally, in situations where a team leader sends a task completion request, a robot will agree to the task if its task queue is not already full and the task is within its capabilities. If either of these conditions is unmet, the robot will not send a rejection (as it would in the Gunn framework, Section 3.6.3) but will instead offer to recruit someone for the task if the robot is not already recruiting for too many tasks—it will send a rejection otherwise. The team leader will agree to this offer of recruitment only if no other robot agrees to take on the task for execution themselves. If the team leader agrees, the robot will add the task to their task queue, marking it to be completed via recruitment. The robot will then perform the task-level recruitment processes described in 3.9.1.1. Figure 3.5 shows the flow of logic used to determine whether a robot should offer to perform task-level recruitment when it receives a task request.

In either concurrent or active recruitment settings, a robot can also recruit for a task it has discovered itself but is completely unsuited for. For example, a low-cost explorer robot discovering a fire, but lacking the necessary extinguishing equipment,

could recruit another robot to put out the fire. In the Gunn framework, this situation would be handled by notifying a team leader of the task (Section 3.6.2). In the ARC framework, the robot will begin a recruitment process instead if the robot determines it is able to do so (i.e., if it is not too busy for an active recruitment task or is using concurrent recruitment).

As described above, robots are able to accept a task with the intention of recruiting someone else to complete it, rather than agreeing to complete it themselves. Because all robots in my framework have the ability to assign tasks via task-level recruitment, this has the effect that a task may be passed from one robot to another until a suitable robot can be found who agrees to execute it. To prevent tasks from being passed back and forth uselessly between two or more robots, robots will not offer to recruit for a task that they have recently assigned to another robot themselves.

When a robot becomes disabled, it is possible that tasks assigned to it can be lost if no other robot has knowledge of those tasks. If no other robots are aware of those tasks, they will need to be rediscovered. If other robots are aware of the outstanding work, however (e.g., a victim that needs to be confirmed, or an area that requires exploration, Section 4.3.2.1), they will generate new tasks and attempt to assign them accordingly.

3.9.2 Role-Level Recruitment

The purpose of role-level recruitment is to acquire a skill (or set of skills) that is judged to be essential for the successful operation of the team. Because of the importance of such skills to the team as a whole, the cost of managing an appropri-

ate robot over the long run is considered acceptable. The concept of an *ideal team* (Section 3.4) is useful here: if a team is missing a member which, according to the ideal team definition, should be present, then the team should be prepared to accept the cost associated with managing the additional member and should expend effort into locating one.

The ARC framework supports role-level recruitment by specifying an additional team-management duty for the team leader (in addition to those already described in Section 3.6.3) called a *missing role check*. The team leader executes missing role checks periodically, and this process involves two steps. First, the team leader determines which roles, if any, are absent from the team, based on its current imperfect knowledge of the team structure. Team leaders are the obvious choice for executing missing role checks since it is assumed that they will have the most complete picture of the current team membership. The most important underfilled role (recall that roles have an *importance* value, Section 3.2) is selected from this missing role list. The team leader then assigns a member of the team with a *role-level recruitment* duty. The recruiting robot then becomes responsible for locating another robot to fill the missing role on the team. This process is shown in Figure 3.6. Recruitment tasks and how they are executed are described in Section 3.8.

Missing role checks are similar in spirit to the *role switch checks* (Section 3.5) that all robots execute periodically to balance roles among team members. The main difference between these two facilities is that missing role checks are only executed by a team leader. Additionally, missing role checks are executed much less frequently (every 6 minutes in my implementation) than role switch checks (which run every

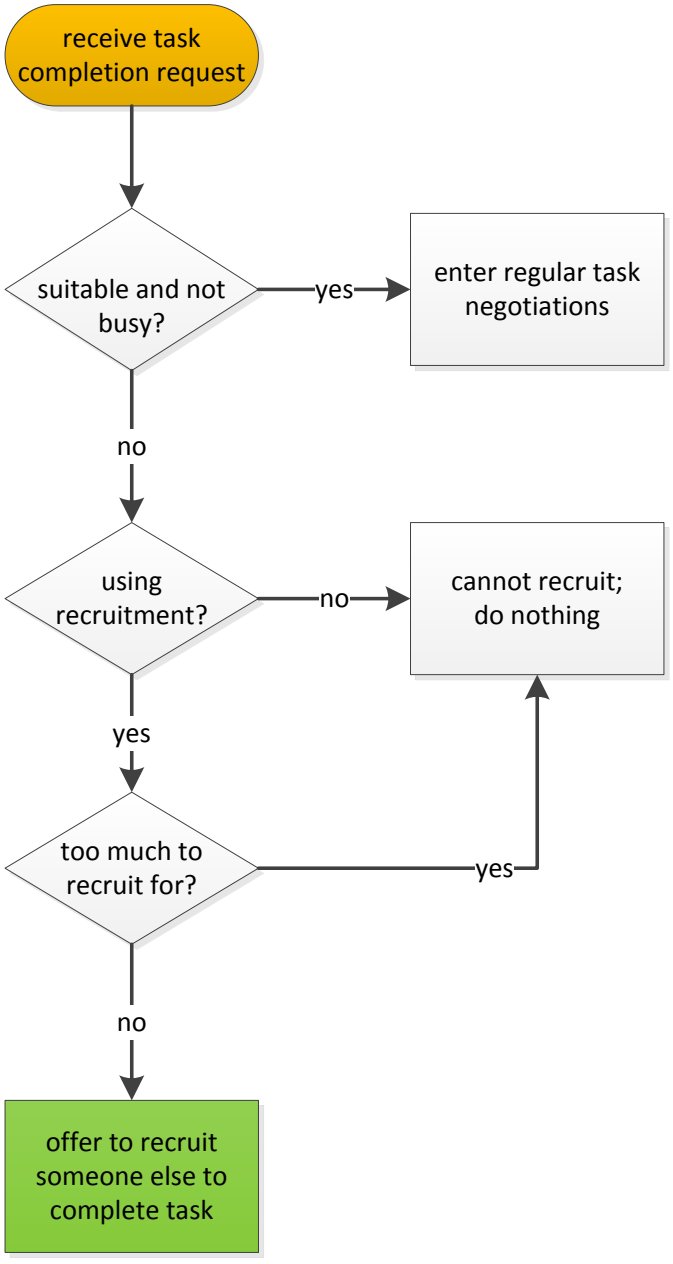

Figure 3.5: Determining whether to offer to recruit for a task.

30 seconds). This ensures that robots will self-balance their roles within a team before a team leader judges it necessary to perform role-level recruitment, and avoids situations whereby a team leader tasks a robot to recruit for a particular role that becomes filled shortly afterwards as a result of a role switch check.

Unreliable communication and inaccurate knowledge can cause a team leader, when executing a missing role check, to mistakenly believe that a critical role is missing from the team because the robot filling that role has not been heard from recently. As a result of this perceived loss, another robot on the team may be assigned a role-recruitment duty to find a replacement for the robot that is assumed to be lost. If a replacement robot is recruited, it is entirely possible that two robots will now fill the same role redundantly on the same team. Subsequent role checks (Section 3.5) may rebalance the roles of these two robots to ensure that only one of them occupies the important role, but this may result in the other robot occupying a role that is less suited to its abilities. This may eventually be resolved by the extra robot being recruited back to another team (possibly its previous team), whose leader will eventually recognize the loss and attempt to acquire a replacement. Thus, the overall effect of unreliable communication is that robots may spend a significant amount of time being passed back and forth between teams. This is not entirely negative: similar to passive encounters (Section 3.5.1) exchanging members between teams using recruitment provides increased opportunities to share operational knowledge. Increased recruitment efforts in active recruitment settings may also result in greater environmental coverage as robots explore and uncover more of the environment in order to find recruits.

The same opportunities exist when robot failures or losses occur: as robots become disabled due to the dangers of the domain, teams will attempt to recruit other robots in order to compensate for these losses. If few desired robots are actually available (possibly due to disablement or a lack of suitable replacements), robots may be recruited back and forth frequently between different teams. This results in greater information sharing as recruited robots pass along the knowledge they have acquired.

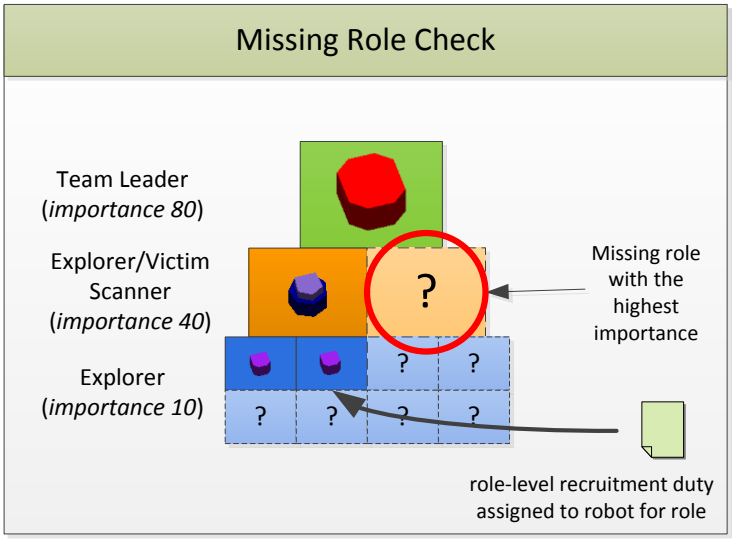

Figure 3.6: Shows a missing role and the role-recruited duty that is generated as a result of a missing role check.

3.9.2.1 Communication and Role Assignment Process

The communication flow for role-level recruitment follows a similar structure to that of task-level recruitment (Section 3.9.1.1). Robots tasked with role-level recruitment duties periodically broadcast (every ten seconds in my implementation) a

message requesting a robot to fill a specific role. The message also contains the number of robots on the recruiting robot's team that are currently known to be filling that role—this allows potential recruits to determine which team is in greater need of robots occupying that role.

A receiving robot with too many tasks on its queue will not respond to a role-level recruitment broadcast. I assume that if a robot is busy then it is currently useful to its team or otherwise has too much important work to complete. Robots with fewer tasks, however, will consider the role switch in the following manner.

If a receiving robot determines that it has lost contact with its team, it will offer its role suitability to the recruiter. If a receiving robot is aware of its team, it considers whether or not it would be more useful on the new team: if it is more suitable in the new role, or the number of robots on the new team occupying that role is less than the number of robots on its own team occupying that role, the robot will offer its suitability to the recruiter. The potential recruit does not contact its leader at all during this process (i.e., does not ask for permission to potentially leave the team). This process is illustrated in Figure 3.7.

After waiting a short period of time to gather responses from robots (two seconds in my implementation), the recruit will select the candidate who indicated the highest suitability for the role. It will send a *role switch request*, indicating that the selected robot should change to the recruiter's team and fill the new role. The recruited robot, upon receiving the message, initiates the role change, sends an acknowledgement to the recruiting robot, and announces its new membership to any nearby robots (including new teammates). This knowledge will eventually spread—the robot's new

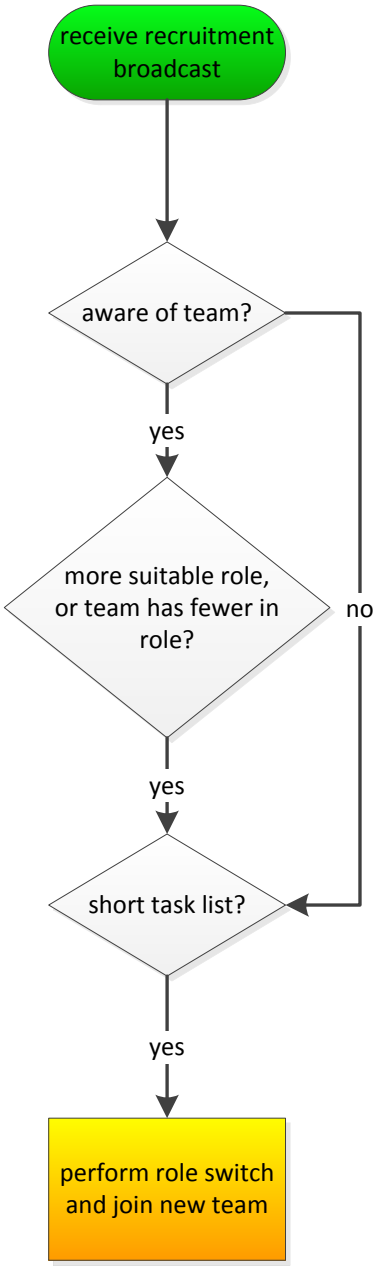

Figure 3.7: Recruitment communication flow from the perspective of a potential recruit.

team leader would gain knowledge of the new robot through membership updates, and the robot's old leader will eventually record the robot's loss either through wireless updates or by realizing that the robot has lost contact with the team (similar to when robots become lost or disabled, Section 3.5.1).

It is possible that a particular robot may find itself being considered for recruitment by more than one robot at the same time. If recruitment broadcasts from robots A and B both reach a robot C, C will offer its suitability to both robots for the desired roles. If only one (or none) of either A or B accept C's offer, there is no conflict—robot C will subsequently join one (or none) of the teams. However, if both A and B accept C's offer, C will only join the team whose recruiter (either A or B) accepts first. C will then send rejection messages to any other robots from which it receives further acceptance messages.

To prevent a robot from continually being recruited uselessly between two or more teams in a short time span, robots in my work that have accepted a role-level recruitment request will ignore further requests for a short period of time (one minute in my implementation). During this time the robot will not send responses of any kind to other recruits.

As with task allocation (Section 3.6.3), poor communication can result in failure to recruit other robots, or can result in duplication of effort. For example, if a new recruit's final acknowledgement is not received, the recruiting robot considers the recruitment to have failed and resumes broadcasting again. This may result in multiple robots being recruited for the same role when only one robot is actually needed. As mentioned previously (Section 3.9.2), further role recruitment efforts

made by other teams help to balance this out. However, as communication becomes increasingly unreliable, teams will dedicate more effort toward recruitment and will become more fluid as members are passed more frequently between teams. While this makes knowledge of current team structure increasingly inaccurate, there are greater opportunities for information sharing as robot memberships change between teams over the course of the mission.

3.10 Conclusion

In this chapter I have described my core methodology, including portions of the Gunn framework that were necessary to implement my own. Chapter 4 describes the details of the implementation I created for the purposes of conducting the evaluation described in Chapter 5. Some of this is based on a previous implementation of the Gunn framework [Gunn, 2011], and Chapter 4 describes the components I have implemented as well as those that my work depends on.

Chapter 4

Implementation

4.1 Implementation Overview

This chapter describes the details of the example implementation that I use to evaluate my framework (described in Chapter 3). Since I have used the Gunn framework (Section 3.1) as the starting point for my implementation, it is necessary to describe some existing components of this framework before I describe those of my own.

The Gunn framework was evaluated in a simulated USAR environment [Gunn, 2011]. To facilitate comparison with this work, I will be evaluating my framework in this domain as well. I begin with a description of the main USAR concepts that are relevant to my work.

4.1.1 USAR Concept and Goals

This section provides a high-level description of the operations that robots in my work are expected to perform. The main goal of my robots is, ultimately, to find and correctly identify as many human casualties and cover as much of the environment as possible in a simulated USAR domain. Realistically, victim and environment information would eventually be conveyed to a human rescue team (which may, for example, enter the area at a later time) who could then extract casualties based on the information provided by the robots.

The effort and cost involved in building and conducting experiments using real physical robots in a sufficiently realistic setting (e.g., debris, size of environment, etc.) would be prohibitive. Additionally, the time required to conduct experiments makes this almost impossible—running the required number of trials for my evaluation (Chapter 5) would take over a year, even when not including factors such as setup time or battery charging. For this reason, my work is performed in simulation, similar to [Gunn, 2011].

4.1.2 Simulated Environment

Stage [Gerkey et al., 2003] is an established multi-robot simulator which, like Gunn [2011], I use as a platform to implement and evaluate my framework. It is widely-known and has been verified against real robots. Conducting my work in simulation has two main advantages, as noted by Vaughan [2008]: experimental runs in Stage are repeatable, and they can be run faster than real-time. This last feature is particularly important due to the large number of experimental trials required to

evaluate my work. Additionally, using a simulator removes complexities and problems that are inherent to physical robots (such as the failure of components like motors or cables) that are not a central part of my work.

Stage provides facilities for constructing virtual environments, robots, and any other objects, as well as an API for programming the behaviours and interactions between these objects. For example, I make use of an abstract *robot sensor* model that allows robots to detect one another—Stage refers to this class of sensor as a *fiducial sensor*, which can be used to simulate RFID technology, robotic vision, or any other sensor based on object identification. Another class of sensor allows rangefinder models such as sonar or lasers. My implementation (and that of Gunn [2011]) make heavy use of Stage's simulated sensors to allow robots to sense the environment, the presence of victims, and other robots. It should be noted that the implementation of the Gunn framework was written using version 3.2 of the Stage API—my implementation was written using the most current version of Stage (4.1, at the time of writing), which required existing elements from [Gunn, 2011] to be ported to the new version of Stage. Although some elements of Stage have changed between versions 3.2 and 4.1, I have ported the Gunn framework in such a way that it remains functionally identical to the 3.2 version.

Figure 4.1 shows an overhead view of an example environment, and Figure 4.2 shows a closer view depicting the significant objects in my work. As in [Gunn, 2011], every experimental trial begins with two teams of robots entering the environment through two separate openings meant to represent doorways or windows. The formation of the robots is chosen to ensure that those with debris-clearing equipment are

able to clear any debris blocking the team's entry, and to ensure that explorer robots have a clear line of sight to the rest of environment (as opposed to being obstructed by the team leader). Replacement robots, which are released into the environment at a later time (Section 3.4), are spaced evenly along the perimeter facing inwards. The four robot types used in my work are described in more detail in Section 4.2.

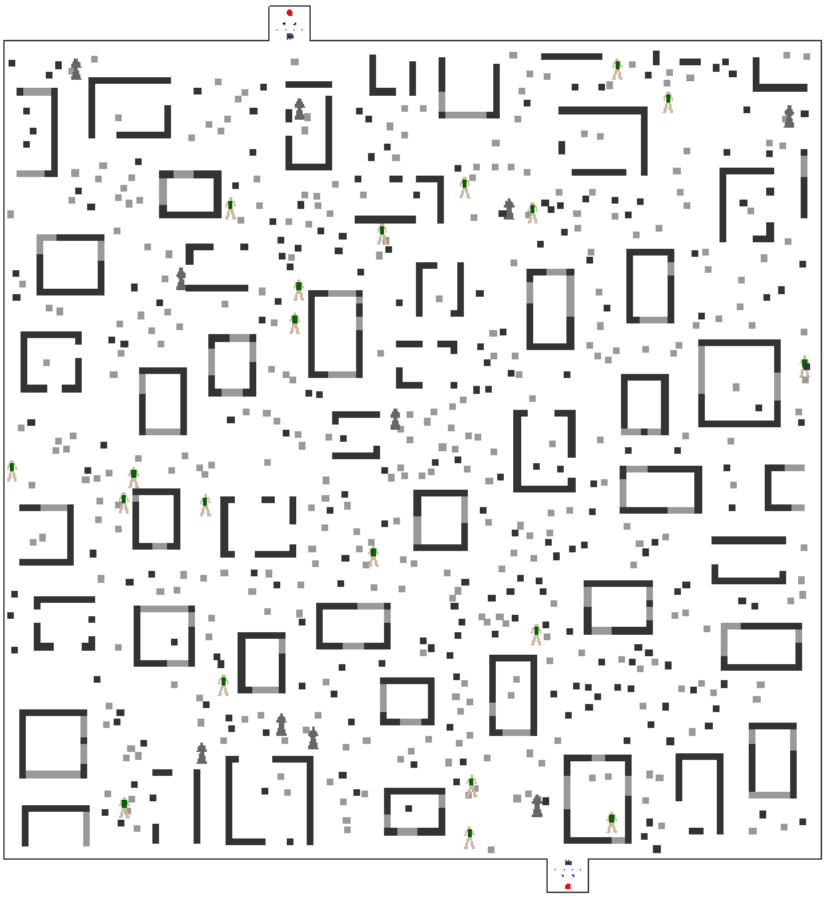

Figure 4.1: An example randomly-generated world.

Light grey objects in Figure 4.2 are low-lying debris configurations such as bricks or pieces of concrete. These are obstacles that robots with a tracked drive system would be able to traverse in the real world, and robots with wheeled drives would be

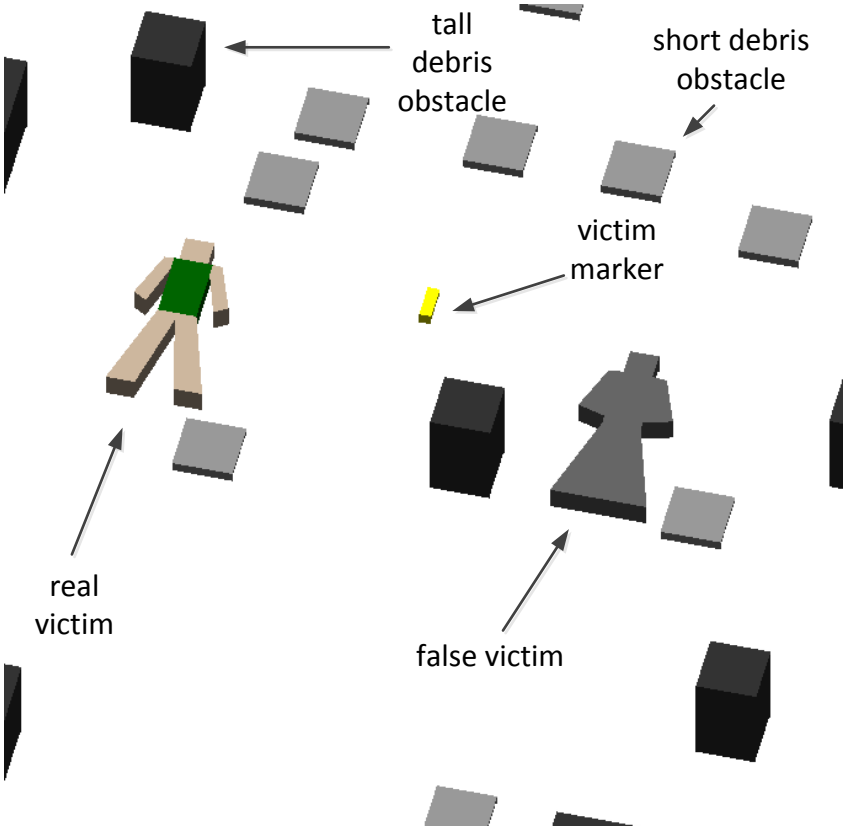

Figure 4.2: Significant environmental objects in my work.

forced to navigate around them. In my implementation, robots with debris-clearing equipment are able to remove low-lying debris from the environment, simulating the ability to break down and render debris passable to other robots.

Dark grey debris configurations are meant to represent larger structures such as walls, support beams, or office equipment (e.g., shelves). Due to their size, these obstacles are impassable to all robot types and cannot be removed with debris-clearing equipment—all robots must navigate around them.

Also present in my environments are a number of randomly-placed *victim* objects.

