Heterogeneous Multi-Robot Cooperation

by

Lynne E. Parker
email: ParkerL@ai.mit.edu

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1994

Heterogeneous Multi-Robot Cooperation

by

Lynne E. Parker

Submitted to the Department of Electrical Engineering and Computer Science
on January 7, 1994, in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

Abstract

This report addresses the problem of achieving fault tolerant cooperation within small- to medium-sized teams of heterogeneous mobile robots. I describe a software architecture I have developed, called ALLIANCE, that facilitates robust, fault tolerant cooperative control, and examine numerous issues in cooperative team design. ALLIANCE is a fully distributed architecture that utilizes adaptive action selection to achieve cooperative control in robot missions involving loosely coupled, largely independent tasks. The robots in this architecture possess a variety of high-level functions that they can perform during a mission, and must at all times select an appropriate action based on the requirements of the mission, the activities of other robots, the current environmental conditions, and their own internal states. Since such cooperative teams often work in dynamic and unpredictable environments, the software architecture allows the team members to respond robustly and reliably to unexpected environmental changes and modifications in the robot team that may occur due to mechanical failure, the learning of new skills, or the addition or removal of robots from the team by human intervention. In addition, an extended version of ALLIANCE, called L-ALLIANCE, incorporates a simple mechanism that allows teams of mobile robots to learn from their previous experiences with other robots, allowing them to select their own actions more efficiently on subsequent trials when working with "familiar" robots on missions composed of independent tasks. This mechanism allows a human system designer to easily and quickly group together the appropriate combination of robots for a particular mission, since robots need not have *a priori* knowledge of their teammates.

The development of ALLIANCE and L-ALLIANCE involved research on a number of topics: fault tolerant cooperative control, adaptive action selection, distributed control, robot awareness of team member actions, improving efficiency through learning, inter-robot communication, action recognition, and local versus global control. This report describes each of these topics in detail, along with experimental results of investigating these issues both in simulated and in physical mobile robot teams.

I am not aware of any other cooperative control architecture that has exhibited the combination of fault tolerance, reliability, adaptivity, and efficiency possible with ALLIANCE and L-ALLIANCE, and which has been successfully demonstrated on physical mobile robot teams.

Thesis Supervisor: Rodney A. Brooks
Title: Professor of Computer Science

Acknowledgements

I express my heartfelt thanks to my thesis advisor, Rod Brooks, who supported and encouraged me throughout my time at MIT. He provided a good mixture of guidance and non-interference, allowing me to explore and develop my own ideas while offering helpful suggestions when they were most needed. I learned much from him about developing ideas and making them work. I also sincerely thank Lynn Stein, who served as one of my thesis readers, for her many helpful suggestions and critiques of my thesis work. Her advice improved my thesis in many areas. Thanks also to Patrick Winston, my third thesis reader, not only for his encouragement during my stay at MIT, but also for his guidance of the AI Lab, which has been a great source of inspiration to me.

I have very much enjoyed being a part of the Mobot Lab. In particular, I have enjoyed discussing various cooperative robotics topics with Maja Mataric, and to commiserate with her when our robots refused to behave. Anita Flynn has been a constant source of encouragement and inspiration, which I greatly appreciate. I offer a special thanks to Ian Horswill for his willingness to offer hacking assistance when needed, to Matt Marjonovic for his efforts on the robots' radio communication system, and to Mike Connell for his efforts in building the R-2s. Three visiting scientists to the Mobot Lab from Japan also provided much help and friendship: Masa Konishi, from the Mazda Motor Corporation, who worked with me on the Seymour robot during my early days at MIT, Yasuo Kagawa, also from Mazda, who built the first version of the multi-robot simulator that I used, and Macky Yamamoto, from Matsushita Research Institute, who was very helpful in debugging the R-2's. I thank all three for their assistance.

While at MIT, I have enjoyed the company of many other AI Lab members, including Joanna Bryson, Tina Kapur, Tanveer Mahmood, Nancy Pollard, Sajit Rao, Aparna Ratan, Jose Robles, Karen Sarachik, Pawan Sinha, Mark Torrance, and Holly Yanco. Many of these people proofed papers, critiqued practice talks, or offered other valuable advice, for which I am grateful. Thanks also to Eric Grimson, who was extremely supportive and encouraging as my academic counselor.

I appreciate the help, patience, and friendliness of IS Robotics' personnel Chuck Rosenberg, Helen Griner and Peter Ning in providing me with working and communicating R-2's and Genghis-II. Although I was not thrilled with the time delays in receiving functional robots, the delays were mostly caused by circumstances out of their control, and were certainly not from lack of effort on their part. Their services saved me from months (years?) of tedious robot-building on my own, and for that I am truly grateful.

While living in the Boston area, my husband and I have been fortunate to meet a number of fellow "transplants from the South," — in particular, Alice and Jeff Hill, Pam and Terry Wright, and Carol and Jon Long — who provided a much-needed escape from the pressures of MIT, and whose companionship we have greatly enjoyed. Although we four couples are scattering across the country now, we have become lifelong friends.

Finally, I express my gratitude for my family — my parents, sister, brother, and grandparents — who believed in me, offered guidance, and supported my decisions no matter what. Most of all, I offer my deepest thanks to my husband Bob for agreeing to move to Boston so that I could pursue this degree, and for supporting me and encouraging me all along the way. He brings immense happiness to my life with his never ending love and devotion, and thus I dedicate this dissertation to him.

Support for this research was provided in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contracts N00014-91-J-4038 and N00014-85-K-0124, in part by the California Institute of Technology Jet Propulsion Laboratory under contract 959333, and in part by Matsushita Research.

To my husband Bob,
with love and gratitude

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A key driving force in the development of mobile robotic systems is their potential for reducing the need for human presence in dangerous applications, such as the cleanup of toxic waste, nuclear power plant decommissioning, extra-planetary exploration, search and rescue missions, and security, surveillance, or reconnaissance tasks; or in repetitive types of tasks, such as automated manufacturing or industrial/household maintenance. The nature of many of these challenging work environments requires the robotic systems to work fully autonomously in achieving human-supplied goals. One approach to designing these autonomous systems is to develop a single robot that can accomplish particular goals in a given environment. However, the complexity of many environments or missions may require a mixture of robotic capabilities that is too extensive to design into a single robot. Additionally, time constraints may require the use of multiple robots working simultaneously on different aspects of the mission in order to successfully accomplish the objective. In some instances, it may actually be easier or cheaper to design cooperative teams of robots to perform some mission than it would be to use a single robot. Thus, we must build teams of possibly heterogeneous robots that can work together to accomplish a mission that no individual robot can accomplish alone.

This report addresses the problem of achieving fault tolerant cooperation within small- to medium-sized teams of heterogeneous mobile robots (say, 2 to 10 robots) by constructing a software architecture, called ALLIANCE, that facilitates cooperative control, and by studying numerous issues in cooperative team design. ALLIANCE is a fully distributed architecture that utilizes adaptive action selection to achieve cooperative control in robot missions involving loosely coupled, largely independent subtasks. The robots in this architecture possess a variety of high-level functions that they can perform during a mission, and must at all times select an appropriate action based on the requirements of the mission, the activities of other robots, the current environmental conditions, and their own internal states. Since such cooperative teams often work in dynamic and unpredictable environments, the software architecture allows the robot team members to respond robustly, reliably, flexibly, and coherently to unexpected environmental changes and modifications in the robot team that may occur due to mechanical failure, the learning of new skills, or the addition or removal of robots from the team by human intervention. In addition, I present an extended version of ALLIANCE, called L-ALLIANCE, that uses a dynamic parameter update mechanism to allow teams of robots to learn from their previous experiences with other robots, and to select their own actions more efficiently on subsequent trials when working with "familiar" robots.

1.1 Cooperative Example: Hazardous Waste Cleanup

On April 26, 1986, at 1:23 AM, the worst civilian nuclear power accident in history occurred at the Chernobyl power station in Ukraine, U.S.S.R. [Woodruff, 1988]. The nuclear reactor was destroyed by an explosion brought on by a series of human operator errors in performing a test, with the resulting graphite fire in the reactor core leading to the contamination of large areas surrounding the plant, requiring the evacuation of 135,000 people. More than two hundred fire fighters and on-site personnel were treated for acute radiation sickness, and many died from direct exposure during the initial emergency response. Although automated solutions to site evaluation were attempted[1], nearly all of the work performed in containing the disaster had to be carried out by humans, many of whom literally sacrificed their lives to stabilize the reactor.

If teams of robots could have been sent into the Chernobyl power station instead of teams of humans, many lives could have been saved. However, the robotics community clearly has a long way to go before we can deal with such applications in which the environment and the capabilities of the robots vary dynamically from moment to moment. A situation as dangerous as that presented at Chernobyl is an ideal application for groups of heterogeneous robots working together to accomplish a mission that is too hazardous for humans. However, notwithstanding the non-trivial radiation-hardening advances needed to allow robots to perform in these environments, many advances must also be made in the software control systems of such robotic teams. This report addresses many of these software control issues of

[1]A German remote-controlled robot was sent to the site to help with the cleanup [Press, 1986], but it failed immediately, presumably due to radiation bombardment. Supposedly a last resort, a small radio-controlled toy car, altered to carry a video camera, turned out to be the most useful automated solution for relaying information to the humans from the remote location.

cooperative robotics that are necessary to achieve this ultimate goal of fault tolerant cooperation in dynamic environments.

A tremendously simplified analog of the Chernobyl application is used here as a descriptive example of the type of cooperation I wish to achieve. Consider the mission illustrated in figure 1-1, in which an artificial hazardous waste spill[2] in a large indoor area must be cleaned up. In this case, the spill consists of a number of small cylindrical objects clustered in one area of the room. I pretend that the spill is dangerous enough that we prefer to avoid the risk of human exposure to toxins, and opt instead to use an automated solution. The mission requires first finding the spill and then moving it to a safe location in the room where we assume the hazardous material can be dealt with more easily by humans. The mission also requires the team to periodically report its progress to the humans monitoring the system by radioing a message from the room entrance. If the monitoring human determines that sufficient progress is not being made, the human sends in another robot or robots to help with the mission.

A difficulty in this mission is that the human monitor does not know the exact location of the spill in robot coordinates, and can only give the robots qualitative information on the initial location of the spill and the final desired location to which the spill must be moved. In this case, the robot or robots are told that the initial location is in the center of the front third of the room, and that the desired final location of the spill is in the back center of the room, relative to the position of the entrance. The ultimate goal is to have the mission completed as quickly as possible without needlessly wasting energy.

For such a mission, it is quite possible that no individual robot possesses all the required capabilities. Thus, we must be able to custom-design a multi-robot team from the available pool of automata such that the group possesses all the required capabilities, and the robots contain a considerable amount of overlap in their abilities. The challenge for each robot, then, is to select the appropriate action to pursue throughout the mission while responding appropriately to the effect of its own actions, the actions of other team members, and dynamic changes in the environment or in the team itself.

Developing a cooperative architecture that allows a team of robots to accomplish this mission involves finding answers to a number of questions. For example, if more than one robot can find the location of the toxic waste spill, how does a robot determine whether or not it should attempt that task? What if one robot is much better at that task than any other team member? What happens if the robot finding the

[2]By *artificial* (rather than *real*), I mean that I pretend that the objects the robots manipulate are hazardous waste. I have not actually applied the robots to real toxic waste spills, since they are simply small laboratory research testbeds, and are in no way designed specifically for such missions.

spill gets stuck in a corner somewhere and is never able to escape; does this mean that the mission doomed to failure? Likewise, if more than one robot can report the progress of the team, how do the robots determine which team member should actually perform that task? How do robots react if a new robot is suddenly added to the team? What happens if the environment suddenly changes and the spill grows much larger? Or, fortuitously, the spill disappears? If the team is using communication, is it destined for failure if the communication mechanism breaks down? What assurance do we have that the robots will be able to accomplish their mission even when no robot failures occur? What if we want to apply the robots to a much larger mission — will their control architecture scale to such a mission? Are robot team members required to have some knowledge of their teammates' capabilities before the start of the mission? Can robot teammates improve the efficiency of their performance from trial to trial when working with "familiar" robots?

This report addresses these and other cooperative control issues by providing mechanisms enabling such cooperation to be accomplished. Throughout the report, I use the results of over 50 logged runs of the physical robots performing this hazardous waste cleanup mission to elucidate the important issues in heterogeneous mobile robot cooperation. Additional experimental results of implementations of ALLIANCE in different cooperative applications are reported in chapter 6, and provide further insight into mobile robot cooperation.

1.2 Design Requirements of a Cooperative Architecture

The difficulties in designing a cooperative team are significant. In [Bond and Gasser, 1988], Bond and Gasser describe the basic problems the field of Distributed Artificial Intelligence must address; those aspects directly related to situated multi-robot systems include the following:

- How do we formulate, describe, decompose, and allocate problems among a group of intelligent agents?
- How do we enable agents to communicate and interact?
- How do we ensure that agents act coherently in their actions?
- How do we allow agents to recognize and reconcile conflicts?

The ALLIANCE architecture described in this report offers one solution to the above questions. In addition to answering these questions, however, one of my primary design goals in developing a multi-robot cooperative architecture is to allow

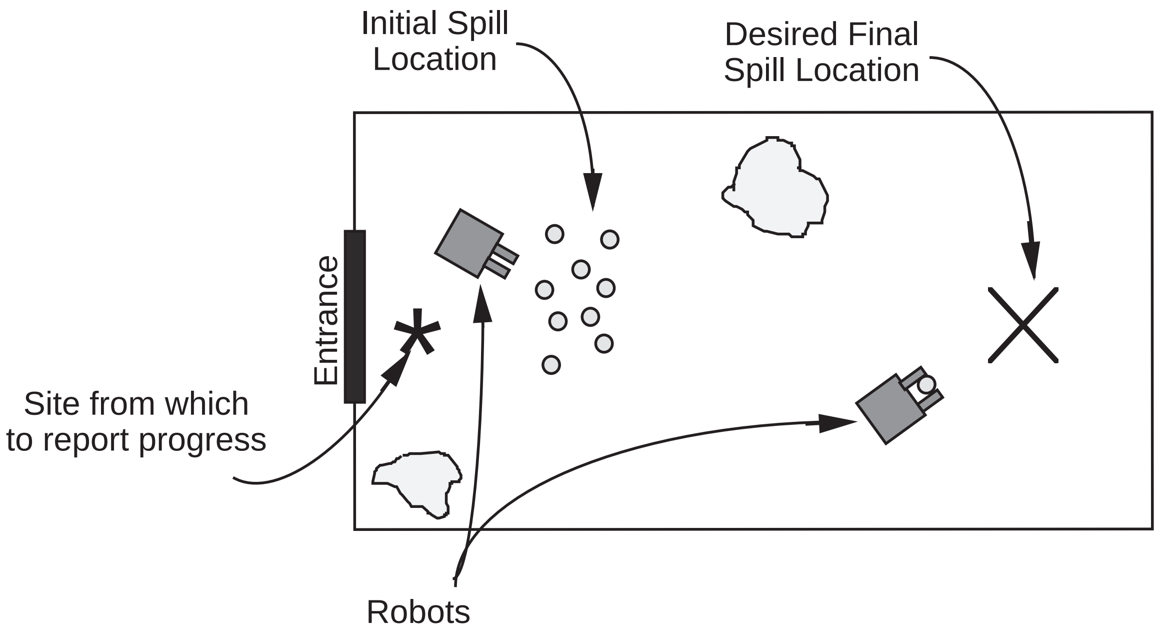
Initial Spill Location
Desired Final Spill Location
Entrance
Site from which to report progress
Robots

Figure 1-1: An example mission to be solved by a cooperative team: hazardous waste cleanup.

the resulting robotic teams to be robust, reliable, and flexible. The following subsections discuss these design requirements that I feel are particularly important for cooperative robotics teams.

1.2.1 Robustness and Fault Tolerance

Robustness refers to the ability of a system to gracefully degrade in the presence of partial system failure; the related notion of *fault tolerance* refers to the ability of a system to detect and compensate for partial system failures. Requiring robustness and fault tolerance in a cooperative architecture emphasizes the need to build cooperative teams that minimize their vulnerability to individual robot outages — a requirement that has many implications for the design of the cooperative team.

To achieve this design requirement, one must first ensure that critical control behaviors are distributed across as many robots as possible rather than being centralized in one or a few robots. This complicates the issue of action selection among the robots, but results in a more robust multi-robot team since the failure of one robot does not jeopardize the entire mission.

Second, one must ensure that an individual robot does not rely on orders from a higher-level robot to determine the appropriate actions it should employ. Relying on one, or a few, coordinating robots makes the team much more vulnerable to individual robot failures. Instead, each robot should be able to perform some meaningful task, up to its physical limitations, even when all other robots have failed.

And third, one must ensure that robots have some means for redistributing tasks among themselves when robots fail. This characteristic of task re-allocation is essential for a team to accomplish its mission in a dynamic environment.

1.2.2 Reliability

Reliability refers to the dependability of a system, and whether it functions properly each time it is utilized. To properly analyze a cooperative robot architecture, one should separate the architecture itself from the robots on which the architecture is implemented. Clearly, if the architecture is implemented on robots that only function 20% of the time, one cannot expect a very dependable result from trial to trial. One measure of the reliability of the architecture is its ability to guarantee that the mission will be solved, within certain operating constraints, when applied to any given cooperative robot team. Without this characteristic, the usefulness of a control architecture is clearly much diminished.

As an example of a reliability problem exhibited in a control architecture, consider a situation in which two robots, r_1 and r_2, have two tasks, t_1 and t_2, to perform. Let us assume that their control architecture leads them to negotiate a task allocation which results in r_1 performing task t_1 and r_2 performing task t_2. Further suppose that r_1 experiences a mechanical failure that neither r_1 nor r_2 can detect. While r_1 valiantly continues to complete task t_1, robot r_2 successfully completes task t_2. However, although r_2 also has the ability to successfully complete task t_1, it does nothing further because it knows that r_1 is performing that task. Thus, the robots continue forever, never completing the mission. One would probably not term such a control architecture *reliable*, since a mere reallocation of the tasks would have resulted in the mission being successfully completed.

1.2.3 Flexibility and Adaptivity

The terms *flexibility* and *adaptivity* refer to the ability of team members to modify their actions as the environment or robot team changes. Ideally, the cooperative team should be responsive to changes in individual robot skills and performance as well as dynamic environmental changes. In addition, the team should not rely on a prespecified group composition in order to achieve its mission.

The first of these requirements — the need for the teams to be responsive to changes in robot performance — reflects the fact that the capabilities of robots change over time due to learning, which should enhance performance, or due to mechanical or environmental causes that may reduce or increase a robot's success at certain tasks. Team members should respond to these changes in performance by taking over tasks that are no longer being adequately performed or by relinquishing those tasks better executed by others. Each robot must decide which task it will undertake based on the actual *performance* of tasks by other robots, rather than on what other robots *say* that they are able to accomplish.

Robots must also exhibit flexibility in their action selection during the mission in response to the dynamic nature of their environment. Obviously, in real environments changes occur that cannot be attributed to the actions of any robot team member or members. Rather, outside forces not under the influence of the robot team affect the state of the environment throughout the mission. These effects may be either destructive or beneficial, leading to an increase or decrease in the workload of the robot team members. The robot team should therefore be flexible in its action selections, opportunistically adapting to environmental changes that eliminate the need for certain tasks, or activating other tasks that a new environmental state requires.

The final flexibility requirement deals with the ease with which the developed architecture can be used by differing groups of robots. The human system designer should not be required to perform a great deal of preparatory work when utilizing different teams of robots for various applications. Rather, I want to build all the robots with the same control architecture and then allow the human designer to form teams as desired from subsets of the available robots. For example, we may have a pool of robots that specialize in various forms of toxic waste cleanup. However, depending upon the particular site to be cleaned up, different groups of robots may be useful for different missions, and thus the team composition varies. The aim is to have these teams perform acceptably the very first time they are grouped together, without requiring any robot to have prior knowledge of the abilities of other team members. However, over time we want a given team of robots to improve its performance by having each robot learn how the presence of other specific robots on the team should affect its own behavior. For example, a robot r_x that prefers to clean floors, but can also empty the garbage, should learn that in the presence of robot r_y, it should automatically empty the garbage, since robot r_y's only abilities are to clean the floors.

1.2.4 Coherence

Coherence refers to how well the team performs as a whole, and whether the actions of individual agents combine toward some unifying goal. Typically, coherence is measured along some dimension of evaluation, such as the quality of the solution or the efficiency of the solution [Bond and Gasser, 1988]. Efficiency considerations are particularly important in teams of heterogeneous robots whose capabilities overlap, since different robots are often able to perform the same task, but with quite different performance characteristics. To obtain a highly efficient team, the control architecture should ensure that robots select tasks such that the overall mission performance is as close to optimal as possible.

A team in which agents pursue incompatible actions, or in which they duplicate each other's actions cannot be considered a highly coherent team. On the other hand, designing a coherent team does not require the elimination of all possible conflict. Rather, the agents must be provided with some mechanism to resolve the conflicts as they arise. A simple example of conflict occurs whenever multiple robots share the same workspace; although they may have the same high-level goals, they may at times try to occupy the same position in space, thus requiring them to resolve their positioning conflict. This can usually be accomplished through a very simple protocol.

Clearly, multi-robot teams exhibiting low coherence are of limited usefulness in solving practical engineering problems. A design goal in building cooperative robot teams must therefore be to achieve high coherence.

1.3 Preview of Results

The primary contribution of this report is ALLIANCE — a novel, fault tolerant cooperative control architecture for small- to medium-sized heterogeneous mobile robot teams applied to missions involving loosely-coupled, largely independent tasks. This architecture is fully distributed at both the individual robot level and at the team level. At the robot level, a number of interacting *motivational behaviors* control the activation of the appropriate sets of behaviors which allow the robot to execute certain tasks. At the team level, control is distributed equally to each robot team member, allowing each robot to select its own tasks independently and without any centralized control. These two levels of distribution allow the ALLIANCE architecture to scale easily to missions involving larger numbers of tasks. The architecture utilizes no form of negotiation or two-way conversations; instead, it uses a simple form of broadcast communication that allows robots to be aware of the actions of their teammates. The control mechanism of ALLIANCE is designed to facilitate fault tolerant cooperation; thus, it allows robots to recover from failures in individual robots or in the communication system, or to adapt their action selections due to changes in the robot team membership or the changes of a dynamic environment. I also show that ALLIANCE is guaranteed to allow robot teams to accomplish their mission for a large class of applications.

My next major contribution is an extension to ALLIANCE, called L-ALLIANCE, that uses a dynamic parameter update mechanism to allow robots to use learned knowledge to improve their performance from trial to trial when working with "familiar" team members — that is, team members with which they have previously worked. I show that, for missions composed of independent tasks, the L-ALLIANCE mechanism allows team members to efficiently select their actions when working with robots that have overlapping capabilities. The significance of L-ALLIANCE is that it (1) eliminates the need for the human designer to adjust the parameters of the architecture manually for each application, (2) allows the human designer to custom-design teams of robots for specific missions, and (3) requires no advance knowledge of the capabilities of robot team members.

I have implemented ALLIANCE and L-ALLIANCE on both simulated and physical robot teams performing a variety of missions, including a hazardous waste cleanup mission, a box pushing demonstration, a janitorial service mission, and a bounding overwatch mission, as well as numerous generic missions. These demonstrations validated this architecture and allowed me to study a number of important issues in cooperative control. The key issues addressed in this report include the following:

- Fault tolerant cooperative control
- Distributed control
- Adaptive action selection
- The importance of robot awareness
- Inter-robot communication
- Improving efficiency through learning
- Action recognition
- Local versus global control of the individual robot

Previous research in fully distributed heterogeneous mobile robot cooperation includes [Noreils, 1993], who proposes a three-layered control architecture that includes a planner level, a control level, and a functional level; [Caloud *et al.*, 1990], who describes an architecture that includes a task planner, a task allocator, a motion planner, and an execution monitor; [Asama *et al.*, 1992], who describes an architecture called ACTRESS that utilizes a negotiation framework to allow robots to recruit help when needed; and [Cohen *et al.*, 1990a], who uses a hierarchical division of authority to perform cooperative fire-fighting. However, these approaches deal primarily with the task selection problem and largely ignore the issues so difficult for physical robot teams, such as robot failure, communication noise, and dynamic environments. Since these earlier approaches achieve efficiency at the expense of robustness and adaptivity, I expect that they will not be able to exhibit the level of fault tolerance required for the types of cooperative control missions I address here. In contrast, my research emphasizes the need for fault tolerant and adaptive cooperative control as a principal characteristic of the cooperative control architecture. I am not aware of any other cooperative control architecture that has exhibited the combination of fault tolerance, reliability, adaptivity, and efficiency possible with ALLIANCE and L-ALLIANCE, and which has been successfully demonstrated on physical mobile robot teams.

1.4 Organization of the Report

The organization of this report is as follows:

Chapter 2: Experimental Testbeds. This chapter describes the two experimental testbeds used in my cooperative robot research: a cooperative robot simulator and a team of real mobile robots.

Chapter 3: ALLIANCE: The Cooperative Robot Architecture. This chapter describes the fully distributed heterogeneous cooperative robot architecture ALLIANCE, including the assumptions of the architecture and a general overview of the approach. I present a formal model of the primary mechanism of ALLIANCE, called the *motivational behavior*, and then provide proofs that, in certain situations, ALLIANCE is guaranteed to allow the robot team to accomplish its mission. I also provide results of implementing ALLIANCE on a team of physical mobile robots performing the artificial hazardous waste cleanup mission.

Chapter 4: L-ALLIANCE: Improving Efficiency. This chapter describes an extended version of ALLIANCE, called L-ALLIANCE, that incorporates an on-line, distributed, dynamic parameter update mechanism that greatly improves the efficiency of cooperative teams performing a mission composed of independent subtasks, while preserving the fault tolerant characteristics of ALLIANCE. I investigate a number of control strategies for improving efficiency and present results from these studies. For those cases in which the optimal solution can be computed, I compare the preferred control strategy with the optimal result, demonstrating the high level of efficiency that can be obtained with this control strategy. I then describe the formal model that incorporates this control strategy into the motivational behaviors of each robot.

Chapter 5: Robot Awareness and Action Recognition. This chapter investigates the importance of *awareness* — the knowledge a robot has about the current actions of its teammates — presenting results from a study of the physical robots performing the hazardous waste cleanup mission.

Chapter 6: Additional Implementations of ALLIANCE. This chapter discusses additional experiments I performed either on physical mobile robots or in simulations using ALLIANCE: a box pushing demonstration, a janitorial service mission, and a bounding overwatch mission.

Chapter 7: Designing Control Laws. This chapter investigates issues in the design of cooperative control laws by examining a case study, Keep Formation, which compares the types of control that are possible with varying degrees of local and global knowledge.

Chapter 8: Related Cooperative Mobile Robot Work. This chapter reviews the current state of the art in cooperative mobile robot systems and relates the work done previously with the research presented in this report. I also provide an analogy between animal societies and current cooperative mobile robot work as an aid to categorization of research.

Chapter 9: Summary and Conclusions. This chapter summarizes the main contributions of this report and describes areas of future work.

Chapter 2

Experimental Testbeds

To investigate issues of cooperative behavior, two experimental environments were utilized in this research: a cooperative robot simulator and a heterogeneous team of physical mobile robots.

The purpose of the simulation experiments was three-fold. First, the simulator was used to to debug the intricacies of the ALLIANCE architecture and to test alternative strategies to its design. This type of debugging and exploration can be quite difficult to perform on physical robots due to the time required to re-download code to each robot, to recharge the robot batteries, to set up the experimental scenarios, and to debug and correct robot mechanical and/or electronic failures. Second, the cooperative robot simulator extended the scope of applications that could be investigated by allowing the construction of a wide variety of robots with various mixtures of sensors and effectors that are not currently available in our laboratory. I made every attempt, however, to keep myself honest by assuming only the existence of sensory and effector devices that are available with current robot technology. And third, the speed of the simulator provided the ability to accelerate "real-time", allowing me a more favorable platform for statistical data collection for many types of experiments.

Of course, years of experience in mobile robot development have shown that approaches to robot control which work in simulated robot worlds are often not successful when applied to real mobile robot teams due to unrealistic assumptions made in the simulations [Brooks, 1991a]. It is therefore important when developing any robot control paradigm to validate the proposed system on real, physical robots. Thus, I also implemented cooperative tasks on our laboratory's team of heterogeneous mobile robots.

The results of my experiments in both of these domains are reported throughout this report. This chapter describes these two testbeds in some detail.

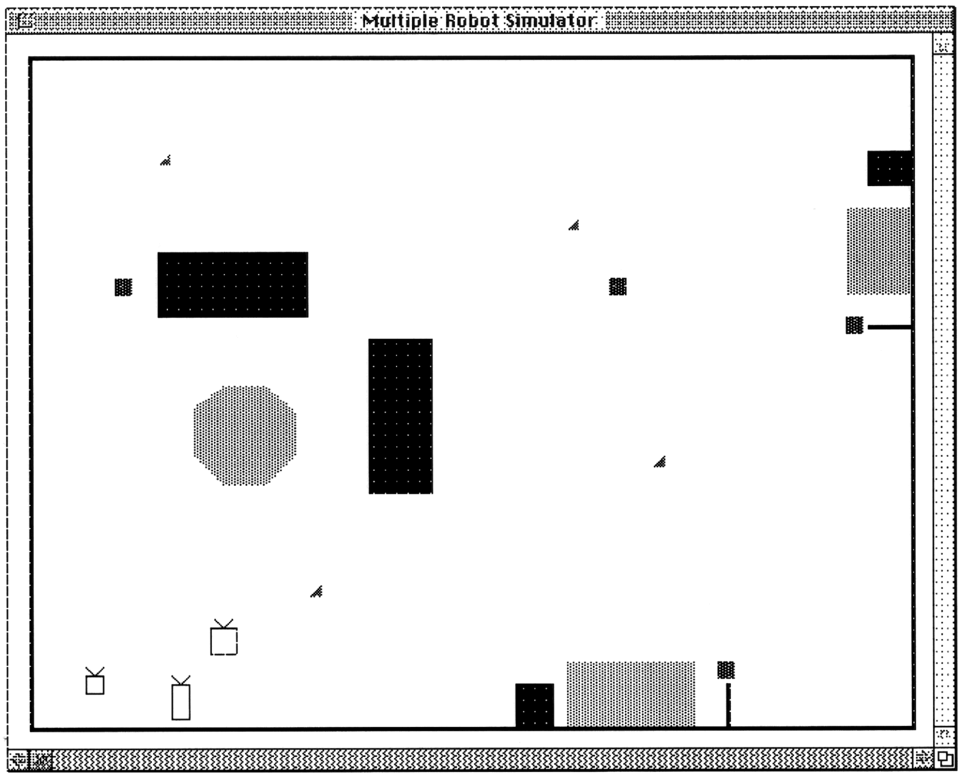

Figure 2-1: A typical indoor environment created using the multi-robot simulator.

2.1 The Cooperative Robot Simulator

The original version of the cooperative robot simulator was developed in the MIT Mobot Laboratory by Yasuo Kagawa, a visiting scientist from the Mazda Corporation. I modified the internal mechanisms of the original version extensively, however, to improve its response time and lower its memory requirements significantly, and to increase the available sensory and effector models. This simulator, written in object-oriented Allegro Common Lisp version 1.3.2, runs in the Macintosh environment and is designed to simulate indoor office-type environments. Figure 2-1 shows a typical office environment which can be created using this simulator. This example is from the janitorial service mission described in chapter 6.

The simulator provides most of the features one would expect in such a cooperative robot simulator: the ability to create a simulated office environment with obstacles and walls, to define sensory and effector devices, to define robots possessing any given combination of sensors and effectors, and the ability for robots to communicate with each other.

The sensors that have been developed are a ranging sensor, an infrared beacon detector, a compass, a microphone, an x, y positioning system, a pyroelectric sensor, and a dust sensor (see [Yamamoto, 1993] for an example of a real-life dust sensor). Each robot's movement is commanded by velocity and turn angle values; additional effectors implemented are a floor vacuum, a garbage dumper, and a duster arm that can be extended either right or left[1]. Note that these additional effectors are not modeled mechanically in any way; they merely act as switches to turn some effect on or off, based on the robot's current position and its proximity to obstacles of certain types. Although all of the sensors and effectors can have a variable amount of random noise added to them, a primary disadvantage of this simulator is that no attempt has been made to accurately model their error profiles. One must therefore keep this in mind when evaluating behaviors that are generated with the simulator. In my experiments, I typically used values of 20% random noise added to the sensors and effectors.

Obstacle objects can be of two types — convex polyhedral objects and one-dimensional wall-type objects. These objects can possess a number of additional characteristics, such as the ability to emit an IR beacon or sound at a specified intensity, or to possess a certain amount of dust, garbage, or heat. A nice feature of the simulator is that objects can be moved around manually during the robot simulation, thus mimicking a dynamic environment.

A major strength of the robot simulator is that the user-written robot control programs are written in the Behavior Language[2] [Brooks, 1990a], which is also the programming language used in the real mobile robot experiments. This allows most of the robot control code developed using the simulator to be easily ported to run on an actual mobile robot. The main exception is, of course, the sensor and effector interfaces, which will be different on a physical robot.

2.2 The Pool of Heterogeneous Robots

A primary goal of this research is to allow a human system designer to create new teams of cooperative robots by selecting, from a pool of available robots, those robots which have the proper mix of capabilities for the current application. To enable the demonstration of this capability, I have composed a pool of heterogeneous mobile robots from which I can create various teams with differing group capabilities. Shown in figure 2-2, the Mobot Laboratory's pool of heterogeneous robots consists of two

[1]These effectors were designed for the janitorial service mission, which is described in chapter 6.

[2]The *Behavior Language* is a modified and extended version of the subsumption architecture [Brooks, 1986] that facilitates real-time robot control.

types[3] of mobile robots — three R-2s and one Genghis-II — all of which were designed and built by IS Robotics Corporation located in Somerville, Massachusetts.

It is important to note, however, that even though our laboratory has duplicate copies of the R-2 robot, significant variations in the sensitivity and accuracy of their sensors and effectors cause them to have quite different true capabilities. It is also possible to modify the morphology of an individual robot in several ways, such as installing a gripper attachment, that greatly affects a robot's capabilities. Thus, a heterogeneous robot team can be composed that consists of only the R-2 type of robot, since the robot behavior varies noticeably.

Of course, when working with specific mobile robots, one is limited in the applications that can be demonstrated by the physical limitations of the available robots. Thus, the physical robot experiments described in this report have been designed specifically with the capabilities of the R-2s and Genghis-II in mind. The ALLIANCE architecture, however, is independent of the specific robot platform on which it is implemented.

The next two subsections describe the capabilities of these robots, followed by a description of the radio communication and positioning system that allows the robots to send messages to each other and to determine their own current x, y position relative to a global frame of reference.

2.2.1 R-2 Robots

Mechanical Design

Shown from four views in figure 2-3, the R-2 robot [IS Robotics, 1992b] is a small, fully autonomous wheeled vehicle measuring 10 inches wide, 12.5 inches deep (including the gripper), and 14 inches tall (including the radio communication and positioning subsystem). The robot has two 3-inch diameter drive wheels that are driven differentially at speeds up to 18 inches per second. Two caster wheels in the rear provide stability. The R-2 also possesses a two degree of freedom parallel jaw gripper with a 3-inch maximum width, a 7-inch maximum lift, and a lifting capacity of two pounds. Onboard rechargeable batteries provide 1400 mAh capacity at 14.4 volts; in practice, this allows about 45 minutes of continuous operation. Three output devices — a piezoelectric buzzer; a four-line, 16 character per line LCD display; and a white LED lamp in the left finger — are available for determining the robot status. Normally, one might think that distinguishing the robots from each other would be difficult,

[3] I should note here that the Mobot laboratory also possesses 20 R-1 robots, which are smaller versions of the R-2, and which I had originally intended to use in these cooperative robot experiments. Unfortunately, the radio communication system on the R-1s is incompatible with that on the other robots and could not be easily or cheaply upgraded.

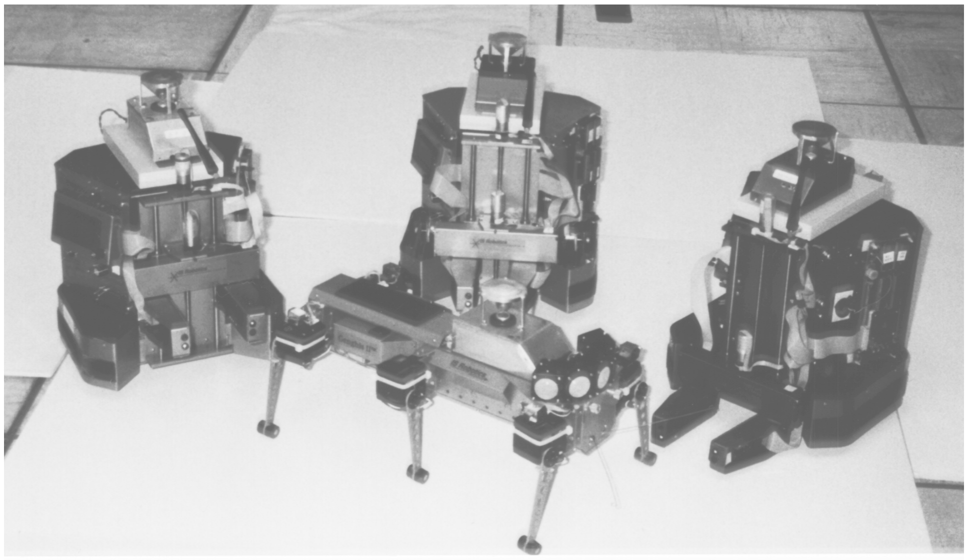

Figure 2-2: The pool of heterogeneous robots — three R-2s and one Genghis-II.

since they are all of the same type. However, identification of individual robots is made easy on this team — each robot has its own color: GREEN, BLUE, and RED[4].

Sensors

The R-2 robot is equipped with several types of sensors. Eight infrared (IR) proximity sensors are positioned at the tips of both fingers and along the sides and back of the

[4]We also have a fourth assembled R-2, GOLD, and the parts to a fifth R-2 robot, PURPLE, but incessant mechanical problems rendered them unusable.

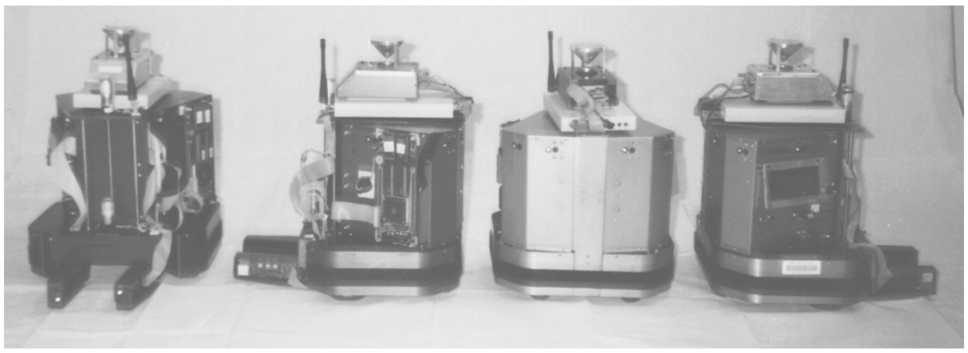

Figure 2-3: Four views of the R-2 robot.

robot. These IR sensors have an effective range of about 8 to 12 inches. Piezoelectric bump sensors sensitive to both force and position are present around the base of the robot for use in collision detection, and inside the fingers for use in measuring gripping force. Shaft encoders are installed on the lift and grip motors and on the drive wheels to enable velocity sensing. Also included are sensors to measure battery current, battery voltage, and motor drive current. Two break beam sensors between the fingers allow detection of objects between the fingers at either the front or at the rear of the gripper.

Computational Organization

To enable real-time control of the R-2, its computational functions are divided across several onboard microprocessors [IS Robotics, 1992b, IS Robotics, 1993a] — a Motorola 68332 master processor and five Motorola MC68HC11 slave processors. All of the processors, plus some hardware function cards, are linked together via a common backplane. The microprocessors communicate along this backplane based on the Synchronous Serial Peripheral Interface facility provided by the MC68HC11.

The 68332 master processor executes the user's behavior language code and thus controls the overall operation of the robot. It uses input received from the robot sensors to send velocity commands to the two wheels and to send position commands to the lift and grip motors. The 68332 processor resides on a board made by Vesta, which is configured with 128K bytes of ROM for the operating system code and 1M byte of RAM for the user code. The 68332 also has 2K bytes of on-chip RAM. The processor runs at 16.777 MHz.

The five MC68HC11 slave processors are responsible for processing data from the sensors and for controlling the actuators. The functionality of the robot is divided across the five slave processors as follows:

1. *Right and Finger Processor*: controls the right drive wheel motor, the finger gripper motor, and the LED lamp.
2. *Left and Lift Processor*: controls the left drive wheel motor and the lift motor.
3. *Radio Processor*: controls the radio communication and positioning subsystem.
4. *Status Processor*: controls the LCD and buzzer output devices and monitors the current and voltage sensors.
5. *Proximity Processor*: controls the infrared and bump sensors.

Each MC68HC11 has 2K of EEPROM that contains a sensor or motor driver, and 256 bytes of RAM.

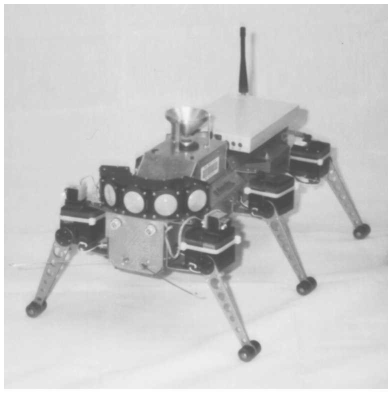

Figure 2-4: The Genghis-II robot.

The 68332 master processor is programmed by the user using *Behavior Language* code [Brooks, 1990a]. Software functions and macros are provided that allow the user program to receive sensory feedback from the slave processors and to send motor control commands to the appropriate slave.

The slave processors are programmed in M68HC11 assembly code and are designed to respond to the master processor whenever requested, either by sending data to the master or by receiving data from the master. The slave is not allowed to request services from the master.

2.2.2 Genghis-II

Mechanical Design

Genghis-II [IS Robotics, 1992a], shown in figure 2-4, is the industrialized version of Genghis, which was built in the Mobot Laboratory at MIT under the direction of Rodney Brooks [Angle, 1991, Brooks, 1989]. This robot is a small legged robot with six two-degree-of-freedom (swing and lift) legs, and in a standing position measures approximately 16 inches long, 11 inches wide, and 9 inches high (including the radio positioning system). Each leg is approximately 6 inches long. Two output devices — a piezoelectric buzzer and a series of 8 LEDs — are available for displaying the robot status.

Two onboard battery sources provide separate power for the electronics and radio systems and the 12 motors. The electronics and radio systems are powered by two non-rechargeable 6-volt lithium batteries in series; the motors are powered by rechargeable NiCad batteries that provide approximately 45 minutes of operation.

Sensors

Genghis-II's sensor suite includes two whiskers at the front for obstacle detection, force detectors on each leg, a passive array of infrared (pyroelectric) heat sensors, three tactile sensors along the robot belly, four near-infrared sensors mounted on the shoulders, and an inclinometer for measuring the pitch of the robot. The robot also has 5 mode switches that can be utilized by the user to interface to Genghis-II.

Computational Organization

Similar to the R-2, Genghis-II's onboard computational resources are divided across a master processor and, in this case, four slave processors. All of these processors are Motorola MC68HC11 microprocessors. In Genghis-II, the master processor is the servo control processor, which also controls the force, whisker, and pyro sensors. The functionality of the four slave processors is distributed as follows:

1. *Behavior Processor*: runs the user's behavior language code.
2. *Infrared Processor*: controls the infrared and inclinometer sensors and the mode switches.
3. *Radio Processor*: controls the radio communication and positioning system.
4. *Status Processor*: controls the piezoelectric buzzer, the LED's, and the tactile sensors along the robot belly.

As with the R-2, the Behavior processor is programmed by the user using the Behavior Language. Software macros provide facilities for interfacing to the sensor and motor drivers for robot control. The remainder of the processors are programmed in M68HC11 assembly code.

2.2.3 Radio Communication and Positioning System

The radio communication and positioning system used with this pool of robots consists of four parts [IS Robotics, 1993c, IS Robotics, 1993b]:

1. A radio transceiver unit attached to each robot.
2. An omnidirectional ultrasonic receiver attached to each robot.
3. A radio base station used to coordinate the radio waves and to control the ultrasonic pingers.

4. Two ultrasonic pingers (called *pinger A* and *pinger B*) used by the robots to triangulate their own global position.

This system serves two purposes: (1) to allow the robots to communicate with each other, and (2) to allow the robots to determine their own current global x, y position.

Communication

Designing a reliable communication system for a multi-robot team can be quite difficult. Ideally, the communication mechanism would allow robots to send messages whenever they desire, to whomever they desire, without experiencing conflicts from other communicating robots. In practice, this goal can be quite difficult to achieve, especially when the robots must share the same communications medium. The radio system designed for the robot teams used in this report prevents conflicts in the use of the airwaves while still providing ample opportunity for robots to send messages frequently.

This collision-free communication is accomplished by having the radio base station coordinate the use of the radio waves by transmitting a message that awards a specific robot a certain slice of time in which to broadcast a 10-byte message. The time slices are determined by assigning each robot a unique identification number and having the base station cycle through the robot identification numbers repeatedly. The current implementation allows each robot to broadcast messages at a rate of once every three seconds.

Positioning

Generating interesting cooperative behavior in robot teams is quite difficult to achieve when using robots that are restricted to sensors with detection ranges less than their own widths. To enhance the capabilities of the individual robots, it is extremely helpful to endow them with the ability to have some sense of physical position, both globally and with respect to other robot team members. This knowledge can greatly extend the capabilities of the individual robots and the team as a whole, and provide a framework from which interesting cooperative behavior can be generated. Thus, this cooperative environment includes a positioning system giving the robot team this increased capability.

As shown in figure 2-5, the primary robot work area measures 26 feet by 12 feet, and is surrounded by a wall boundary detectable by the robots. The two ultrasonic pingers, A and B, are placed 7.5 feet apart at the short end of the work area, and emit approximately a 100-degree ultrasonic cone with a range of about 27 to 30 feet. Each robot is told the baseline distance between the two pingers. This arrangement

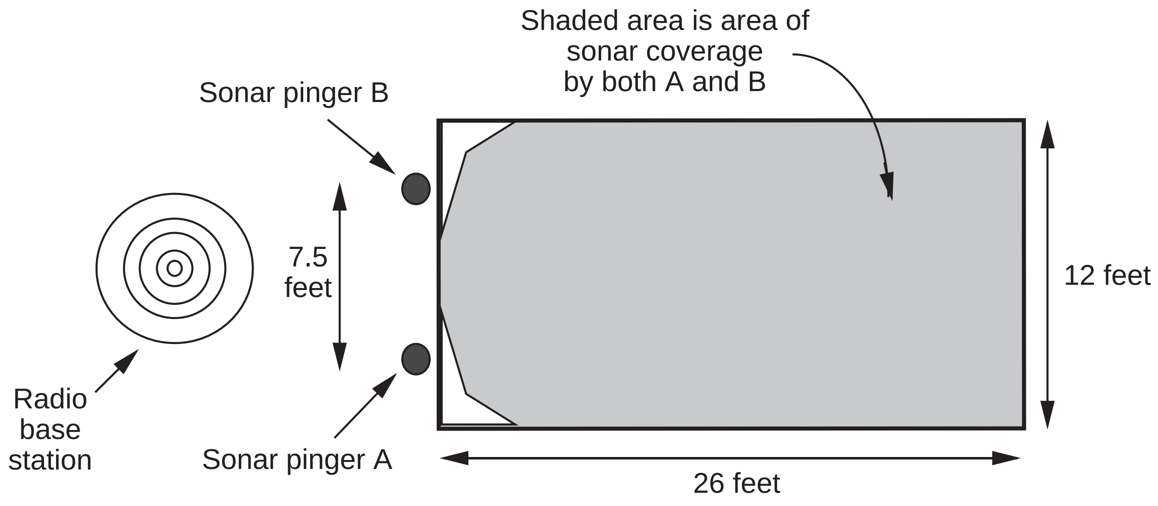

Figure 2-5: Physical layout of the radio communication and positioning system. The rectangular area is the multi-robot work area.

of the pingers gives almost complete coverage in the work area by both pingers; those areas not covered are the unshaded areas shown in figure 2-5.

The positioning system is coordinated by the radio base station, which broadcasts a message to all robots that says, in essence, "listen for pinger A", while simultaneously commanding pinger A to send out an ultrasonic signal. Upon receipt of this radio message, each robot listens for the ultrasonic signal, and then calculates its distance to pinger A based upon the velocity of sound in air. This is repeated for pinger B, at which time the robots can calculate their own positions by using triangulation and the known distance between the two pingers. If the robot does not detect a signal from one of the pingers within a certain amount of time, a special value of -1 for the range to that pinger is returned. The user's behavior language code can then recognize when it has incomplete information on its position and act accordingly. In my experiments, these dead zones have not been a problem because the robots back away from the surrounding wall obstacle in such a way that they return to an area receiving both sonar pings fairly quickly.

This positioning system is quite accurate and gives the robots the ability to localize their position to within 1/2 inch when stationary. The position is somewhat noisy when the robot is moving, however, due to interference from the robot motors, giving a practical accuracy of about 6 inches. The position updates occur once every 600 milliseconds.

2.3 Summary

In this chapter, I have described the two primary testbeds used in this report for studying alternative approaches to cooperative control — the cooperative robot simulator and a pool of physical robots. My experience in researching this report had led me to the conclusion that both simulated robot and physical robot testbeds are vital to the research of general cooperative control issues such as those studied in this report. Whereas the physical robots kept my research firmly rooted in the real world and prevented the use of unrealistic assumptions, the simulated robots provided the ability to thoroughly investigate the characteristics of the ALLIANCE architecture through variation of robot capabilities and robot applications, and through collection of large amounts of data on team performances that would have been impossible to collect on any given physical robot team. Neither of these testbeds could provide all of these features alone. In the following chapters, I describe the results of the ALLIANCE implementations which use these testbeds.