These are meant to represent human casualties in need of assistance. Victims are detectable by robots equipped with the appropriate *victim sensor* fiducial model (Section 4.1.2.1). To provide an enhanced level of realism, environments will also contain a number of *false victims*. These are debris configurations that resemble actual humans and require advanced sensors in order to be distinguished from true victims.

4.1.2.1 Victim Identification

The primary goal of my robots is to locate and identify human casualties in the environment. In reality, this is a task that requires the input and fusion of multiple types of sensor readings [Kadous et al., 2006; Tunwannarux et al., 2008; Murphy et al., 2000]. I use Stage's fiducial sensor model to abstract away these complexities since real-world victim identification is not the focus of my work.

To reflect the fact that lower cost robots will possess limited sensory capabilities and expensive robots will have more advanced sensors, I make use of two types of victim-detection sensors, implemented by Gunn [2011]: the *basic victim sensor* and the *advanced victim sensor*. The basic version of the victim sensor has a range of 4.0 meters and a field of view of 180 degrees. Within this range, robots equipped with this sensor can detect the potential presence of human victims but not confirm them (i.e., to the basic victim sensor, both true and false victims will appear the same). The advanced robot sensor has a field of view of 180 degrees and has a maximum range of 6.0 meters, at which distance it can detect the presence of a potential victim. Within a distance of 4.0 meters, it can correctly distinguish between true and false

victim sensor type	fov	identification range	detection range
basic	180°	n/a	4.0m
advanced	180°	4.0m	6.0m

Table 4.1: Characteristics of the two victim sensor types used in my work.

victims. These victim sensor types are summarized in Table 4.1.

4.1.2.2 Robot Detection

Some robots in [Gunn, 2011] and my work are equipped with *robot detectors* that simulate visual identification of robots in close proximity. This is facilitated by the use of the fiducial sensor model provided by Stage. Robot sensors have a range of 6.0m and a field of view of 180 degrees.

4.1.2.3 Area Exploration

In order to locate victims, robots must explore the environment. As in [Gunn, 2011], the exploration process is guided by the robot occupying the team leader role. All robots maintain an occupancy grid (a 2D map defining which parts of the environment are passable or not) of the environment, and robots with sufficient computing facilities will identify *frontiers*, which are boundaries between explored and unexplored space. Robots with the appropriate software control modules (Section 4.3.1.2 can assign *frontier exploration* tasks to other robots so that these areas can be explored.

Frontier exploration tasks consist of a robot moving to the indicated location,

exploring randomly for a short period of time, and then reporting back to the assigning robot with the map data acquired (Section 4.3.2.1). This may not always be successful, however: the robot could become stuck, disabled, or recruited to another team before it is able to complete the task or report back to a leader.

4.1.2.4 Managing Domain Knowledge

The Gunn framework provides facilities for managing and updating robot knowledge [Gunn, 2011]. To the degree that onboard computation and memory allows, each robot is responsible for maintaining a copy of operational domain knowledge. This includes the locations of victims, the state of the robot's occupancy grid, and knowledge of other robots in the domain. Robots acquire this knowledge either by discovering significant information themselves (e.g., exploring an area or finding a victim) or by receiving messages from other robots who have transmitted the information as a result of a successful task completion: robots will listen for information updates from other team members to update their own knowledge. It is assumed that the robot filling the role of team leader will have the most complete set of operational knowledge due to its role in assigning tasks and receiving the results.

Spreading knowledge among robots as much as possible is desirable since it reduces the chance that important knowledge becomes lost when robot failures occur (Section 4.1.2.6). Limited wireless communications range, temporary communication failure, or robot failure may result in robots (including a team leader) failing to receive critical information from other members. This inconsistency in team knowledge can result in duplication of effort (e.g., a robot may be sent to confirm a victim that

was already examined by another robot whose resulting update was not received). It is highly unlikely that any two robots on a team possess the exact same set of knowledge, and the failure of a leader might mean that some information is lost—however, the team would not have to re-start the mission since the new leader would presumably possess at least some of the former leader's knowledge.

4.1.2.5 Unreliable Communication

Every robot in my implementation is equipped with short-range wireless communication modules. As in [Gunn, 2011], I use the ITU indoor radio model (provided by Stage) to simulate wireless radio transmission. I also use the same wireless parameters (specifically: signal power, receiver sensitivity, and power loss), modelling communication ranges of approximately 20m. Although commercially available wireless 2.4GHz transmitters can achieve much greater distances in open areas, I assume that the configuration of the environment (debris, etc.) limits the effective range of communication.

Additions by Gunn [2011] have extended Stage to support random communication failures. Each simulated world specifies a communication success parameter, describing what percentage of wireless messages are successfully transmitted. For every message m to be sent from robot s to all robots R in radio range, the simulator generates a random number from 0 to 100. If the random number is less than or equal to the success parameter, the message is sent to all robots in range. Otherwise, the message does not reach any of the robots.

4.1.2.6 Robot Failure

The Gunn framework [Gunn, 2011] supports the concept of random robot failures to simulate the real-world risk of robot disablement, either temporarily or permanently. Every robot type defines a *failure probability* value, specifying the probability that it will experience a temporary or permanent failure. Temporary failure durations vary randomly between 3 and 4 minutes, and permanent failures last for the remaining duration of the mission. As in [Gunn, 2011], failures in my work are total, i.e., I do not simulate the failure of individual robot components. While experiencing a failure, a robot is not able to move, communicate, or perform computations of any kind.

4.1.2.7 Fiducial Victim Markers

In addition to locating potential victims, robots in my implementation are also responsible for placing *fiducial victim markers* next to potential or confirmed victims which they discover. These markers are detectable by all robots and are meant to help human rescue workers (entering after the completion of the robots' mission) in locating casualties. Markers can also double as medical kits or protein bars that can be used by victims. In a real-world setting, a marker might be a small supply bag with an embedded radio beacon and a small quantity of food. Every robot carries two markers and will drop one marker near a detected victim when no other markers have been detected nearby. A victim without nearby markers indicates that no one has discovered it yet. The limited complement of markers that a robot carries means that they may run out quickly—my implementation includes facilities allowing

robots to physically donate a marker to another robot if needed (Section 4.3.2.2). The *marker manager* module handles marker placement and donation, and is described in Section 4.6.1.

4.2 Robot Types

In this section I describe in detail the four robot types in my implementation. The first three (the MaxBot, Midbot, and MinBot) are the same as used in [Gunn, 2011]. I have added a fourth type of robot to my implementation, the DebrisBot. Figure 4.3 shows each of these robots.

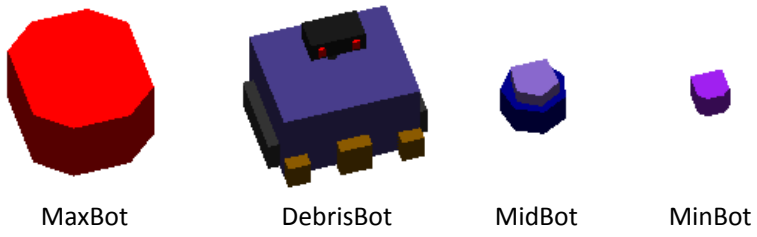

Figure 4.3: The robot types used in my work. They are kept geometrically simple to avoid performance impacts when examining simulation runs by hand.

4.2.1 MinBots

The MinBot is an inexpensive robot whose main purpose is exploration and victim discovery. It has a limited complement of sonar sensors to avoid obstacles and to update its own map of the environment. MinBots lack robot detection sensors

(Section 4.1.2.2) and only use sonar to sense the environment, and therefore cannot distinguish between robots and environmental obstacles. This limits the level of interaction that MinBots can have with other more capable robots.

MinBots cannot detect frontiers, generate frontier exploration tasks, assign tasks, combine shared environment maps, or perform pathfinding algorithms. This makes them poorly-suited for leadership positions. Importantly, the original implementation of the MinBot [Gunn, 2011] did not permit it to assign tasks to other robots. I maintain this restriction, but allow MinBots (and all other robots) to perform recruitment because this capability is central to my work. Recruitment duties are less computationally intense than full task assignment duties—recall from Chapter 3 that task-based recruitment only makes use of the exhaustive task assignment capabilities (as opposed to both the role-based and exhaustive capabilities) of the task assignment system already implemented by Gunn [2011]. The computational effort required for role-level recruitment is also relatively light. However, the MinBot's lack of a planner or shared team map means that it cannot provide path information to other robots when recruiting for tasks. Therefore, there is a greater likelihood that a recruited robot may become stuck or lost due to the lack of path information that would guide it around potentially hazardous areas.

4.2.2 MidBots

The MidBot is larger than the MinBot and also has a wheeled drive. Its main purpose is exploration and victim confirmation. It also has a more powerful sonar array than the MinBot.

This robot has advanced victim sensors, allowing it to detect the presence of a potential victim at a range of 6.0m, and it can confirm victims at a range of 4.0m. It can also maintain a combined team environment map, detect frontiers, and can generate frontier tasks. The MidBot is also capable of performing task assignment. These capabilities mean that the MidBot (unlike the MinBot or DebrisBot) is somewhat suited towards filling team leadership positions. However, the MidBot is not capable of path-planning; therefore, task assignments performed by the MidBot will not come with path information, meaning that assigned robots are more likely to become stuck in the environment while navigating to assigned task locations.

4.2.3 MaxBots

The MaxBot is a large robot with a tracked drive system, enabling it to traverse low-lying debris. It uses a sonar array to detect low-lying debris, and a laser range finder with a field of view of 180 degrees to identify all other obstacles. It has a robot detector and a basic victim sensor only. Its computational facilities (i.e., the ability to detect frontiers, maintain shared team maps, and assign tasks) make it the most suitable robot for leadership roles. Since the MaxBot also has a path planner, it is able to communicate path information for other robots when assigning tasks, if a path can be found using its accumulated knowledge of the environment.

4.2.4 DebrisBots

I have implemented a highly-specialized robot called the DebrisBot. Its main purpose is to clear low-lying debris from the environment. It is equipped with a *debris*

remover which is used to break down and remove low-lying debris, eventually clearing the space the debris occupied so that other robots can pass through. The DebrisBot will attempt to clear debris when executing either the *unguided debris removal* or the *guided debris removal* tasks (Section 4.3.2). Debris removal is achieved through extra facilities I have implemented in the Stage simulator allowing a robot to apply cumulative damage to certain objects (such as debris) in the environments. This is used to simulate the process of breaking down and eventually removing debris. The range of this equipment is 1m, meaning that the DebrisBot must directly approach a debris object in order to remove it. All other object types (e.g., tall debris, robots, or victims) are unaffected by this equipment and cannot be removed or damaged.

This robot has two sonar arrays: one of them is at ground level and the other is 20cm higher. The different heights of the sonar arrays allow the DebrisBot to distinguish between low-lying, removable debris, and taller obstacles that cannot be removed. The presence of a basic victim sensor is useful not only because it allows the detection of potential victims, but because it also allows the DebrisBot to distinguish between victims and low-lying debris. Since mistaking a victim for a piece of removable debris is obviously undesired, the DebrisBot uses its victim sensor to disqualify any detected debris that is within 2.0m of a detected victim. In more realistic settings, this prevents humans from getting directly injured as well as lowering the chances that removing debris near a victim could injure them (e.g., by causing structural collapse). To avoid situations where robots can potentially be mistaken as debris, the DebrisBot applies a similar strategy: it uses a robot detector to disqualify any debris that is nearby another detected robot (within 25cm).

The DebrisBot is highly specialized for debris-removal tasks and is poorly-suited for leadership. It lacks path planning abilities, does not maintain shared map knowledge, and cannot detect frontiers. Much like the MinBot, it does not have task-assignment capabilities but is still able to perform recruitment.

4.3 Attributes, Tasks, and Roles

4.3.1 Attributes

As mentioned in Chapter 3, every robot is described with a series of computational, physical, and sensor attributes. These allow robots to have an estimate of their own abilities, as well as the abilities of others. They are divided into three main categories: physical (Section 4.3.1.1), computational (Section 4.3.1.2), and sensory (Section 4.3.1.3) attributes.

4.3.1.1 Physical Attributes

Physical attributes define properties such as robot speed, size, how they traverse the environment, and any special equipment they possess. Below is a table describing these attributes and their values for each robot type in my implementation. I have added the *debris remover* property to the original implementation by Gunn [2011] to enable robots to have self-knowledge regarding any debris-removal equipment they carry. I have also implemented the *marker count* attribute to track the number of victim markers each robot has. All robots start the mission with 2 markers, and cannot carry more than 2. The actual number of markers a robot possesses will vary

	MinBot	MidBot	MaxBot	DebrisBot
locomotion	wheeled	wheeled	tracked	tracked
width/length	10cm x 10cm	20cm x 20cm	38cm x 44cm	50cm x 40cm
expendability	1.0	0.25	0.05	0.1
debris remover	none	none	none	yes
marker count	2	2	2	2

Table 4.2: Physical attributes for the robots in my work.

over the course of a mission as markers are dropped next to victims (Section 4.1.2.7), donated to other robots, or received by donating robots (Section 4.3.2.2).

4.3.1.2 Computational Attributes

Computational attributes define the components that exist in the robot control software. In a physical implementation of my framework, cheaper robots would possess limited CPU power, and this would place significant constraints in terms of what those robots could be expected to compute in real-time (the results of these computations may also be beyond the robot's memory capacity). For example, a physical implementation of the MinBot could conceivably be designed around the well-known Atmel ATmega328 processor [Atmel Corporation, 2015], which has a maximum safe clock speed of 20MHz and only 2KB of RAM—on this platform, storing and calculating the results of shared team map data, for example, would be impossible. A MaxBot based on the popular Pioneer platform, on the other hand, would not experience these limitations.

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

Table 4.3: Computational attributes for the robots in my work.

	MinBot	MidBot	MaxBot	DebrisBot
victim sensor	basic	advanced	none	basic
robot sensor	no	yes	yes	yes
sonar sensors	5	10	3	14
sonar range	4m	6m	2m	6m
laser rangefinder	none	none	yes	none
victim marker detector	yes	yes	yes	yes

Table 4.4: Sensory attributes for the robots in my work.

4.3.1.3 Sensory Attributes

Sensory attributes define the types of sensors that robots are equipped with. Cheaper robots would be equipped with fewer (or simpler) sensors in order to keep their cost low, and advanced robots would have greater and/or more complex sensing facilities.

Although *laser rangefinder* is used as a sensory attribute, Stage 4.1 (recall from Section 4.1.2 that I ported the Gunn framework from Stage 3.2) no longer supports the laser rangerfinder component. Instead, Stage 4.1 offers a general-purpose *ranger* component to simulate both sonar and laser sensors. I have re-implemented the laser rangefinder using the Stage 4.1 ranger facilities such that it performs identically to that used in [Gunn, 2011].

Additionally, the *victim marker detector* attribute is not an explicit attribute in my implementation, although it is listed above; all robots are assumed to have the appropriate sensors for detecting victim markers.

4.3.2 Task Types

This section describes the task types that are used in my implementation. The first seven of these (*explore*, *explore frontier*, *find team*, *find victim*, *confirm victim*, *manage team*, and *encounter*) are implemented already as part of [Gunn, 2011], and the remaining ones (*begin role recruitment*, *find robot*, *unguided debris removal*, *guided debris removal*, *donate marker*, and *wait for marker*) I have developed as part of my implementation.

4.3.2.1 Gunn Framework Tasks

The following tasks have already been implemented as part of the Gunn framework [Gunn, 2011] and are used directly in my own work:

- *Explore*: This task involves undirected, randomized exploration. This is the lowest priority task which all robots except the DebrisBot will perform when

no other tasks are available.

- *Explore Frontier*: This task involves moving to a specific location, exploring the area for a brief period, and reporting any results of the search to a team leader. Tasks of this type can only be generated by a robot with the frontier finding software module (Section 4.3.1.2).

- *Find Team*: This task is used by replacement robots when initially released into the environment. Robots executing this task will travel inwards for a total of five minutes, or until another team is encountered which it can join (or possibly form a new team with another robot).

- *Find Victim*: This task is not explicitly placed on a robot's task queue, but is used to determine a robot's suitability for filling roles that require the use of the victim tracking software module (Section 4.3.1.2).

- *Confirm Victim*: This task is used to confirm the presence of a victim where a potential victim is located. When executing this task, a robot will move to a specified location and confirm whether or not a true victim is present. Only robots with an advanced victim detection sensor (Section 4.3.1.3) are suited for this task, and only robots with the victim tracking software module can generate instances of this task (Section 4.3.1.2).

- *Manage Team*: This task is not explicitly placed on a robot's task queue, but is used to represent the set of capabilities that are required for a robot to manage a team (assign tasks, etc.).

- *Encounter*: This task is used to guide the team merge and redistribution process that occurs when teams encounter each other in the environment. It is executed by the two team representatives cooperatively.

4.3.2.2 ARC Framework Tasks

I have implemented the following tasks to support the example implementation of the ARC framework:

Begin Role Recruitment

This is a task that is only used to initialize the role-recruitment process in active recruitment settings. Recall from Section 3.8.2 that active recruitment treats recruitment duties as an explicit task that offsets regular work. When a robot starts this task, it activates the role-level recruitment communications system (Section 4.5), removes this task from its task queue, and begins executing the *find robots* task described below, in order to find a suitable robot to fill a specified role.

Find Robot

The *find robot* task is used in active recruitment settings to locate other robots. When a robot begins this task, it activates the role-level recruitment facilities (in the case of role-level recruitment, Section 4.5) or the appropriate task-level recruitment facilities (in the case of task-level recruitment, Section 4.5). The robot will then begin a random walk to increase the chances of locating a useful robot.

Unguided Debris Removal

This is a highly-specialized task that only robots equipped with debris removing equipment (Section 4.3.1.1) can perform—only the DebrisBot can execute it. This is

the idle task for the DebrisBot, and is executed only in the absence of other useful work. It involves performing a random walk, approaching, and removing any low-lying debris within a certain field of view. This task relies on the *detect obstacle* perceptual schema (Section 2.6.1) to distinguish between low-lying and non-removable debris objects.

A robot executing this task performs a random walk around the environment, avoiding obstacles such as victims, tall debris, or other robots, and moves towards short debris objects (which can be removed with the appropriate equipment). It uses the input of several different sensors to differentiate between these objects to ensure that only low-lying debris is targeted. If a low-lying debris object is detected, the robot will move towards the debris and stop a short distance away (within 1.0m in my implementation). It will then activate its debris-removal equipment, which breaks down the debris using the damage facilities I have added onto Stage (Section 4.2.4). If after six seconds (the debris-removal process takes approximately five seconds in my approach) the debris is not destroyed, the robot executing the task will abandon the current debris object (failure to remove the debris can be caused by the robot approaching a debris object slightly misaligned) and perform a random walk for two seconds that treats all objects as obstacles. This helps to prevent the robot from approaching the same debris object at the same unsuccessful angle. The robot then resumes the process of looking for debris to remove. If the debris removal was successful, the robot immediately continues to look for more debris.

Guided Debris Removal

This task is similar to the *unguided debris removal* task described in the sec-

tion above. Instances of this task are generated by robots that detect they have become physically stuck on a piece of debris, as indicated by the localization schema (Section 4.4.1). This task involves moving to a specified location and attempting to remove a piece of debris, presumably to free a robot that has become stuck at that location. If the debris-removing robot successfully navigates to the specified location, it approaches the debris (as in the *unguided debris removal* task) and begins the removal process. After an attempt to remove the debris (whether successful or not), the robot discards the task and resumes other work. If after a short period of time (30 seconds in my implementation) the robot fails to either reach the debris location or reaches the location but does not find any debris, the task is removed and the robot continues with other tasks.

Donate Marker

The *donate marker* task guides the marker donation process in the case where one robot A has agreed to donate a fiducial victim marker to another robot B (Section 4.1.2.7). A robot's suitability for this task is determined by whether or not the robot is equipped with a robot detector, and the number of markers the robot currently has (Section 4.3.1.1). A robot with an empty complement of markers is considered unsuitable for this task. Tasks of this type are assigned either directly by recruitment (where robot B requiring a marker attempts to locate a robot who can donate one), or by B passing the task to a team leader for subsequent reassignment when recruitment is not available. Upon activating this task, robot A broadcasts a wireless message to robot B, indicating that A is ready to donate. If no response is received from B after a short period of time, robot A discards the task. Otherwise,

A will activate the *turn in place* schema, attempting to visually locate B. If B cannot be located, A discards the task. If B is located, robot A activates the *move to robot* motor schema (Section 4.4.2) and approaches robot B. If after a short period of time, A has not physically reached B (e.g., as a result of becoming stuck on an obstacle), A will discard the task. If A successfully reaches B, A will give one of its markers to B and resume other work.

Wait For Marker

The *wait for marker* task controls the actions of a robot waiting to receive a marker donation from another robot. The robot halts all motion and waits for an indication from a potential donor. If after a short period of time no message has been received, the robot discards the task and resumes other work. If a donor message is received, the robot waits a certain amount of time for the donating robot to arrive and physically donate a marker. If the receiver spends too long waiting for a donation after receiving a donor message (15 seconds in my approach), it is assumed that the donating robot has become lost or stuck, and the waiting robot gives up the task and resumes normal work. Thus, the victim is not marked until it is encountered again by a robot with a marker. If the donation is successful, the receiving robot drops the marker near the victim and considers the task complete. Future work could include implementing a *drop marker at location* task, where appropriately-equipped robots are explicitly tasked with dropping a marker near a victim. Although marker dropping already occurs in my approach as a result of exploration and victim confirmation tasks, it may be useful for a robot lacking markers to specifically task another robot (either through recruitment or via a team leader) with dropping a marker at a known

victim location.

Figure 4.4 depicts the marker donation process.

4.3.3 Role Types

As mentioned previously (Section 3.2), roles are used as a heuristic to simplify the task-assignment process. This section describes the four roles that are used in my implementation. The first three (*team leader*, *coordinator/explorer*, and *explorer*) have already been implemented as part of [Gunn, 2011]. The fourth type, *debris remover*, I have added as a part of my implementation. Table 4.5 shows the calculated suitabilities of each robot for every role in my implementation. Suitability values are calculated for a particular robot-role combination by examining a robot's attributes (Section 3.2) and determining how well they satisfy the requirements of each of the tasks required by that role [Gunn, 2011]. Tasks in each role are assigned hand-tuned weights indicating the importance of that task to the role. For example, a Coordinator/Explorer role places a high emphasis on the *confirm victim* task (Section 4.3.2.1): robots with the appropriate capabilities (e.g., the MidBot has an advanced victim sensor) will thus score a higher suitability for the Coordinator/Explorer role. In the extreme case, robots failing to meet the minimum suitability requirements for all tasks associated with a role will score a suitability of zero. For example, the only expected tasks of a DebrisBot relate to debris removal (although it can perform other tasks if requested to do so, as explained in Section 3.6.5)—since all other robots do not meet the minimum requirements for debris removal tasks, they receive a suitability score of zero for the *debris remover* role.

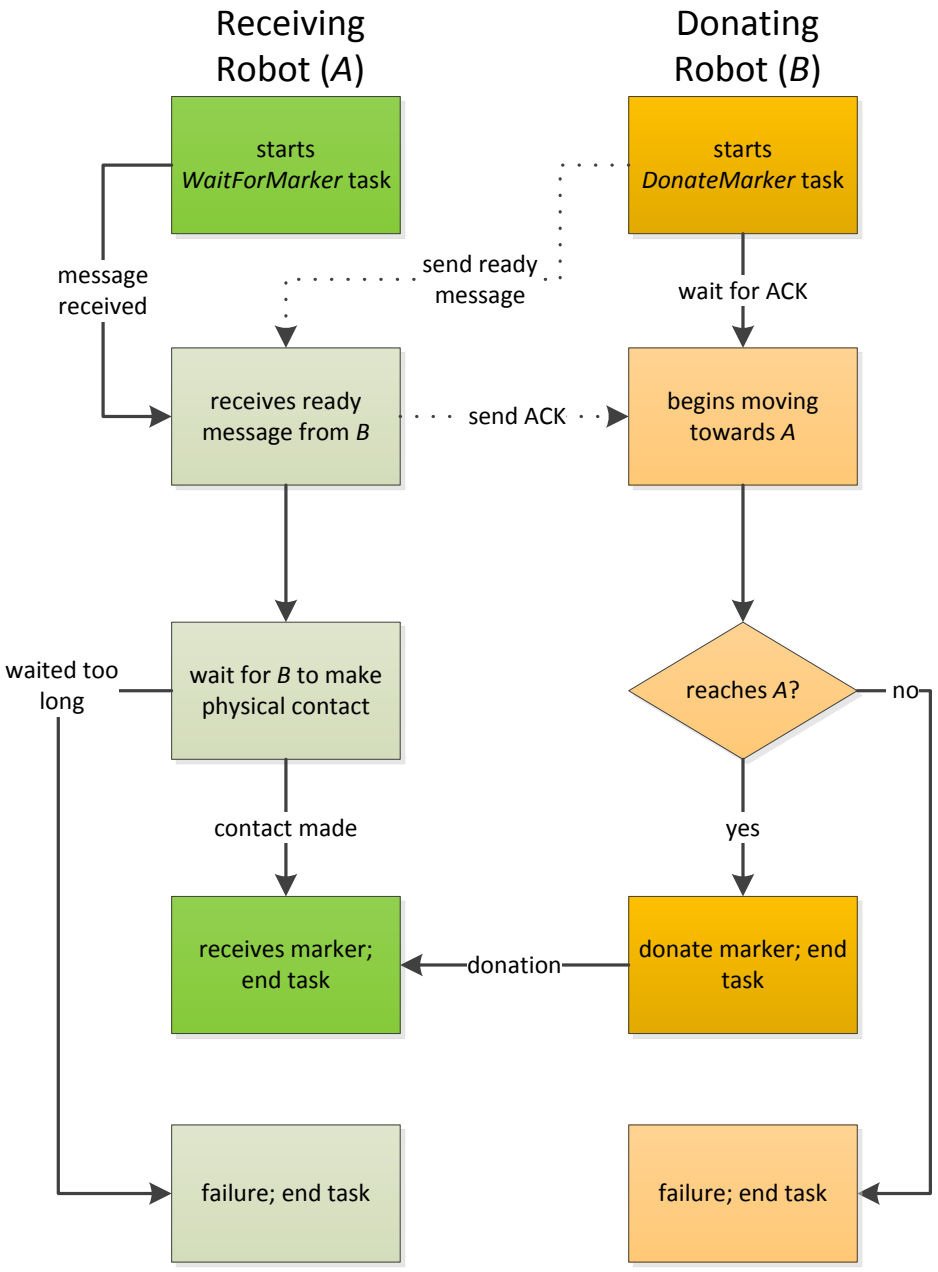

Figure 4.4: Logic flow of the marker donation process between two robots.

robot type	Leader	Coordinator/Explorer	Explorer	Debris Remover
MinBot	11	24	124	0
MidBot	73	86	56	0
MaxBot	96	42	52	0
DebrisBot	16	22	52	200

Table 4.5: Suitability values for each robot type to fill each role.

4.3.3.1 Team Leader

The team leader is responsible for directing the efforts of the robots on its team. This includes assigning tasks, detecting and assigning frontiers to be explored, and maintaining an overall map of the environment that the team has explored.

Out of the four robot types in my implementation, the MaxBot is best-suited for this particular role. It possesses frontier-detection software which enables it to direct exploration efforts more efficiently than simply assigning random exploration tasks. The planning module on the MaxBot also enables it to build paths to locations of interest such as potential victims or frontiers, so that robots can be explicitly guided to them. This greatly reduces the chances that a robot will become stuck or lost while navigating to a given location. The MaxBot also has the ability to assign tasks, which is a critical capability in terms of managing a team.

The MidBot is adequately suited for leadership. It possesses the same set of computational abilities as the MaxBot, with the exception of the planning module. Thus, the MidBot is not able to calculate path information when assigning tasks,

increasing the risk that an assigned robot becomes stuck or lost.

The MinBot is utterly unsuited for the team leader role. It lacks all leadership capabilities except the victim-tracking software. Because it lacks a frontier finder, it is not able to generate frontier exploration tasks and must perform exploration using a random walk. It is also unable to assign tasks, meaning that any other team members may find themselves short of useful work (or the team may break up altogether). Role-switch checks, however, will ensure that the leadership role is ceded to the more capable robot on the team, if one exists or is discovered later on (Section 3.5). The DebrisBot, much like the MinBot, lacks the basic capabilities required for this role, and is similarly limited.

4.3.3.2 Coordinator/Explorer

Robots occupying this role are responsible for performing exploration and victim verification tasks, as well as guiding the team distribution process when two teams encounter one another.

The MidBot is best-suited for this task since it is the only robot that possesses advanced victim-sensing capabilities enabling it to confirm the presence of victims. The MaxBot is poorly-suited for this task since it only has basic victim-sensing capabilities, but the MaxBot is equipped to handle the team distribution process. The MinBot is unable to perform both victim verification and team redistribution, making it a poor fit for this role. The DebrisBot lacks victim verification abilities, but has a robot detector that will enable it to peform team redistribution if necessary.

4.3.3.3 Explorer

Robots filling this role are expected to perform exploration tasks as guided by a team leader, for the purpose of detecting potential victims in previously unexplored areas.

The high expendability of the MinBot (i.e, it is cheap and easy to replace) makes it an ideal choice to fill this role. Exploration tasks are potentially dangerous and may result in a robot becoming lost or damaged. While the other robot types possess the capabilities this role requires, their suitabilities for this role are lower because they are better suited to fill more advanced or specialized roles.

4.3.3.4 Debris Remover

This is a highly specialized role that involves clearing low-lying debris from the environment. Specifically, robots in this role are expected to perform the *unguided debris removal* and *guided debris removal* tasks (Section 4.3.2.2). Debris removal equipment is required to perform these tasks, meaning that only the DebrisBot is suited to fill this role in my implementation. All other robots are utterly unsuited for it. The nature of the existing team management facilities implemented by Gunn [2011], however, dictate that role switches (Section 3.5) may result in a robot taking up the debris remover role on its own even if it is not equipped for it. Such a robot would not be able to perform debris-removal tasks since it lacks the necessary equipment. This results in the robot occupying the debris remover role without actually being able to perform any of the tasks required of the role, resulting in wasted effort that could be spent elsewhere (e.g., searching for victims). Subsequent encounters with

DebrisBots (either alone or as part of a team) will help to correct these issues as team rebalancing takes place.

4.3.4 Ideal Team

Recall from Section 3.4 that the *ideal team* definition describes the desired number of each robot role in a team, and is used as a guide when deciding how to restructure a team or integrate new members. Similar to [Gunn, 2011], the ideal team definition in my implementation requires 1 *team leader*, 1-2 *coordinators/verifiers*, and 3-10 robots as *explorers*. My definition also specifies that a team should possess 1-2 *debris removers*. This is depicted in Figure 4.5.

The number of robots filling the first three roles in my definition were based on values obtained through experimentation in [Gunn, 2011]. These values ensure that a team only has one leader, that the team has advanced robots capable of confirming victims or stepping in as a team leader (if necessary), and that a large number of cheap, expendable robots are available for simple exploration. I have chosen to specify a minimum of 1 and a maximum of 2 *debris removers* in my ideal team definition to ensure that stray DebrisBots are more likely to be integrated into existing teams when they are encountered. For example, restricting a team to only one DebrisBot prevents teams from integrating additional DebrisBots that might otherwise be adopting team leader roles if they become lost (Section 3.5). By allowing more than one DebrisBot on a team at one time, there is a greater probability that DebrisBots in the domain will be able to perform more useful work related to their intended role.

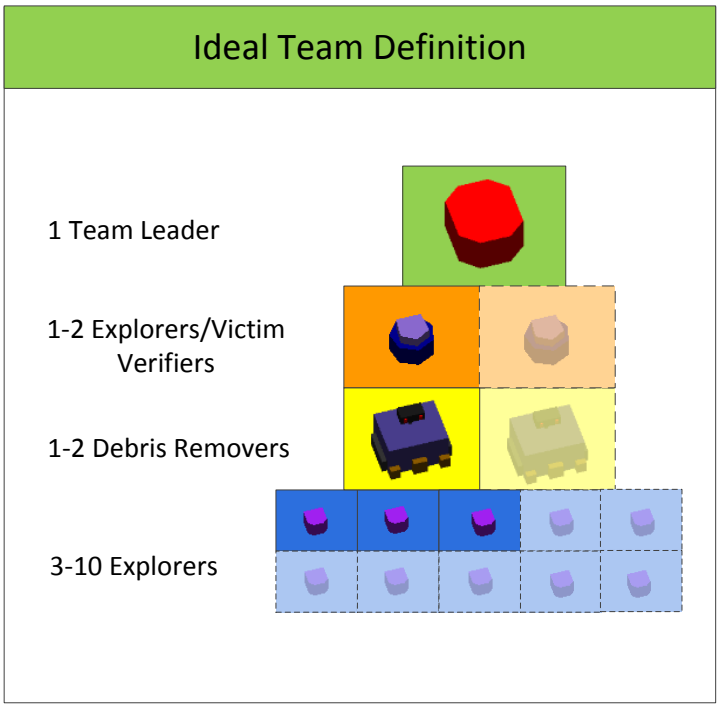

Figure 4.5: Depicts an ideal robot team as used in my implementation.

4.4 Autonomous Control

This section provides a brief description of the low-level control mechanisms for the robots in my work. With the exception of the *detect markers* schema, these have all been implemented as part of [Gunn, 2011]. They are divided into two categories: *perceptual schemas* and *motor schemas* (Section 2.6.1). I have made extensions to some of these schemas to support my work.

4.4.1 Perceptual Schemas

Perceptual schemas are used to collect raw sensor data from rangefinders, robot detectors, and victim detectors to build meaningful representations of the immediate environment.

- *Localization*: The localization schema tracks the robot's position and orientation. All team members share the same coordinate system (centered where the team began in the environment), but different teams will have different translational origins. Robots appropriately equipped can use robot detection sensors to establish translations between observed robots and their own coordinate system, so that location information (victims, frontier locations, etc.) can be passed between members of different teams. The localization schema is also used to determine if a robot has become stuck on debris. I have made extensions so that if after a short period of time (5.0 seconds) the robot is not able to free itself, it will generate a *guided debris removal* task where it is stuck and attempt to pass it along to either a team leader or another robot via recruitment (when recruitment is available).

- *Process Range Data*: This schema processes data returned from the rangefinding sensors (sonar or laser) to gather information about nearby obstacles such as debris or walls.

- *Detect Debris*: This schema uses the localization schema to detect whether or not the robot is stuck, and then records an obstacle at the robot's location if it is. This information is used to avoid the obstacle again in the future (if the

robot can free itself or is freed by a DebrisBot).

- *Detect Obstacles*: This schema uses data from the *process range data*, *detect debris*, and *detect robots* schemas to generate a list of obstacles to avoid. I have made additions to this schema allowing robots to distinguish between low-lying and high obstacles based on the height of the robot's ranger sensors. This is useful for robots that must distinguish between removable and non-removable debris objects (e.g., the DebrisBot).

- *Detect Robots*: This schema uses input from the robot detector sensor to build a list of currently observed robots. This schema can only be used by robots equipped with a robot sensor.

- *Detect Victims*: This schema uses the victim detector sensor to build a list of victims currently observed. The output of this schema is fed into the victim tracker module (Section 4.6) to generate victim confirmation tasks.

- *Detect Lost*: This schema tracks the cumulative distance the robot travels when moving to a new location. After a robot travels five times the distance required without reaching its goal, the robot removes the current task from its queue and begins work on the next one, with one exception: robots getting lost during victim confirm tasks (Section 4.3.2.1) re-queue the task for later, rather than deleting them, due to their importance.

- *Detect Markers*: I have implemented this schema to use input from the marker detector sensor to build a list of currently observed victim markers. The output

of this schema is used by the *marker manager* (Section 4.6.1) to determine if there have been any recently detected victim markers in the area.

4.4.2 Motor Schemas

Recall from Section 2.6.1 that motor schemas are low-level behaviours that use perceptual schema data to generate vectors that guide robot motion.

The *avoid obstacles*, *move to location*, *turn in place*, *random*, and *recover stuck* motor schemas have been implemented in [Gunn, 2011]. I have implemented the *seek debris* and *move to robot* motor schemas.

- *Avoid Obstacles*: This schema generates a repulsive force away from detected objects. This force grows exponentially as the robot gets closer to the perceived obstacles.

- *Move To Location*: This schema enables a robot to move to a specific location using a path (if available), or normal reactive motion if a path is not available.

- *Turn In Place*: This schema generates a motion vector commanding the robot to move in a clockwise direction. This is used to allow the robot to spin in place and look for a robot that has requested an encounter or a marker donation.

- *Random*: This schema generates a small random motion vector that is added onto the robot's current motion. This ensures that the robot does not become stuck when multiple active motor schemas generate motions that cancel each other out.

- *Recover Stuck*: This schema uses the localization perceptual schema to detect whether or not the robot is stuck. When this occurs, the *recover stuck* schema generates a backwards wiggling motion to attempt to free the robot.

- *Seek Debris*: I have implemented this schema to assist debris removal robots in targeting removable debris. It generates a motion vector attracting the robot to short obstacles (as determined by the *detect obstacles* perceptual schema, Section 4.4.1) while repelling the robot away from non-removable debris or objects that should not be targeted (such as other robots or victims).

- *Move To Robot*: I have implemented this motor schema to support the *donate marker* and *wait for marker* tasks (Section 4.3.2.2), which require that two robots physically meet in order for a marker donation to take place. This schema generates a motion vector guiding one robot towards another robot that is visible with the robot detection schema (Section 4.4.1).

4.5 Framework-Specific Modules

This section describe aspects of the robot software that specifically support my framework. With the exception of the *recruitment manager*, all of these have been implemented as a part of [Gunn, 2011], although I have made additions to several of them to support my framework.

- *Encounter Manager*: The *encounter manager* module provides facilities for controlling the team merge and distribution process when two robots encounter one another in the domain.

- *Knowledge Manager*: This module tracks any information known to the robot. This includes the robot's self-attributes, the attributes of other robots encountered, and the current structure of the team (which is unlikely to be perfect except at the outset of a mission). Knowledge regarding other robots is considered to be 'stale' after a robot has not been heard from after three minutes. Stale robot knowledge is not used in role-switch or task assignments, but can be reactivated by reestablishing communication. If a robot does not hear from any of its team mates after three minutes, for example, the robot considers itself to be on a team of one, and subsequent role switch checks will result in that robot adopting a leadership role for its own single-robot team. Subsequent encounters with other robots correct this as teams merge or redistribute themselves.

- *Communication Manager*: The *communication manager* is responsible for processing wireless messages received from other robots involving team redistribution commands, task assignments, and status updates regarding victims or completed tasks. My work has required minor extensions to this module to support recruitment-related messages.

- *Role Manager*: This module is responsible for handling role- and team-related processes such as the role-switch check. I have made extensions to this module implementing the *missing role check* process (Section 3.9.2). I have also extended the role manager to handle the role-level recruitment process as described in Section 3.9.2.1.

- *Task Manager*: The *task manager* handles robot work loads. It manages the

task queue (Section 3.6.1), handles task preemption, and manages all communications and processes related to task assignment. I have made additions to this module enabling robots to use the already-existing *exhaustive task assignment* (Section 3.6.5) facilities to perform task-level recruitment (Section 3.9.1 in both active and concurrent recruitment settings (Section 3.8).

- *Active Recruitment Manager*: This module manages active recruitment states and tracks the path the robot has travelled (when recruiting actively) in order to locate another robot; when the robot completes its recruitment tasks, it uses this path information to guide itself back to its team's last known location. Path-tracking facilities are not necessary in concurrent recruitment settings since robots perform regular work while recruiting (Section 3.8.1).

4.6 Mission-Specific Modules

This section describes the modules that are specific to the USAR implementation of my framework. All of these modules with the exception of the *marker manager* have already been implemented by Gunn [2011].

Importantly, not every robot will have instances of all modules due to differences in computational ability. Computational attributes (Section 4.3.1.2) are used to determine which facilities are available on a robot.

- *Environment Mapper*: This module is responsible for integrating the results of exploration tasks to maintain a shared team map of the environment. This requires considerable computation and storage facilities and is not present on

every robot.

- *Frontier Finder*: The *frontier finder* module examines a robot's map of the environment and uses an occupancy grid-based approach to locate areas of exploration (Section 4.1.2.3) and generate *explore frontier* tasks (Section 4.3.2.1). This requires considerable computation and storage facilities and is not present on every robot.

- *Victim Tracker*: This module is responsible for maintaining and updating a robot's list of currently known victims and their statuses. Since victim detection is central to my work, all robots in my implementation possess this module.

- *Planner*: The *planner* is used to compute paths to significant locations (e.g., victims, frontiers, etc.) that can be given to robots as part of a task assignment. Because of the computational effort required by the pathfinding algorithm, only the MaxBot possesses a *planner*.

4.6.1 Marker Manager

I have implemented the *marker manager* system to handle the process of releasing markers when victims are detected, as well as to control when and how often a robot should ask for markers when its own supply is low.

4.6.1.1 Releasing Markers

To avoid wasting victim markers by constantly deploying them whenever a victim is detected, robots in my implementation use the marker manager to track how long

it has been since the robot has observed a victim marker in the environment. The marker manager also tracks the time that has passed since the robot has dropped a marker near a victim. When a robot detects a victim close by (1.5 meters in my work), the marker manager goes through the following process: if a sufficiently long time has passed since the last marker has been seen (30 seconds), and if a marker has not been dropped by the robot recently (60 seconds), then the robot will drop a marker. If any of these conditions are not met the robot will not drop a marker.

The type of victim sensor a robot has also plays a part in this process. Robots with basic victim sensors cannot distinguish between true and false victims; as a result, these robots may drop markers near false victims. Robots with advanced victim detection capabilities, however, will only consider dropping a marker if the detected victim is in fact a real victim. Future work could include giving robots with sophisticated victim sensors the ability to pick up markers deposited near false victims that were dropped by robots with only basic victim sensors.

The marker manager also prevents robots from dropping markers during the execution of a marker donation task (Section 4.3.2.2), to ensure that a robot agreeing to donate a marker actually has a marker to donate by the time it reaches the receiving robot.

4.6.1.2 Requesting Marker Donations

If a robot without any markers wishes to drop a marker near a victim, and the conditions for doing so are met (as outlined above in Section 4.6.1.1), the robot will make a request to receive a marker donation from another robot. It does this

by creating a *donate marker* task, and attempts to either (*a*) recruit someone for that task (when recruitment is enabled), or (*b*) pass the task along to a team leader for delegation to another team member. To prevent an overload of potential task assignments or recruitment processes, robots requiring a marker donation will only create *donate marker* tasks every 45 seconds in my implementation.

4.7 Conclusion

In this chapter I have described the implementation of my framework in a simulated USAR domain. The following chapter describes the experiments I have designed and performed to evaluate my framework.

Chapter 5

Evaluation

5.1 Introduction

To evaluate the effectiveness of my methodology (Chapter 3), I conducted experiments in a simulated USAR domain, comparing the effectiveness of the *concurrent* and *active* recruitment configurations against that of the baseline *passive* recruitment configuration (Section 3.8). In my evaluation I consider factors such as the reliability of wireless communication and the probability that any given robot can experience temporary or permanent failures during a mission.

The next section reiterates the research questions my work aims to answer. In the sections that follow, I describe the evaluation metrics employed and the process I used to generate experimental USAR environments. Following this, I describe my experimental design (one major factorial experiment and two follow-up experiments), and then present the results of these experiments and discuss my findings.

5.2 Review of Research Questions

Recall my research questions from Section 1.6.

1. **To what degree does the use of various recruitment strategies improve or hamper the overall performance of teams of heterogeneous robots in dangerous domains?**

2. **What factors (e.g., availability of replacement robots, wireless reliability, probability of robot failure) determine the recruitment strategy that should be used in a given situation?**

3. **Which recruitment strategies, if any, result in overall best performance of teams of heterogeneous robots if wireless reliability levels are not known?**

I answer my first research question by conducting a set of experiments in which I vary the recruitment strategy and parameters affecting communication reliability and robot failure. The second research question is answered by examining the relative success of different recruitment strategies given these hazards, in order to determine how these factors should influence the choice of recruitment strategy when they are known. I answer the third question by examining the performance of my recruitment strategies in environments with varying levels of communication reliability.

5.3 Evaluation Metrics

To evaluate the performance of the USAR implementation of my framework, I record four metrics over the duration of each mission: *percentage of victims successfully identified to leaders*, *percentage of area coverage*, *debris items removed*, and *successful marker donations*. My implementation captures these values at the end of every experimental trial.

My software tracks the total number of successfully identified victims known to each individual robot. A victim is considered successfully identified when it has been confirmed as either a positive or negative victim. For the purpose of my evaluation, however, I only consider victims that are known to team leaders, since such robots would be expected to have the most complete picture of a team's operational knowledge. I do not take into account victim knowledge acquired by replacement robots (recall from Section 3.5 that single robots will eventually become their own team leaders) that have yet to join a team. Additionally, I do not process victim knowledge acquired by MinBots due to their inability to merge and maintain team knowledge (Section 4.2.1). There is also the possibility that a team may lose a leader near the end of an experiment, leading to a further loss of information as the leader can no longer receive information from other robots. Statistical information is still captured from disabled leaders, however. These metrics make my results more conservative than they would be if I counted all victims identified by all robots, and this is the same approach used by Gunn [2011].

Environmental coverage is recorded by merging map exploration data from the same set of robots that victim information is captured. This is the *union* of area

coverage data that has been successfully communication to team leaders as a result of exploration tasks (Section 4.3.2.1). This is the same area coverage metric used by Gunn [2011]: as with victim statistics, I do not count coverage by replacement robots that have yet to join a larger team, and there may be individual robots that possess individual information that has not been communicated to a team leader at the end of any trial. Additionally, the loss of a leader near the end of an experiment can cause leaders to miss important information that would otherwise be included in statistics capturing. Once again, this makes this metric more conservative than simply counting all coverage by all individuals. All other statistics, which I describe below, are gathered from all robots.

Debris removal tracking is divided into two separate categories: (a) debris removed by unguided removal, and (b) debris removed by guided removal (these tasks are explained in Section 4.3.2.2). This allows me to examine the performance of the DebrisBot and determine how various factors (communication success rate, robot failures, etc.) affect certain types of debris removal tasks.