Chapter 3

ALLIANCE: The Cooperative Robot Architecture

This chapter describes the novel fault tolerant control architecture, ALLIANCE[1], which enables heterogeneous mobile robot cooperation. I begin by discussing the assumptions that were made, and those that were explicitly *not* made, in developing this architecture, and then give the formal definition of the problem addressed by ALLIANCE. I provide an overview of the approach taken in ALLIANCE, and then discuss the details of the primary action selection mechanism of the architecture — the *motivational behavior* — along with a formal model of the mechanism. I then provide proofs that, in certain situations, ALLIANCE is guaranteed to allow the robot team to accomplish its mission. I describe the results of implementing ALLIANCE on a physical robot team performing the hazardous waste cleanup mission introduced in chapter 1. I conclude this chapter by returning to the design requirements described in chapter 1 and discussing how ALLIANCE meets these requirements.

3.1 Assumptions Made in ALLIANCE

In the design of any control scheme, it is important to make explicit those assumptions underlying the approach. I list here the assumptions made in ALLIANCE, and then

[1]I chose the name *ALLIANCE* to emphasize the analogy between the cooperation exhibited by robots under this architecture and the cooperation exhibited by groups of political nations which form an alliance for the mutual benefit of all the participants. In a political alliance, individual nations are able to survive to some extent without other nations; yet they form an alliance in order to achieve more as a group than as individuals. Likewise, in this cooperative control architecture, individual robots are able to survive and perform useful tasks completely on their own; yet, by joining with other robots, the team is able to accomplish more as a group than is possible with individual robots alone.

discuss the implications of these assumptions.

I assume:

1. The robots on the team can detect the effect of their own actions, with some probability greater than 0.
2. Robot r_i can detect:
   - the actions of other team members for which r_i has redundant capabilities, with some probability greater than 0; these actions can be detected through any available means, including explicit broadcast communication.
   - the effect of the above actions.
3. The robots share a common language.
4. Robots on the team do not lie and are not intentionally adversarial.

Since robots that have no idea how well they are executing their task cannot exhibit much robustness, flexibility, reliability, or coherence, I make the first assumption to ensure that robots have some measure of feedback control and do not perform their actions purely with open-loop control. However, I do not require that robots be able to measure their own effectiveness with certainty, because I realize this rarely happens on real robots. As we shall see in the following sections, ALLIANCE provides mechanisms to facilitate detection of robot failures and difficulties in task performance.

The second assumption deals with the problem of *action recognition* — the ability of a robot to observe and interpret the behavior of another robot. Without the ability for robots to perform action recognition, it is impossible to create cooperative teams. However, it is not required that robots determine these actions through passive observation, which can be quite difficult to achieve (see section 5.1 for a further exploration of this issue). Instead, it is quite acceptable for robots to learn of the actions of their teammates through an explicit communication mechanism, whereby robots broadcast information on their current activities to the rest of the team. However, I also recognize that communication can be disrupted, so ALLIANCE provides methods of creating robust and reliable cooperative behavior even in the absence of complete knowledge of the actions of other teammates.

It is important to note that in the second assumption I distinguish between the detection of the current actions of team members and the detection of the *effects* of those actions. I make this distinction because I want robots to respond not only to the *intentions* of those teammates, but also to the environmental changes caused by the actions of those teammates. Thus, robots should allow team members to continue their tasks only if they demonstrate their ability to successfully accomplish those tasks through their effect on the world.

This second assumption implies that the third assumption must be true — that is, that the robots must share an unambiguous common language, to the extent that their capabilities overlap, whether they interpret the actions of other robots passively or actively. If the actions are interpreted passively, the robots must in essence share a common *body language*, whereas the use of an explicit communication mechanism implies the presence of a more traditional language, including a vocabulary and usage rules. Of course, the robots need not share a language concerning capabilities that are not shared by other robots.

Finally, I assume that the robots are built to work on a team, and are neither in direct competition with each other, nor are attempting to subvert the actions of their teammates. In particular, it is important in ALLIANCE that when robots broadcast information on their current activities, they can be assumed to be telling the truth. Although at a low level conflicts may arise due to a shared workspace, for example, I assume that at a high level the robots share compatible goals.

3.2 Assumptions Not Made in ALLIANCE

The ALLIANCE architecture explicitly does not make certain assumptions that are often made in cooperative architectures addressing a similar application domain. Enumerating these "non-assumptions" helps clarify the differences between ALLIANCE and other architectures for heterogeneous robot cooperation.

ALLIANCE does *not* make the following assumptions:

1. The communications medium is guaranteed to be available.

2. The robots possess perfect sensors and effectors.

3. Whenever a robot fails, it can communicate its failure to its teammates.

4. A centralized store of complete world knowledge is available.

A primary philosophy in the design of ALLIANCE is that the cooperative team should be able to continue to function as much as possible even in the midst of dynamic changes in the abilities of the robot team members, or in the state of the environment. Because of this philosophy, I want the robots in ALLIANCE to be able to function to some extent even when their communications medium has failed. Naturally, when communication is available, robot teams should take advantage of it; however, I do not want the team to experience total breakdown when communication becomes unavailable. (See [Arkin *et al.*, 1993] for a similar philosophy towards communication.)

For similar reasons, I likewise do not assume that the sensors and effectors of the robot team members are perfect. Robots must have some ability to monitor their own progress and the progress of their teammates. However, I do not carry this assumption to the extreme position in which I assume that a robot can always detect its own failure and will always be able to communicate its failure to the other robots on the team. Thus, robots should be somewhat skeptical of the ability and claims of other robots to accomplish certain tasks unless those robots demonstrate their ability through the world to actually achieve those tasks.

Finally, I cannot assume that robots have access to some centralized store of world knowledge, or that some centralized agent is monitoring the state of the entire robot environment and can make controlling decisions based upon this information. Clearly, once robots begin to depend upon any type of centralized control or centralized global information, they become much more vulnerable to failures of that centralized agent. Additionally, the centralized store or source of control can be a bottleneck that severely constrains the abilities of the robot team.

3.3 Formal Problem Definition

It is important to note here that this report deals strictly with cooperative robot missions that are composed of loosely coupled subtasks that are largely independent of each other. By "largely" independent, I mean that tasks can have fixed ordering dependencies, but they cannot be of the type of "brother clobbers brother" [Sussman, 1973], where the execution of one task required by the mission undoes the effects of another task required by the mission. Even with this restriction, however, this report covers a very large range of missions for which cooperative robots are useful. As we shall see in this and later chapters, a wide variety of applications have been implemented and are reported that fall into this domain of loosely coupled, largely independent subtasks.

I now define formally the problem addressed in this report. Let the set $R = \{r_1, r_2, ..., r_n\}$ represent the set of n heterogeneous robots composing the cooperative team, and the set $T = \{task_1, task_2, ..., task_m\}$ represent m independent subtasks which compose the *mission*. I use the term *high-level task-achieving function* to correspond intuitively to the functions possessed by individual robots that allow the robots to achieve tasks required in the mission. These functions map very closely to the upper layers of the subsumption-based control architecture [Brooks, 1986]. In the hazardous waste cleanup mission, the high-level functions are: *find-initial-final-locations*, *move-spill*, and *report-progress*. Table 3.1 gives examples of what I consider to be the high-level task-achieving functions of a number of previously reported robots. In the ALLIANCE architecture, the control code of each robot

Robot	High-Level Functions
Allen [Brooks, 1986]	Wander
Attila/Hannibal [Ferrell, 1993]	Keep walking
Genghis [Brooks, 1989]	Keep walking
George/HARV [Arkin, 1990]	Reactively navigate
Herbert [Connell, 1989]	Collect empty soda cans
Hilare [Giralt *et al.*, 1983]	Map office environment
Polly [Horswill, 1993]	Give 7th floor AI Lab tours
Rocky III [Miller *et al.*, 1992]	Search for soft soil; acquire soil sample; return sample to home
Rocky IV [Gat *et al.*, 1993]	Collect soil sample; chip rocks; deploy instruments; return sample to home
RPV [Bonasso, 1991]	Reactively navigate underwater
Squirt [Flynn *et al.*, 1989]	Eavesdrop
Toto [Mataric, 1992b]	Map office environment; go to goal

Table 3.1: High level task-achieving functions of various robots.

is organized into groups of behaviors called *behavior sets*, each of which supplies the robot with a high-level task-achieving function. Thus, in the ALLIANCE architecture, the terms *high-level task-achieving function* and *behavior set* are synonymous.

I refer to the high-level task-achieving functions, or behavior sets, possessed by robot r_i in ALLIANCE as the set $A_i = \{a_{i1}, a_{i2}, ...\}$. Since different robots may have different ways of performing the same task, we need a way of referring to the task a robot is working on when it activates a behavior set. Thus, I define the set of n functions $\{h_1(a_{1k}), h_2(a_{2k}), ..., h_n(a_{nk})\}$, where $h_i(a_{ik})$ returns the task in T that robot r_i is working on when it activates behavior set a_{ik}.

I now define the notions of *goal-relevant capabilities* and *task coverage.*

Definition 1 *The* goal-relevant capabilities *of robot r_i, GRC_i, are given by the set:*

$$GRC_i = \{a_{ij} | h_i(a_{ij}) \in T\}$$

where T is the set of tasks required by the current mission.

In other words, the capabilities of robot r_i that are relevant to the current mission (i.e. *goal*) are simply those high-level task-achieving functions which lead to some task in the current mission being accomplished.

The term *coverage* has been used in a number of contexts, such as DNA physical mapping in computational biology [Lander and Waterman, 1988] and in multiple robotics applications dealing with spatial distribution of robots [Gage, 1993]. Here, I use the term *task coverage* to give a measure of the number of capabilities on the team that may allow some team member to achieve a given task. This is the measure that is used by human (or automated) designers to collect robots from the available pool of robots to form a team with the required capabilities to accomplish a given mission. However, we cannot always predict robot failures; thus, at any point during a mission, a robot may reach a state from which it cannot achieve a task for which it has been designed. This implies that the expected task coverage for a given task in a mission may not always equal the *true* task coverage once the mission is underway.

Definition 2 Task coverage *is given by:*

$$task\_coverage(task_k) = \sum_{i=1}^{n} \sum_{j} \left\{ \begin{array}{ll} 1 & \text{if } (h_i(a_{ij}) = task_k) \\ 0 & \text{otherwise} \end{array} \right\}$$

Note that if any one robot has more than one way to accomplish $task_k$, that redundancy is included in the task coverage value. An interesting side-effect of this definition is that in homogeneous robot teams, the task coverage for all tasks in T is always a positive integral multiple of the number of robots n. That is,

$$\forall k. \exists (c \in \mathbf{N}). task\_coverage(task_k) = c \times n$$

where $\mathbf{N}$ is the set of natural numbers. Note that this is not a sufficient condition for homogeneity, since the n robots could have n different ways of accomplishing task $task_k$.

The task coverage measure is useful for composing a team of robots to perform a mission from the available pool of heterogeneous robots. At a minimum, we need the team to be composed so that the task coverage of all tasks in the mission equals 1. This minimum requirement ensures that, for each task required in the mission, a robot is present that has some likelihood of accomplishing that task. Without this minimum requirement, the mission simply cannot be completed by the available robots. Ideally, however, the robot team is composed so that the task coverage for all tasks is greater than 1. This gives the team a greater degree of redundancy and overlap in capabilities, thus increasing the reliability and robustness of the team amidst individual robot failures. Although it is often most useful to have a uniform task coverage across the mission, some missions may include a subset of tasks that are of particular importance; this subset of tasks may therefore require higher levels of task coverage than the remainder of the tasks.

Within this application domain, two key problems must be addressed. First, a control architecture must be developed that allows robots on these cooperative teams to properly select tasks during the mission such that the collective actions of the team lead to mission completion. This is the problem that the cooperative architecture

ALLIANCE addresses. Second, a control architecture must be developed that not only allows a robot team to complete its mission, but to complete it *efficiently*. The enhanced cooperative architecture L-ALLIANCE, which is discussed in chapter 4, addresses this problem.

3.4 Overview of ALLIANCE

ALLIANCE is a fully distributed architecture for fault tolerant, heterogeneous robot cooperation that utilizes adaptive action selection to achieve cooperative control. Under this architecture, the robots possess a variety of high-level task-achieving functions that they can perform during a mission, and must at all times select an appropriate action based on the requirements of the mission, the activities of other robots, the current environmental conditions, and their own internal states. Adaptive action selection is achieved through the selfish interests of individual robots, modified by their analyses of the current and previous performance of other team members. The *performance* of a robot is determined solely by how that robot affects the world, and is not dependent upon explicit, often artificial skill metrics.

Adaptive action selection is facilitated in this architecture by designing the robots to be somewhat selfish and lazy. They are selfish in that they only do what they "want"[2] to do and what they "think" is in their own best interests, as determined by their motivations and the environment. The *best interests* of a robot are defined from the local point of view of that robot, not from some omniscient onlooker with a global view. These interests are defined very simply as the action(s) the robot is most motivated to perform at each point in time. The purpose of this approach is to maintain a purely distributed cooperative control scheme which affords an increased degree of robustness; since individual robots are always fully autonomous, they have the ability to perform useful actions even amidst the failure of other robots.

The robots in this architecture are lazy in the sense that, although they want certain tasks to be accomplished, they do not care if some other robot performs those tasks for them. For example, a baggage-handling robot may want both (1) the baggage to be removed from an airplane, and (2) the removed baggage to be placed on a cart. However, it is fine with this robot if another robot does one or both of

[2]Throughout this report, I use terms that seem to attribute intent, cognition, and a will to the robots designed under the ALLIANCE and L-ALLIANCE architectures — terms such as "think", "want", "know", "believe", "decide", and "forget". These qualities are used solely to simplify the discussion and should be viewed strictly as characteristics imposed by an observer watching the behavior of the robots; they usually do not reflect the internal organization of the robots. (See [Braitenberg, 1984] for an interesting discussion of this distinction.) The robots in ALLIANCE explicitly do *not* use belief systems typical in many multi-agent systems.

these tasks, as long as the tasks get done. This characteristic prevents the needless waste of energy due to replication of effort.

In ALLIANCE, individual robots are designed using a behavior-based approach [Brooks, 1986]. Under the behavior-based construction, as shown in figure 3-1, a number of task-achieving behaviors are active simultaneously, each receiving sensory input and controlling some aspect of the actuator output. The lower-level behaviors, or competences, correspond to primitive survival behaviors such as obstacle avoidance, while the higher-level behaviors correspond to higher goals such as map building and exploring. The output of the lower-level behaviors can be *suppressed* or *inhibited* by the upper layers when the upper layers deem it necessary. When the output of one layer, L_1, is *inhibited* by another layer, L_2, L_1's output is prevented from reaching its destination. When layer L_1's output is *suppressed* by Layer L_2, not only is L_1's output prevented from reaching its destination, but it is also replaced by an output from L_2. Within each layer of competence may be a number of simple modules interacting via inhibition and suppression to produce the desired behavior, as shown in figure 3-2. (In this and the previous figure, the small circles indicate either inhibition or suppression of the behavior outputs. When we want to distinguish between the two types of interaction, we place an "I" within the circle to indicate inhibition, and an "S" to indicate suppression.) This approach has been used successfully in a number of robotic applications, several of which are described in [Brooks, 1990b].

Extensions to this approach are necessary, however, when a robot must select among a number of competing actions — actions which cannot be pursued in parallel. Unlike typical behavior-based approaches, ALLIANCE delineates several *behavior sets* that are either active as a group or hibernating. Figure 3-3 shows the general architecture of ALLIANCE and illustrates three such behavior sets. Each behavior set a_{ij} of a robot r_i corresponds to those levels of competence required to perform some high-level task-achieving function. Because of the alternative goals that may be pursued by the robots, the robots must have some means of selecting the appropriate behavior set to activate. This action selection is controlled through the use of *motivational behaviors*, each of which controls the activation of one behavior set. Due to conflicting goals, only one behavior set should be active at any point in time. This restriction is implemented via cross-inhibition of behavior sets, represented by the arcs at the top of figure 3-3, in which the activation of one behavior set suppresses the activation of all other behavior sets. However, other lower-level competences such as collision avoidance may be continually active regardless of the high-level goal the robot is currently pursuing. Examples of this type of continually active competence are shown in figure 3-3 as layer 0, layer 1, and layer 2.

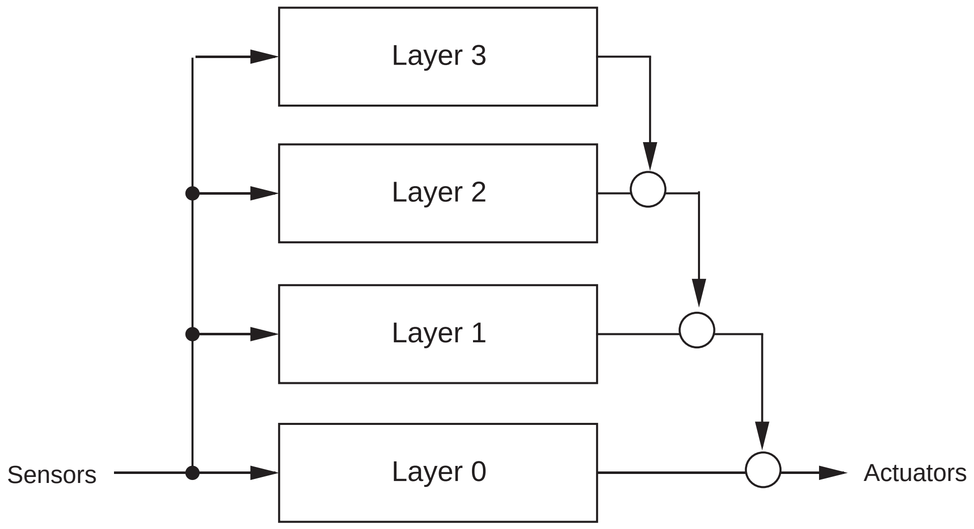

Figure 3-1: The organization of a typical behavior-based architecture. Each layer is responsible for some level of competence of the robot. For example, a lower layer (Layer 0) might provide obstacle avoidance capabilities, while an upper layer (Layer 3) might provide a map-building competence. The circles indicate inhibition or suppression of lower layer outputs by upper layers. Typically, either an "I" or an "S" is placed within each circle to distinguish between inhibition and suppression of outputs.

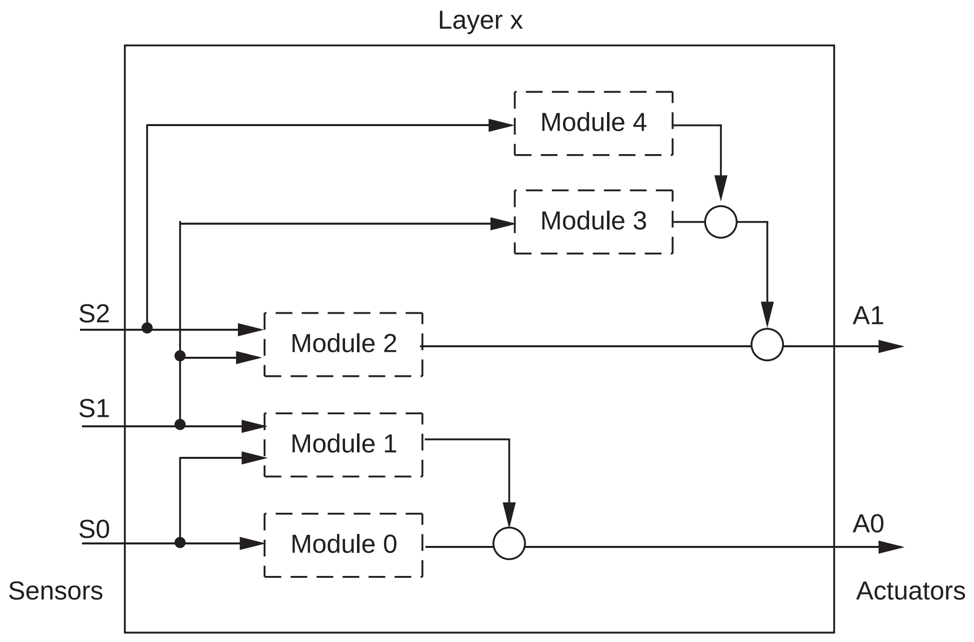

Figure 3-2: Within each layer of competence are usually several simple modules (dashed rectangles) interacting to produce the desired behavior. The circles indicate suppression or inhibition of module outputs.

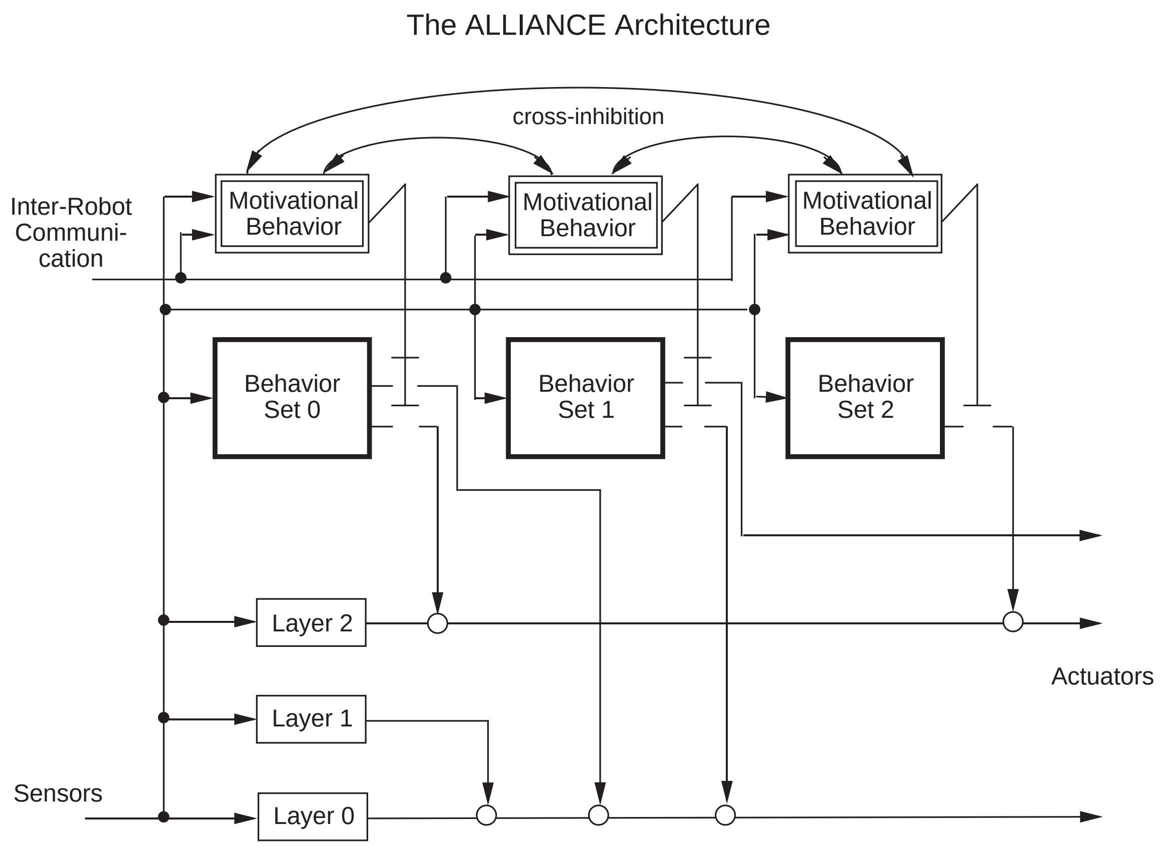

Figure 3-3: The ALLIANCE architecture, implemented on each robot in the cooperative team, delineates several behavior sets, each of which correspond to some high-level task-achieving function. The robot must have some mechanism allowing it to choose which high-level function to activate; the primary mechanism providing this ability is the *motivational behavior*. Within each behavior set are a number of layers of competence organized like those shown in figure 3-1, and within each layer of competence are a number of simple interacting modules as shown in figure 3-2. The symbols in the current figure that connect the output of each motivational behavior with the output of its corresponding behavior set (vertical lines with short horizontal bars) indicate that a motivational behavior either allows all or none of the outputs of its behavior set to pass through to the robot's actuators. The non-bold, single-bordered rectangles correspond to individual layers of competence that are always active.

3.5 Motivational Behaviors

The primary mechanism for achieving adaptive action selection in this architecture is the *motivational behavior*. At all times during the mission, each motivational behavior receives input from a number of sources, including sensory feedback, inter-robot communication, inhibitory feedback from other active behaviors, and internal motivations called *robot impatience* and *robot acquiescence*. The output of a motivational behavior at a given point in time is the activation level of its corresponding behavior set, represented as a non-negative number. When this activation level exceeds a given threshold, the corresponding behavior set becomes active. Once a behavior set is activated, other behavior sets in that robot are suppressed so that only one behavior set is active in an individual robot at a time.

Intuitively, a motivational behavior works as follows. Robot r_i's motivation to activate any given behavior set, a_{ij}, is initialized to 0. Then over time, robot r_i's motivation $m_{ij}(t)$ to perform a behavior set a_{ij} increases at a fast rate of impatience as long as the task corresponding to that behavior set (i.e. $h_i(a_{ij})$) is not being accomplished, as determined from sensory feedback. For example, in the hazardous waste cleanup mission a robot with the ability to move toxic waste should have an increasing motivation to move that waste as long as the waste remains at the initial spill location. On the other hand, if the sensory feedback indicates that the behavior set is not applicable, then the motivation to perform that behavior set should go to zero. Thus, the waste-moving robot should not be motivated to move waste if it has already been moved, as evidenced through the robot's sensors.

Additionally, the robots should be responsive to the actions of other robots, adapting their task selection to the activities of team members. Thus, if a robot r_i is aware that another robot r_k is working on task $h_i(a_{ij})$, r_i should be satisfied for some period of time that that task is going to be accomplished even without its own participation in the task, and thus go on to some other applicable action. Robot r_i's motivation to activate behavior set a_{ij} continues to increase, but at a slower rate. This characteristic prevents robots from replicating each other's actions and thus wasting needless energy. Of course, detecting and interpreting the actions of other robots is not a trivial problem, and often requires perceptual abilities that are not yet possible with current sensing technology (see section 5.1). As it stands today, the sensory capabilities of even the lower animals far exceed present robotic capabilities. Thus, to enhance the robots' perceptual abilities, this architecture utilizes a simple form of broadcast communication to allow robots to inform other team members of their current activities, rather than relying totally on sensing through the world. At some pre-specified rate, each robot r_i broadcasts a statement of its current action, which other robots may listen to or ignore as they wish. No two-way conversations are employed in this architecture.

Each robot is designed to be somewhat impatient, however, in that a robot r_i is only willing for a certain period of time to allow the communicated messages of another robot to affect its own motivation to activate a given behavior set. Continued sensory feedback indicating that a task is not getting accomplished thus overrides the statements of another robot that it is performing that task. This characteristic allows robots to adapt to failures of other robots, causing them to ignore the activities of a robot that is not successfully completing its task. As an example of this feature, consider two robots, RED and BLUE , that want to locate a hazardous waste spill in a room. RED responds first to the need to locate the spill and begins broadcasting its actions of spill localization. BLUE is then satisfied that the spill will be located and sits patiently waiting on that task to be completed. However, assume that RED experiences a mechanical failure that prevents it from successfully locating the spill. In the meantime, BLUE's impatience to locate the spill has been increasing because its sensory feedback indicates the task still needs to be performed. Initially, this impatience increased at a slow rate due to the communication from RED. After a period of time, however, the impatience begins to increase at a faster rate because RED does not demonstrate its ability to accomplish that task. Eventually BLUE's motivation to activate its *find-spill* behavior set passes the threshold of activation and BLUE proceeds to locate the spill itself. In this manner, impatience helps robots to adapt to dynamic changes in the environment or in the actions or failures of other robots.

A complementary characteristic in these robots is that of *acquiescence*. Just as the impatience characteristic reflects the fact that other robots may fail, the acquiescence characteristic indicates the recognition that a robot itself may fail. This feature operates as follows. As a robot r_i performs a task, its willingness to give up that task increases over time as long as the sensory feedback indicates the task is not being accomplished. As soon as some other robot r_k indicates it has begun that same task and r_i feels it (i.e r_i) has attempted the task for an adequate period of time, the unsuccessful robot r_i gives up its task in an attempt to find an action at which it is more productive. However, even if another robot r_k has not taken over the task, robot r_i may give up its task anyway if the task is not completed in an acceptable period of time. This allows r_i the possibility of working on another task that may prove to be more productive rather than becoming stuck performing the unproductive task forever.

With this acquiescence characteristic, therefore, a robot is able to adapt its actions to its own failures. This is illustrated by continuing the previous example — once BLUE announces its activity to localize the spill, RED acquiesces that task to BLUE if RED has attempted the task for a sufficient length of time without success. Once again, this prevents the replication of effort that leads to the use of excessive energy.

The design of the motivational behaviors also allows the robots to adapt to unexpected environmental changes which alter the sensory feedback. The need for additional tasks can suddenly occur, requiring the robots to perform additional work, or existing environmental conditions can disappear and thus relieve the robots of certain tasks. In either case, the motivations fluidly adapt to these situations, causing robots to respond appropriately to the current environmental circumstances.

3.6 Discussion of Formal Model of ALLIANCE

Now that the basic philosophy behind the ALLIANCE architecture has been presented, let us look in detail at how this philosophy is incorporated into the motivational behavior mechanism by examining a formal model of the motivational behavior. As these details are discussed, a number of parameters are introduced that are included in the formal model of ALLIANCE. For now, however, the question of determining the proper parameter settings is deferred so that I can convey the general framework within which cooperative control is achieved. I return to the important issue of determining these proper parameter settings in chapter 4, which discusses the learning enhancement to ALLIANCE. I also note that all of the implementations of this model have been accomplished using features of the Behavior Language [Brooks, 1990a], in both the physical robot experiments (hazardous waste cleanup mission and box-pushing demonstration) and in the simulated robot experiments (janitorial service mission and bounding overwatch mission).

In this section I first discuss the threshold of activation of a behavior set, and then describe the five primary inputs to the motivational behavior. In this discussion, it is helpful to keep in mind the definition of the $h_i(a_{ij})$ functions defined in section 3.3. I conclude this section by showing how these inputs are combined to determine the current level of motivation of a given behavior set in a given robot. Refer to appendix A for a summary of the ALLIANCE formal model[3].

Threshold of activation

The threshold of activation is given by one parameter, θ. This parameter determines the level of motivation beyond which a given behavior set will become active. Although different thresholds of activation could be used for different behavior sets and for different robots, in ALLIANCE I assume that one threshold is sufficient since the rates of impatience and acquiescence can vary across behavior sets and across robots.

[3]The model described in this section and in appendix A is an updated version of that provided in [Parker, 1992].

Sensory feedback

The sensory feedback provides the motivational behavior with the information necessary to determine whether its corresponding behavior set needs to be activated at a given point during the current mission. Although this sensory feedback usually comes from physical robot sensors, in realistic robot applications it is not always possible to have a robot sense the applicability of tasks through the world — that is, through its sensors. Often, tasks are information–gathering types of activities whose need is indicated by the values of programmed state variables. For example, in the hazardous waste cleanup mission, the robots are first required to find the location of the spill. However, they do not have a physical sensor that provides binary "found/not found" feedback to indicate whether the spill has been located. Instead, the robot maintains a memory flag which indicates the need to find the spill. This value is updated by the robot when it finds the spill, or when a communicated message from another robot announces the spill location. The value of the memory location, therefore, serves as a type of *virtual* sensor which serves some of the same purposes as a physical sensor.

At times, it is quite possible that the sensory feedback provides erroneous information to the robot. This erroneous information can lead the robot to assume that a task needs to be executed when, in fact, it does not (false positive), or that a task does *not* need to be performed when, in fact, it does (false negative). Although higher task coverages and redundancy in individual robot sensors can help reduce this problem, at some point the levels of redundancy become exhausted, leading to robot failure. Thus, sensory failures as well as effector errors can lead to the team's failure to accomplish its mission.

I define a simple function to capture the notion of sensory feedback as follows:

$$sensory\_feedback_{ij}(t) = \begin{cases} 1 & \text{if the sensory feedback in robot } r_i \text{ at time } t \\ & \quad \text{indicates that behavior set } a_{ij} \text{ is applicable} \\ 0 & \text{otherwise} \end{cases}$$

Note that this use of sensory feedback serves the same purpose as "precondition lists" in traditional planning systems, such as STRIPS [Fikes and Nilsson, 1971], or in situated agent planning systems, such as Maes' spreading activation networks [Maes, 1989]. In these planning systems, the precondition lists are collections of symbolic state descriptions that must hold true before a given action can be performed. One could impose a similar symbolic description on the required sensory feedback of each motivational behavior in ALLIANCE to make the environmental requirements of behavior set activation more explicit.

Inter-robot communication

The inter-robot broadcast communication mechanism utilized in ALLIANCE serves a key role in allowing robots to determine the current actions of their teammates. As I noted previously, the broadcast messages in ALLIANCE substitute for more complex passive action interpretation, or action recognition, which is quite difficult to achieve (see section 5.1).

Two parameters are utilized in ALLIANCE to control the broadcast communication among robots: ρ_i and τ_i. The first parameter, ρ_i, gives the rate at which robot r_i broadcasts its current activity. The second parameter, τ_i, provides an additional level of fault tolerance by giving the period of time robot r_i allows to pass without receiving a communication message from a specific teammate before deciding that that teammate has ceased to function. While monitoring the communication messages, each motivational behavior a_{ij} of robot r_i must also note when a team member is pursuing task $h_i(a_{ij})$. For example, in the hazardous waste cleanup task, the motivational behavior controlling the *report-progress* behavior set remembers the identification of any other robot that is currently pursuing the task of reporting the team's progress. However, this motivational behavior ignores robot messages concerning any other task, since those tasks are controlled by other motivational behaviors.

To refer to this type of monitoring in the formal model, the function *comm_received* is defined as follows:

$$comm\_received(i,k,j,t_1,t_2) = \begin{cases} 1 & \text{if robot } r_i \text{ has received message from} \\ & \quad \text{robot } r_k \text{ concerning task } h_i(a_{ij}) \text{ in the} \\ & \quad \text{time span } (t_1,t_2) \text{, where } t_1 < t_2 \\ 0 & \text{otherwise} \end{cases}$$

Suppression from active behavior sets

When a motivational behavior activates its behavior set, it simultaneously begins inhibiting other motivational behaviors within the same robot from activating their respective behavior sets. At this point, a robot has effectively "selected an action". The first motivational behavior then continues to monitor the sensory feedback, the communication from other robots, and the levels of impatience and acquiescence to determine the continued need for the activated behavior set. At some point in time, either the robot completes its task, thus causing the sensory feedback to no longer indicate the need for that behavior set, or the robot acquiesces the task either to another robot or because the robot is giving up on itself. In either case, the need for this behavior set eventually goes away, causing the corresponding motivational behavior to inactivate this behavior set. This, in turn, allows another motivational behavior within that robot the opportunity to activate its behavior set.

One additional detail has to be handled here to avoid problems when two or more motivational behaviors share exactly the same rate of impatience and which activate at the same instant. Although this situation is unlikely, if it ever occurs it can lead to the robot thrashing between the state in which multiple behavior sets are active and the idle state[4]. To remedy this potential problem, a fixed priority among behavior sets is established, with the higher-priority behavior set "winning" in the case of simultaneous behavior set activations. This is implemented by treating the suppression across motivational behaviors as a message that includes the priority of the suppressing behavior set. If a motivational behavior receives a suppression message after it has activated its behavior set, it compares its own priority with that of the suppression message. If it discovers that its own priority is lower than that indicated in the suppression message, it deactivates its behavior set. In the formal model, however, I ignore this detail and simply refer to the cross-behavior set suppression with the following function:

$$activity\_suppression_{ij}(t) = \begin{cases} 0 & \text{if another behavior set } a_{ik} \text{ is active, } k \neq j \text{, on} \\ & \quad \text{robot } r_i \text{ at time } t \\ 1 & \text{otherwise} \end{cases}$$

This function says that behavior set a_{ij} is being suppressed at time t on robot r_i if some other behavior set a_{ik} is currently active on robot r_i at time t.

Robot impatience

Three parameters are used to implement the robot impatience feature of ALLIANCE: $\phi_{ij}(k,t)$, $\delta\_slow_{ij}(k,t)$, and $\delta\_fast_{ij}(t)$. The first parameter, $\phi_{ij}(k,t)$, gives the time during which robot r_i is willing to allow robot r_k's communication message to affect the motivation of behavior set a_{ij}. Note that robot r_i is allowed to have different ϕ parameters for each robot, r_k, on its team, and that these parameters can change during the mission (indicated by the dependence on t). This allows r_i to be influenced more by some robots than others, perhaps due to reliability differences across robots.

The next two parameters, $\delta\_slow_{ij}(k,t)$ and $\delta\_fast_{ij}(t)$, give the rates of impatience of robot r_i concerning behavior set a_{ij} either while robot r_k is performing the task corresponding to behavior set a_{ij} (i.e. $h_i(a_{ij})$) or in the absence of other robots performing the task $h_i(a_{ij})$, respectively. I assume that the fast impatience parameter corresponds to a higher rate of impatience than the slow impatience parameter for a given behavior set in a given robot. The reasoning for this assumption should be clear — a robot r_i should allow another robot r_k the opportunity to accomplish its

[4]The robot returns to the idle state after multiple simultaneous behavior set activations because all the active behavior sets send suppression messages, thus causing all the behavior sets to be deactivated.

task before becoming impatient with r_k; however, there is no reason for r_i to remain idle if a task remains undone and no other robot is attempting that task.

The question that now arises is the following: what slow rate of impatience does a motivational behavior controlling behavior set a_{ij} use when more than one other robot is performing task $h_i(a_{ij})$? The method used in ALLIANCE is to increase the motivation at a rate that allows the slowest robot r_k still under its allowable time $\phi_{ij}(k,t)$ to continue its attempt. This seems reasonable, since r_i would have allowed this slowest robot the time $\phi_{ij}(k,t)$ to perform the task if it were the only robot performing task $h_i(a_{ij})$.

The specification of when the impatience rate for a behavior set a_{ij} should grow according to the slow impatience rate and when it should grow according to the fast impatience rate is given by the following function:

$$impatience_{ij}(t) = \begin{cases} \min_k(\delta\_slow_{ij}(k,t)) & \text{if } (comm\_received(i,k,j,t-\tau_i,t)=1) \\ & \quad \text{and} \\ & \quad (comm\_received(i,k,j,0,t-\phi_{ij}(k,t))=0) \\ \delta\_fast_{ij}(t) & \text{otherwise} \end{cases}$$

Thus, the impatience rate will be the minimum slow rate, $\delta\_slow_{ij}(k,t)$, if robot r_i has received communication indicating that robot r_k is performing the task $h_i(a_{ij})$ in the last τ_i time units, but not for longer than $\phi_{ij}(k,t)$ time units. Otherwise, the impatience rate is set to $\delta\_fast_{ij}(t)$.

The final detail to be addressed is to cause a robot's motivation to activate behavior set a_{ij} to go to 0 the first time it hears about another robot performing task $h_i(a_{ij})$. This is accomplished through the following:

$$impatience\_reset_{ij}(t) = \begin{cases} 0 & \text{if } \exists k.((comm\_received(i,k,j,t-\delta t,t)=1) \text{ and} \\ & \quad (comm\_received(i,k,j,0,t-\delta t)=0)), \\ & \quad \text{where } \delta t = \text{time since last communication check} \\ 1 & \text{otherwise} \end{cases}$$

This reset function causes the motivation to be reset to 0 if robot r_i has just received its first message from robot r_k indicating that r_k is performing task $h_i(a_{ij})$. This function allows the motivation to be reset no more than once for every robot team member that attempts task $h_i(a_{ij})$. Allowing the motivation to be reset repeatedly by the same robot would allow a persistent, yet failing robot to jeopardize the completion of the mission.

Robot acquiescence

Two parameters are used to implement the robot acquiescence characteristic of ALLIANCE: $\psi_{ij}(t)$ and $\lambda_{ij}(t)$. The first parameter, $\psi_{ij}(t)$, gives the time that robot r_i wants to maintain behavior set a_{ij} activation before yielding to another robot. The second parameter, $\lambda_{ij}(t)$, gives the time robot r_i wants to maintain behavior set a_{ij} activation before giving up to possibly try another behavior set.

The following *acquiescence* function indicates when a robot has decided to acquiesce its task:

$$acquiescence_{ij}(t) = \begin{cases} 0 & \text{if [(behavior set } a_{ij} \text{ of robot } r_i \text{ has been active for more} \\ & \quad \text{than } \psi_{ij}(t) \text{ time units at time } t\text{) and} \\ & \quad (\exists x.comm\_received(i, x, j, t-\tau_i, t) = 1)] \\ & \text{or} \\ & \quad \text{(behavior set } a_{ij} \text{ of robot } r_i \text{ has been active for more} \\ & \quad \text{than } \lambda_{ij}(t) \text{ time units at time } t\text{)} \\ 1 & \text{otherwise} \end{cases}$$

This function says that a robot r_i will not acquiesce behavior set a_{ij} until one of the following conditions is met:

- r_i has worked on task $h_i(a_{ij})$ for a length of time $\psi_{ij}(t)$ and some other robot has taken over task $h_i(a_{ij})$
- r_i has worked on task $h_i(a_{ij})$ for a length of time $\lambda_{ij}(t)$

Motivation calculation

All of the inputs described above are now combined into the calculation of the levels of motivation as follows:

$$\begin{aligned} m_{ij}(0) &= 0 \\ m_{ij}(t) &= [m_{ij}(t-1) + impatience_{ij}(t)] \\ &\quad \times sensory\_feedback_{ij}(t) \\ &\quad \times activity\_suppression_{ij}(t) \\ &\quad \times impatience\_reset_{ij}(t) \qquad (3.1) \\ &\quad \times acquiescence_{ij}(t) \qquad (3.2) \end{aligned}$$

Initially, the motivation to perform behavior set a_{ij} in robot r_i is set to 0. This motivation then increases at some positive rate $impatience_{ij}(t)$ unless one of four situations occurs: (1) the sensory feedback indicates that the behavior set is no longer needed, (2) another behavior set in r_i activates, (3) some other robot has just taken over task $h_i(a_{ij})$ for the first time, or (4) the robot has decided to acquiesce the task. In any of these four situations, the motivation returns to 0. Otherwise, the motivation grows until it crosses the threshold θ, at which time the behavior set is activated and the robot can be said to have selected an action. Whenever some behavior set a_{ij} is active in robot r_i, r_i broadcasts its current activity to other robots at a rate of ρ_i.

3.7 Proofs of Termination

When evaluating a control architecture for multi-robot cooperation, it is important to be able to predict the team's expected performance using that architecture in a wide variety of situations. One should be justifiably wary of using an architecture that can fail catastrophically in some situations, even though it performs fairly well on average. At the heart of the problem is the issue of reliability — how dependable the system is, and whether it functions properly each time it is utilized. As I noted in chapter 1, to properly analyze a cooperative robot architecture we should separate the architecture itself from the robots on which the architecture is implemented. Even though individual robots on a team may be quite unreliable, a well-designed cooperative architecture could actually be implemented on that team to allow the robots to very reliably accomplish their mission, given a sufficient degree of task coverage. On the other hand, an architecture should not be penalized for a team's failure to accomplish its mission even though the architecture has been implemented on extremely reliable robots, if those robots do not provide the minimally acceptable level of task coverage. A major difficulty, of course, is defining reasonable evaluation criteria and evaluation assumptions by which an architecture can be judged. Certain characteristics of an architecture that extend its application domain in some directions may actually reduce its effectiveness for other types of applications. Thus, the architecture must be judged according to its application niche, and how well it performs in that context.

ALLIANCE is designed for applications involving a significant amount of uncertainty in the capabilities of robot team members which themselves operate in dynamic, unpredictable environments. Within this context, the first key point of interest is whether the architecture allows the team to complete its mission at all, even in the presence of robot difficulties and failure. This section examines this issue by first evaluating the performance of ALLIANCE in the most tractable domain — teams composed of reliable, aware robots — and then considering the more complex (and more realistic) domain — teams of robots with limited reliability in a dynamic environment.

3.7.1 Reliable, Aware Robot Teams

Let us first consider the situation in which the team is composed of reliable, aware robots. Clearly, one would like the architecture to be able to perform well in this simplest of domains, since we prefer that performance gains in challenging environments not preclude acceptable performances in the tamer applications. By the term "reliable robot", I mean that if a robot r_i is designed to accomplish a task $h_i(a_{ij})$ when it activates behavior set a_{ij}, then it really can accomplish that task, with probability equal to 1. Formally, a reliable robot is defined as follows, using the notation of section 3.3:

Definition 3 *A* reliable robot*, r_i, is a robot for which both of the following conditions hold:*

- $\forall(a_{ij} \in GRC_i) \quad . \quad p(r_i \textit{ achieves task } h_i(a_{ij}) \textit{ in finite time when } a_{ij} \textit{ is activated }) = 1,$

  where p stands for the probability function.

- *The sensory feedback of r_i provides an accurate view of the state of world and the state of the tasks, T, required by the current mission.*

This definition says that the outcome of activating a goal-relevant behavior set a_{ij} from any world state is that the task $h_i(a_{ij})$ is successfully accomplished. Note that in making this definition, I imply that nothing can happen in a reliable robot's environment that causes the robot to fail at its task; otherwise the robot would not satisfy the definition of *reliable*.

I now define the notion of a reliable robot team:

Definition 4 *A* reliable robot team *is a group of robots, R, for which the following condition holds:*

$$\forall(r_i \in R).r_i \textit{ is a } \text{reliable robot}$$

I also define the notion of an *aware* robot team as follows:

Definition 5 *An* aware robot team *is a group of robots, R, in which all members, r_i, know what the current actions of all robots $r_j \in R$ are, at all times t.*

Although this awareness is assumed to come from the ALLIANCE broadcast communication mechanism that allows each robot to inform its teammates of its own current action, any method of achieving awareness (via action recognition, for instance) is sufficient.

Additionally, the robots are useful only if they can be motivated to perform some action. Thus, I define the concept of an *active* robot team:

Definition 6 *An* active *robot team is a group of robots, R, such that:*

$$\forall(r_i \in R).\forall(a_{ij} \in GRC_i).\forall(r_k \in R).\forall t. \\ [(\delta\_slow_{ij}(k,t) > 0) \bigwedge (\delta\_fast_{ij}(t) > 0) \bigwedge (\theta \textit{ is finite})]$$

In other words, an *active* robot has a monotonically increasing motivation to perform any task of the mission which that robot has the ability to accomplish. Additionally, the threshold of activation of all behavior sets of an *active* robot is finite.

Finally, I define the notion of an *idle* robot:

Definition 7 *An* idle *robot is a robot, r_i, which is* active *at time t, but for which the following condition is true:*

$$\forall j. m_{ij}(t) < \theta$$

In other words, all of the motivations of an *idle* robot are currently below the threshold of activation.

I now define two conditions that are useful in studying these robot teams.

Condition 1 (Sufficient task coverage):

$$\forall(task_k \in T).(task\_coverage(task_k)) \geq 1$$

This condition ensures that all tasks required by the mission should be able to be accomplished by some robot on the team.

Secondly, since reliable robots are assumed to never fail, the robots can be designed to be maximally patient and minimally acquiescent. In terms of the formal problem model, we have:

Condition 2 (Maximal patience, minimal acquiescence):

$$\forall(r_i \in R).\forall(a_{ij} \in GRC_i).\forall(r_k \in R).\forall t.[(\phi_{ij}(k,t) = \infty) \bigwedge (\psi_{ij}(t) = \infty) \bigwedge (\lambda_{ij}(t) = \infty)]$$

The following can now be proved about reliable, aware robot teams:

Theorem 1 *Let R be a* reliable, aware, active robot team*, and M be the mission to be solved by R, such that Conditions 1 and 2 hold. Then the probability that ALLIANCE enables R to accomplish M equals 1.*

Proof: Let us examine the calculation of the motivation of robot r_i to perform behavior set a_{ij} (equation 3.2 from section 3.6). At time t, the motivation $m_{ij}(t)$ to perform behavior set a_{ij} either (1) goes to 0, or (2) changes from $m_{ij}(t-1)$ by the amount $impatience_{ij}(t)$.

The motivation goes to 0 in one of four cases: (1) if the sensory feedback indicates that the behavior set is no longer applicable, (2) if another behavior set becomes active, (3) if some other robot has taken over task $h_i(a_{ij})$, or (4) if the robot has acquiesced its task. Since a *reliable* robot's sensory feedback is assumed to be accurate, then if the sensory feedback indicates that the behavior set is no longer applicable, we know that the task $h_i(a_{ij})$ must be successfully accomplished. If another behavior set a_{ik} becomes active in r_i, we know from the definition of a reliable robot that r_i will complete behavior set a_{ik} in finite time, thus allowing r_i to activate behavior set a_{ij}. If some other robot has taken over task $h_i(a_{ij})$, then since that robot is a reliable robot, we know that it will eventually accomplish task $h_i(a_{ij})$, thus eliminating the need to activate task a_{ij}. Since I assume that Condition 2 holds, then the robot will not be acquiescing the task. Therefore, all the factors causing the motivation to perform behavior set a_{ij} to go to 0 will, at some point, be eliminated unless the behavior set is no longer necessary.

Since r_i is *active*, then we know that $impatience_{ij}(t)$ is greater than or equal to $\min_k(\delta\_slow_{ij}(k,t))$, which is greater than 0. Therefore, we can conclude that an *idle*, yet *active* robot always has a strictly increasing motivation to perform some incomplete task. At some point, the finite threshold of activation, θ, will be surpassed for some behavior set, causing r_i to activate that behavior set, which will then lead the robot to accomplish the task $h_i(a_{ij})$. Finally, since we assume that R is *aware* and that Condition 2 holds, then once a robot r_i has begun execution of a task, no other robot will attempt to interfere with its execution of that task.

Since Condition 1 holds, sufficient robots are present to complete all tasks required by the mission, and thus all tasks will eventually be successfully completed. $\square$

3.7.2 Limitedly-Reliable Robot Teams

Now let us consider teams of robots that are not always able to successfully accomplish their tasks; I use the term *limitedly-reliable* robot to refer to these robots. A limitedly-reliable robot is simply any robot team member, whether or not that robot meets the definition of *reliable* from the previous section. This situation is much more realistic than the earlier assumption of completely reliable robot teams; in fact, this situation is a primary assumption upon which ALLIANCE is designed. However, the uncertainty in the expected effect of robots' actions clearly makes the control problem much more difficult. Ideally, ALLIANCE's impatience and acquiescence factors will allow a robot team to successfully reallocate actions as robot failures or dynamic changes in the environment occur. With what confidence can we know that this will happen in general? As we shall see below, in many situations ALLIANCE is guaranteed to allow a limitedly-reliable robot team to successfully accomplish its mission.