Successful marker donations are tracked by counting the number of times a marker donation has successfully taken place (Section 4.6.1.2). This metric is particularly important because marker donations require coordination between two robots, and the success of any cooperative actions are likely to be heavily influenced by the environmental conditions and the recruitment strategy used.

5.4 Environment Generation

I evaluated my implementation in multiple simulated environments to reduce the effects that bias inherent in any single environment would have on my results. To support this, I used a two-phase approach similar to that of Gunn [2011] to generate environments in which to evaluate my implementation.

5.4.1 Initial World Generation

Using a software tool developed by Wegner [2003] and further modified by Gunn [2011], I generated 40 simulated environments as candidates for my experimental evaluation. I made modifications to this tool to support the inclusion of DebrisBots. In all other respects, environment parameters were identical to those used by Gunn [2011]: all environments measure 60m x 60m, with approximately 13% of the environment covered by debris and obstacles, and 60% of these debris configurations are passable only by robots with tracked drives (i.e., MaxBots and DebrisBots can traverse these types of debris, but other robots cannot) and can be removed by robots with the appropriate equipment (i.e., DebrisBots).

Victim placement is identical to the approach used by [Gunn, 2011]: 10 negative victims and 20 positive victims (the difference between these is discussed in Section 4.1.2.1) are distributed randomly in the environment such that victims are at least 0.8 meters away from a wall and do not overlap each other.

The placement of robot teams also matches that of [Gunn, 2011]: two openings are built into opposite ends of each environment, allowing one-way entry of two separate teams at the onset of the mission (indicated by red circles in Figure 5.1). The

distribution of replacement robots is also the same: replacements are released from equally-spaced positions along the inner perimeter (indicated by blue dots in Figure 5.1).

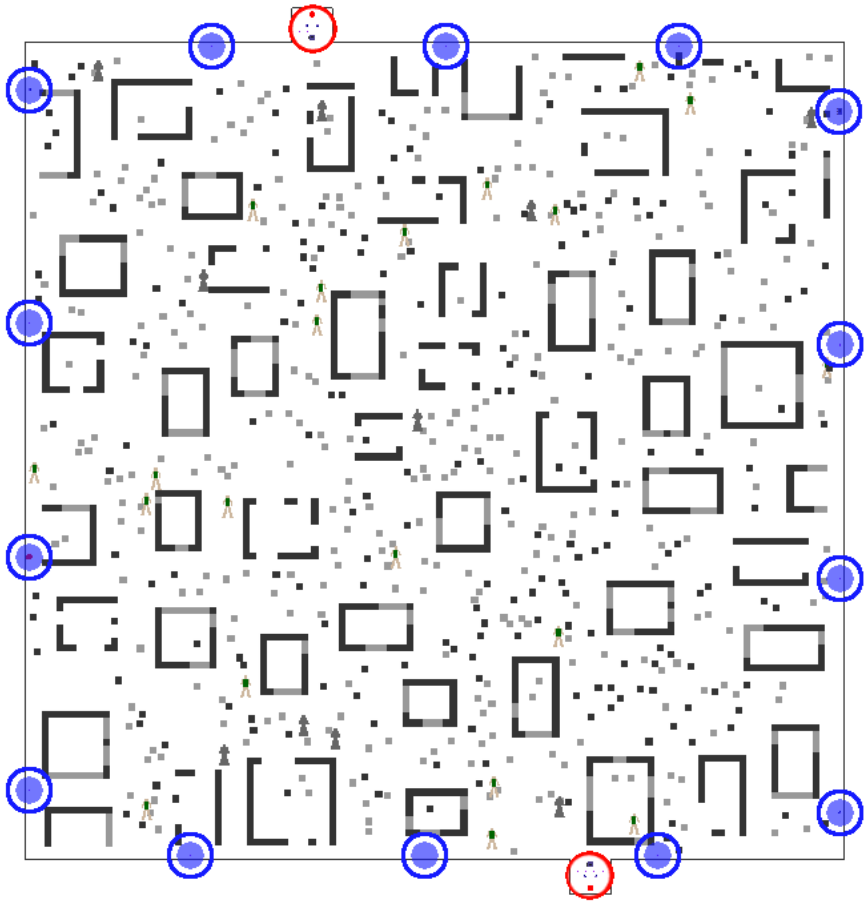

Figure 5.1: An example randomly-generated world.

After the initial environment generation, I selected 3 environments to use for running my experiments. My selection criteria were similar to those of Gunn [2011]:

- I discarded any environment with significant debris blocking the area in which a team is initially deployed, since the time necessary to navigate such debris would contribute to lower team performance over the course of the half-hour

missions. Unlike Gunn [2011], however, I allow limited removable debris objects to be present in front of a team's starting position due to the added presence of a DebrisBot on each team. The DebrisBot is able to remove this debris immediately upon the start of a mission and prevent it from hindering the team's progress.

- I discarded any environments where the presence of large obstacles blocked a replacement robot's starting position, to avoid wasted time as the robot attempts to navigate around it.

- Any environment with highly clustered debris or obstacles was discarded, since the environments should be uniformly covered with debris.

- Environments with highly clustered victim populations were not used, in order to ensure that each environment required roughly the same level of effort when locating victims.

The output of this first phase was 3 simulated USAR worlds (represented as Stage *.world* files) that I used as a starting point for generating my experimental trials. All of my environments are shown in Appendix A.

5.5 Experimental Design

I used a factorial experiment design to evaluate the effectiveness of my recruitment strategies, employing the 3 selected worlds I generated as described in Section 5.4.1. Limitations of the Stage simulator dictate that experimental parameters (e.g., wireless

communication success rate, probabilities of robot failure, random seeds, etc.) must be specified in the *.world* files themselves—this required me to generate one *.world* file for every experimental configuration.

Stage (as well as the Gunn and ARC frameworks) guarantee repeatability through the use of random seed integers embedded in a *.world* file. Running any particular *.world* file will generate the same results each time, since the random seed in a *.world* file cannot be changed. This is useful when re-running trials for observation. To ensure sufficient breadth of events in my simulations, I used identical sets of 50 random seeds for each experimental configuration, resulting in 50 *.world* files for every combination of independent variables (described in detail in Section 5.5.1). This helped to ensure that differences in robot team performance across different experimental configurations could not be attributed to random numbers that inadvertently favoured a group of trials. This ultimately resulted in 8100 *.world* files, which were run concurrently (approximately 36 at a time) using Amazon Web Services. Figure 5.2 shows my experimental breakdown. The variables depicted in the figure are explained further in the following subsection.

5.5.1 Independent Variables

With the exception of the *recruitment configuration* variable, the independent variables used in my factorial experiment are taken from [Gunn, 2011] in order to facilitate comparison with that work:

- *Replacement Robots*: This variable controls the availability of replacement robots in my experiments—either they are present or they are not. When replacement

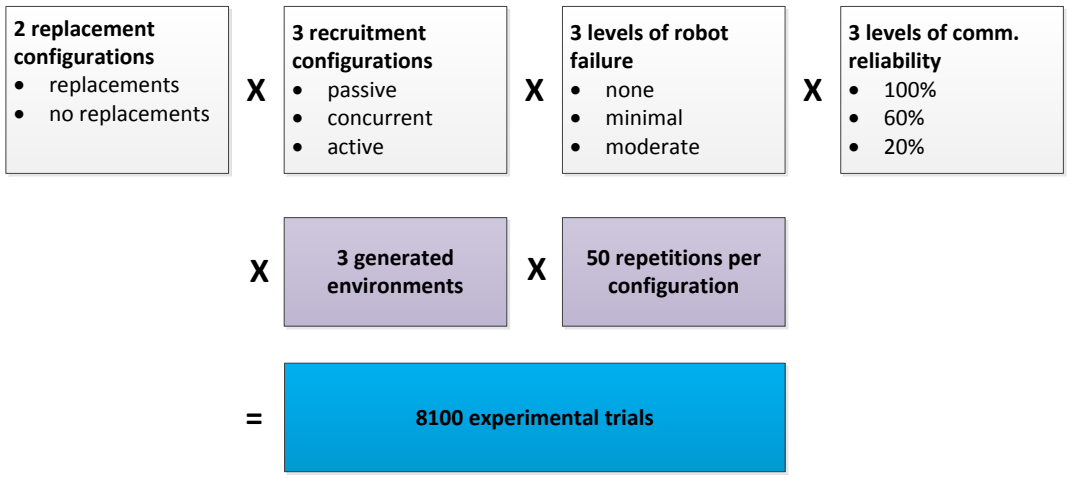

Figure 5.2: Depicts my experimental breakdown.

robots are used, 10 additional MinBots, 2 replacement MidBots, 1 DebrisBot, and 1 MaxBot are available. They are introduced into the environment after five simulated minutes and are spread equidistantly around the inner perimeter of the environment, facing inwards. With the exception of the DebrisBots, this setup is identical to that of [Gunn, 2011].

- *Recruitment Configuration*: This defines the recruitment mode used—this will be *active*, *concurrent*, or *passive* recruitment (Section 3.8). Recall from Section 3.9 that task-level and role-level recruitment techniques will both be enabled together in active or concurrent recruitment settings.

- *Robot Failure*: This variable controls the probability that a particular type of robot will experience a temporary or permanent failure (Section 4.1.2.6). Temporary and permanent failure probabilities are specified separately for each robot type, as shown in Table 5.1. These are the same values as used in [Gunn,

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%**
	DebrisBot	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%**
	DebrisBot	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%**
	DebrisBot	0.002	0.014	**25.1%**

Table 5.1: Robot failure probability configurations.

2011].

- *Communication Reliability*: This defines the probability that any given wireless message will be successfully delivered to its intended recipients (Section 4.1.2.5). As in [Gunn, 2011], I use communication success rates of 100%, 60%, and 20%.

5.6 *Additional DebrisBots* Experiment

I conducted a second experiment in which I inserted additional DebrisBots into the environment as replacements. This *Additional DebrisBots* experiment was done to study the effect that extra DebrisBots would have on the number of successful guided debris removals (Section 4.3.2.2), since this particular metric is difficult to interpret given the small number of DebrisBots in my main experiments. I substituted two replacement MinBots with DebrisBots, resulting in a total of five DebrisBots in the environment. All other experimental parameters were identical to those of my main experiments, with the exception of the presence of replacements—since I was specifically concerned with testing the effect of multiple replacement DebrisBots, I did not run any trials where replacements were not present. This resulted in 4050 experimental trials. Figure 5.3 shows the breakdown for this experiment. The results of this experiment are given in Section 5.9.

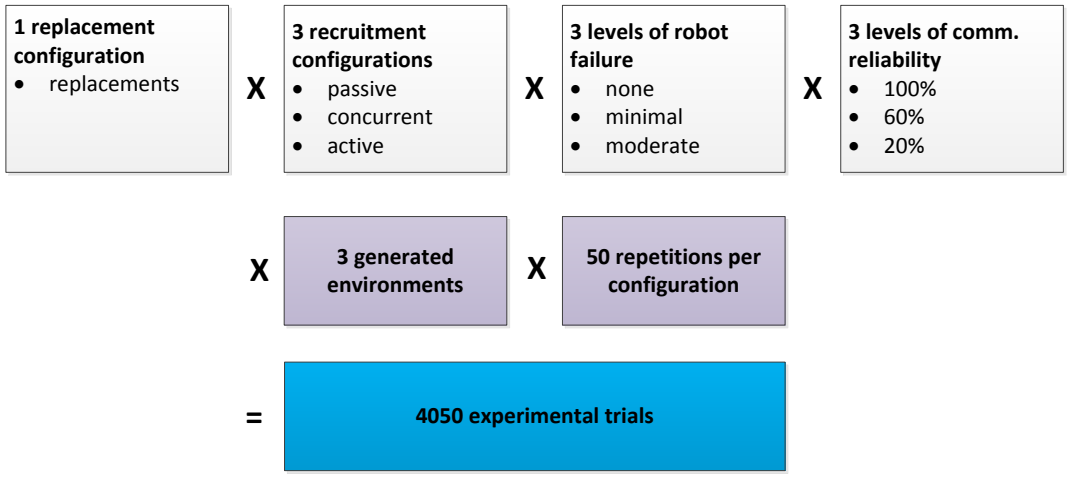

Figure 5.3: Depicts the breakdown for my *Additional DebrisBot* experiment.

5.7 *Minimal Team* Experiment

I performed a third, *Minimal Team* experiment to compare area coverage levels and the number of victims identified for all three recruitment strategies when operating with a minimal team and no replacement robots. The purpose of this experiment was to determine how robot performance was impacted when few robots were available for recruitment or task assignment in general. In this experiment, robot teams only contained 1 MinBot, 1 MidBot, 1 DebrisBot, and 1 MaxBot. All robots were given 4 victim markers each (Section 4.1.2.7) instead of the usual 2, to offset the lower number of markers available to the team as a whole—this ensured that robots in this experiment had the same number of victim markers (16) as they did in my main experiment. For this particular experiment, the ideal team definition (Section 4.3.4) was changed to exactly match the initial make-up of each team (1 of each type of robot). This was done to ensure that robots would not focus too much effort into recruiting robots that were unlikely to be available given the smaller teams and the (known) lack of replacements. Note, however, that this does not preclude the possibility of role-level recruitment taking place if a team detects that a member has become lost. Additionally, task-level recruitment is still possible. All other experimental parameters were identical to the previous two experiments. The breakdown of this experiment is also identical to that of the *Additional DebrisBots* experiment (Figure 5.3), yielding 4050 trials.

5.8 Main Experiment Results

In this section I describe the results I obtained by running the trials for my main experiment, as described in Section 5.5. Statistics were collected at the end of every trial, including the metrics described in Section 5.3. The results presented in the following subsections show the impact of communication unreliability and robot failures, as well as the effect of the presence of replacement robots. Error bars on all charts that follow indicate a confidence interval of 95%.

It is important to note that victim identification results (Section 5.8.1) and area coverage results (Section 5.8.2) in passive settings are similar, although slightly higher than, those of Gunn [2011] in ideal communication conditions with no robot failures. As conditions deteriorate, area coverage and the percentage of victims identified in passive settings do not degrade as severely as they do in [Gunn, 2011]. This is likely due to the added presence of DebrisBots (Section 4.2.4), who are able to traverse short obstacles and perform unguided debris removal in the absence of more important work (Section 4.3.2.2). This enables robot teams to gain additional coverage due to the extra robot in general, and the DebrisBot also clears debris for the other team members. While it is possible that the improved equation for determining role switch weights (Section 3.5) has an effect on these results, I do not believe this is a significant factor, for the same reason given in [Gunn and Anderson, 2015]: the number of situations that the altered equation affects is small. The addition of DebrisBots (1 on each team, plus 1 replacement, if replacements are used), however, is a significant factor and is likely the cause of these differences. In all other respects, my experimental setup is identical to that of Gunn [2011], and so I believe that the

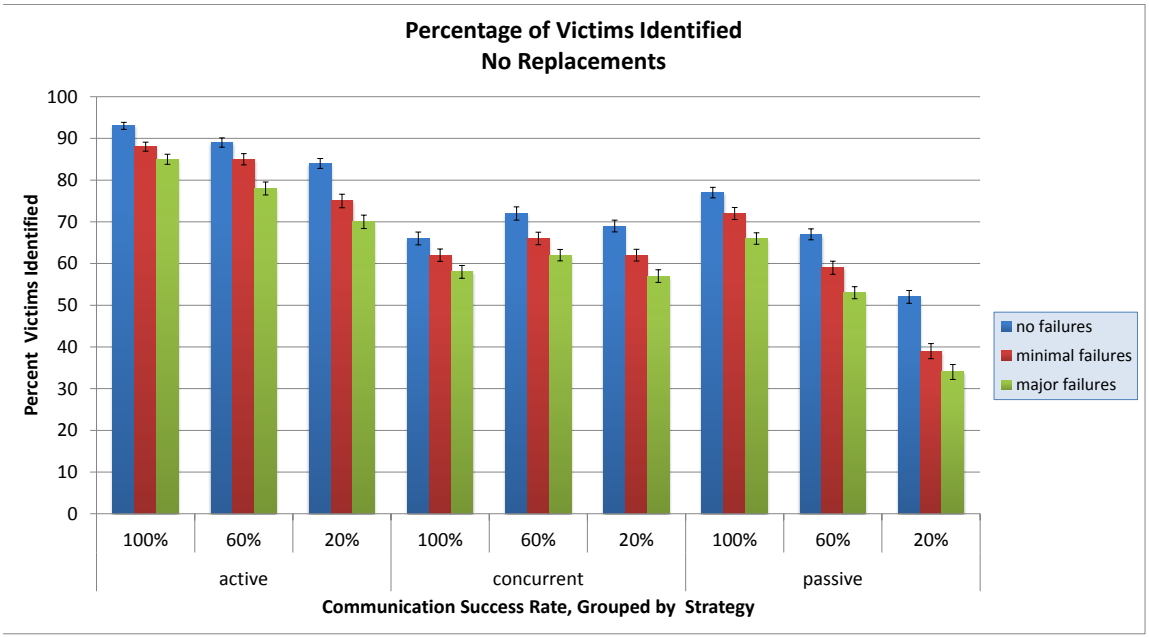

Figure 5.4: Percentage of identified victims, with no replacement robots available.

results shown in the subsequent sections are directly comparable to that work.

5.8.1 Victims Identified to Leaders

Figures 5.4 and 5.5 shows the percentage of victims identified to a team leader, for each recruitment strategy under varying communication success rates and robot failure rates.

The largest improvements between my new recruitment strategies and passive recruitment are seen when communication success rates fall to 20%. A major weakness of [Gunn, 2011] was that extreme communication failures caused teams to completely break down, indicating that extremely low communication success rates were not sufficient to maintain adequate team performance—as communications failed in passive settings, fewer victim confirmation tasks (Section 4.3.2.1) could be assigned by a team

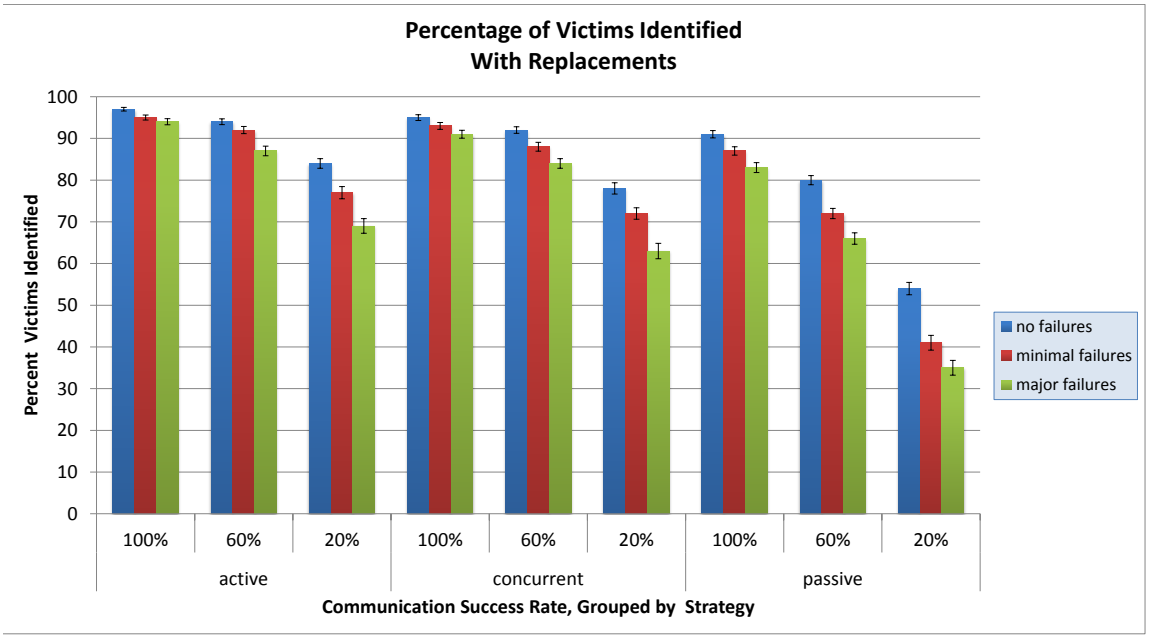

Figure 5.5: Percentage of identified victims, with replacement robots available.

leader. Active recruitment strategies, however, allow other robots to perform task assignment as well (i.e., task-level recruitment, Section 3.9.1), allowing more opportunities for victims to be confirmed. Role-level recruitment (Section 3.9.2) also provides additional opportunities for information exchange, since robots changing teams bring new victim knowledge with them. Additionally, the physical act of searching during active recruitment allows recruiting robots to cover more ground (area coverage is discussed in Section 5.8.2) and discover more victims.

In conditions of 100% communication success rate and no robot failures, active and concurrent recruitment performed similarly when replacements were available. Given that there were no failures, there should be less need for replacements (although robots can still leave a team by getting lost). The replacement robots available essentially become a bonus, and both concurrent and active recruitment perform

equally well at finding and integrating them. When replacements were not available, active recruitment provided more noticeable gains over concurrent recruitment. In the case where communications are 100% reliable, robot failures do not occur, and replacements are not available, the only robots that can be recruited to replace those lost from a team must either be from an existing team, or previously lost robots. In either case, a more active approach is of greater use.

Beyond that one case, active recruitment outperformed passive in all experimental conditions, regardless of robot failure levels or the communication success rate. Overall, active recruitment outperformed passive in all experimental conditions, regardless of the presence of replacement robots. These improvements likely arise as a result of the physical searches that robots perform when actively recruiting: as mentioned above, physical exploration increases the chances of encountering other robots and also provides opportunities to discover new tasks, such as victims that need to be confirmed. Robots encountering one another as a result of recruitment also have the opportunity to fill more useful roles on other teams and exchange useful information as a result.