Obviously, a robot's selection of actions is very much dependent upon the parameter settings of the motivational behaviors — particularly, the settings of $\phi_{ij}(k,t)$ (time before impatient), $\psi_{ij}(t)$ (time before acquiescing to another robot), and $\lambda_{ij}(t)$ (time before giving up current task). If these parameter are set to very small values, the robots tend to "thrash" back and forth between tasks, exhibiting very short attention spans. If the parameters are set to very large values, then the robots can be viewed either as showing remarkable perseverance, or as wasting incredible amounts of time.

In practice, finding the proper parameter settings is not difficult. The ALLIANCE architecture has been implemented on a number of quite different robotic applications, reported primarily in this chapter and in chapter 6, and parameter tuning did not prove to be a problem. Clearly, however, some attention should be paid to the parameters, as they do have a significant influence on the action selection of the robots. Ideally, the robots on the cooperative team should be able to adapt these values with experience to find the right parameter settings that moderate between the two extremes, rather than relying on human tuning. The learning system L-ALLIANCE described in chapter 4 provides mechanisms that allow the robots to obtain the proper parameter settings; thus, I do not dwell on this issue here.

However, it is interesting to note that in certain restricted circumstances the exact values to which the ALLIANCE parameters are set does not affect the ability of the team to complete its mission. I describe these circumstances here, and prove that in these situations the robot team is guaranteed to be able to complete its mission under the ALLIANCE architecture.

Let us now define a condition that holds in some multi-robotic applications.

Condition 3 (Progress when Working):
Let z be the finite amount of work remaining to complete a task w. Then whenever robot r_i activates a behavior set corresponding to task w, either (1) r_i remains active for a sufficient, finite length of time ϵ such that z is reduced by a finite amount which is at least some constant δ greater than 0, or (2) r_i experiences a failure with respect to task w. Additionally, if z ever increases, the increase is due to an influence external to the robot team.

Condition 3 ensures that even if robots do not carry a task through to completion before acquiescing, they still make some progress toward completing that task whenever the corresponding behavior set is activated for some time period at least equal to ϵ. One exception, however, is if a robot failure has occurred that prevents robot r_i from accomplishing task w, even if r_i has been designed to achieve task w. For example, in the hazardous waste cleanup mission, all the robots on the team are designed to have the ability to move spill objects from the start location to the desired final spill location. However, if a robot's gripper breaks it is no longer able to complete this task for which it has been designed, and thus a robot failure with respect to moving the spill objects has occurred. The effect of this failure on the remaining capabilities of that robot depends upon the settings of the impatience and acquiescence parameters, and on the task coverage. If another robot is present that is able to move the spill and has not experienced a failure, then that task will be accomplished in spite of r_i's failure. On the other hand, if r_i's impatience to perform the *move-spill* task is higher than its impatience to perform another task, and no other robot takes over the *move-spill* task, then r_i will be stuck forever trying to accomplish the *move-spill* task. This in turn causes other tasks that r_i could perform to remain incomplete. However, a careful method of updating the impatience and acquiescence parameters can circumvent this problem; chapter 4 describes such a method used in L-ALLIANCE.

Condition 3 also implies that if more than one robot is attempting to perform the same task at the same time, the robots do not interfere with each others' progress so badly that no progress towards completion of the task is made. The rate of progress may be slowed somewhat, or even considerably, but some progress is made nevertheless.

Finally, Condition 3 implies that the amount of work required to complete the mission never increases as a result of robot actions. Thus, even though robots may not be any help towards completing the mission, at least they are not making matters worse. Although this may not always hold true,[5] in general, this is a valid assumption to make. As we shall see, this assumption is necessary to prove the effectiveness of ALLIANCE in certain situations. Of course, this does not preclude dynamic environmental changes from increasing the workload of the robot team, which ALLIANCE allows the robots to handle without problem.

What I now show is that whenever conditions 1 and 3 hold for a limitedly-reliable robot team, then either ALLIANCE allows the robot team to accomplish its mission, or some robot failure occurs. Furthermore, if a robot failure occurs, then we can know that any task that remains incomplete at the end of the mission is either a task that the failed robot was designed to accomplish, or a task that is dependent upon the capabilities of that robot. As an example of the difference between these types of potentially incomplete tasks, consider a robot team performing the hazardous waste

[5]For instance, a robot could go haywire and start scattering toxic waste already moved to the goal destination all over the room, making it difficult, if not impossible, for the remaining robots to repair the damage.

cleanup mission that consists of five robots that can move the spill and report the progress, and one robot, r_i, that can only find the initial spill location. This team does have satisfactory task coverage to accomplish its mission. However, if r_i fails, then not only is r_i's task (finding the spill) incomplete at the end of the mission, but so are the tasks of moving the spill and reporting the progress, since they depend upon the spill to be located before they can be activated. Thus, the extent of the tasks that can remain incomplete after a robot failure may be more extensive than just the tasks the failed robot was designed to perform.

I can now show the following:

Theorem 2 *Let R be a* limitedly-reliable, active *robot team, and M be the mission to be solved by R, such that Conditions 1 and 3 hold. Then either (1) ALLIANCE enables R to accomplish M, or (2) a robot failure occurs. Further, if robot r_f fails, then the only tasks of M that are not completed are some subset of (a) the set of tasks r_f was designed to accomplish, unioned with (b) the set of tasks dependent upon the capabilities of r_f.*

Proof: The proof of theorem 1 showed us how the calculation of the motivational behavior guarantees that each robot eventually activates a behavior set whose sensory feedback indicates that the corresponding task is incomplete. Thus, I can focus here on the higher-level robot interactions to prove that either the mission becomes accomplished, or a robot failure occurs.

PART I (Either ALLIANCE succeeds or a robot fails):
Assume no robot fails. Then after a robot r_i has performed a task w for any period of time greater than ϵ, one of five events can occur:

1. Robot r_j takes over task w, leading robot r_i to acquiesce.
2. Robot r_i gives up on itself and acquiesces w.
3. Robot r_j takes over task w, but r_i does not acquiesce.
4. Robot r_i continues w.
5. Robot r_i completes w.

Since Condition 3 holds, we know that the first four cases reduce the amount of work left to complete task w by at least a positive, constant amount δ. Since the amount of work left to accomplish any task is finite, the task must eventually be completed in finite time. In the fifth case, since task w is completed, the sensory feedback of the robots no longer indicates the need to perform task w, and thus the robots will go on to some other task required by the mission.

Thus, for every task that remains to be accomplished, either (1) a robot able to accomplish that task eventually attempts the task enough times so that it becomes complete, or (2) all robots designed to accomplish that task have failed.

PART II (Incomplete tasks are dependent upon a failed robot's capabilities):
Let F be the set of robots that fail during a mission, and A_F be the union of (a) the tasks that the robots in F were designed to accomplish and (b) those tasks of the mission that are dependent upon a task that a robot in F was designed to accomplish.

First, I show that if a task is not in A_F, then it will be successfully completed. Let w be some task required by the mission that is not included in A_F. Since Condition 1 holds and this robot team is active, there must be some robot on the team that can successfully accomplish w. Thus, as long as w remains incomplete, one of these successful robots will eventually activate its behavior set corresponding to the task w; since condition 3 holds, that task will eventually be completed in finite time. Thus, all tasks not dependent upon the capabilities of a failed robot are successfully completed in ALLIANCE.

Now, I show that if a task is not completed, it must be in A_F. Let w be a task that was not successfully completed at the end of the mission. Assume by way of contradiction that w is not in A_F. But we know from Part I that all tasks w not in A_F must be completed. Therefore, task w must be in A_F.

I can thus conclude that if a task is not accomplished, then it must be a task for which all robots with that capability have failed, or which is dependent upon some task for which all robots with that capability have failed. $\square$

Note that it is not required here that robot team members be *aware* of the actions of their teammates in order to guarantee that ALLIANCE allows the team to complete its mission under the above conditions. However, awareness does have an effect on the *quality* of the team's performance, both in terms of the time and the energy required to complete the mission. These effects on team performance are discussed in chapter 5.

3.8 Experimental Results: Hazardous Waste Cleanup

Recall the introductory example from chapter 1, illustrated in figure 1-1. A hazardous waste spill in an enclosed room must be cleaned up by a team of robots. In this mission, the robot team must locate the spill and move it to a safe location, while also periodically reporting its progress to humans monitoring the system by returning a robot representative to the room entrance occasionally to radio the team's current mission completion status. In my experimental setup, the spill consists of 10 spill objects (i.e. small cylindrical "pucks" measuring 2 inches in diameter and 1.5 inches high) at the initial spill location and the room in which the robots work is rectangular, with sides parallel to the axes of the global coordinate system.

Further recall that a difficulty in this mission is that the human monitor does not know the exact location of the spill in robot coordinates, and can only give the robot team qualitative information on the initial location of the spill and the final desired location to which the robots must move the spill. Thus, the robots are told that the initial location is in the center of the front third of the room, and that the desired final location of the spill is in the back, center of the room, relative to the position of the entrance. The robot team's goal is to complete this mission as quickly as possible without needlessly wasting energy.

3.8.1 The Robot Team

I studied this task with two different teams of robots: one team was composed of two R-2 robots, RED and GREEN, while the second team was composed of three R-2 robots, RED, BLUE, and GREEN. As noted earlier, although these robots are of the same type and thus have the potential of maximum redundancy in capabilities, mechanical drift and failures have caused them to have quite different actual abilities. For example, RED has use of its side infrared (IR) sensors which allow it to perform wall-following, whereas the side IR sensors of BLUE and GREEN have become dysfunctional. The L-ALLIANCE learning system described in chapter 4 gives these robots the ability to determine from trial to trial which team member is best suited for which task.

Each robot has been preprogrammed to have the following behavior sets, which correspond to high-level tasks that must be achieved on this mission: *find-initial-final-locations-methodical*, *find-initial-final-locations-wander*, *move-spill*, and *report-progress*. A low-level *avoid-obstacles* behavior is active at all times in these robots except during portions of the *move-spill* task, when it is suppressed to allow the robot to pick up the spill object. The organization of the behavior sets for this mission is shown in figure 3-4.

Two behavior sets are provided which both accomplish the task of finding the initial and final spill locations — *find-initial-final-locations-methodical* and *find-initial-final-locations-wander* — both of which depend upon the workspace being rectangular and on the sides of the room being parallel to the axes of the global coordinate system. Because of these assumptions, these behavior sets do not serve as generally applicable location-finders. However, I made no attempt to generalize these algorithms, since the point of this experiment is to demonstrate the characteristics of ALLIANCE. Shown in more detail in figure 3-5, the methodical version of finding the spill location

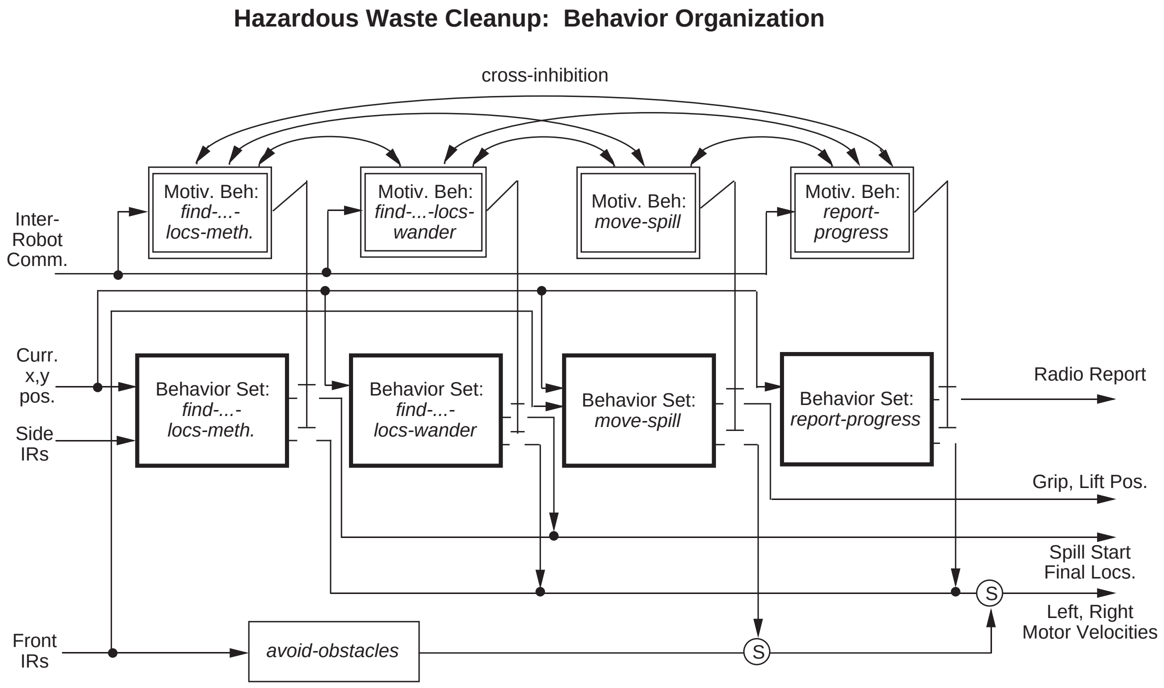
Hazardous Waste Cleanup: Behavior Organization
cross-inhibition
Motiv. Beh: find-...-locs-meth.
Motiv. Beh: find-...-locs-wander
Motiv. Beh: move-spill
Motiv. Beh: report-progress
Inter-Robot Comm.
Curr. x,y pos.
Side IRs
Front IRs
Behavior Set: find-...-locs-meth.
Behavior Set: find-...-locs-wander
Behavior Set: move-spill
Behavior Set: report-progress
avoid-obstacles
S
S
Radio Report
Grip, Lift Pos.
Spill Start Final Locs.
Left, Right Motor Velocities

Figure 3-4: The ALLIANCE-based control of each robot in the hazardous waste cleanup mission. Not all sensory inputs to the behavior sets are shown here. In this figure, the high-level task achieving functions *find-initial-final-locations-methodical* and *find-initial-final-locations-wander* are abbreviated as *find-...-locs-meth* and *find-...-locs-wander*, respectively. Refer to figures 3-5 through 3-8 for more detail.

is much more reliable than the wander version, and involves the robot first noting its starting (or home) x, y position and then following the walls of the room using its side IRs until it has returned to its home location while tracking the minimum and maximum x and y positions it reaches. It then uses these x, y values to calculate the coordinates of the center of the front third of the room (for the initial spill location) and the back center of the room (for the final spill location). These locations are then made available to the *move-spill* behavior set, which requires this information to perform its task.

The wander version of finding the initial and desired final spill locations, shown in figure 3-6, avoids the need for side IR sensors by causing the robot to wander in each of the four directions (west, north, east, and south) for a fixed time period. While the robot wanders, it tracks the minimum and maximum x and y positions it discovers. Upon the conclusion of the wandering phase, the robot calculates the desired initial and final locations from these minimum and maximum x, y values.

The *move-spill* behavior set, shown in more detail in figure 3-7, can be activated whenever there are spill objects needing to be picked up and the locations of the initial and final spill positions are known. It involves having the robot (1) move to the vicinity of the initial spill location, (2) wander in a straight line through the area of the spill while using its front IR sensors to scan for spill objects, (3) "zero in" on a spill object once it is located to center it in the gripper, (4) grasp and lift the spill object, (5) move to the vicinity of the final spill location, and then (6) lower and release the spill object.

The *report-progress* behavior set, shown in figure 3-8, corresponds to the high-level task that the robot team is required to perform about every 4 minutes during the mission. This task involves returning to the room entrance and informing the human monitoring the system of the activities of the robot team members and some information regarding the success of those activities. Note that this task only needs to be performed by the team as a whole every 4 minutes, not by all team members. In a real-life application of this sort, the progress report would most likely be delivered via a radio message to the human. However, in this experiment no actual progress information was maintained (although it could easily be accomplished by logging the robot activities), and delivering the report consisted of playing an audible tune on the robot's piezoelectric buzzer from the room entrance rather than relaying a radio message.

3.8.2 Instantiating the Formal Problem Definition

With the information given thus far in this section, the hazardous waste cleanup mission can be stated in terms of the formal problem definition presented in section 3.3. The set of robots in the two-robot ($n = 2$) version of this mission is $R = \{$RED,

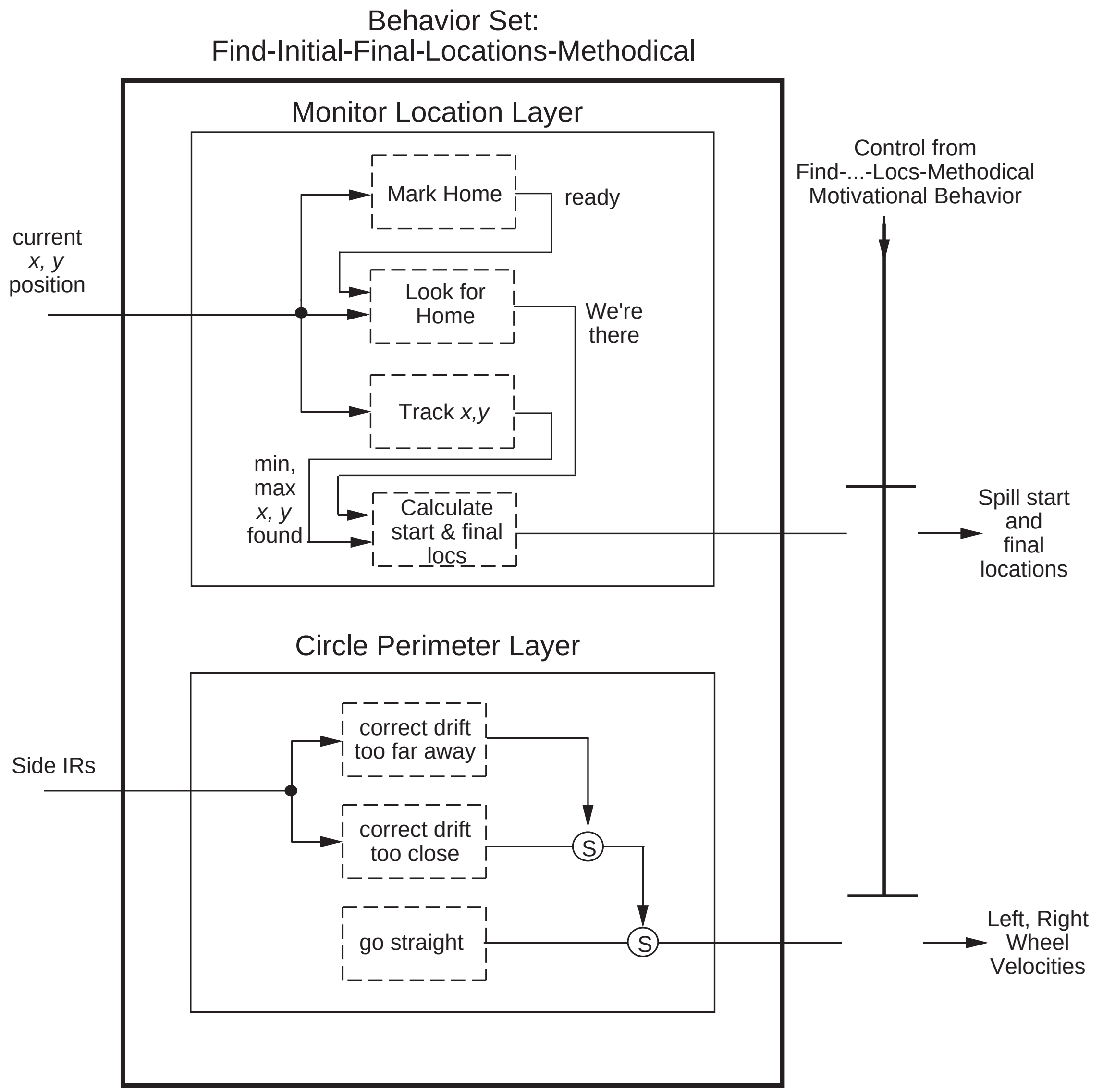
Behavior Set:
Find-Initial-Final-Locations-Methodical
Monitor Location Layer
current *x, y* position
Mark Home
ready
Look for Home
We're there
Track *x,y*
min, max *x, y* found
Calculate start & final locs
Control from Find-...-Locs-Methodical Motivational Behavior
Spill start and final locations
Circle Perimeter Layer
Side IRs
correct drift too far away
correct drift too close
go straight
S
S
Left, Right Wheel Velocities

Figure 3-5: The robot control organization within the *find-initial-final-locations-methodical* behavior set.

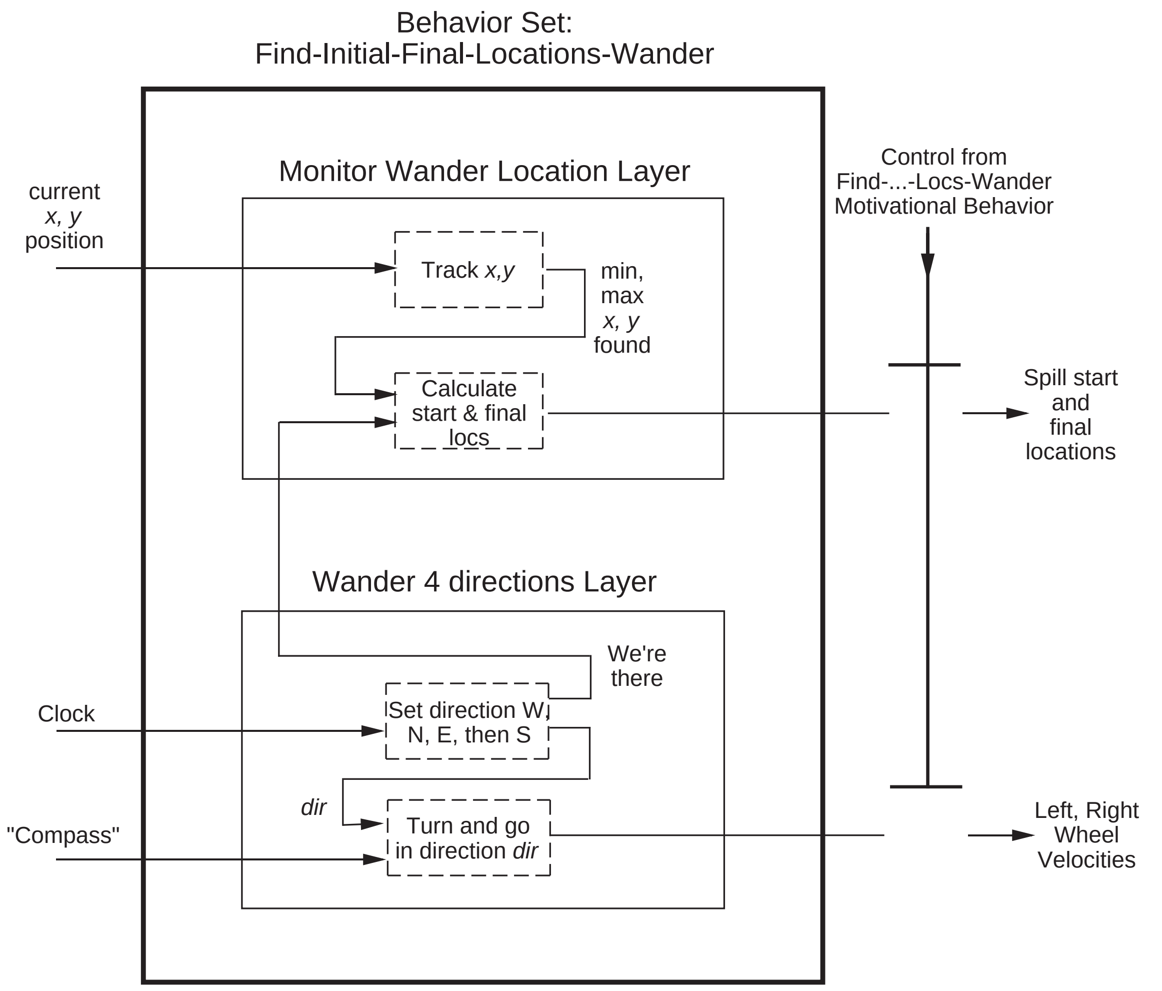
Behavior Set:
Find-Initial-Final-Locations-Wander
Monitor Wander Location Layer
current *x, y* position
Track *x,y*
min, max *x, y* found
Calculate start & final locs
Control from Find-...-Locs-Wander Motivational Behavior
Spill start and final locations
Wander 4 directions Layer
We're there
Clock
Set direction W, N, E, then S
dir
"Compass"
Turn and go in direction *dir*
Left, Right Wheel Velocities

Figure 3-6: The robot control organization within the *find-initial-final-locations-wander* behavior set.

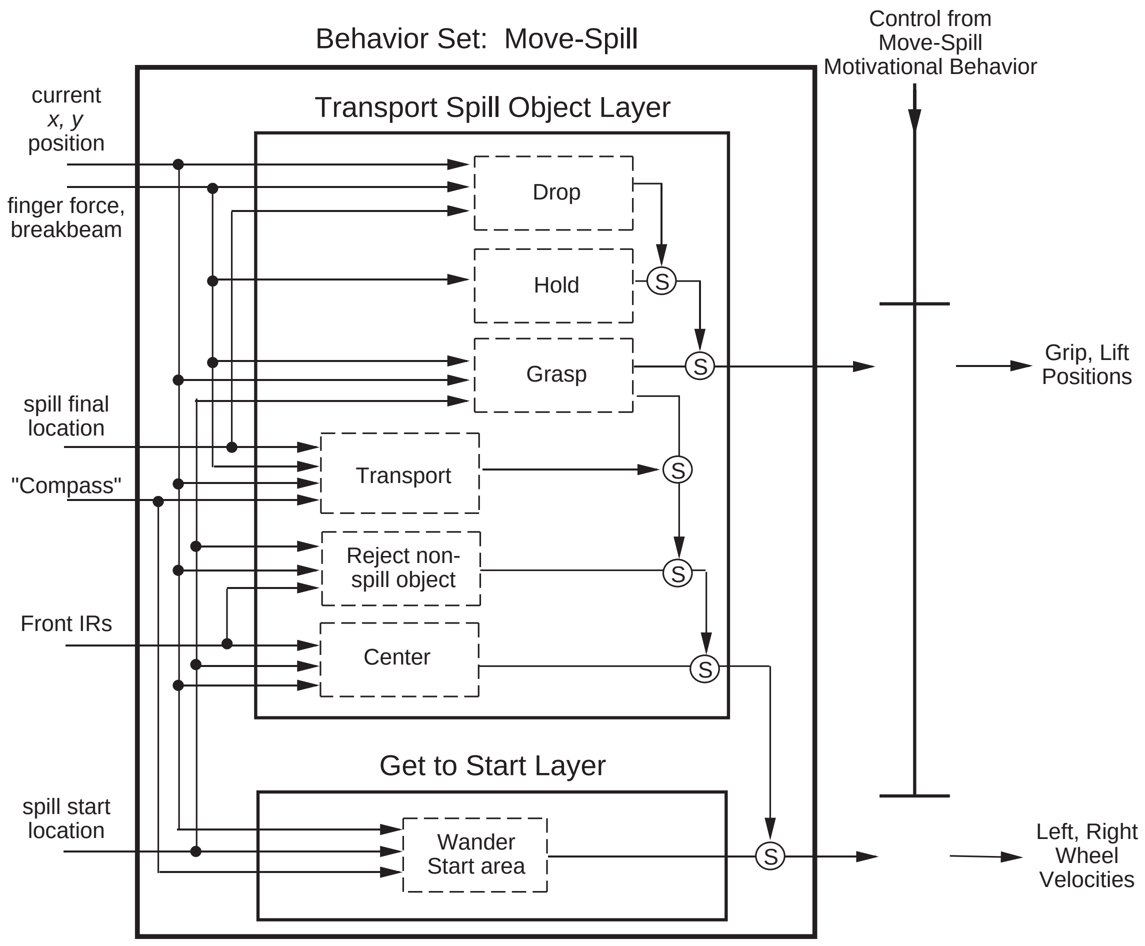

Figure 3-7: The robot control organization within the *move-spill* behavior set.

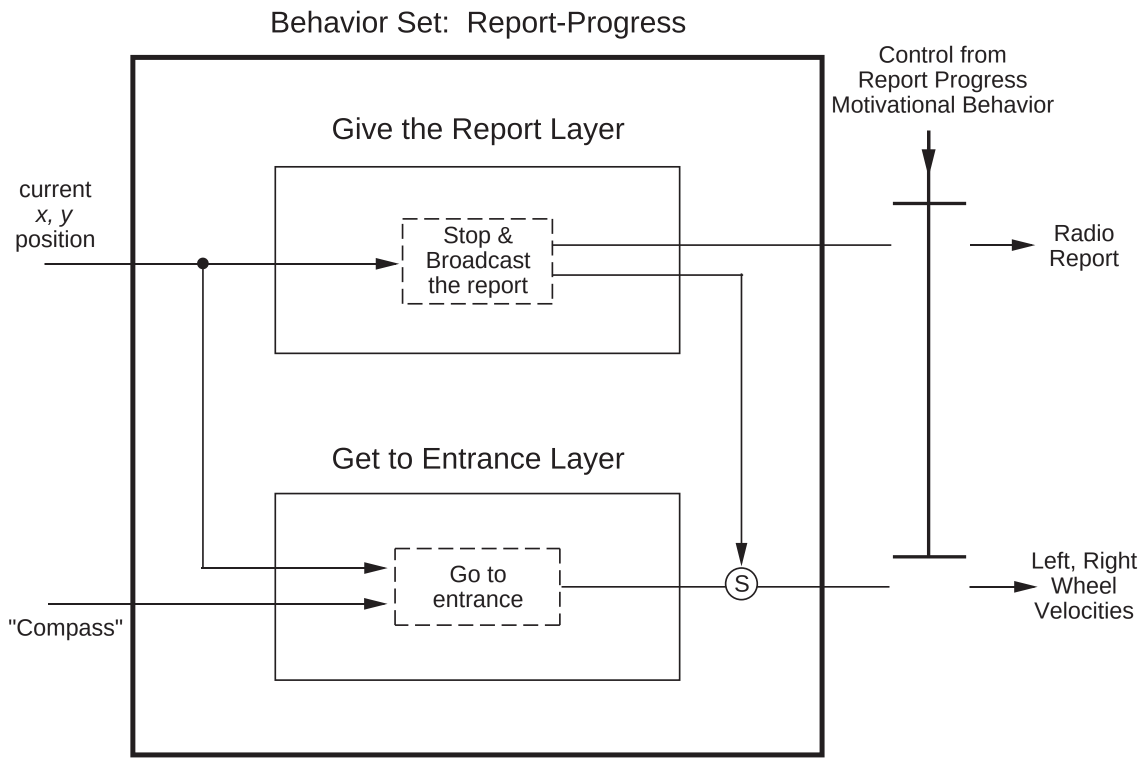
Behavior Set: Report-Progress
Control from Report Progress Motivational Behavior
Give the Report Layer
current *x, y* position
Stop & Broadcast the report
Radio Report
Get to Entrance Layer
Go to entrance
S
"Compass"
Left, Right Wheel Velocities

Figure 3-8: The robot control organization within the *report-progress* behavior set.

GREEN}, while in the three-robot ($n = 3$) version, $R = \{$RED, BLUE, GREEN$\}$. The set of tasks composing the mission, T, is defined as follows:

$$\begin{aligned} T \;=\; & \{\textit{find-initial-final-locations},\ \textit{move-spill-to-final-location}, \\ & \textit{report-progress-every-4-minutes}\} \end{aligned}$$

The sets of high-level task-achieving functions possessed by these robots are identical:

$$\begin{aligned} A_{RED} = A_{BLUE} = A_{GREEN} \;=\; & \{\textit{find-initial-final-locations-methodical}, \\ & \textit{find-initial-final-locations-wander},\ \textit{move-spill}, \\ & \textit{report-progress}\} \end{aligned}$$

Since RED has all the sensors and effectors required to accomplish these functions, its goal-relevant capabilities are identical to its high–level task–achieving functions; that is, $GRC_{RED} = A_{RED}$. However, since BLUE and GREEN's side IRs are broken, we have:

$$\begin{aligned} GRC_{BLUE} = GRC_{GREEN} \;=\; & \{\textit{find-initial-final-locations-wander}, \\ & \textit{move-spill},\ \textit{report-progress}\} \end{aligned}$$

The tasks achieved by the behavior sets in each robot are as expected:

$$\begin{aligned} & h(\text{RED's } \textit{find-initial-final-locations-methodical}) \\ & \qquad = \{\textit{find-initial-final-locations}\} \end{aligned}$$

$$\begin{aligned} & h(\text{RED's } \textit{find-initial-final-locations-wander}) = \\ & h(\text{BLUE's } \textit{find-initial-final-locations-wander}) = \\ & h(\text{GREEN's } \textit{find-initial-final-locations-wander}) \\ & \qquad = \{\textit{find-initial-final-locations}\} \end{aligned}$$

$$\begin{aligned} & h(\text{RED's } \textit{move-spill}) = \\ & h(\text{BLUE's } \textit{move-spill}) = \\ & h(\text{GREEN's } \textit{move-spill}) = \{\textit{move-spill-to-final-location}\} \end{aligned}$$

$$\begin{aligned} & h(\text{RED's } \textit{report-progress}) = \\ & h(\text{BLUE's } \textit{report-progress}) = \\ & h(\text{GREEN's } \textit{report-progress}) \\ & \qquad = \{\textit{report-progress-every-4-minutes}\} \end{aligned}$$

Note that even though the behavior sets *find-initial-final-locations-methodical* and *find-initial-final-locations-wander* are distinct, they accomplish the same task.

For $R = \{\text{RED, GREEN}\}$, the task coverage of this mission is given by:

$$\begin{aligned}
\mathit{task\_coverage(find\text{-}initial\text{-}final\text{-}locations)} &= 3 \\
\mathit{task\_coverage(move\text{-}spill\text{-}to\text{-}final\text{-}location)} &= 2 \\
\mathit{task\_coverage(report\text{-}progress\text{-}every\text{-}4\text{-}minutes)} &= 2
\end{aligned}$$

and for $R = \{\text{RED, BLUE, GREEN}\}$:

$$\begin{aligned}
\mathit{task\_coverage(find\text{-}initial\text{-}final\text{-}locations)} &= 4 \\
\mathit{task\_coverage(move\text{-}spill\text{-}to\text{-}final\text{-}location)} &= 3 \\
\mathit{task\_coverage(report\text{-}progress\text{-}every\text{-}4\text{-}minutes)} &= 3
\end{aligned}$$

Obviously, since $\mathit{task\_coverage}(\mathit{task}_i)$ does not equal a multiple of n for all task_i in either the two-robot or the three-robot team, these teams do not qualify as homogeneous as set forth in section 3.3.

3.8.3 Results of Baseline Experiments

The first set of experiments I report here are what I call the two- and three-robot *baseline* experiments. In this set of experiments, I equipped the robot teams described above with the fully functional ALLIANCE architecture, complete with explicit communication for allowing robots to be aware of the activities of their teammates. Using this communication system, robots broadcasted a statement of their current actions to the rest of the team at a pre-specified rate. I ran over 50 logged trials of these experiments, along with several variations reported in chapter 5, all of which allowed the study of a number of interesting cooperative robot issues. The results presented in this section for the numerous baseline experiments illustrate the ability of ALLIANCE to meet the design requirements of robustness, reliability, flexibility, and coherence.

Let us examine the results of these experiments. Figure 3-9 shows the actions selected by each robot on a typical trial of this experiment with the two-robot team, and figure 3-10 shows the actions selected by each robot on a typical trial with the three-robot team. Figure 3-11 shows the trace of the robot movements during a typical two-robot mission[6]. Prior to both the two- and three-robot trials, the L-ALLIANCE learning system described in chapter 4 has allowed the team members to determine that GREEN (or GREEN and BLUE, in the three-robot case) cannot successfully ac-

[6]This movement trace data was created by hand from a videotape of a two-robot team performing this mission.

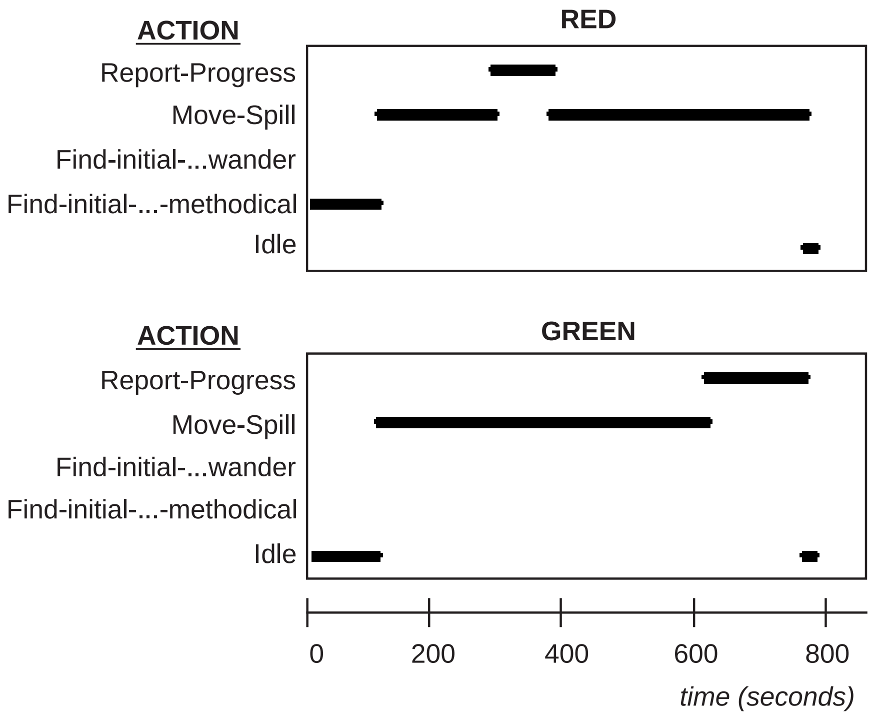

Figure 3-9: Robot actions selected during experiment with two robots (RED and GREEN) using the ALLIANCE architecture with full awareness of other team member actions. This is one instance of many runs of this mission.

complish the task corresponding to the *find-initial-final-locations-methodical* behavior set, and that RED is quicker at finding the spill.

At the beginning of a typical three-robot mission, RED has the highest motivation to perform behavior set *find-initial-final-locations-methodical*, causing it to initiate this action, as shown in figure 3-12. (In this and the following photographs, the starting location of the GREEN and BLUE robots (center, rear of the first photograph) is the room entrance, the spill location is at the right, center of the photograph (where the small, cylindrical objects are located), and the goal location is at the near, right end of the photograph.) This causes GREEN and BLUE to be satisfied for a while that the initial and final spill locations are going to be found; since no other task can currently be performed, they sit waiting for the two locations to be found. However, they do not sit forever waiting on the locations to be found. As they wait, they become more and more impatient over time, which can cause one of BLUE or GREEN to activate its own *find-initial-final-locations-wander* if RED does not successfully locate the spill. Note that BLUE and GREEN do not activate their own *find-initial-final-locations-methodical* because they have learned in previous trials that

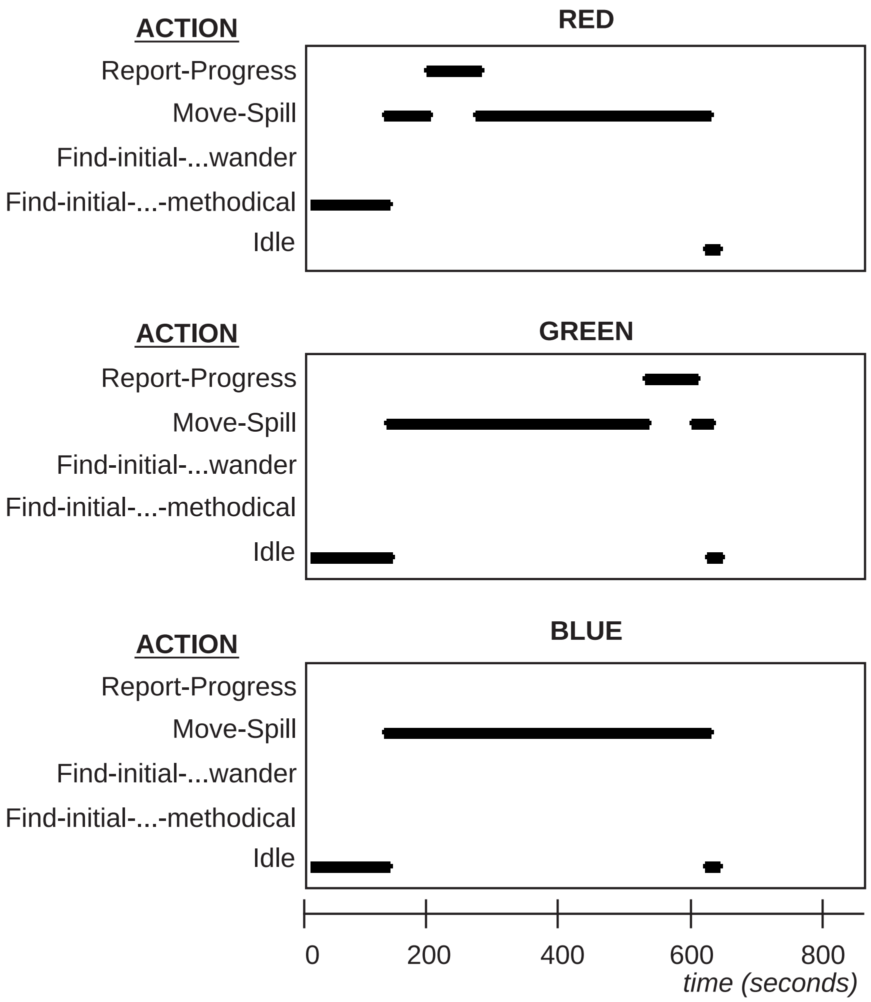

Figure 3-10: Robot actions selected during experiment with three robots (RED, BLUE, and GREEN) using the ALLIANCE architecture with full awareness of the current actions of other team members. This is one instance of many runs of this mission.

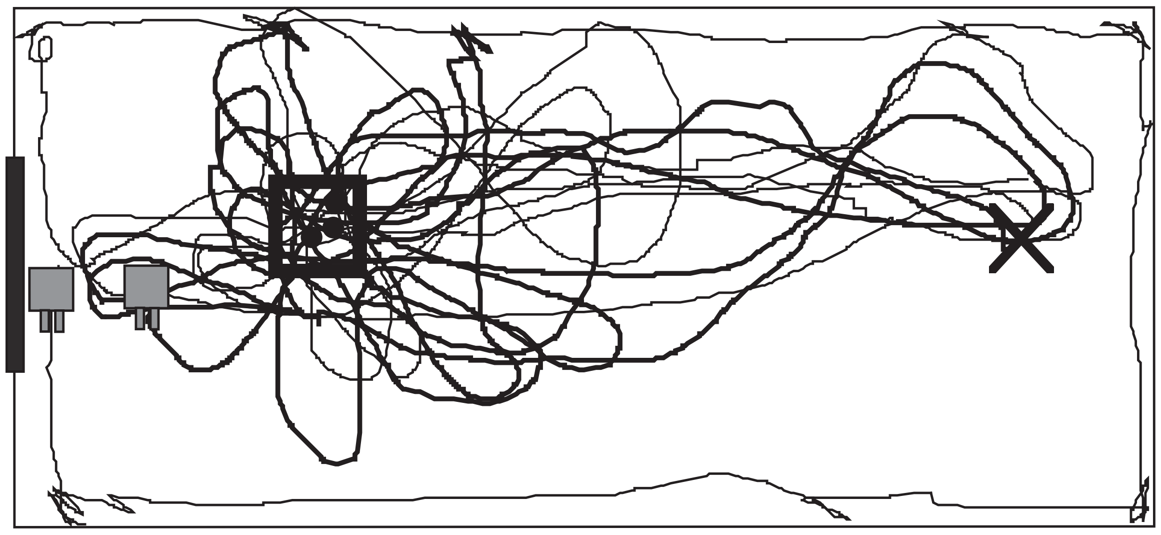

Figure 3-11: A trace of the movements of a typical two-robot team performing the cleanup mission. The two robots at the left end of the picture show the robot starting locations. The square with the heavy bold outline towards the left, center of the room is the starting spill location, while the bold "X" at the right indicates the desired final spill location. The location from which progress reports are to be given is in the vicinity of the robot starting positions. The bold trace gives the movements of GREEN; the non-bold trace indicates the movements of RED. In this example, RED first performs the *find-locations* task, followed by both GREEN and RED moving spill objects — three objects each, on three transports each. GREEN ends up reporting progress two times, while RED reports the team progress once.

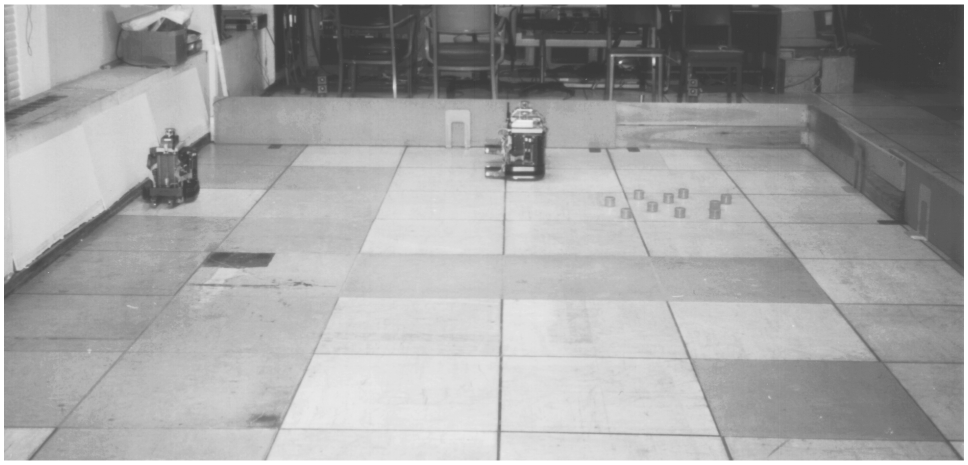

Figure 3-12: The beginning of the hazardous waste cleanup mission. The left-most robot (RED) has activated its *find-initial-final-locations-methodical* behavior set, which causes it to circle the perimeter of the room. The remaining two robots (aligned with each other at the room entrance) are patiently waiting on RED to complete its task. Note the spill objects (small cylindrical "pucks") at the right center of the photograph.

that action does not achieve the desired effect. Indeed, as shown in figure 3-13 for a two-robot experiment, GREEN did overtake RED at this task when I intentionally interfered with RED's progress. In that case, RED acquiesced its attempt to find the initial and final locations to GREEN, since RED realized it was encountering difficulties of some sort. In either case, the robot finding the initial and final spill locations reports these two locations to the rest of the team.

At this point, the environmental feedback and knowledge of the initial and final spill locations indicate to the robots that the *move-spill* behavior set is applicable. Since this is a task that can be shared, the robots begin searching for a spill object in the initial spill area, as shown in figure 3-14 for a typical three-robot experiment. Once a spill object has been grasped, the robot carries it to the goal location. Figure 3-15 shows a close-up of a robot grasping a spill object and beginning its transport to the goal location. In this photograph, note the extra spill object caught in the cavity under the gripper, which allows this robot to move more than one spill object on this trip to the goal location. Figure 3-16 shows the transport from a wider perspective.

In the meantime, the robots' motivations to report the team's progress are increasing. Once a robot has delivered a spill object to the destination, that robot becomes motivated to report the team's progress, and thus activates the *report-progress* behavior set. Figure 3-17 shows a robot (at the far, center of the photograph) reporting the team's progress at the room "entrance" during a typical mission. This reporting

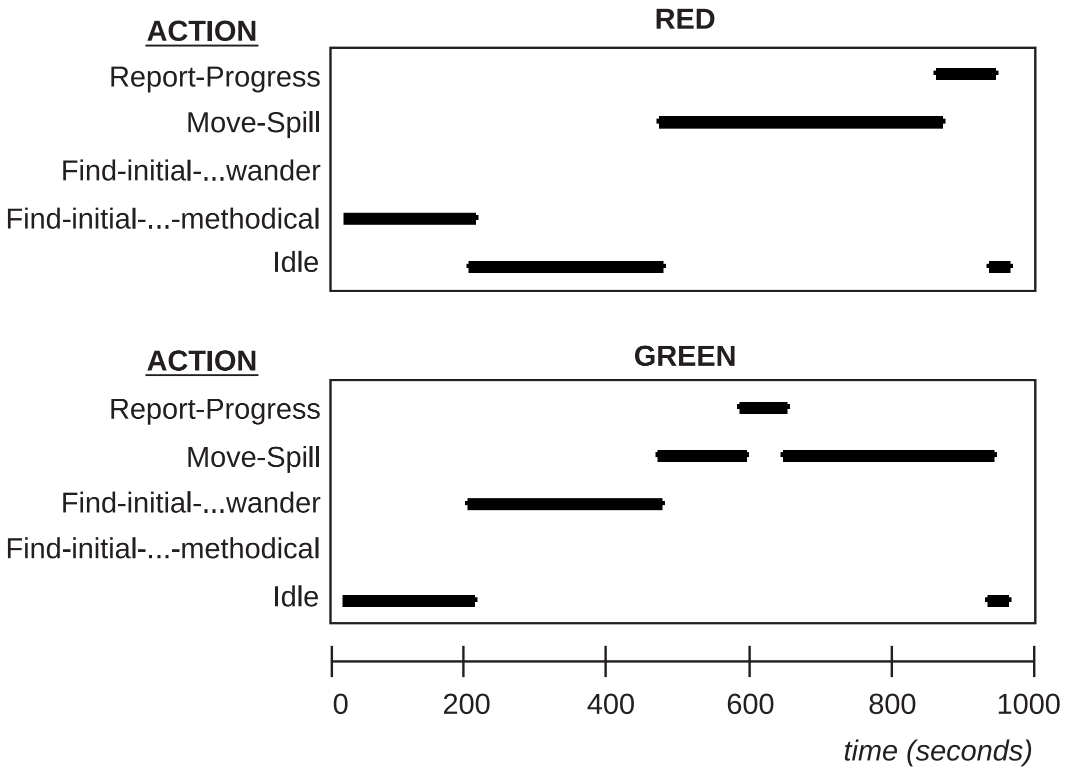

Figure 3-13: Robot actions selected during experiment with two robots (RED and GREEN) with full awareness of other team member actions, and when RED fails in its task to find the initial and final spill locations. This is one instance of many runs of this mission.