It is interesting to note that in ideal conditions with no replacements, concurrent recruitment was outperformed by passive recruitment. The likely cause of this phenomenon is the slight duplication of effort that occurs as non-leader robots in concurrent settings assign victim-confirmation tasks to other robots without the intervention of a team leader. Without the broader perspective that the leader possesses (i.e., which victims are confirmed and which are still unknown), victim confirmation tasks may be unnecessarily assigned to more than one robot, resulting in wasted

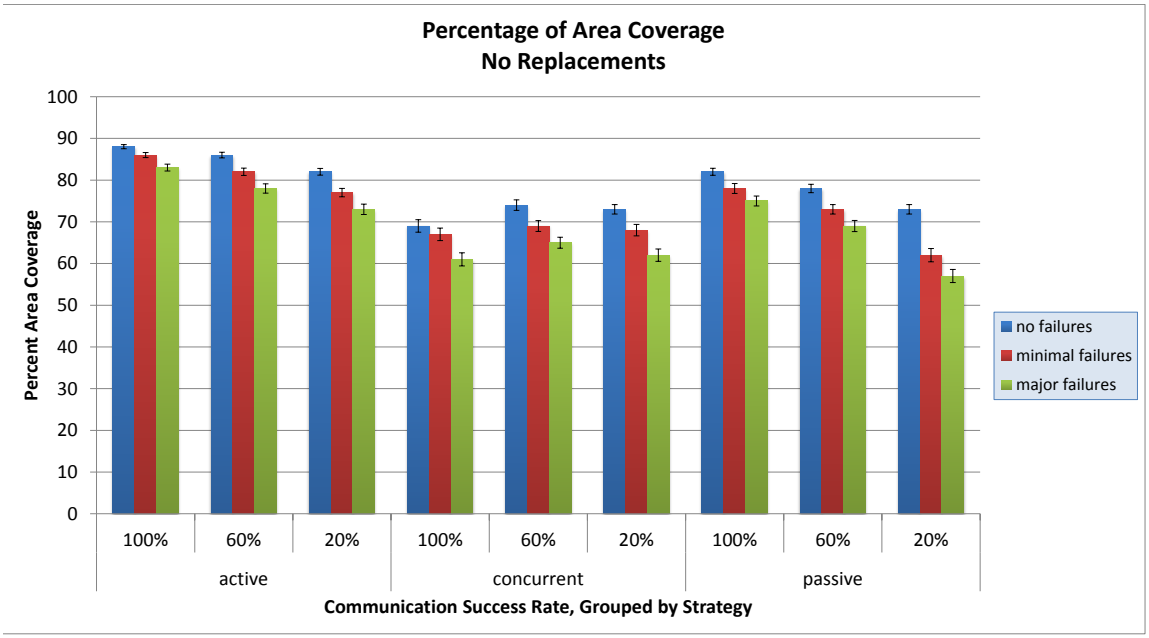

Figure 5.6: Percent area coverage, with no replacement robots available.

robot efforts that should be directed elsewhere. Since passive recruitment relies on a single leader to make task assignment decisions, this duplication of effort is much less likely to occur—the leader assigns tasks using knowledge accumulated from the other members of the team. However, this reliance on a leader and communications with it is also shown to be a vulnerability. As communications degrade or as robots fail, the redundancy in robot efforts provided by concurrent recruitment becomes advantageous and results in more victims identified than in passive settings.

5.8.2 Area Coverage

Figures 5.6 and 5.7 show the percentage of robot area coverage for each recruitment strategy, under varying communication success and robot failure rates.

With the exception of conditions with extremely low communication success rates,

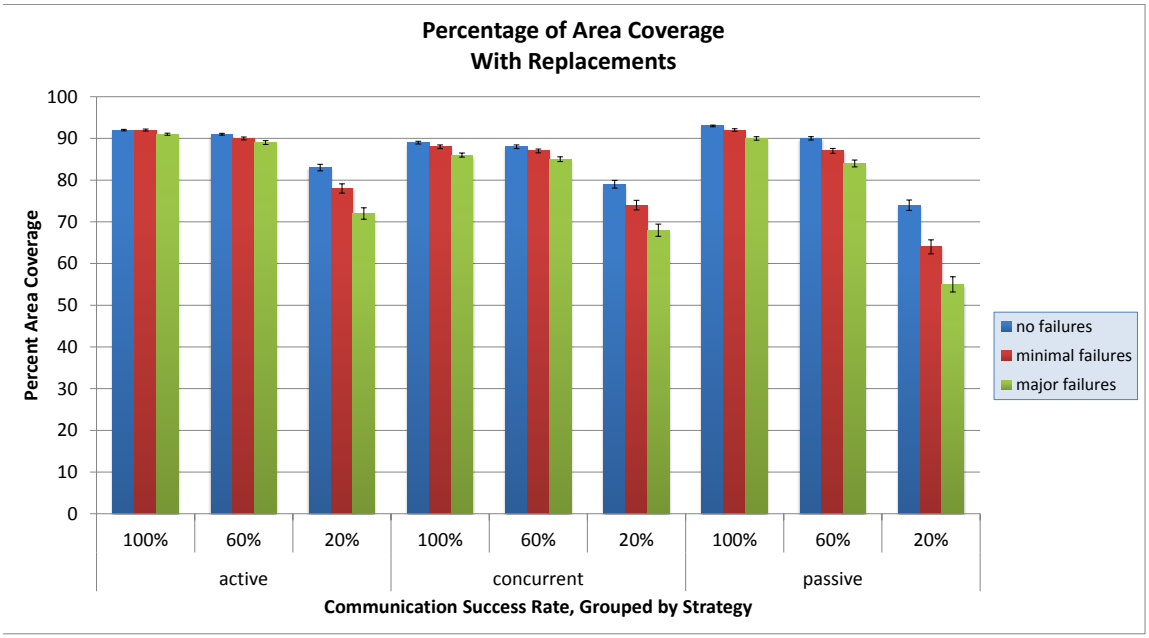

Figure 5.7: Percent area coverage, with replacement robots available.

there is not much difference in area coverage across recruitment strategies when replacements are available. A possible reason for this could be that 30 minutes is not sufficient time to allow robots to cover the entire area, and the area coverage levels shown in Figure 5.7 represent the maximum area that can be covered in that time. It is also possible that my experimental environments, shown in Appendix A, contain hard-to-reach areas that make it difficult for my robots to achieve more than 90% environmental coverage. This could also be reflective of my conservative tracking of environmental coverage, since the remaining area may have actually been explored by isolated robots or not communicated to a team leader. As communication success rates fall to 20%, and as robot failures become more significant, the effect of using active and concurrent recruitment over passive recruitment becomes more obvious. With a 20% communications success rate, not only is it more difficult to communicate

exploration results to a team leader, but it becomes much more difficult to effectively manage a team (e.g., receive task assignments, accept assignments, negotiate team rebalancing, etc.), and the ability to more actively recruit individuals has a greater impact.

Similarly, greater failure rates have a corresponding impact. The fact that there is only a small difference in area coverage between 100% and 60% communication success rates and low levels of failure indicates that the environment is forgiving enough in these conditions to allow near-complete coverage of the environment during a trial without significant recruitment. Less-forgiving conditions, however, are more reflective of USAR environments and show the utility of my recruitment techniques. It is probable that real robots in actual USAR conditions will exhibit a greater level of failure than is used in my experiments, and will further demonstrate the usefulness of these strategies.

My recruitment strategies also show their effectiveness when no replacement robots are available, and active recruitment results in greater coverage of the environment compared to concurrent and passive recruitment in all experimental configurations. This is due to the physical searches that take place in these settings, as well as the improved likelihood of recruiting new team members (possibly to replace failed robots): as robot teams exchange members over the course of a trial (e.g., a robot from team 1 may join team 2, and a robot from team 2 may eventually join team 1), useful environmental information can be shared.

Interestingly, concurrent recruitment performed poorer than passive recruitment in terms of area coverage, except in conditions of severe communication failure. The

likely explanation for this is a duplication of exploration effort, as noted in Section 5.8.1. Recall from Section 4.3.1.2 that MidBots have the ability to detect frontiers (Section 2.6.2). Since in concurrent recruitment settings robots are able to assign tasks, MidBots are thus able to assign *explore frontier* (Section 4.3.2.1) tasks to other robots via recruitment when frontiers are discovered. In passive recruitment settings, however, only team leaders are able to assign exploration tasks. Since these tasks originate from a leader with a unified view of the environment, exploration tasks will be allocated so as to minimize duplicate area coverage. MidBots in concurrent recruitment settings, however, are not likely to have as complete a picture of the environment as a team leader. Subsequent exploration tasks (assigned by MidBots) result in some duplication of effort as robots are tasked with exploring areas that have already been examined.

Figures 5.9 and 5.8 show area coverage in situations of major robot failure and 20% communication success rates, with and without replacement robots respectively, for each of the three recruitment strategies. When replacements were not available, concurrent recruitment performed slightly better than passive due to the fact that exploration tasks could be assigned by non-leader robots. When replacements were available, however, concurrent recruitment resulted in much greater area coverage than passive recruitment, since my recruitment strategies are better able to leverage the existence of replacements in severe environmental conditions, when compared to passive recruitment. Regardless of the presence of replacement robots, active recruitment resulted in the highest area coverage over both concurrent and passive recruitment in these conditions. This is likely due to the combination of physical

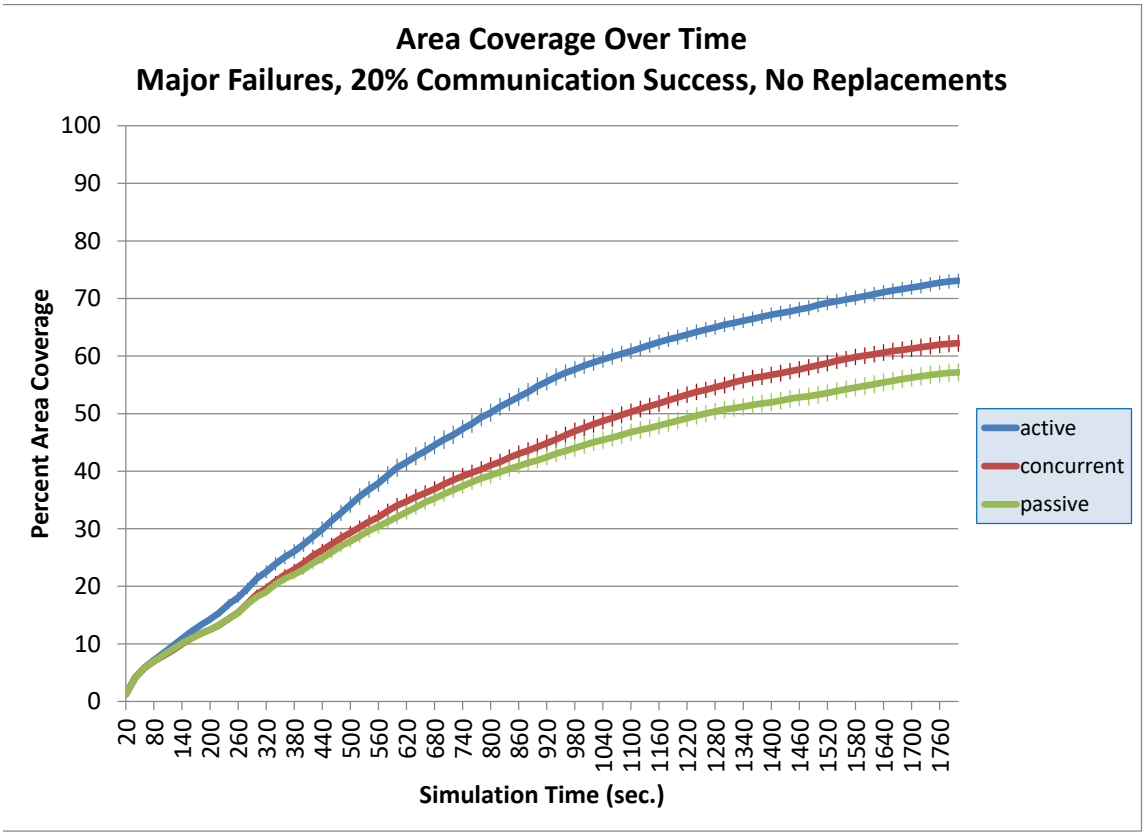

Figure 5.8: Percent area coverage over time, with major failures and 20% communication success rates, with no replacement robots available.

searches that take place in active recruitment settings as well as the ability of non-leader robots to assign exploration tasks.

5.8.3 Debris Removed

Recall that DebrisBots are able to clear small debris objects as a result of their *unguided debris removal* or *guided debris removal* tasks (Recall from Section 4.3.2.2 that unguided debris removal is performed by an idle DebrisBot, and guided debris removal is executed at the request of a robot that has become stuck). My simulation

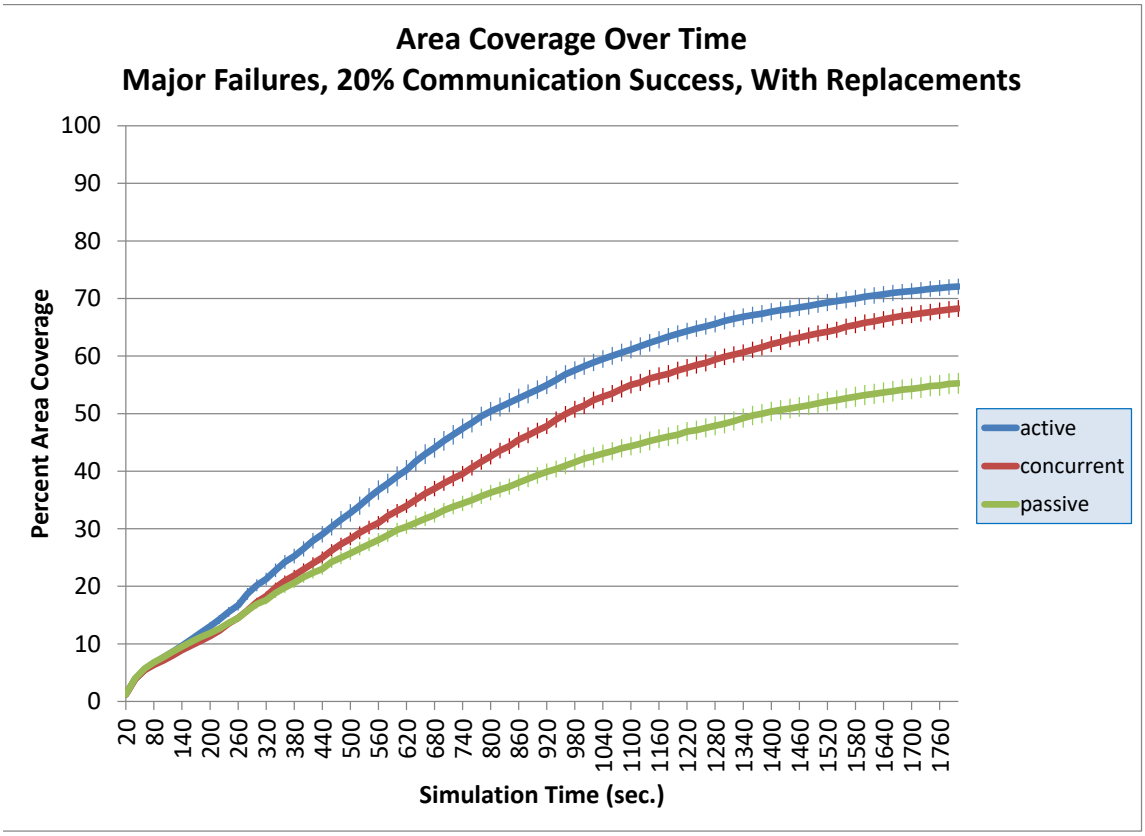

Figure 5.9: Percent area coverage over time, with major failures and 20% communication success rates, with replacement robots available.

software tracks the success of these two types of debris removals separately since this yields more useful information about the behaviour of the DebrisBots. The next two sections describe these results.

5.8.3.1 Unguided Debris Removal

Figures 5.10 and 5.11 show the number of debris objects removed as a result of unguided debris removal, by recruitment method, communication success rate, and probability of robot failure.

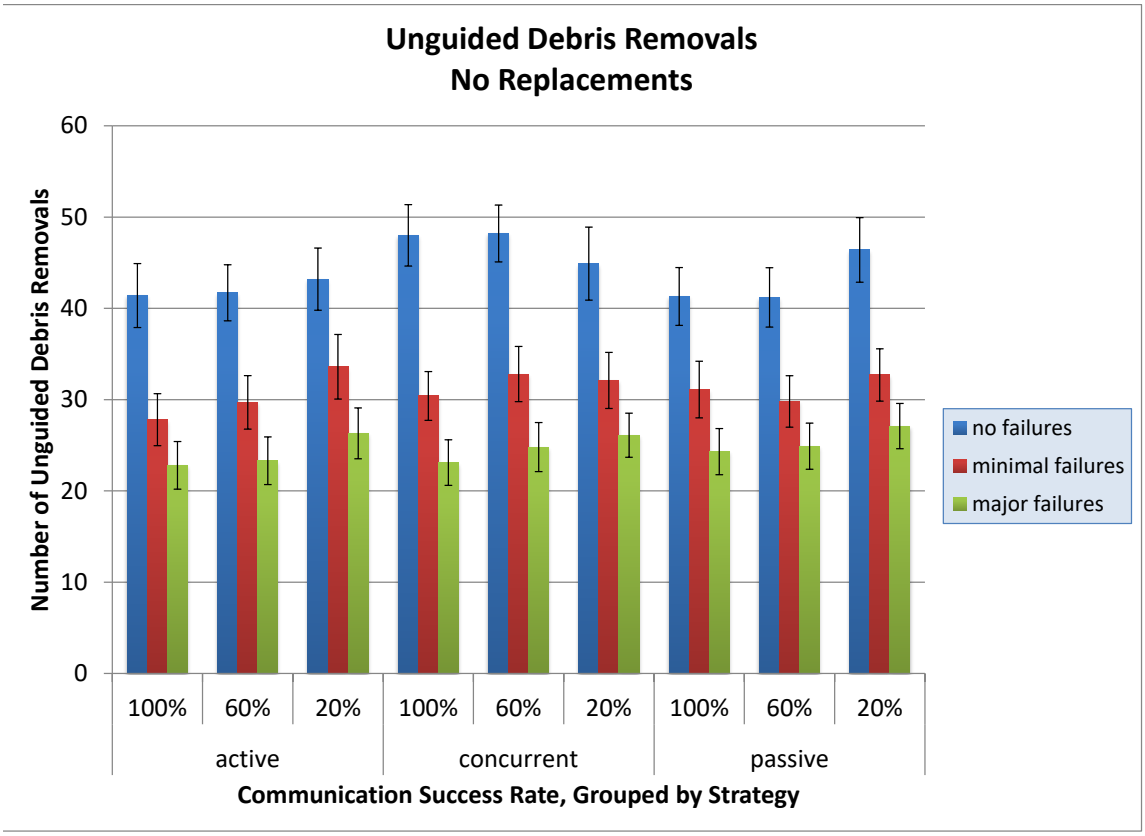

Figure 5.10: Amount of debris removed during execution of the *unguided debris removal* task, with no replacements available.

The amount of debris removed was slightly greater in concurrent recruitment settings than in passive settings. This is likely because in passive settings, DebrisBots are unable to assign tasks via recruitment to other robots, and must spend more time executing tasks other than debris removal. The amount of debris removed in concurrent settings was also higher than in active settings. This is due to the fact that DebrisBots performing active recruitment must perform a physical search at the exclusion of other useful work, i.e., DebrisBots cannot perform unguided debris removal tasks while they are actively recruiting. Concurrent recruitment does not suffer from

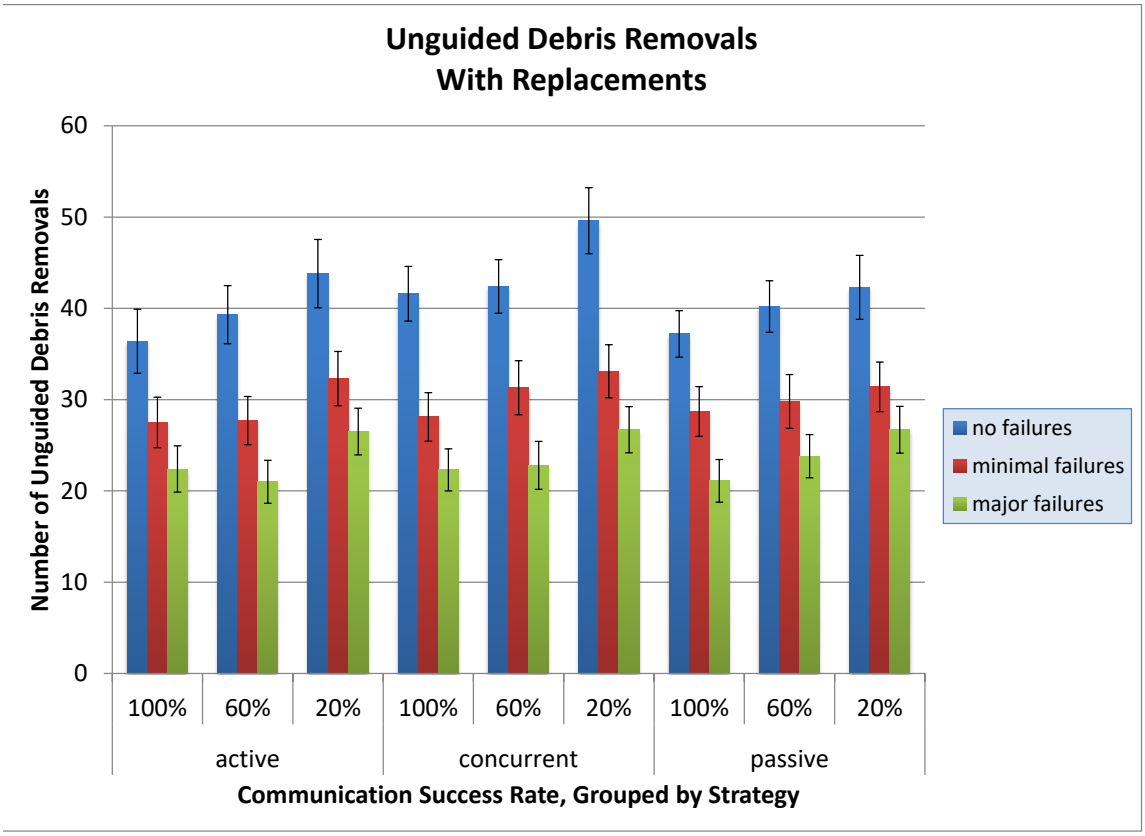

Figure 5.11: Amount of debris removed during execution of the *unguided debris removal* task, with replacements available.

this problem, since there is nothing preventing a DebrisBot from continuing useful work (e.g., unguided debris removal) while simultaneously broadcasting recruitment requests.

The addition of replacement robots resulted in more debris removal when communication was poor, since robot replacements include one DebrisBot (Section 5.5.1) and lower communication reliability provides extra opportunities for DebrisBots to look for debris in the absence of higher-priority task assignments.

Another phenomenon can be observed when robot failures occur: the number of

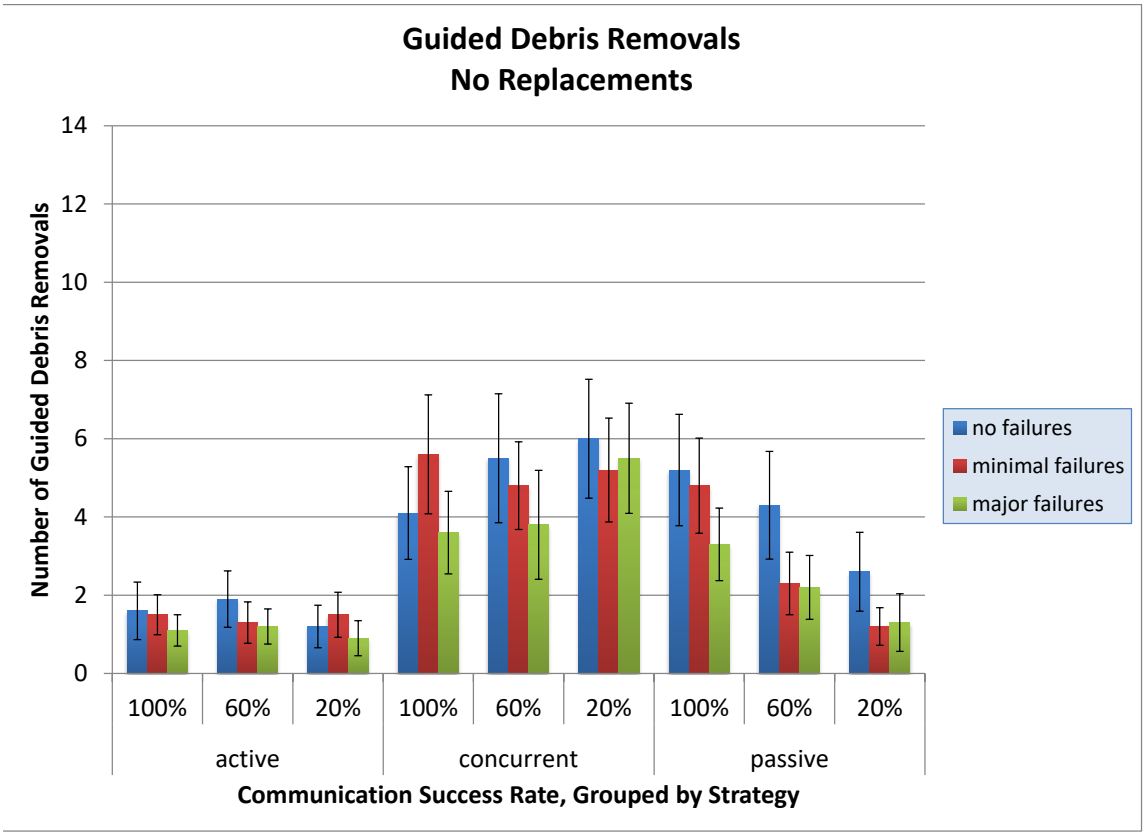

Figure 5.12: Amount of debris removed during execution of the *guided debris removal* task, with no replacements available.

unguided debris removals decreases as robot failures increase. This is because the higher number of robot failures results in DebrisBots being assigned tasks that would otherwise be assigned to other available robots. In real-world USAR conditions, this effect would likely be seen even more strongly due to the fact that robots would fail very often.

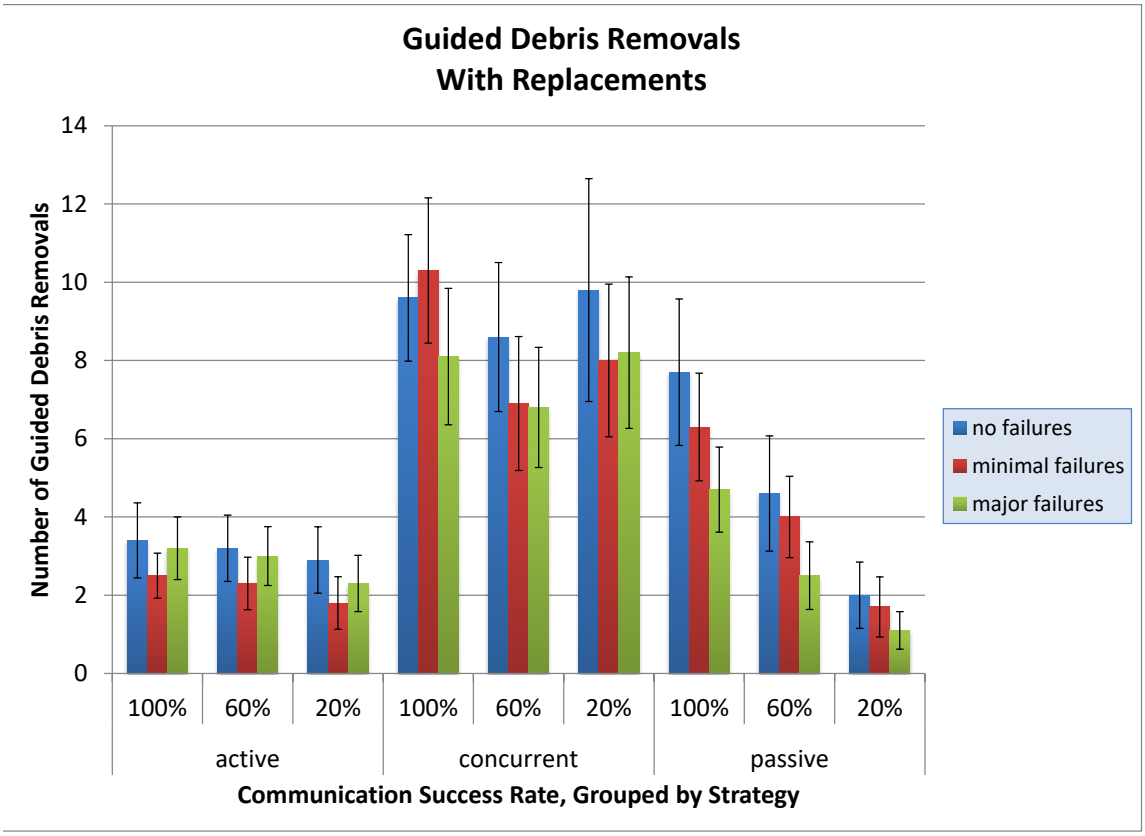

Figure 5.13: Amount of debris removed during execution of the *guided debris removal* task, with replacements available.

5.8.3.2 Guided Debris Removal

Figures 5.12 and 5.13 show the number of debris objects removed as a result of guided debris removal, by recruitment method, communication success rate, and probability of robot failure.

Recall from Section 4.3.2.2 that instances of *guided debris removal* tasks are only created when a robot has detected that it has become stuck. A debris object that is removed relatively close by to a stuck robot is considered a success, but does not necessarily indicate that the robot has been freed. This is because the robot may

have become stuck on an obstacle that cannot be removed (such as a wall—visual observations of some simulations indicate that robots are more likely to become stuck against walls that form tight corners), and the DebrisBot may simply have removed debris that happened to be very close by. These numbers are lower in comparison to the number of *unguided* debris removals (Section 5.8.3.1) due to the infrequency with which robots become stuck, compared to the amount of debris that is removed during unguided debris removal. The infrequency and variability in robots becoming stuck means there is significant variability with which guided debris removals occur, and this is reflected in the wide confidence intervals seen in Figures 5.12 and 5.13. This makes it more difficult to draw useful conclusions regarding the effect of various recruitment strategies and environmental conditions. However, some useful observations can still be made.

Active recruitment configurations resulted in the lowest number of guided debris successes. This is likely because robots recruiting for targeted debris tasks in active settings perform physical searches that take them further away from the stuck robot who originally created the task. While a physical search may ultimately result in finding a DebrisBot sooner, the DebrisBot receiving the task may have to travel across a significant portion of the environment in order to reach the stuck robot. This increases the probability that the DebrisBot may be intercepted and recruited for more important duties along the way (thus moving the guided debris tasks further down the robot's task queue), as well as increasing the possibility that the DebrisBot becomes stuck on (non-removable) obstacles. Concurrent and passive recruitment do not involve physical searches, making it more likely in those settings that a targeted

debris removal task will ultimately be successful. As communications degrade in a passive setting, however, the likelihood of a guided debris removal task being conveyed to a DebrisBot lowers signficantly (due to the required intervention of a team leader), accounting for the lower number of guided debris removals seen in Figure 5.12 when communications fall to 20% in passive settings.

The presence of replacement robots results in a greater number of guided debris removals. A combination of factors likely contributes to this. The addition of a replacement DebrisBot provides redundancy and increases the probability that a stuck robot will be assisted. Also, the greater number of non-disabled robots (due to the presence of replacements) means that there are more robots that can become stuck, potentially increasing the number of targeted debris removal tasks. Lastly, the presence of replacement robots means that more robots are available in general to perform work that might otherwise be assigned to a DebrisBot—thus, the DebrisBot is more available for guided debris removal tasks.

5.8.4 Marker Donations

Figures 5.14 and 5.15 show the number of fiducial victim markers that were successfully transferred to robots who requested them (Section 4.1.2.7), by recruitment strategy, robot failure rate, and communication reliability rate. These numbers are generally low because of the high degree of coordination and communication that is required. Unlike other tasks, marker donation requires the efforts of two robots, with one robot executing the *give marker* task and another executing the *wait for marker* task, and assumes that neither robot has more important work to complete.

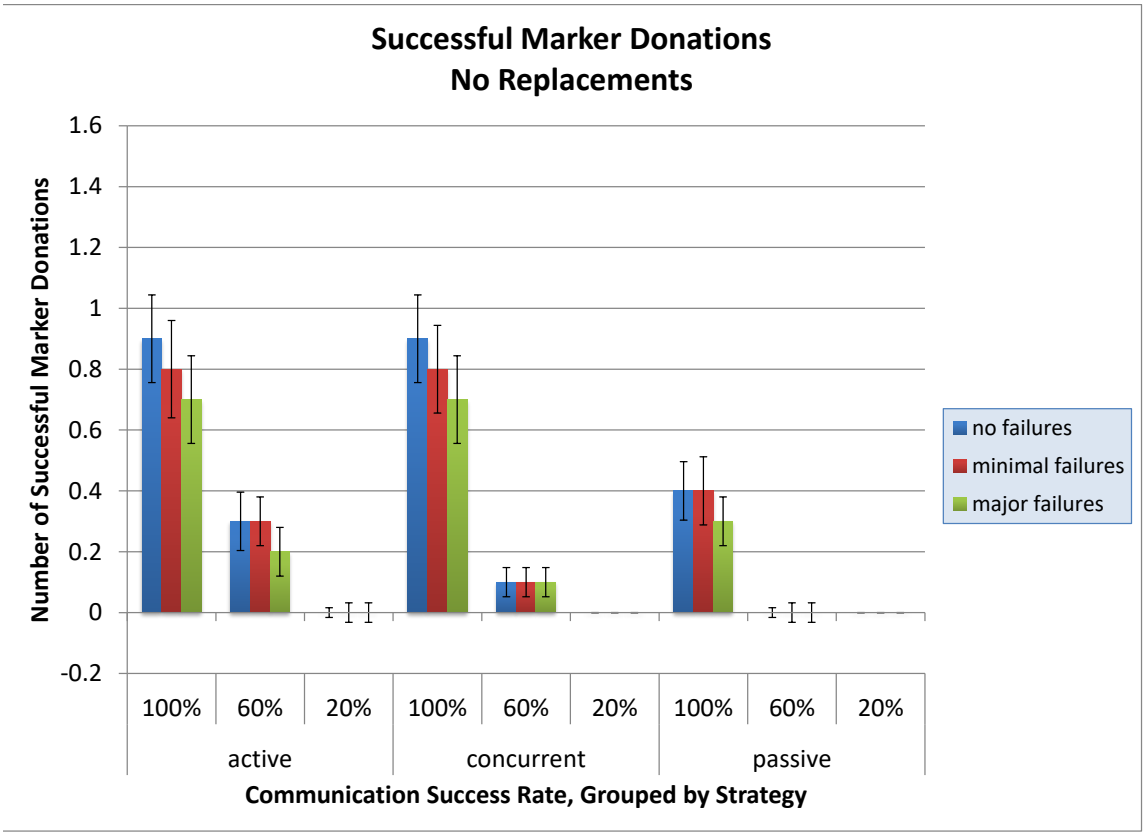

Figure 5.14: Number of successful marker donations, with no replacements.

Additionally, the donating robot must have established line-of-sight with the robot requiring a marker, which will not occur frequently due to the number of obstacles in the environment. It is also possible that each robot carried too few victim markers, making it likely that they would have none to donate after having previously encountered even a small number of victims (recall from Section 5.4.1 that every environment contains 10 negative and 20 positive victims).

Communication success rates had a significant impact on the number of fiducial victim markers successfully donated to robots requiring them. The tasks involved in a marker donation require a high degree of coordination, and communication fail-

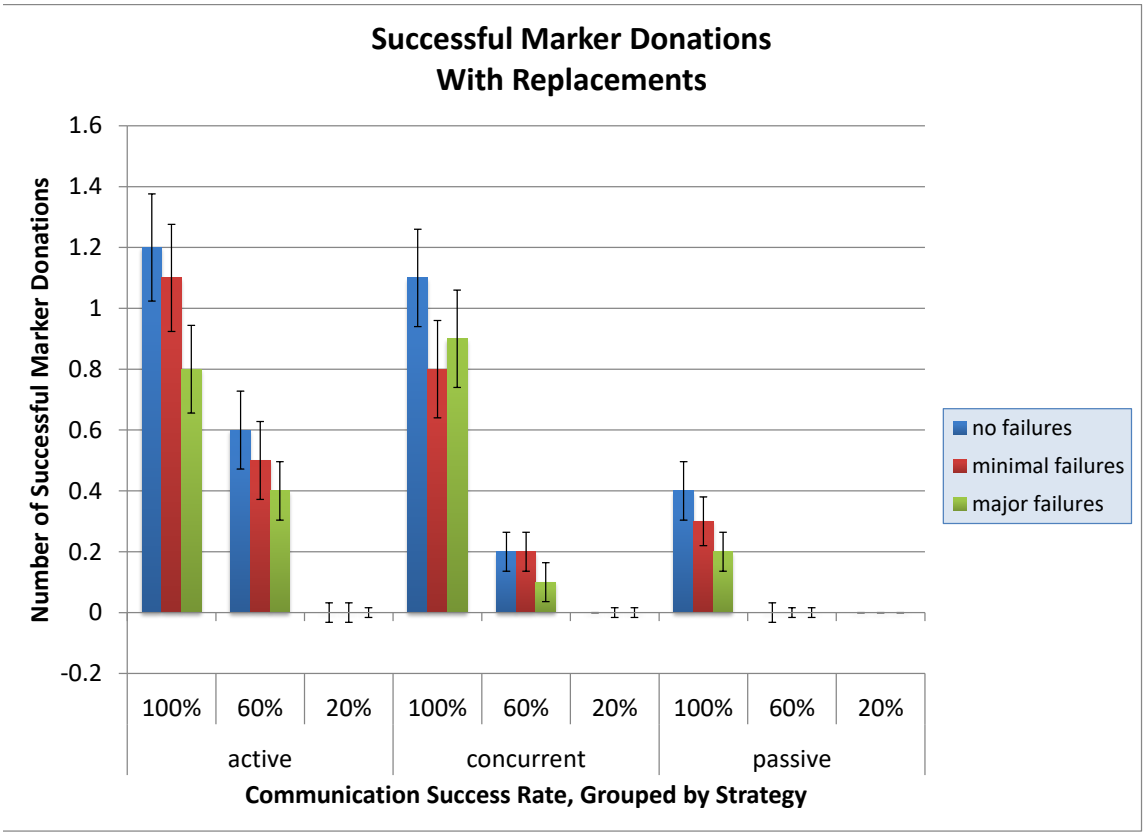

Figure 5.15: Number of successful marker donations, with replacements.

ures occurring with either robot at any point causes this process to fail. Extremely low communication success rates (20%) resulted in virtually no successful marker donations.

Robot failures also had an effect on the number of marker donations that could be completed. This is caused by the lower availability of working robots to provide marker donations, as well as the lower number of working robots that can request them.

Both active and concurrent recruitment performed better than passive recruitment when communications were reliable. This is because recruitment facilities provided

robots with extra opportunities to successfully pass on a *give marker to robot* task themselves to a unit that was able to donate. Passive recruitment, on the other hand, required a robot to pass on the task to a team leader, who would then assign the task to another robot. The team leader therefore acts as a bottleneck, which is problematic if they are out of wireless range or otherwise unavailable (i.e., due to radio or robot failure).

The presence of replacement robots resulted in a slight increase in the number of successful marker donations due to the larger number of robots available. The gains, however, are small. A possible reason for this is that the majority of replacement robots were MinBots, which only carry a single victim marker each. While a robot with one marker is still able to donate, any MinBot encountering an unmarked potential victim (e.g., shortly after being deployed as a replacement) will cause the MinBot to drop its only marker, precluding the possibility of making a donation.

5.8.5 Effect of Failures and Communication Success Rates

Robot failures and communication success rates both had significant impacts on the effectiveness of my recruitment techniques, and these two variables affected my results in different ways depending on the metric being observed. Victim identification (Section 5.8.1) and area coverage (Section 5.8.2) were most greatly impacted by communication failures, since these metrics relied on information being passed to team leaders. While robot failures did have an effect on these two metrics, the effect of failures was less than that of communication success rates, which was the limiting factor in these results.

Unguided debris removal was most greatly affected by robot failures. As mentioned in Section 5.8.3.1, more robot failures led to more tasks being assigned to DebrisBots. This resulted in fewer opportunities for unguided debris removal since DebrisBots would be busy with tasks that would otherwise be assigned to other robots. It is difficult to determine the effects of communication success rates and robot failures on *guided* debris removals due to the wide confidence intervals shown, as discussed in Section 5.8.3.2.

Marker donation (Section 5.8.4) was slightly affected by robot failures due to the varying number of working robots who could donate victim markers, but this metric was most strongly affected by communication success rates. The high degree of coordination required to complete this joint task made marker donations more difficult to accomplish with communication success rates of 60%, and virtually impossible to complete at communication success rates of 20%.

5.9 *Additional DebrisBots* Experiment Results

Figure 5.16 shows the results of my *Additional DebrisBots* experiment (Section 5.6), by recruitment strategy, communication success rate, and robot failure probability. Error bars indicate a 95% confidence interval.

The results of this experiment are similar to those of the main experiment's guided debris removal amounts (Section 5.8.3.2). While the amount of debris removed is higher in the *Additional DebrisBots* experiment because of the increased number of robots available to clear debris, it is clear that in all experiments, concurrent recruitment consistently outperforms passive recruitment, and passive recruitment

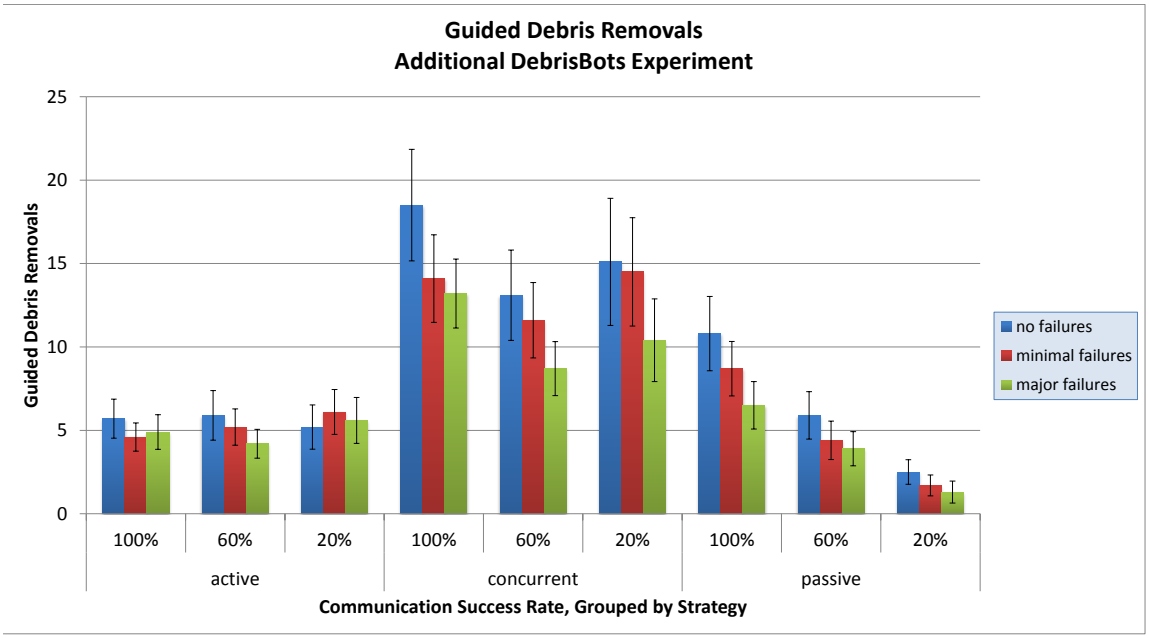

Figure 5.16: Amount of debris removed during execution of the *guided debris removal* task, during my *Additional DebrisBots* trials.

outperforms active recruitment (except when communication is poor). This suggests that for uncommon tasks, active recruitment is not always an ideal strategy. As recruiters move away from the uncommon task's location to find a robot to perform the task, there is a greater risk that a recruit may not be able to navigate to the task's location since it is further away, or that the recruit may be intercepted for other duties along the way. Also similar to the main experiment's guided debris results, there is significant variability in the number of guided debris removals in any trial, leading to wide 95% confidence intervals. The discussion regarding wide confidence intervals presented in Section 5.8.3.2 similarly applies to these results.

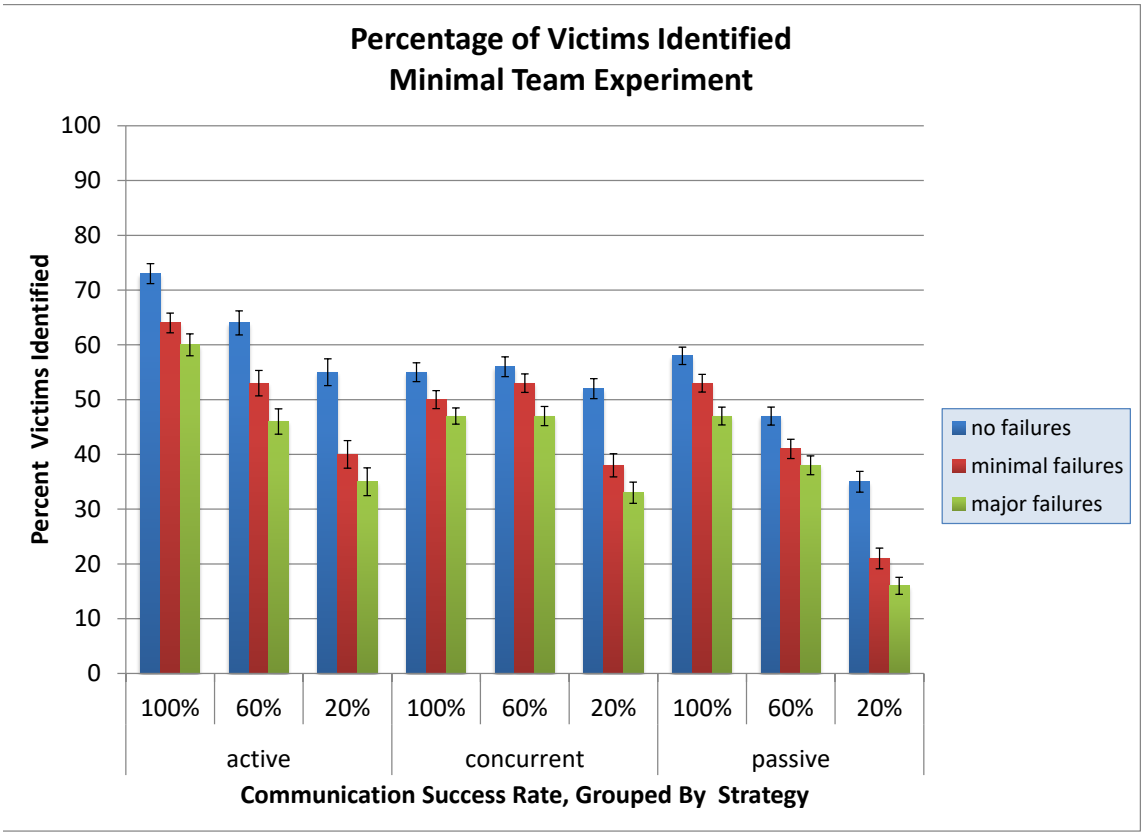

Figure 5.17: Percentage of victims identified, during the *Minimal Team* experiment.