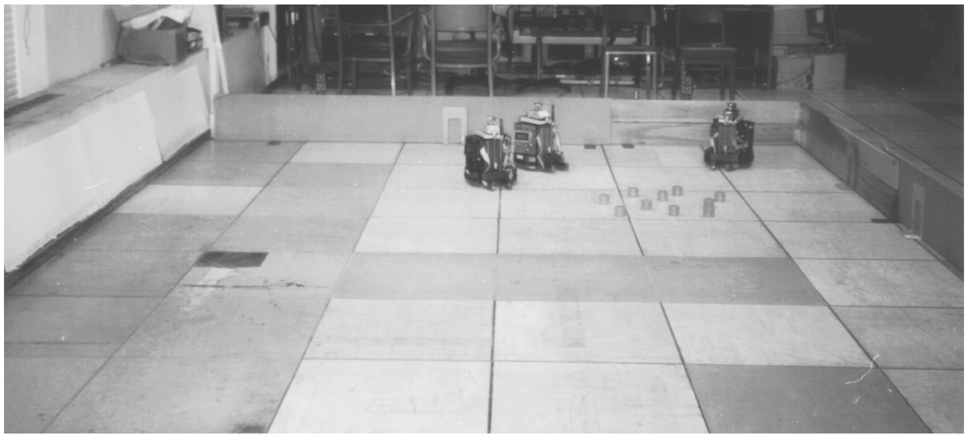

Figure 3-14: Now knowing the location of the spill, the R-2 robots are attempting to find spill objects to move to the goal location.

satisfies the rest of the team, so the remaining robots re-activate their *move-spill* behavior set. This series of actions is repeated until all of the spill is moved and the mission is complete.

3.8.4 Discussion of Baseline Experiments

This experiment illustrates a number of primary characteristics I consider important in developing cooperative robotic teams. First of all, the cooperative team is robust, in that robots are allowed to continue their actions only as long as they demonstrate their ability to have the desired effect on the world. This was illustrated in the two-robot experiment by GREEN's becoming gradually more impatient with RED's search for the spill. If RED did not locate the spill in a reasonable length of time then GREEN would take over that task, with RED acquiescing the task. Secondly, the cooperative team is able to respond autonomously to many types of unexpected events either in the environment or in the robot team without the need for external intervention. At any time in this mission, I could disable one of the robots, causing the remaining robot to perform those tasks that the disabled robot would have performed, assuming the task was within the working robot's capabilities. Clearly, I could also have easily increased or decreased the size of the spill during the mission and the robots would not be adversely affected. Third, the cooperative team need have no *a priori* knowledge of the abilities of the other team members to effectively accomplish the task. However, the learning system, L-ALLIANCE, does allow the team to improve its efficiency on subsequent trials whenever familiar robots are present. This was illustrated in GREEN's willingness to allow RED to attempt to find the spill because of GREEN's

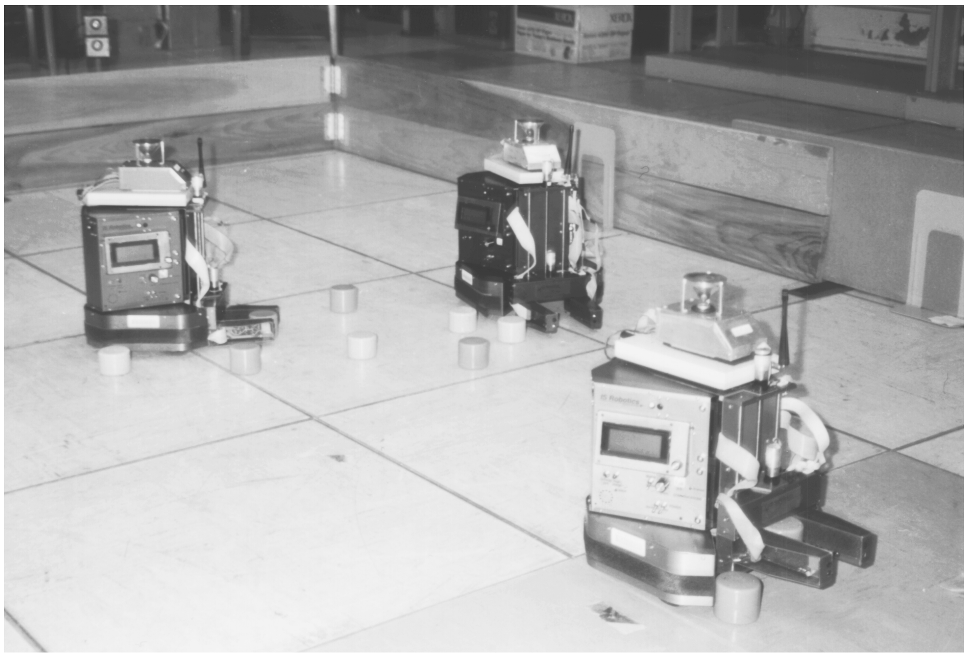

Figure 3-15: The rightmost R-2 in this photograph has just grasped a spill object and is taking it to the goal location. Note the additional puck caught in the cavity below the gripper, which allows this robot to transport more than one spill object on this trip.

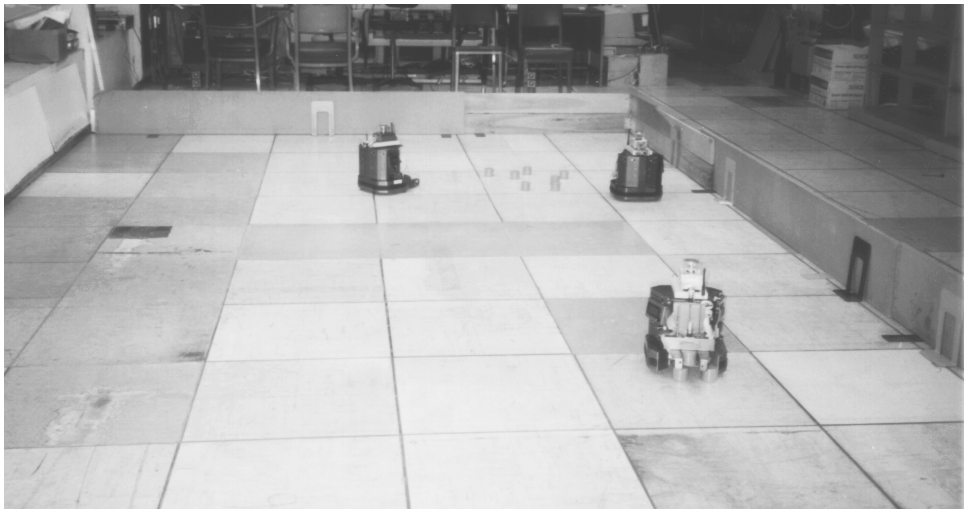

Figure 3-16: One R-2 (the closest to the viewer) has found and grasped a spill object, and is now transporting it to the goal location. This goal location is at the near, right foreground of this photograph.

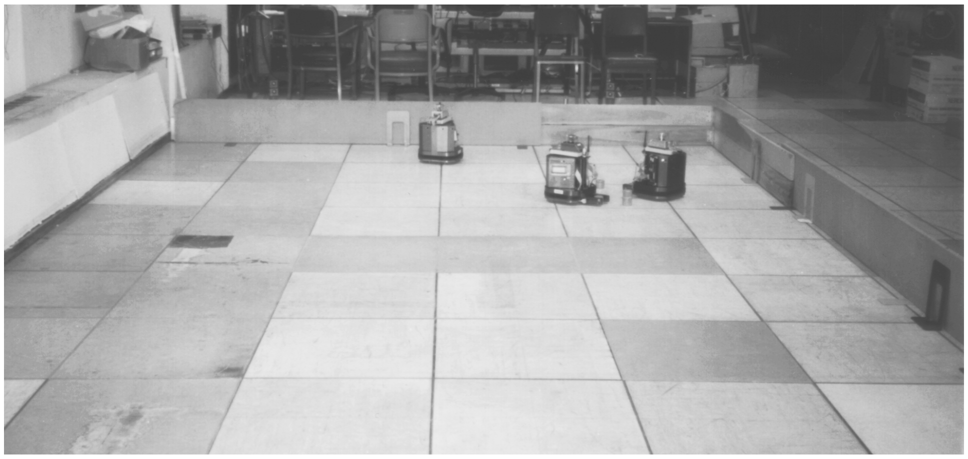

Figure 3-17: Reporting progress in the hazardous waste cleanup mission. The robot in the far, center of this photograph is reporting the team's progress at the "entrance" to the room.

knowledge that RED was superior in this task. In a similar vein, the learning system would also allow GREEN and RED to learn about GREEN's improved performance in finding the spill if GREEN's faulty sensors were repaired between missions.

The types of problems I had in implementing this application revolved mostly around the weaknesses of the R-2 sensors, and dealt with issues of behavior set design rather than issues of cooperative control. Collisions between robots were reduced by placing white paper around the lower edge of the robots, which can be more easily detected by the robots' infrared sensors. However, the robots still jammed their shoulders at times, and had to be manually rescued. Another type of robot interference that occasionally occurred in the process of searching for pucks was when two robots headed straight toward each other in the zone of the spill, and interpreted each other's fingers as spill objects. They would then proceed to "hold hands", grasping each other's gripper and lifting. Although this was quite entertaining, I tried to preempt this as much as possible, to prevent damage to the robots' grippers. A final problem in this mission was that the robots would occasionally become lodged on a spill object when depositing another spill object at the goal location. Recall that when a robot initially grasps a spill object, it lifts it up during the transport to the goal. However, since the robot cannot sense spill objects underneath its fingers, if it releases its spill object and lowers its gripper while another spill object is underneath the gripper, the robot pushes up on the spill object, raising its wheels off the ground and becoming permanently stuck.

One interesting emergent behavior can also be reported that resulted from more than one robot searching the initial spill area at the same time. Occasionally, if the robot positions were just right, the result of trying to home in on a spill object actually caused one robot to follow another robot through the spill area. This emergent "follow the leader" behavior is not of particular use here, however, since the result is usually for the following robot to be lead away from the spill objects, rather than toward them.

3.9 Summary and Contributions

This chapter has presented a fully distributed architecture for fault tolerant heterogeneous mobile robot cooperation. I discussed the assumptions I made, and did not make, in the development of ALLIANCE, and then described the primary mechanism facilitating adaptive action selection in this architecture — the motivational behavior. I presented a formal model of this mechanism, which combines input from robot sensors, inter-robot communication, active behavior sets, and internal robot motivations, to calculate a motivational level for each of the robot's behavior sets. I proved that ALLIANCE is guaranteed to terminate in mission completion for reliable, aware robots which are maximally patient and minimally acquiescent, or for limitedly-reliable robot teams when the Progress When Working condition is true. I then presented the results of implementing this architecture on physical teams of robots performing the hazardous waste cleanup mission.

Let us now review the initial design requirements outlined in chapter 1 and examine the extent to which ALLIANCE meets these design goals.

3.9.1 Meeting Design Requirements

Recall that my primary design goal outlined in chapter 1 was to develop a cooperative architecture that allowed heterogeneous robots to cooperate to accomplish a mission while exhibiting robustness, reliability, flexibility, and coherence. As I review these issues here, I note that the development of a cooperative robot architecture can actually be viewed as the development of an individual robot control architecture that facilitates a single robot's cooperation with other similarly-designed robots. Thus, I describe how each of these performance issues are addressed both from the view of an individual robot control strategy and from the view of a collective team strategy.

Robustness and Fault Tolerance

Fault tolerance and *robustness* refer to the ability of a system to detect and gracefully compensate for partial system failures. In ALLIANCE, each individual robot is designed using a behavior-based approach which ensures that lower levels of competence continue to work even when upper levels break down. In addition, individual robots can be given multiple ways to perform certain tasks, allowing them to explore alternative approaches when met with failure.

From the viewpoint of the team, ALLIANCE first enhances robustness by being fully distributed. Unlike hierarchical architectures, since no individual robot in ALLIANCE is responsible for the control of other robots, the failure of any particular robot is not disproportionally damaging. Secondly, ALLIANCE enhances team robustness by providing mechanisms for robot team members to respond to their own failures or to failures of teammates, leading to a reallocation of tasks to ensure that the mission is completed. Third, ALLIANCE allows the robot team to accomplish its mission even when the communication system providing it with the awareness of team member actions breaks down (see chapter 5 for a deeper discussion of this issue). Although the team's performance in terms of time and energy may deteriorate, at least the team is still able to accomplish its mission. Finally, ALLIANCE enhances team robustness by making it easy for robot team members to deal with the presence of overlapping capabilities on the team. The ease with which redundant robots can be incorporated on the team provides the human team designer the ability to utilize physical redundancy to enhance team robustness.

Reliability

Reliability refers to the dependability of a system and whether it functions properly each time it is utilized. In ALLIANCE, reliability is measured in terms of the architecture's ability to have the robot team accomplish its mission each time the mission is attempted. I have shown that under certain conditions ALLIANCE is guaranteed to allow the robot team to complete its mission, except when robot failures eliminate required capabilities from the team (from which no architecture could recover). The ALLIANCE action selection mechanism thus gives a means for the robot team to achieve its mission reliably and consistently.

Flexibility and Adaptivity

Flexibility and *adaptivity* refer to the ability of robots to modify their actions as the environment or robot team changes. The motivational behavior mechanism of ALLIANCE constantly monitors the sensory feedback of the tasks that can be performed by an individual agent, adapting the actions selected by that agent to the current environmental feedback and the actions of its teammates. Whether the environment changes to require the robots to perform additional tasks or to eliminate the need for certain tasks, ALLIANCE allows the robots to handle the changes fluidly and flexibly. ALLIANCE enhances the adaptivity and flexibility of a robot *team* by providing mechanisms for robots to work with any other robots designed using ALLIANCE; the robots are not required to possess advance knowledge of the capabilities of the other robots.

As we shall see in chapter 4, L-ALLIANCE also helps with the issues of flexibility and adaptivity by allowing robots to learn about their own abilities and the abilities of their teammates in order to improve their performance on subsequent trials of similar missions whenever familiar agents are present.

Coherence

Coherence refers to how well the actions of individual agents combine towards some unifying goal. For individual agents, ALLIANCE causes robots to work only on those tasks which the environmental feedback indicates need to be executed. Thus, ALLIANCE will not cause an individual agent to work on some task that is not required by the mission, nor consistent with the current state of the environment.

Obtaining coherence at the team level requires that robots have some means of determining the actions of other robots and/or the effect of those actions on the environment. Without this knowledge, the robots become a collection of individuals pursuing their own goals in an environment that happens to contain other such robots. While we certainly want the robots to be able to accomplish something useful even without knowledge of other robots on the team, ideally each robot should take into account the actions of other robots in selecting their own actions.

Determining the actions of other robots can be accomplished through either passive observation or via explicit communication. Since passive action recognition is very difficult and is a major research topic in itself, ALLIANCE augments the observation skills of the robot team members through the use of one-way broadcast communication that provides each robot with an awareness of the actions of other robots, plus the ability to act on that information. With this awareness, robots do not replicate the actions of other robots, thus giving them more coherence. I note the importance of this mechanism to achieve team coherence, since when the communications mechanism is unavailable, team coherence is reduced. Refer to chapter 5 for a discussion of this issue.

3.9.2 Contributions

The design of ALLIANCE embodies many characteristics that facilitate fault tolerant cooperative control of small- to medium-sized heterogeneous mobile robot teams as applied to missions involving loosely-coupled, largely independent tasks. These characteristics are summarized as follows:

- Fully distributed (at both the individual robot level and at the team level).

- Applicable to robot teams having any degree of heterogeneity.
- Uses no negotiation or two-way conversations.
- Recovers from failures in individual robots or in the communication system.
- Allows new robots to be added to the team at any time.
- Allows adaptive action selection in dynamic environments.
- Eliminates replication of effort when communication is available.
- Provably terminates for a large class of applications.
- Scales easily to large missions.

I note that a number of issues regarding the efficiency of ALLIANCE were not addressed in this chapter. Among these issues include questions of how long robots remain idle before activating a task, how to ensure that robots failing at one task go on to attempt another task they might be able to accomplish, how robots deal with having more than one way to accomplish a task, and so forth. Since these issues are handled with dynamic parameter update mechanisms, I answer these questions as I present L-ALLIANCE in chapter 4.

Chapter 4

L-ALLIANCE: Improving Efficiency

This chapter describes an extended version of ALLIANCE, called L-ALLIANCE (for Learning ALLIANCE), that preserves the fault tolerant features of ALLIANCE while incorporating on-line, distributed control strategies that greatly improve the efficiency of the cooperative robot team performing a mission composed of independent tasks. These strategies allow each individual robot to learn about the quality with which robot team members perform certain tasks, and then to use this learned knowledge to determine the appropriate action to activate at each point in time. This chapter first provides the motivation for why this learning is necessary, discusses the assumptions made in L-ALLIANCE, and then provides a formal description of the learning problem. I show in section 4.3 that this learning problem is NP-hard, concluding that requiring the robot team to derive the optimal selection of actions through learning is unrealistic.

In section 4.4, I discuss the scalability of L-ALLIANCE as the size of the robot team and the size of the mission grows, showing that through parallelism, the ALLIANCE and L-ALLIANCE techniques are independent of the size of the mission, and grow linearly with the number of robots on the team. I then discuss the mechanism used in L-ALLIANCE to allow robots to learn about the performance levels of teammates, and describe various distributed control strategies I investigated for using this learned knowledge to improve the efficiency of the team. In section 4.5, I present the empirical results of these investigations in simulation for a large space of possible cooperative robot teams that vary in the number of robots, the size of the mission to be performed, the degree of task coverage, the degree of heterogeneity across robots, and the degree to which the Progress When Active condition introduced in chapter 3 holds. I then compare the results of the best distributed control strategy to the optimal solution and show that this final control strategy performs quite close to the optimal allocation of tasks to robots for those examples in which the optimal solution can be derived. In section 4.6, I provide the details of how the learning approach is incorporated into the L-ALLIANCE motivational behaviors. Finally, I conclude in section 4.7 by returning to my original design requirements for the cooperative architecture, describing the contributions L-ALLIANCE makes towards meeting those requirements.

4.1 Motivation for Efficiency Improvements via Learning

As described in chapter 3, the ALLIANCE architecture allows robots to adapt to the ongoing activities and environmental feedback of their current mission. However, ALLIANCE does not address a number of efficiency issues that are important for cooperative teams. These issues include the following: How do we ensure that robots attempt those tasks for which they are best suited? Can we enable the robot team to increase its performance over time? Does failure at one task imply total robot failure? How does a robot select a method of performing a task if it has more than one way to accomplish that task? How to we minimize robot idle time?

The L-ALLIANCE enhancement to ALLIANCE addresses these issues of efficiency by incorporating a dynamic parameter update mechanism into the ALLIANCE architecture. This parameter update mechanism allows us to preserve the fault tolerant features of ALLIANCE while improving the efficiency of the robot team performance. A number of benefits result from providing robots with the ability to automatically adjust their own parameter settings to improve efficiency:

1. **Relieve humans of the parameter adjusting task:**

   As described in chapter 3, ALLIANCE requires human programmer tuning of motivational behavior parameters to achieve desired levels of robot performance. Although finding good parameter settings is often not difficult in practice, the cooperative architecture would be much simpler to use if the human were relieved of the responsibility of having to tune numerous parameters.

2. **Improve the efficiency of the mission performance:**

   Related to the previous item is the issue of the efficiency of the robot team's performance of its mission. As human designers, it is often difficult to evaluate a given robot team performance to determine how best to adjust parameters to improve efficiency. However, if the robots were controlled by an automated action selection strategy that has been shown to result in efficient group action selection in practice, then the human designer can have confidence in the robot team's ability to accomplish the mission autonomously, and thus not feel the need to adjust the parameters by hand.

3. **Facilitate custom-designed robot teams:**

   Providing the ability for robot teams to carry over their learned experiences from trial to trial would allow human designers to successfully construct unique teams of interacting robots from a pool of heterogeneous robot types for any given mission without the need for a great deal of preparatory work. Although ALLIANCE allows newly constructed teams to work together acceptably the first time they are grouped together, automated parameter adjusting mechanisms would allow the team to improve its performance over time by having each robot learn how the presence of other specific robots on the team should affect its own behavior. For example, two robots pursuing the hazardous waste cleanup mission that can both find the location of the spill, but with different task completion times, should learn which robot performs the task quicker and allow that robot to find the spill location on future missions, as long as it continues to demonstrate superior performance. Through this learning, the robots should thus allow the mere presence of other team members to affect their subsequent actions.

4. **Allow robot teams to efficiently adapt their performance over time:**

   During a mission, a robot team's environment and the ability of its members may change dynamically. However, in the basic ALLIANCE architecture, parameter settings do not change after the start of the mission. Thus, these robot teams would be quite vulnerable to calibration problems due to drifts in the environment and in robot capabilities. The ability to automatically update parameters during a mission is therefore of critical importance.

Providing a robot team with the ability to automatically update its own motivational behavior parameters requires solutions to two problems:

- How to give robots the ability to obtain knowledge about the quality of team member performances

- How to use team member performance knowledge to select a task to pursue

Solutions to the first problem require a robot to learn not only about the abilities of its teammates, but also about its own abilities. Although each robot "knows" the set of behaviors that it has been programmed to perform, it may perform poorly at certain tasks relative to other robots on the team. Robots must thus learn about these relative performance differences as a first step toward efficient mission execution. However, learning these relative performance quality differences is only a first step in improving efficiency. The next, more major, question, is how robots use the performance knowledge to efficiently select their own actions. This chapter describes the L-ALLIANCE approach to these problems.

4.2 Assumptions Made in L-ALLIANCE

Two key assumptions are made in the development of L-ALLIANCE, as follows:

- A robot's average performance in performing a specific task over a few recent trials is a reasonable indicator of that robot's expected performance in the future.
- If robot r_i is monitoring environmental conditions C to assess the performance of another robot r_k, and the conditions C change, then the changes are attributable to robot r_k.

Without the first assumption, it will be quite difficult for robots to learn anything at all about their own expected performance, or the performance of their teammates, since past behavior would provide no clues to the expected behavior in the future. The trick, of course, is determining which aspects of a robot's performance are good predictors of future performance. In L-ALLIANCE, I have used the simple measure of the *time* of task completion, which has served to be a good indicator of future performance.

My second assumption deals with the well-known credit assignment problem, which is concerned with determining which process should receive credit (or punishment) for the successful (or unsuccessful) outcome of an action. The assumption I make in L-ALLIANCE is that the only agents which affect the properties of the world that a robot r_i is interested in are the robots that r_i is monitoring. Thus, if a robot r_k declares it is performing some task, and that task becomes complete, then the monitoring robot will assume that r_k caused those effects. This assumption is certainly not always true, since external agents really can intrude on the robots' world. However, since this issue even causes problems for biological systems, which often have difficulty in correctly assigning credit, I do not concern myself greatly with this oversimplification.

4.3 The Efficiency Problem

I now formally define the efficiency problem with which I am concerned. As in chapter 3, let $R = \{r_1, r_2, ..., r_n\}$ represent the set of n robots on the cooperative team, and the set $T = \{task_1, task_2, ..., task_m\}$ represent the m independent tasks required in the current mission. Each robot in R has a number of high-level task-achieving functions (or behavior sets) that it can perform, represented by the set $A_i = \{a_{i1}, a_{i2}, ...\}$. Since different robots may have different ways of performing the same task, I need a way of referring to the task a robot is working on when it activates a behavior set. Thus, as in chapter 3, I define the set of n functions $\{h_1(a_{1k}), h_2(a_{2k}), ..., h_n(a_{nk})\}$, where $h_i(a_{ik})$ returns the task that robot r_i is working on when it activates behavior set a_{ik}.

Now, I define a metric evaluation function, $q(a_{ij})$, which returns the "quality" of the action a_{ij} as measured by a given metric. Typically, we consider metrics such as the average time or average energy required to complete a task, although many other metrics could be used. Of course, robots unfamiliar with their own abilities or the abilities of their teammates do not have access to this $q(a_{ij})$ fuction. Thus, an additional aspect to the robot's learning problem is actually obtaining the performance quality information required to make an "intelligent" action selection choice.

Finally, I define the tasks a robot will elect to perform during a mission as the set $U_i = \{a_{ij} | \text{robot } r_i \text{ will perform task } h_i(a_{ij}) \text{ during the current mission}\}$.

In the most general form of this problem, the following condition holds:

Condition 4 (Different Robots are Different):
Different robots may have different collections of capabilities; thus, I do not *assume that* $\forall i. \forall j. (A_i = A_j)$. *Further, if different robots can perform the same task, they may perform that task with different qualities; thus, I do* not *assume that if* $h_i(a_{ix}) = h_j(a_{jy})$, *then* $q(a_{ix}) = q(a_{jy})$.