5.10 *Minimal Team* Experiment Results

In this section, I describe the results of my *Minimal Team* experiment. Figures 5.17 and 5.18 show the number of victims known to team leaders and area coverage, respectively, by recruitment method, communication success rate, and probability of robot failure. As with my main experiment, these statistics are captured from team leaders.

Active recruitment outperformed both concurrent and passive recruitment in terms of the percentage of victims identified, in almost all experimental configurations. This

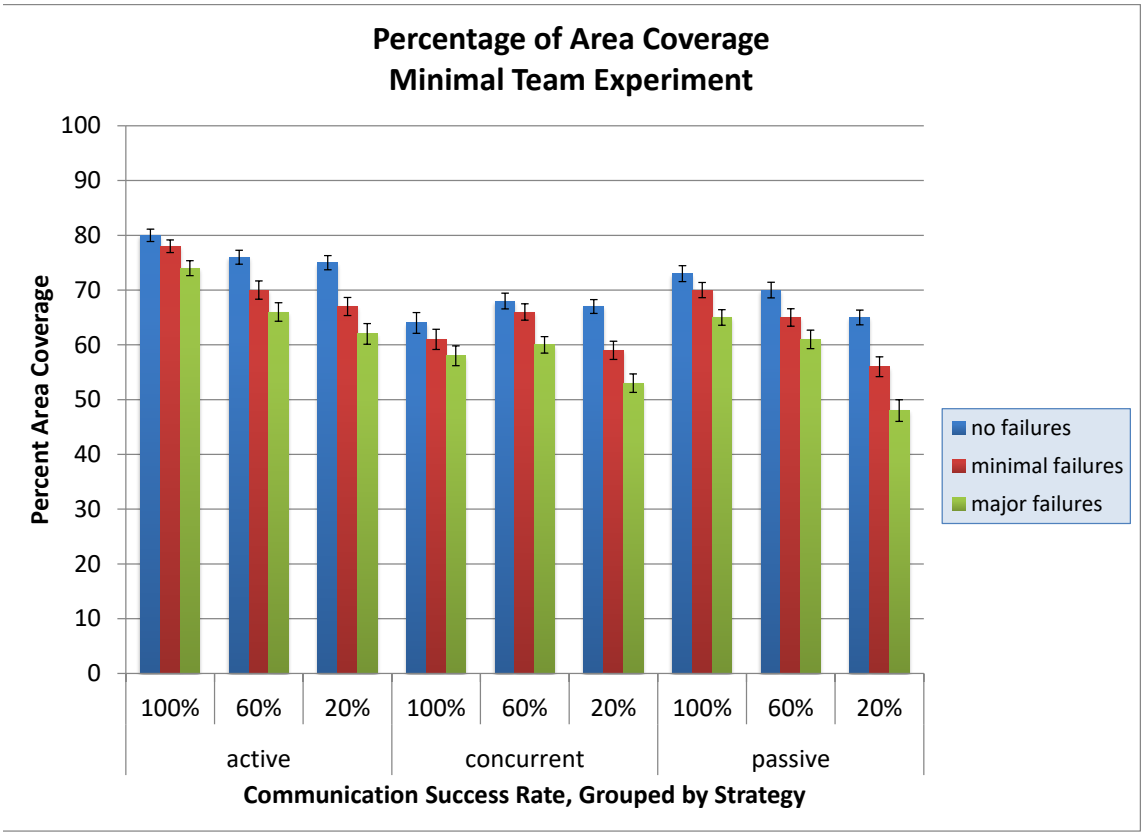

Figure 5.18: Percentage of area coverage, during the *Minimal Team* experiment.

is due to the greater level of physical exploration that takes place in active recruitment. More exploration provides opportunities to cover more of the environment, discover more victims, and encounter other robots with useful information regarding the status of victims. Area coverage was also highest in active settings due to the physical searches that active recruitment supports.

As in my main experiments (Section 5.8.2), concurrent recruitment performed slightly poorer than passive recruitment in terms of area coverage, except in conditions of severe communication failure. A similar phenomenon can be seen when examining the percentage of victims identified: concurrent recruitment performs more

poorly than passive recruitment, except in conditions of greater robot failure and communication unreliability (i.e., the types of conditions typical to domains such as USAR). As with area coverage, this is likely due to some duplication of effort that occurs in terms of area exploration and victim confirmation tasks (Section 4.3.2.1). As conditions degrade, however, the effectiveness of passive recruitment declines and is surpassed by concurrent recruitment.

There was not a large difference in the percentage of victims found in concurrent settings until communication success rates fell to 20%. This suggests that the ability to assign tasks without a team leader provides redundancy and prevents robot performance from degrading.

Robot failures had a stronger effect on these results than in my main experiment (Section 5.8). This shows, intuitively, that the performance of small robot teams can degrade significantly in dangerous domains when losses occur and replacements are not available.

Interestingly, the amount of area coverage in this experiment is only slightly less than that of my main experiment where replacement robots are not available (see Figure 5.4). This suggests that a relatively small team of robots is sufficient to cover a significant portion of these evironments, regardless of the recruitment strategy, and larger teams (or the availability of replacement robots) provide few improvements beyond this. As mentioned previously (Section 5.8.2), it is possible that my experimental environments (Appendix A) contain hard-to-reach areas that make it difficult for my robots to achieve full coverage, and the presence of larger teams or replacements only helps to provide slightly more coverage in those areas. It is also possible that the

remaining area is explored by isolated individuals or not communicated to a team leader. (A map of area coverage over my experimental trials would have provided useful insight, but these statistics were not captured by my simulator.) However, active recruitment still outperformed the other two strategies in this regard due to the physical searches it involves.

5.11 Analysis

Now that I have described the results of my experiments, I will now make some general observations of my methodology and identify areas of strength as well as certain areas that could use improvement.

My framework clearly shows that active and concurrent recruitment strategies provide significant performance gains in many cases. The ability to assign tasks to other robots without the involvement of a team leader removes a layer of indirection from the task assignment process, and enables robots to pass tasks from one unit to another until a successful recruitment takes place (Section 3.9.1.2). Role-level recruitment also provides significant benefits because it allows robots to more aggressively search for desired team members, who likely contain new information that can be useful to the team (such as the locations of victims). This spread of information also increases redundancy across multiple robots, meaning that the failure of a particular unit does not result in as great a loss of information.

My recruitment strategies also demonstrate that they are much more capable of handling situations where communication success rates are extremely low, or when robot failures are likely to occur. Passive recruitment strategies rely heavily on chance

encounters with other robots and rely on team leaders to perform task assignments, and it is clear that this strategy does not perform well under challenging conditions (also noted by Gunn [2011]). The overall high performance of my recruitment strategies indicates that more decentralized decision making is useful when operating conditions are harsh.

Similarly, active and concurrent recruitment often result in greater performance improvements when replacement robots are not available. These gains are lessened when replacement robots are present, indicating that with enough robots in ideal conditions, recruitment strategies provide little benefit and can even be detrimental to team performance (as noted in Section 5.8.2). However, it is unrealistic to expect such ideal conditions when operating in hazardous environments.

A weakness of concurrent recruitment is that it can result in duplication of effort. This is advantageous when operating conditions are harsh, but can be detrimental to robot team performance under ideal conditions due to the fact that, without the intervention of a team leader (who possesses a larger view of the work that needs to be performed), robots are able to assign tasks in a slightly overlapping fashion. For example, two different robots independently assigning exploration tasks might result in an area unknowingly being explored more than once.

A weakness of active recruitment is that it takes robot efforts away from immediately useful work. This is not problematic for types of work that are easily completed, either because there are many such tasks available, or because many robots are available that are capable of performing that particular type of task. However, task types which occur infrequently tend to become neglected as robots spend significantly more

time recruiting than executing these types of tasks. Work such as exploration will still occur as a side-effect of active searches and likewise, new tasks might be discovered as well. This suggests that robots capable of performing important tasks which occur less frequently should use concurrent recruitment to avoid spending too much time recruiting actively instead of executing important tasks. Therefore, the decision to use concurrent versus active recruitment depends on the level of task specialization, the number of robots available to complete the task, the importance of the task, and the overall likelihood of such a task being created.

A related weakness of active recruitment is that since it involves physical searching, robots actively recruiting for a specialized task at a particular location are more likely to recruit the required robot further away from the physical location of the task, since active searching (which takes a robot away from its original location) is more likely to result in successful recruitment. This might make it more difficult for a recruited robot to reach the original task location—it may be intercepted, or become stuck or lost along the way. As seen in my main experiment (Section 5.8.3.2) and in my *Additional DebrisBots* experiment (Section 5.9), it would be more useful to perform less aggressive searches in these cases, and future research will need to determine where this balance lies.

5.12 Conclusion

In this chapter, I have provided an overview of my research questions and the approach used to answer them. I described my evaluation metrics and how I generated my experiments, as well as my experiment breakdown and the independent variables

in my study. Following this, I presented and discussed my results. Finally, I concluded with a discussion of my methodology's strengths and weaknesses.

Chapter 6

Conclusion and Future Work

6.1 Introduction

I begin this chapter by reviewing my research questions (Section 5.2) and discussing how my experimental results from Chapter 5 answer them. I provide an outline of the main contributions of my thesis in Section 6.3, and discuss directions for future work in Section 6.4.

6.2 Answers to Research Questions

My research questions were originally presented in Section 1.6. In this section, I review these questions and discuss how my results answer them.

1. **To what degree does the use of various recruitment strategies improve or hamper the overall performance of teams of heterogeneous robots in dangerous domains?**

My results show that active and concurrent recruitment strategies provide significant performance improvements over passive recruitment approaches. This is particularly true when operating under the challenging conditions that are expected of dangerous environments. It appears that my recruitment strategies become increasingly useful as the environment becomes more dangerous. This is particularly true of active recruitment, which often provided the greatest gains in these conditions. In situations where conditions are highly favourable (e.g., plenty of robots available, with few failures, and reliable communication) recruitment provides little benefit and can even hamper team performance.

2. **What factors (e.g., availability of replacement robots, wireless reliability, probability of robot failure) determine the recruitment strategy that should be used in a given situation?**

   As environmental conditions become less favourable, recruitment strategies become increasingly useful. The availability of replacement robots, wireless reliability, and robot failure probabilities all contribute to this: unfavourable values of any of these factors should be taken as an indication that more active recruitment strategies are necessary for most tasks. There are exceptions to this, however: for example, guided debris removal was shown to be less effective in active recruitment settings (Section 5.8.3.2). An additional factor to consider is the existence of highly-specialized work that does not need to occur often. In active recruitment settings, highly-specialized robots often spent a significant amount of time performing recruitment at the cost of performing lesser amounts of highly-specialized work. When deciding on a recruitment strategy, therefore,

the importance and frequency of such work, as well as the availability of robots able to complete such work, must be taken into account.

3. **Which recruitment strategies, if any, result in overall best performance of teams of heterogeneous robots if wireless reliability levels are not known?**

   Active and concurrent recruitment outperformed passive recruitment in several cases, particularly when communication reliability was low. This makes these two strategies ideal choices if wireless reliability levels are not known. As noted above, the choice to use either active or concurrent recruitment also depends on the number and availability of robots that are able to perform highly-specialized work, the importance of the work to the team's overall mission, and how often that particular type of work is required (one particular example is guided debris removal, Section 5.8.3.2, which received the least benefits from active recruitment).

6.3 Contributions

My thesis research provides a number of useful contributions, including:

1. A methodology for dynamic team management that provides increased performance for teams of heterogeneous robots in dangerous environments, particularly when environmental conditions become highly unfavourable. This includes techniques for more actively acquiring new team members and promoting the spread of useful information, as well techniques for assigning tasks among other

robots on a team without the direct involvement of a team leader.

2. The recruitment spectrum: a useful definition of the types of recruitment strategies and their proven effects on team performance in dangerous settings.

3. Insight into the usefulness of various recruitment strategies in varying environmental conditions.

4. An example implementation of my methodology, demonstrating the effectiveness of my framework in simulated disaster environments in a repeatable manner.

5. An experimental domain that is useful for other future work.

6.4 Future Work

Although the evaluation of my framework clearly demonstrated significant benefits of my methodology, my work has raised several useful questions that further contribute to this field. There are numerous ways in which my research can be improved, and it encourages several avenues of future work. This includes improvements to my implementation, enhancements to my methodology, and possible directions for further research.

6.4.1 Implementation Improvements

First and foremost, a physical implementation of my work using real robots would yield additional insight into the effectiveness of my methodology. Given that my

recruitment strategies result in greater gains under more challenging conditions, the difficulty of operating in real-world environments with real robots may indicate even more utility than what is shown in simulations. Issues such as localization, sensor error, and the physical shifting of debris would be more complex in the real world. Additionally, robots would become stuck more often in a physical implementation, even if only in a minor way. Experiments with real robots along the lines of those done in my thesis are needed to see the effects that these real world conditions would have on my framework.

Conducting experiments in simulation is still important, however, because of the difficulty of maintaining a large number of real robots consistently over a large number of experimental trials. For this reason, it is also important for simulation realism to be improved. In this section, I describe various ways to improve my implementation.

A possible improvement to my implementation would be to enforce real-time constraints on robot processing. Currently, robots can perform arbitrarily complex computations in a single simulation time step; in a physical implementation, larger computations would take greater time to complete. My simulator could be made more realistic by imposing restrictions on how many computations a robot can complete in a single time step. My results are still conservative in this aspect, however, as the limitations placed on the MinBots (Section 4.3.1.2) are likely small compared to the type of affordable computational power that could be placed on an expendable robot platform (e.g., a cell phone).

Future work could also include making improvements to my simulator allowing robots to physically move small debris objects by bumping into them. This would

make debris removal more realistic than it is implemented in my work and, also realistically, may increase the chances of robots becoming stuck as debris is shifted. Although Stage does not currently support object dynamics [Vaughan, 2008], this could conceivably be implemented, or simulators offering more complex physics, such as ARGoS [Pinciroli et al., 2012] or Gazebo [Koenig and Howard, 2004] could be used. This would also enable other robots to shift debris somewhat, even if not properly equipped to do so. Porting my implementation for evaluation using another simulator in general (such as USARsim [Carpin et al., 2007], which was originally designed with USAR in mind) would also help to more rigorously evaluate my framework.

My results indicated that victim marker donation tasks were infrequent events due to the difficulty of coordinating in the environment (Section 5.8.4). As a result, many robots would not have been able to mark victims when required, due to a lack of victim markers. One possible solution to this problem is to allow robots the ability to pick up markers that are detected near false victims, since those markers must have been dropped by a robot lacking advanced victim-sensing capabilities. Another solution would be to introduce a *mark victim* task, where an assignee would be instructed to drop a victim marker at a particular location.

My simulation environment also deals with robot failures at a very coarse level: either a robot is operational or it is not. Realistically, individual robot hardware components fail often [Carlson and Murphy, 2003], and these failures will affect a robot's capabilities. My implementation could be extended to allow the failure of individual components, and robots would be required to detect these in order to determine if a particular task lies within their capabilities. As components fail (likely

permanently) as a mission progresses, robots would be required to request assistance via recruitment as their abilities degrade. Additionally, robots with failed components would be less likely to be tasked with important work, and fully-functional robots would be more likely to receive such work.

Since real robots always have to deal with the imprecision associated with physical sensors, it would be interesting to explore the impact of varying levels of sensor noise. Extensions to my simulator could be made that would allow for adjustable ranges of sensor error, in order to determine how this would affect robot performance in different recruitment configurations.

Lastly, it would be interesting to see information passing implemented as a task itself. Low-cost robots with large stores of memory could be tasked with acquiring and sharing useful information with the teams they encounter in the environment. This would help to spread out knowledge even more.

6.4.2 Directions for Future Research

My work has raised many questions about recruitment and team management, and this is a valuable contribution. In this section, I discuss broader avenues of future research that my work has revealed.

Some of my experimental results (Sections 5.8–5.10) indicate that not all types of tasks or robots are well-suited for active recruitment. As noted in Section 5.11, active recruitment can have a negative impact on the execution of highly-specialized tasks when few robots with the appropriate capabilities are available, when compared to other recruitment strategies. It would be interesting to allow robots to indivdually

learn the appropriate recruitment strategy to use during a mission, based on their knowledge of (a) their own importance and the importance of the tasks they can execute, and (b) the number of similarly-equipped units present in the environment.

Recall from Section 3.9 that task-level and role-level recruitment are always enabled together in my work, even though they are considered separate mechanisms. A possible direction for future work might involve exploring the usefulness of these two techniques separately. Depending on environmental conditions or the requirements of the tasks at hand, for example, it may be beneficial to enable only one of those techniques when more active recruitment strategies are being used. Future work should explore this possibility further.

My results indicate that recruitment becomes increasingly useful as environmental conditions become less favourable or if small teams of robots are employed. In ideal conditions with many robots, however, recruitment can become detrimental to performance. It would be very interesting to explore this balance further and determine which factors (e.g., number and type of robots, communication success levels, and chances of robot failure) most strongly contribute to this. Another factor to consider is the risk associated with searching: performing physical searches might result in a robot becoming lost, or might result in wasted time if a useful robot cannot be found. There is a trade-off between performing searches and executing immediately useful work, and future research should endeavour to explore this balance further. Additionally, my work assumes that an environment will be uniformly hazardous. In reality, certain areas of a disaster site may be safer (or more dangerous) than others. In these scenarios, robots could assess the conditions of the area in which they are

operating, in order to determine an ideal team management strategy.

Marker donation (Section 4.3.2.2) is unique in my work in that it is the only type of task that requires explicit coordination between two robots. Robots working on tasks together is an important problem and is worth exploring further, particularly in challenging environments. The results I have gathered for marker donation successes (Section 5.8.4) indicate that operating conditions such as poor communication and robot failures often increase the difficulty of joint tasks to the degree that they become impossible to successfully complete. Realistically, many tasks in hazardous environments might require coordination between multiple robots (e.g., safely moving a victim, putting out large fires, etc.). Coordination becomes increasingly difficult as the number of robots increases, and hazardous operating conditions create further challenges. While I have shown (Section 5.8.4) that recruitment can be beneficial in these situations, my implementation tests a very limited range of possible scenarios and coordination tasks. One direction for future work might involve the development of more complex recruitment mechanisms, where robots would be able to search for and "build up" a group of differently-equipped robots in order to complete a larger task together in a hazardous setting. The possibility of robot and communication failures in these environments makes for a much richer set of problems that would help to further advance knowledge of recruitment and its utility in such settings.

As mentioned previously (Section 3.2), my framework supports subtask division to a limited degree in the form of victim marker donation (Section 4.3.2.2), and this division of subtasks is fixed between two robots. This restriction is similar to that in the work of Kiener and Von Stryk [2007]. It would be useful to investigate dynamic

task refactoring, where subtask assignments could change based on perceived robot or communication failures in order to improve robot performance when executing joint tasks in hazardous environments.

It would be interesting to investigate methods of recruitment that do not involve direct communication. While my implementation assumes that robots can only communicate via radio, robots could also be made to harness *stigmergy* [Beckers et al., 1994] to guide the completion of work. Robots could leave behind instructions for other robots using elements taken from the environment and arranged in such a way that they convey a meaningful message. A particular configuration of bricks, for exmple, could indicate a dangerous area or a victim nearby that still needs to be identified.

My framework operates under the assumption that all robots belong to a particular team (even if only a team of one), and that robots should fill roles on those teams so as to match the ideal team definition as closely as possible. It does not consider the possibility that certain robots may be more useful globally if they do not join a team. A rare, highly-specialized robot capable of carrying other robots over debris, for example, would certainly be useful to the team which it belongs to, but may not be sufficiently accessible to other teams in the environment. In these situations, it may be more useful for such robots to refuse to join any team whatsoever, so that all teams can potentially benefit equally from its unique capabilities. Furthermore, this behaviour could be learned over time. For example, the number of times a robot has been recruited by members of different teams can be used as an indicator of whether or not the robot should avoid belonging to a team altogether.

A limitation of my work is the duplication of effort in terms of exploration and victim confirmation in concurrent recruitment settings, when communication success rates are high and robot failures do not occur. This is due to the fact that the intervention of a team leader (who has a greater perspective of outstanding work) is not required in concurrent recruitment settings—as a result, tasks can be assigned redundantly across multiple robots, resulting in wasted effort. Future work should explore effective methods for performing frontier exploration when such tasks can originate from multiple robots with varying perspectives.

Lastly, another limitation of my work is that it only involves ground-based (wheeled or tracked-drive) robots. The demands of robotic USAR suggest that other types of locomotion (wall-climbing or flying) may provide significant advantages and would add further richness to the study of robotic USAR. Wall-climbing robots would be particularly useful since they can avoid ground-based debris and access high-reaching vantage points to survey an area (e.g., for victims). Flying robots (e.g., micro-quadcopters) provide similar benefits and do not need to climb over obstacles at all. Because of their abilities to observe wide areas from higher above, flying or climbing robots would be able to provide navigational information, direct teams efforts, and act as beacons or waypoints for ground-based robots.

6.5 Conclusion

The development of useful autonomous robots is a challenging endeavour that is made even more complicated by the extreme conditions under which they are often expected to operate. My research has demonstrated that active recruitment tech-

niques provide significant performance gains for robots operating in these environments, compared to passive recruitment techniques. My work provides insight into the types of strategies that should be embraced when autonomous robots are used for dangerous work in the future. I have also presented interesting avenues of future work that would help to further enhance the methodology I have developed.

Appendix A

Experimental Environments

Figures A.1, A.2, and A.3 show the three environments in which I conducted my experiments (Chapter 5).

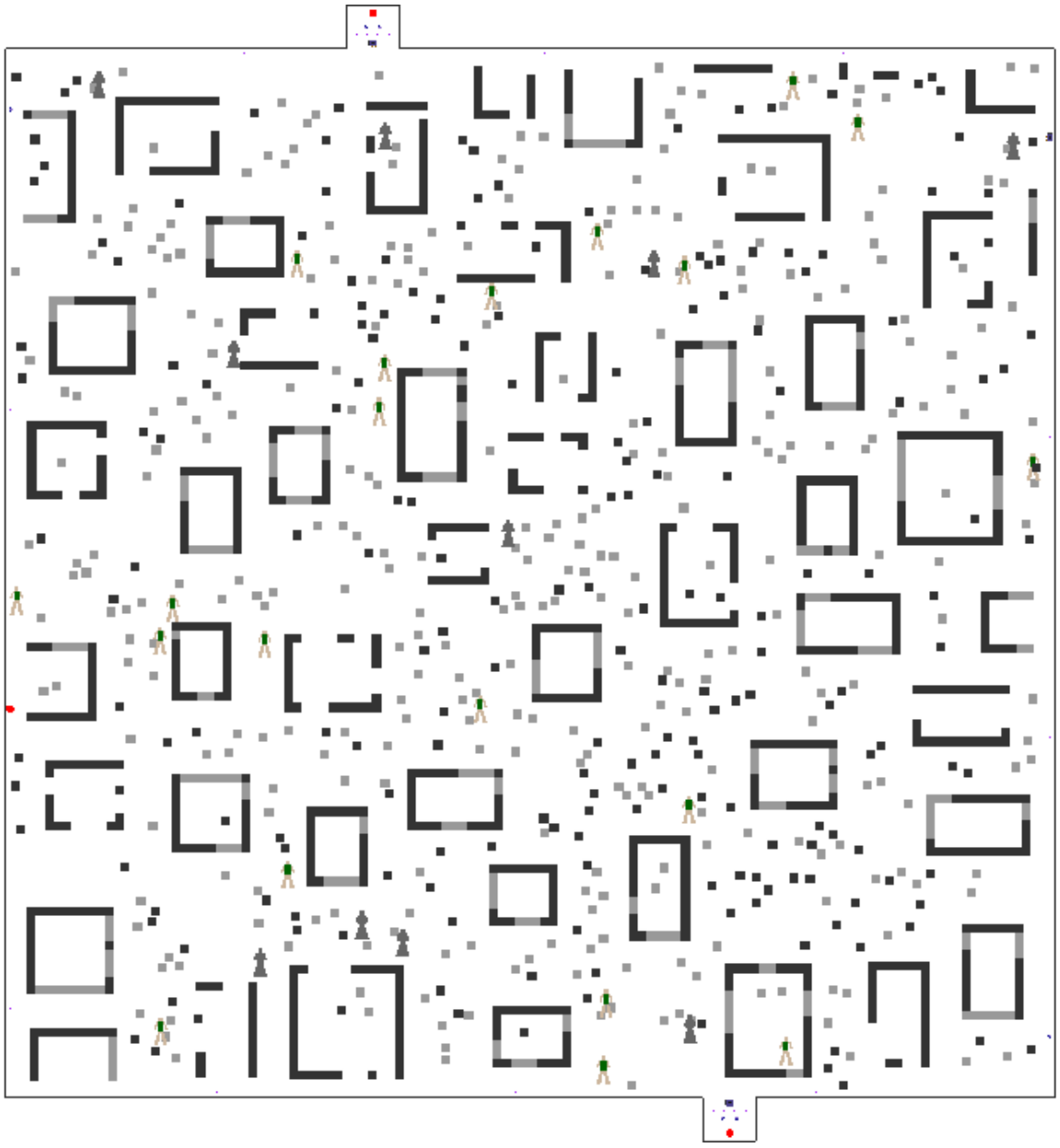

Figure A.1: Experimental environment 1.

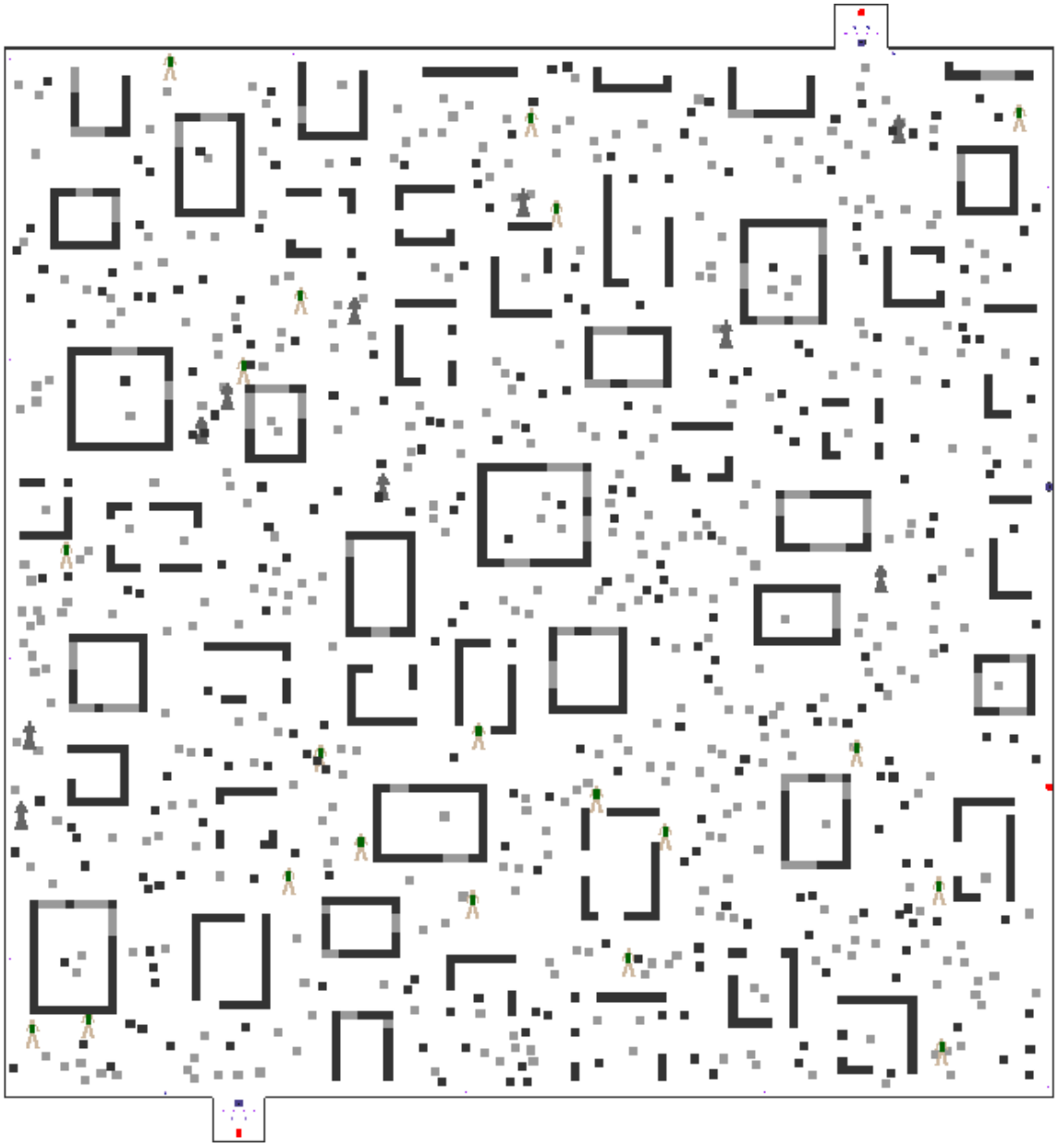

Figure A.2: Experimental environment 2.

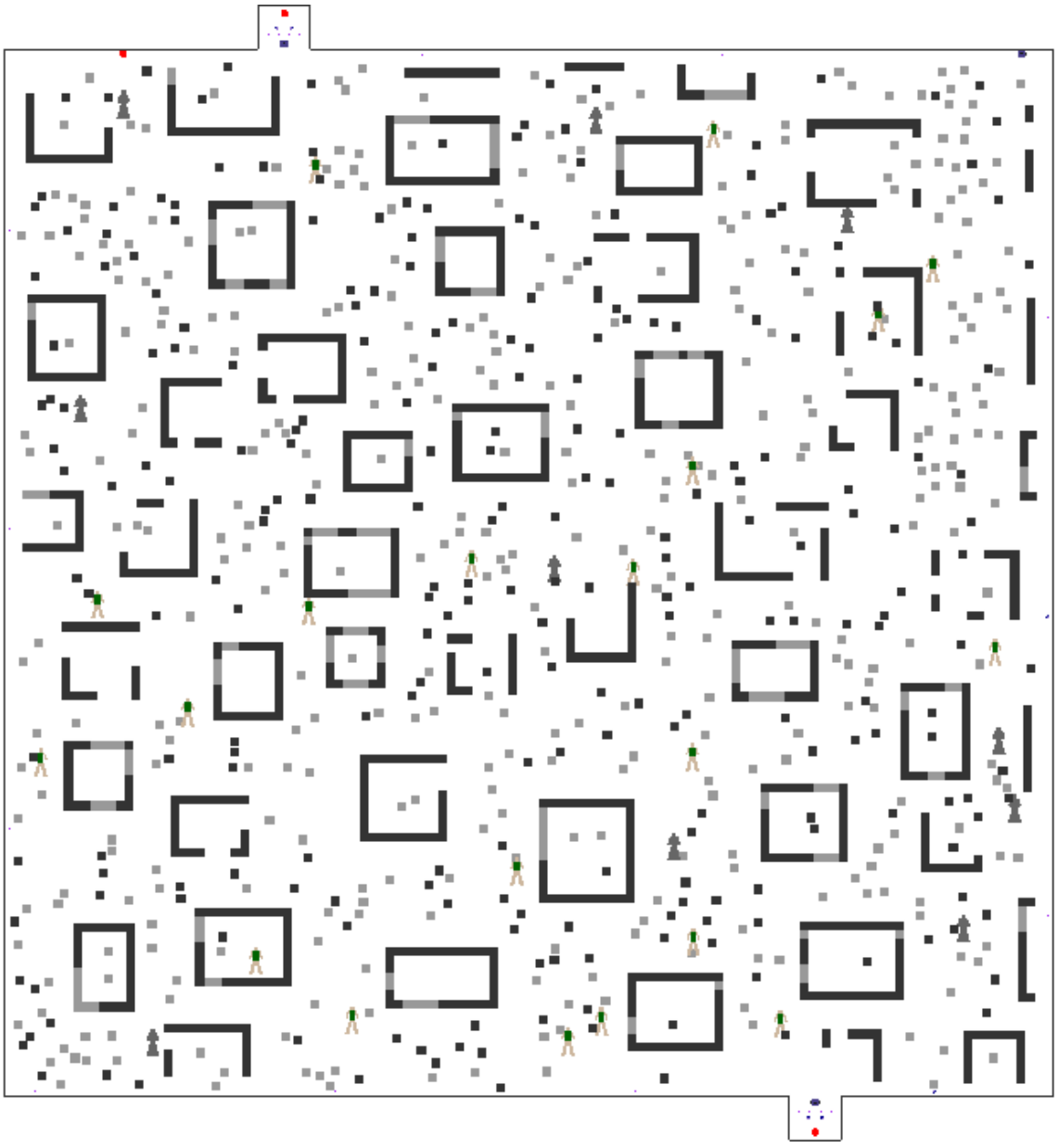

Figure A.3: Experimental environment 3.

Bibliography

R. Arkin. Motor schema based navigation for a mobile robot: An approach to programming by behavior. In *Proceedings of the 1987 IEEE International Conference on Robotics and Automation (ICRA 1987)*, volume 4, pages 264–271. IEEE, 1987.

Atmel Corporation. Atmega48a/pa/88a/pa/168a/pa/328/p datasheet. `http://www.atmel.com/devices/atmega328p.aspx`, 2015.

M. Baker, R. Casey, B. Keyes, and H. A. Yanco. Improved interfaces for human-robot interaction in urban search and rescue. In *IEEE Transactions on Systems, Man, and Cybernetics (SMC 2004)*, volume 3, pages 2960–2965. IEEE, 2004.

T. Balch and R. C. Arkin. Behavior-based formation control for multirobot teams. *IEEE Transactions on Robotics and Automation*, 14(6):926–939, 1998.

R. Beckers, O. Holland, and J.-L. Deneubourg. From local actions to global tasks: Stigmergy and collective robotics. In *Artificial Life IV*, volume 181, page 189, 1994.

R. A. Brooks. A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation*, 2(1):14–23, 1986.

J. Carlson and R. R. Murphy. Reliability analysis of mobile robots. In *Proceedings*

of the 2003 IEEE International Conference on Robotics and Automation (ICRA 2003), volume 1, pages 274–281. IEEE, 2003.

J. Carlson and R. R. Murphy. How UGVs physically fail in the field. *IEEE Transactions on Robotics*, 21(3):423–437, 2005.

D. Carnegie. A three-tier hierarchical robotic system for urban search and rescue applications. In *Proceedings of the 2007 IEEE International Workshop on Safety, Security and Rescue Robotics (SSRR 2007)*, pages 1–6. IEEE, 2007.

S. Carpin, M. Lewis, J. Wang, S. Balakirsky, and C. Scrapper. Usarsim: a robot simulator for research and education. In *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA 2007)*, pages 1400–1405. IEEE, 2007.

J. Casper. Human-robot interactions during the robot-assisted urban search and rescue response at the world trade center. Master's thesis, University of South Florida, 2002.

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 (SMC 2003)*, 33(3):367–385, 2003.

J. L. Casper and R. R. Murphy. Workflow study on human-robot interaction in USAR. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation (ICRA 2002)*, volume 2, pages 1997–2003. IEEE, 2002.

A. Comi, L. Fotia, and D. Rosaci. Improving the social capital of trust-based competitive multi-agent systems by introducing meritocracy. In *Proceedings of the 2014 Federated Conference on Computer Science and Information Systems (FedCSIS 2014)*, pages 1461–1466. IEEE, 2014.

E. D. Costa, P. M. Shiroma, and M. F. Campos. Cooperative robotic exploration and transport of unknown objects. In *Proceedings of the 2012 Robotics Symposium and Latin American Robotics Symposium (SBR-LARS 2012)*, pages 56–61. IEEE, 2012.

J. W. Crandall and M. A. Goodrich. Characterizing efficiency of human robot interaction: A case study of shared-control teleoperation. In *Proceedings of the 2002 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2002)*, volume 2, pages 1290–1295. IEEE, 2002.

P. Dasgupta and K. Cheng. Dynamic multi-robot team reconfiguration using weighted voting games. *Journal of Experimental & Theoretical Artificial Intelligence*, pages 1–22, 2015.

M. B. Dias, T. K. Harris, B. Browning, E. G. Jones, B. Argall, M. M. Veloso, A. Stentz, and A. I. Rudnicky. Dynamically formed human-robot teams performing coordinated tasks. In *AAAI Spring Symposium: To Boldly Go Where No Human-Robot Team Has Gone Before*, pages 30–38, 2006.

F. Dos Sos and A. L. Bazzan. Towards efficient multiagent task allocation in the robocup rescue: A biologically-inspired approach. *Autonomous Agents and Multi-Agent Systems*, 22(3):465–486, 2011.

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

A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989.

E. Elkind, L. A. Goldberg, P. Goldberg, and M. Wooldridge. Computational complexity of weighted threshold games. In *Proceedings of the 22nd National Conference on Artificial Intelligence (AAAI 2007)*, volume 22, page 718. Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999, 2007.

A. Ferworn, A. Sadeghian, K. Barnum, H. Rahnama, H. Pham, C. Erickson, D. Ostrom, and L. Dell'Agnese. Urban search and rescue with canine augmentation technology. In *Proceedings of the 2006 IEEE/SMC International Conference on System of Systems Engineering*, pages 5–pp. IEEE, 2006.

J. Fredslund and M. J. Mataric. A general algorithm for robot formations using local sensing and minimal communication. *IEEE Transactions on Robotics and Automation*, 18(5):837–846, 2002.

A. Gage and R. R. Murphy. Affective recruitment of distributed heterogeneous agents. In *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI 2004)*, pages 14–19, 2004.

S. Garnier, F. Tâche, M. Combe, A. Grimal, and G. Theraulaz. Alice in pheromone land: An experimental setup for the study of ant-like robots. In *Proceedings of*

the 2007 IEEE Symposium on Swarm Intelligence Symposium (SIS 2007), pages 37–44. IEEE, 2007.

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 (ICAR 2003)*, volume 1, pages 317–323, 2003.

T. Gunn. Dynamic heterogeneous team formation for robotic urban search and rescue. Master's thesis, Department of Computer Science, University of Manitoba, Winnipeg, Canada, December 2011.

T. Gunn and J. Anderson. Effective task allocation for evolving multi-robot teams in dangerous environments. In *Proceedings of The 2013 IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2013)*. IEEE Computer Society Press, 2013.

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.

B. Hölldobler, R. C. Stanton, and H. Markl. Recruitment and food-retrieving behavior in novomessor (formicidae, hymenoptera). *Behavioral Ecology and Sociobiology*, 4 (2):163–181, 1978.

A. Howard, L. E. Parker, and G. S. Sukhatme. Experiments with a large heterogeneous mobile robot team: Exploration, mapping, deployment and detection. *The International Journal of Robotics Research*, 25(5-6):431–447, 2006.

Y. Hu, L. Wang, J. Liang, and T. Wang. Cooperative box-pushing with multiple autonomous robotic fish in underwater environment. *Control Theory & Applications, IET*, 5(17):2015–2022, 2011.

M. Johnson, K. Intlekofer Jr, H. Jung, J. M. Bradshaw, J. Allen, N. Suri, and M. Carvalho. Coordinated operations in mixed teams of humans and robots. In *Proceedings of the 1st IEEE Conference on Distributed Human-Machine Systems (DHMS 2008)*, 2008.

M. W. Kadous, R. K.-M. Sheh, and C. Sammut. Effective user interface design for rescue robotics. In *Proceedings of the 1st ACM SIGCHI/SIGART Conference on Human-Robot Interaction*, pages 250–257. ACM, 2006.

J. Kiener and O. Von Stryk. Cooperation of heterogeneous, autonomous robots: A case study of humanoid and wheeled robots. In *Proceedings of the 2007 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2007)*, pages 959–964. IEEE, 2007.

H. Kitano and S. Tadokoro. Robocup rescue: A grand challenge for multiagent and intelligent systems. *AI magazine*, 22(1):39, 2001.

N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, volume 3, pages 2149–2154. IEEE, 2004.

M. Koes, I. Nourbakhsh, K. Sycara, M. Koes, K. Sycara, I. Nourbakhsh, M. Koes,

I. Nourbakhsh, K. Sycara, S. D. Ramchurn, et al. Heterogeneous multirobot coordination with spatial and temporal constraints. In *Proceedings of the 20th National Conference on Artificial Intelligence (AAAI 2005)*, pages 1292–1297, 2005.

M. J. Krieger and J.-B. Billeter. The call of duty: Self-organised task allocation in a population of up to twelve mobile robots. *Robotics and Autonomous Systems*, 30 (1):65–84, 2000.

A. Lein and R. T. Vaughan. Adapting to non-uniform resource distributions in robotic swarm foraging through work-site relocation. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2009)*, pages 601–606. IEEE, 2009.

M. Long, A. Gage, R. Murphy, and K. Valavanis. Application of the distributed field robot architecture to a simulated demining task. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA 2005)*, pages 3193–3200. IEEE, 2005.

A. Mahmood and Y. Kim. Leader-following formation and heading control of networked quadcopters. In *Proceedings of the 14th International Conference on Control, Automation and Systems (ICCAS 2014)*, pages 919–921. IEEE, 2014.

M. J. Matarić. Reinforcement learning in the multi-robot domain. In *Robot colonies*, pages 73–83. Springer, 1997.

M. J. Matarić, M. Nilsson, and K. T. Simsarin. Cooperative multi-robot box-pushing.

In *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 1995)*, volume 3, pages 556–561. IEEE, 1995.

N. Mathews, A. L. Christensen, R. O'Grady, P. Rétornaz, M. Bonani, F. Mondada, and M. Dorigo. Enhanced directional self-assembly based on active recruitment and guidance. In *Proceedings of the 2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2011)*, pages 4762–4769. IEEE, 2011.

D. E. McGovern. Experience and results in teleoperation of land vehicles. In *Pictorial Communication in Virtual and Real Environments*, pages 182–195. Taylor & Francis, Inc., 1991.

R. Murphy, J. Casper, J. Hyams, M. Micire, and B. Minten. Mobility and sensing demands in USAR. In *Proceedings of the 26th Annual Conference of the IEEE Industrial Electronics Society (IECON 2000)*, volume 1, pages 138–142. IEEE, 2000.

L. E. Parker. ALLIANCE: An architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.

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

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

L. Pitonakova, R. Crowder, and S. Bullock. Understanding the role of recruitment in collective robot foraging. 2014. URL `http://eprints.soton.ac.uk/364829/`.

S. K. Robson and J. F. Traniello. Resource assessment, recruitment behavior, and organization of cooperative prey retrieval in the ant formica schaufussi (hymenoptera: Formicidae). *Journal of Insect Behavior*, 11(1):1–22, 1998.

P. Scerri, A. Farinelli, S. Okamoto, and M. Tambe. Allocating tasks in extreme teams. In *Proceedings of the 4th International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 727–734. ACM, 2005.

S. Sen and P. S. Dutta. The evolution and stability of cooperative traits. In *Proceedings of the 1st International Joint Conference on Autonomous Agents and Multiagent Systems: Part 3*, pages 1114–1120. ACM, 2002.

D. A. Shell and M. J. Mataric. On foraging strategies for large-scale multi-robot systems. In *Proceedings of the 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2006)*, pages 2717–2723. IEEE, 2006.

T. B. Sheridan. *Telerobotics, Automation, and Human Supervisory Control*. MIT press, 1992.

Z. Song and R. T. Vaughan. Sustainable robot foraging: Adaptive fine-grained multi-robot task allocation for maximum sustainable yield of biological resources. In *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2013)*, pages 3309–3316. IEEE, 2013.

Z. Song, S. A. Sadat, and R. T. Vaughan. MO-LOST: Adaptive ant trail untangling in

multi-objective multi-colony robot foraging. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems*, volume 3, pages 1199–1200. International Foundation for Autonomous Agents and Multiagent Systems, 2012.

M. Statheropoulos, A. Agapiou, G. Pallis, K. Mikedi, S. Karma, J. Vamvakari, M. Dandoulaki, F. Andritsos, and C. P. Thomas. Factors that affect rescue time in urban search and rescue (USAR) operations. *Natural Hazards*, 75(1):57–69, 2015.

V. Tereshko and A. Loengarov. Collective decision making in honey-bee foraging dynamics. *Computing and Information Systems*, 9(3):1, 2005.

S. Tunwannarux, A. Tunwannarux, M. Demiralp, W. Mikhael, A. Caballero, N. Abatzoglou, M. Tabrizi, R. Leandre, M. Garcia-Planas, and R. Choras. The CEO mission IV rescue robot for world robocup and USAR field. In *Proceedings of the 2008 WSEAS International Conference on Mathematics and Computers in Science and Engineering*, number 10. World Scientific and Engineering Academy and Society, 2008.

M. Van De Vijsel and J. Anderson. Coalition formation in multi-agent systems under real-world conditions. In *Proceedings of Association for the Advancement of Artificial Intelligence (AAAI 2004)*, 2004.

M. van de Vijsel and J. Anderson. Increasing realism in coalition formation. In *Proceedings of the 3rd International Conference on Computational Intelligence, Robotics, and Autonomous Systems (CIRAS 2005)*, Singapore, December 2005.

G. Vásárhelyi, C. Virágh, G. Somorjai, N. Tarcai, T. Szorenyi, T. Nepusz, and T. Vicsek. Outdoor flocking and formation flight with autonomous aerial robots. In *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)*, pages 3866–3873. IEEE, 2014.

R. Vaughan. Massively multi-robot simulation in stage. *Swarm Intelligence*, 2(2-4): 189–208, 2008.

R. Wegner. Balancing robotic teleoperation and autonomy in a complex and dynamic environment. Master's thesis, Department of Computer Science, University of Manitoba, July 2003.

J. Wong, C. Robinson, et al. Urban search and rescue technology needs: Identification of needs. *Federal Emergency Management Agency (FEMA) and the National Institute of Justice (NIJ)*, (207771), 2004.

B. Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings of the 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation (CIRA 1997)*, pages 146–151. IEEE, 1997.