Then I can define the formal efficiency problem under condition 4 as follows:

ALLIANCE Efficiency Problem (AEP):

For each robot, r_i:
Given T, A_i, and $h_i(a_{ik})$, determine the set of actions U_i such that

- $\forall i. U_i \subseteq A_i$
- $\forall j. \exists i. \exists k. ((task_j = h_i(a_{ik})) \text{ and } (a_{ik} \in U_i))$

and one of the following is minimized, according to the desired performance metric:

- $max_i(\sum_{a_{ik} \in U_i} q_{time}(a_{ik}))$ (metric is *time*)

- $\sum_{i} \sum_{a_{ik} \in U_i} q_{energy}(a_{ik})$ (metric is *energy*)

The first two constraints of the learning problem ensure that each task in the mission is assigned to some robot that can actually accomplish that task. The final two constraints ensure that either the time or the energy required to complete the mission is minimized. Since robot team members usually perform their actions in parallel during a mission, the total mission completion time is the time at which the last robot finishes its final task. Thus, when our performance metric is *time*, we want to minimize the maximum amount of time any robot will take to perform its set of actions. On the other hand, when we are concerned with energy usage, parallelism does not help us, since robots use energy whenever they are actively performing some task. In this case, we must minimize the total amount of energy that all robots take to perform their sets of tasks during the mission.

It is important to note here that for reasons outlined in chapter 1 and elsewhere in this report, the efficiency problem must be solved by a distributed group of robots rather than a centralized decisionmaker. I do not want to sacrifice the advantages of robustness, fault tolerance, and flexibility offered by a distributed solution for a centralized controller which greatly reduces or even eliminates these desirable characteristics. In fact, as we shall see, an appropriately designed algorithm can scale much better than a centralized approach as the size of the mission increases. Thus, rather than having some controlling robot derive the task allocation for the entire team, the *ideal* solution involves each robot choosing actions individually such that the globally optimal result is obtained.

However, even if one assumes that the robots have good information on their own abilities and the abilities of their teammates, how realistic is it to require the robots to derive the optimal action selection policy? As it turns out, the efficiency problem, AEP, can be easily shown to be NP-hard by restriction to the well-known NP-complete problem PARTITION [Garey and Johnson, 1979]. The PARTITION problem is as follows: given a finite set W and a "size" $s(w) \in Z^+$ for each $w \in W$, determine whether there is a subset $W' \subseteq W$ such that $\sum_{w \in W'} s(w) = \sum_{w \in W - W'} s(w)$. We then have the following:

Theorem 3 *The ALLIANCE efficiency problem (AEP) is NP-hard in the number of tasks required by the mission.*

Proof: By restriction to PARTITION:

Allow only instances of AEP where $n = 2$, $A_1 = A_2 = W$, $\forall i.\forall j.(h_1(a_{ij}) = task_j)$, and $\forall j.(q(a_{1j}) = q(a_{2j}) = s(w_j))$, for $w_j \in W$. Then since PARTITION is a special case of AEP, AEP must be NP-hard. $\Box$

Since the PARTITION problem is stated in terms of finding two equally-sized subsets of tasks W and W', the proof of this theorem restricts AEP to those instances involving two robots with identical capabilities and qualities of capabilities. Furthermore, each robot has the same one-to-one mapping of behavior sets to tasks, meaning that all robots use the same behavior set to accomplish the same task, and all behavior sets are needed to accomplish the mission. These AEP instances are then instances of PARTITION, so that, if we could solve AEP, we could solve PARTITION.

Thus, since this efficiency problem is NP-hard, I cannot expect the robot teams to be able to derive an optimal action selection policy in a reasonable length of time. Thus, I focus my efforts on heuristic approximations to the problem that work well in practice.

4.4 The L-ALLIANCE Learning and Efficiency Mechanism

The L-ALLIANCE approach to improving efficiency while preserving fault tolerance is based on a key assumption stated in section 4.2: the quality of a robot's recent performance at a task is a reasonable indicator of its expected performance in the future. To measure the quality of task performance, I use the metric of the *time* required to perform a task, which in this case I assume to be equivalent to the energy required to perform a task. Although other performance metrics are certainly possible, it is crucial that the chosen quality be observable by robots on the team, since each robot must assess the performance of its teammates in order to detect improvements in performance or robot failures, and thus alter its action selection accordingly. However, as has been stressed repeatedly in this report, robots will indeed experience failures or changes in capabilities during a mission, or across missions; thus the measure of past performance cannot be guaranteed to predict future performance. Robots must therefore use their knowledge about previous performance only as a guideline, and not as an absolute determinant of the abilities of robot team members.

An integral part of the L-ALLIANCE learning mechanism, then, requires each robot team member to use the given quality metric to monitor its own performance and the performance of its teammates as each robot executes its selected task. Since environmental variations and sensory and effector noise undoubtably cause performance to differ somewhat each time a task is executed, L-ALLIANCE requires the robots to maintain an average and a standard deviation of performance for each robot, for each task it performs, for a small number of trials. Determining how many trials, μ, over which to maintain this data depends upon the desired characteristics of the robot team [Pin *et al.*, 1991]. Maintaining an average over too many trials results in a slow response to changes in robot performance. On the other hand, maintaining an average over too few trials does not provide a reasonable predictor of future performance. My experiments have shown than an average over about 5 trials results in good predictive capability, while still allowing the robots to be responsive to failures.

Let us now assume that robots have managed to learn this information about the abilities of their teammates. How do they use this information to help them select their own actions? A number of factors make this question challenging to answer. The information robots have on their teammates' expected performances is uncertain information. Nevertheless, I want the robots to respond to the failures of other robots, while not being too quick to interrupt another robot's activities. Likewise, I want a robot to acquiesce its current activity if it has indeed failed, but I do not want it to be too "meek", when it could actually complete the task if it had just a little more time. How does a robot decide whether to interrupt another robot that is working on a task that the first robot thinks it can do better, even though the second robot is still making progress towards task completion?

To address this problem, I examined three issues:

1. How does a robot obtain performance quality measurements?

2. In what order should a robot select tasks to execute?

3. How does a robot know when to give up on others or on itself?

Subsections 4.4.2, 4.4.3, and 4.4.4 below address the L-ALLIANCE general approach to answering these three questions. Subsection 4.6 describes the explicit mechanisms for implementing this general approach in the L-ALLIANCE architecture by describing the L-ALLIANCE formal model. However, I first comment on the distributed nature of ALLIANCE and L-ALLIANCE, emphasizing its differences with a more centralized robot controller. It is important to keep this distinction in mind as the issues of learning and efficiency are discussed in the remainder of the chapter.

4.4.1 The Distributed Nature of ALLIANCE and L-ALLIANCE

When discussing the control strategies of L-ALLIANCE, it is very tempting to describe the mechanisms from a global perspective, both at the level of the entire team's selected actions, and, in particular, at the level of an individual robot's action selection. This is understandable, since as humans, we tend to want to attribute a centralized *will* or decision-making capability to automatons exhibiting complex behavior. Thus, when a robot controlled with ALLIANCE or L-ALLIANCE activates some behavior set, it seems natural to describe that process as the result of a centralized decision-maker. One might imagine that such a decision-maker considered all of the tasks that the robot could perform, analyzed the possibilities in light of the current actions and potential capabilities of the robot's team members and the current environmental feedback, perhaps used some objective function to optimize some metric, and then selected the action that maximizes the objective function. As described in chapter 3, however, the mechanism used in ALLIANCE and L-ALLIANCE for action selection within an individual robot is not a centralized mechanism, but rather a distributed mechanism in which the motivational behaviors interact to cause a robot to select its next action. Thus, the phrase "the motivational behaviors interact to cause a robot to select its next action" is more indicative of the underlying process than "the robot selects its next action". Nevertheless, for the sake of conciseness, I often use the later phrase in this report as a shorthand for the former.

It is important to note that although the L-ALLIANCE approach could be implemented on each robot as a centralized controlling behavior, doing so would violate the robustness and fault tolerant design principles set forth in chapter 1. Having any sort of centralized process responsible for obtaining all of the performance quality measurements of robot team members, then using this information to update the parameters of motivational behaviors would place the robot at risk of complete breakdown if the one controlling module were to fail. In addition, a centralized decision-maker does not scale well for larger numbers of tasks, since each additional task that could be performed must be considered in light of the robot's previous abilities and all other team member capabilities. Fairly quickly, then, the centralized decision-maker has too much work to do to effectively control the robot, leading to the classical problems of traditional robot control architectures (see chapter 8 for a further discussion of this issue). Distributing the control mechanism in ALLIANCE and L-ALLIANCE actually makes it quite easy to handle increasingly complex robot missions; one needs simply to provide additional processors over which the motivational behaviors can be divided to allow arbitrarily large numbers of tasks to be monitored and controlled. With ALLIANCE and L-ALLIANCE one thus eliminates the control nightmare of software modules growing arbitrarily large to handle increased mission sizes.

Of course, distributing the knowledge across many motivational behaviors can make the control problem much more difficult. How does one cause the motivational behaviors to interact such that each robot selects the actions it is most suited for, and so that all tasks become complete? The challenge is in the two layers of emergence, or what I refer to as the "doubly-distributed" nature of ALLIANCE and L-ALLIANCE: the interaction of the motivational behaviors on an individual robot must be designed to allow the emergent interaction of the team members' actions to result in the most efficient execution of the mission possible. This is the challenge addressed in the remainder of this chapter.

4.4.2 Obtaining Performance Quality Measurements

Of central importance to the learning mechanism used in L-ALLIANCE is the ability of robots to monitor, evaluate, and catalog the performance of team members in executing certain tasks. Without this ability, a robot must rely on human-supplied performance measurements of robot team members. In either case, once these performance measurements are obtained, the robot team members have a basis for determining the preferential activation of one behavior set over any other either for the sake of efficiency, or to determine when a robot failure has occurred.

The degree to which robot team members can actively pursue knowledge concerning team member abilities depends on the type of mission in which they are engaged. If they are on a *training* mission, whose sole purpose is to allow robots to become familiar with themselves and with their teammates, then the robots have more freedom to explore their capabilities without concern for possibly not completing the mission. On the other hand, if the robots are on a *live* mission, then the team has to ensure that the mission does get completed as efficiently as possible. Even so, as they perform the mission, they should take advantage of the opportunity to find out what they can about the robot capabilities that are demonstrated.

Thus, one of two high-level control phases are utilized for robot team members under L-ALLIANCE, depending upon the type of the team's mission. During training missions, the robots enter the *active learning* phase, whereas during live missions, they enter the *adaptive learning* phase.

Active Learning Phase

Clearly, the only way robots can independently learn about their own abilities and the abilities of their teammates is for the robots to activate as many of their behavior sets as possible during a mission, and to monitor their own progress and the progress of team members during task execution. Of course, on any given mission not all of the available behavior sets may be appropriate, so it is usually not possible to learn complete information about robot capabilities from just one mission scenario. However, the *active learning* phase allows the team to obtain as much information as possible through the active exploration of robot abilities. In this phase, the robots' motivational behaviors interact to cause each robot to select its next action randomly from those actions that are: (1) currently undone, as determined from the sensory feedback, and (2) currently not being executed by any other robot, as determined from the broadcast communication messages.

While they perform their tasks, the robots are maximally patient and minimally acquiescent, meaning that a robot neither tries to preempt another robot's ongoing task, nor does it acquiesce its own current action to another robot. Since robots at the beginning stages of learning do not yet know how long it may take them to perform their tasks, this maximal patience/minimal acquiescence feature allows them to try as long as needed to accomplish their tasks. Of course, if a robot has the ability to detect failure with certainty, then it can give up failed tasks to another team member.

During this active learning phase, each of a robot's motivational behaviors keeps track of the average time plus one standard deviation in the time required for that robot to perform the task corresponding to that motivational behavior's behavior set[1]. The motivational behavior is also responsible for cataloging the times and standard deviations required by other robots to perform that same task. In the case of robot failure, the actual time attributed to the failed robot is some penalty factor (greater than 1) times the actual attempted time. As stated earlier, the number of trials, μ, over which the average is maintained is fairly small; in my experiments, maintaining information over about 5 trials provided good results. It is important to note here that a robot r_i does *not* keep track of the task completion times for capabilities of other robots that r_i does not share. This allows the L-ALLIANCE architecture to scale favorably as the mission size increases.

Adaptive Learning Phase

When a robot team is applied to a "live" mission, it cannot afford to allow members to attempt to accomplish tasks for long periods of time with little or no demonstrable progress. The team members have to make a concerted effort to accomplish the mission with whatever knowledge they may have about team member abilities, and must not tolerate long episodes of robot actions that do not have the desired effect on the world. Thus, in the *adaptive learning* phase, the robots acquiesce (give up tasks) and become impatient (take over tasks) according to their learned knowledge and the control strategies described in the remainder of this chapter, rather than being maximally patient and minimally acquiescent as they are in the active learning phase. However, the motivational behaviors of each robot continue to monitor the robot's and other's performances during this phase, and update the average task completion times and standard deviations for the most recent μ trials.

4.4.3 Ordering Task Execution Based on Quality Measurements

Once a robot has learned quality measurements of its own performances and those of its teammates, how do the motivational behaviors of that robot interact to cause the

[1]The average time plus one standard deviation required for robot r_k to perform task $h_i(a_{ij})$, as measured by robot r_i, is referred to as $task\_time_i(j,k)$ in the L-ALLIANCE formal model.

robot to select its next action to pursue? For now, I deal strictly with the problem of how a robot determines which task to perform from a set of incomplete tasks that are not currently being attempted by any other robot. The next subsection deals with the issue of interrupting a robot's actions to recover from failures or to further improve the efficiency of the mission.

The answer to this action selection question largely determines the efficiency with which the robot team can perform its mission. The ideal is for the motivational behaviors to interact to cause each robot to select its tasks in such a way that the team as a whole minimizes the time or energy required to accomplish its mission. However, each robot is working with incomplete global information, since it at best knows solely about its own abilities to perform certain tasks and the quality with which its teammates perform those same tasks. In addition, each robot has a restricted view of the scope of the mission, since it can only sense the need for those actions that it is able to perform; robots are completely ignorant of any other tasks required by the mission that teammates may have to execute. But as I have already noted, the ALLIANCE learning problem is NP-hard, and thus we could not expect the robots to be able to derive an optimal selection of actions even if they did possess complete global information.

I investigated a number of approaches to this task ordering problem. My overriding concern in evaluating these approaches is the degree of vulnerability of the robot team to any type of component failure — either the failure of robots or, in particular, the failure of the communication system. If the robots are absolutely dependent upon the communication system to perform anything useful, then all of my efforts in creating robust, reliable, flexible, and coherent teams are lost with one component failure. Indeed, communication failure is not a problem to be taken lightly, as applications performed in the real-world offer many more challenges to the communication system than are present in, say, multi-processor communication[2]. Thus, robots cannot be required to wait to be "awarded a bid" (see chapter 8), or to receive permission from some other robot via a communicated message before starting on a task, because if the communication mechanism failed, the robots would accomplish nothing.

I therefore investigated three approaches in which each robot's next action selection is based upon the expected execution time of the tasks it is able to perform, or upon a random selection of actions. The following subsections describe these three task ordering approaches, which I call Longest Task First, Modified Shortest Task First, and Random Task Selection. Section 4.5.2 investigates the relative perfor-

[2]As anecdotal evidence of this problem, at a recent AAAI robot competition [Dean and Bonasso, 1993, pg. 39] held in what most would consider to be a very controlled environment, communication failure due to extreme RF noise from portable microphones, transmitters, two-way radios, and halogen lighting dimmers and starters caused havoc for several of the competing robots.

mances of these approaches.

Longest Task First

In the multi-processor scheduling community, a centralized greedy approach called Descending First Fit has been shown to result in mission completion times within 22% of optimal [Garey and Johnson, 1979] for identical processors. In this approach, the tasks are assigned to processors in order of non-increasing task length. Thus, I first attempted a distributed version of Descending First Fit to determine its effectiveness for our multi-robot application domain. The distributed version, which I call "Longest Task First", requires each robot to select as its next task that which is expected to take the robot the longest length of time to complete. The mechanism utilized to implement this approach is to have the $\delta\_fast_{ij}(t)$ and $\delta\_slow_{ij}(k,t)$ parameters of each motivational behavior to grow at a rate proportional to the expected task completion time (i.e. larger task times imply faster rates of impatience). The philosophy behind the Longest Task First approach is that the mission cannot be completed any quicker than the time required to execute the longest task in the mission. Thus, the team may as well start with the longest task and perform as many of the shorter tasks in parallel with that task as possible.

However, this distributed Longest Task First approach turned out to be disastrous for heterogeneous cooperative teams in which robot failures can occur. Recall that robots in the most general cooperative teams satisfy condition 4, which states that different robots are different, and thus may perform the same task with quite different levels of performance. The result for these teams using the Longest Task First approach was that in general, each task in the mission was completed by the robot team member with the worst ability to accomplish that task.

Modified Shortest Task First

As a logical next step to this approach, I studied the dual of the Longest Task First approach — Shortest Task First — in which the motivational behaviors interact to cause each robot to select as its next action that which it expects to perform the quickest. The centralized version of this greedy approach for identical multi-processors has been shown to result in minimizing the *mean flow* of the mission, which means that the average completion time of the tasks in the mission is minimized [Conway *et al.*, 1967]. However, I do modify the pure Shortest Task First technique a bit to compensate for the fact that heterogeneous robots have different sets of tasks which they are able to pursue. If a mission includes tasks that can only be accomplished by one robot, then it makes sense for that robot to first select the actions which it alone is able to accomplish. Extending this principle even further, I can require a robot to first select from among those actions which it expects to perform better than any other robot on the team, and only after these tasks are complete continue on to select tasks which the robot expects other robots on the team could accomplish quicker. In this second case, I prefer a robot to at least attempt tasks that it may not perform as well as other robot team members rather than remaining idle while the better robots are working on other tasks. Even with their inferior capabilities, the slower robots may still be able to complete tasks during the time in which the better robots are occupied with other tasks, thus reducing the overall mission completion time.

Thus, the interaction of the motivational behaviors under the Modified Shortest Task First approach effectively divides the tasks a robot can perform into two categories:

1. Those tasks which robot r_i expects to be able to perform better than all other robots present on the team.

2. All other tasks r_i can perform.

This two-category mechanism works via the *learned_robot_influence* function defined in the formal L-ALLIANCE model in section 4.6, which initially "blinds" the robot to those tasks in the second category. This causes the robot to first select from among those actions which it feels it can perform better than any other robot team member. If no tasks remain in the first category, the robot is initially satisfied that the tasks will be accomplished by other team members. However, I do not want the robot to sit around doing nothing forever just because other team members might possibly be able to accomplish the tasks in the second category. I remedy this by having each robot also be motivated by a boredom factor, which increases whenever the robot is doing nothing. Once the boredom factor gets high enough, it causes the robot to "forget" that another robot is present that can perform one of the actions in the second category, thus leading the robot to select some pertinent action. The robot then continues task execution in this manner until the mission is complete.

The selection of the shortest task within each category is accomplished by two parameters in the L-ALLIANCE formal model: $\delta\_slow_{ij}(k,t)$ and $\delta\_fast_{ij}(t)$. These two parameters provide the rate of impatience of a robot to perform some task either when another robot is performing the task, or when no other team member has begun the task, respectively. Thus, to cause a robot to select the task it expects to perform the quickest, these rates of impatience for each behavior set should grow at a rate inversely proportional to the expected task completion time — that is, small task completion times have large rates of impatience. Section 4.6 discusses the details of how this is implemented.

Modified Random Task Selection

As a baseline against which to compare the other approaches, I also studied a random selection of tasks. In this case, the motivational behaviors of the robots effectively divide the tasks into the same two categories used in the Modified Shortest Task First approach. However, in this case, the motivational behaviors work together in such a way that tasks are randomly selected, initially from the first category, and then from the second category of tasks. The results from this study, described in section 4.5.2, were actually quite enlightening, leading to the task ordering approach discussed next.

4.4.4 Knowing When to Give Up

The third major issue in L-ALLIANCE is providing each robot team member with the ability to determine when it should become impatient with other robot performances, and when it should acquiesce its own current action. This issue affects not only the robot team's response to failures and difficulties in the environment, but also the efficiency of the action selection. If these impatience and acquiescence factors are set too low, then the robot team thrashes between tasks, perhaps seriously degrading the team efficiency. On the other hand, if these factors are set too high, then the robot team wastes time, and perhaps energy, waiting on a failed robot to complete a task.

Three primary parameters in L-ALLIANCE determine a robot's response to its own or to other robot performances: $\phi_{ij}(k,t)$ (robot impatience), $\psi_{ij}(t)$ (robot acquiescence to another robot), and $\lambda_{ij}(t)$ (robot acquiescence to try another task). The first two parameters concern a robot's response to the actions of its teammates, whereas the third ($\lambda_{ij}(t)$) affects a robot's response to its own performance in the absence of impatient team members. A number of different strategies for setting these impatience and acquiescence rates can be used, all of which are based upon the knowledge each robot gains about its own abilities and/or the abilities of its teammates. Predicting the relative merits of these strategies, however, is more difficult, since a number of factors influence the team's performance, such as the size of the team, the size of the mission, the degree of task coverage, and the extent of robot heterogeneity. Therefore, I conducted a number of studies of these strategies in combination with the various approaches to ordering task execution discussed in the previous section. Section 4.5 discusses the results of these investigations. However, I first introduce the three strategies I investigated in the following subsections; these strategies are summarized in table 4.1.

Three Impatience/Acquiescence Update Strategies

Strategy	Impatience ($\phi_{ij}(k,t)$)	Acquiescence ($\psi_{ij}(t)$)
I	own time	own time
II	own time	minimum time of team
III	time of robot performing the task	own time

Table 4.1: Basis for setting the impatience and acquiescence parameters for a given task within a given robot, for each of three strategies.

Strategy I: Distrust Performance Knowledge about Teammates

The first impatience/acquiescence parameter update strategy takes a minimalist approach to the problem by only requiring the robot to use the knowledge it learns about its own performance; robots are not required to know anything about the capabilities of their teammates. This strategy is the one most likely to be used when a robot team is first formed — that is, before the team members have had an opportunity to learn about their teammates' capabilities. This strategy can also be used when robots have little confidence in the knowledge that they have learned about other robots, perhaps due to significant environmental changes that have rendered earlier quality measurements invalid.

Under strategy I, a robot holds other robots to the same standard by which it measures itself. Thus, if a robot r_i knows that it should be able to complete a certain task $h_i(a_{ij})$ in a certain period of time t, then it becomes impatient with any other robot r_k that does not complete $h_i(a_{ij})$ in that same period of time. Of course, since r_i is holding itself to its own standards, then it is willing to acquiesce its task after working on it for a period of time t without task completion.

The expected group behavior resulting from strategy I is for better robots to begin execution of tasks being pursued by worse robots, but only after the worse robots have attempted their tasks for a period of time determined by the better robots' own expected performance time. However, a worse robot will not be willing to give up its task until it feels it has had a fair chance to complete the task according to its own performance expectations. As it turns out, this mismatch between impatience and acquiescence rates across robots leads to energy inefficiencies due to more than one robot working on the same task at the same time.

Strategy II: Let the Best Robot Win

The second strategy for setting the impatience and acquiescence factors endows the robot team with the character of "striving for the best". Under this strategy, a robot holds itself to the performance standard of the best robot it knows about in the group, for each task to be accomplished. Thus, if a robot r_i has learned that the quickest expected completion time required by a robot team member for a task $h_i(a_{ij})$ is t, then r_i will acquiesce task $h_i(a_{ij})$ to another robot if r_i has attempted $h_i(a_{ij})$ for a time longer than t. On the other hand, robot r_i will become impatient with a robot r_k which is performing task $h_i(a_{ij})$ only after r_k has attempted the task for a longer period of time than r_i believes that it, itself, needs to accomplish $h_i(a_{ij})$.

Implicit in this strategy is the assumption by an acquiescing robot that other robots know their own performance levels better than does the acquiescing robot. Their behavior can be summarized with the statement: "If I think I'm not doing very well, and you think you can do better, then I'll give up." In this strategy, the acquiescing robot r_k does not compare its own expected performance with its knowledge about the expected performance of the impatient robot, r_i. If it did, r_k might at times find that it expects r_i to actually perform the task worse than r_k could. However, since r_k assumes that r_i has better knowledge about r_i's abilities than r_k does, r_k gives up its task.

The expected group behavior resulting from strategy II, then, is for better robots to take over tasks from worse robots, with the worse robots giving up their tasks when they feel that (1) they are not successful, and (2) that another robot on the team can do a better job.

Strategy III: Give Robots a Fighting Chance

The third strategy that I introduce for updating the impatience and acquiescence factors results in a "kinder and gentler" robot team that judges performances of robot team members based on each team member's own individual expected performance, rather than its comparison to other team members' performances. Under strategy III, a robot r_i becomes impatient with robot r_k's performance only after r_k begins performing worse than its (r_k's) normal abilities. Otherwise, robot r_i will not become impatient with r_k, even if r_i expects that it could perform r_k's task much better. Likewise, each robot expects the same courtesy, and is therefore unwilling to acquiesce its own action until it believes it has had a fair chance to accomplish the task, according to its own expected performance requirements.

Thus, the expected group behavior resulting from strategy III is for robots to exhibit a first-come-first-served approach to action selection, not interrupting other agents nor acquiescing to other agents until deteriorated functionality is demonstrated.

4.5 Experimental Investigations of Control Strategies

The two key issues that must be resolved in L-ALLIANCE are determining the relative performances of the three impatience/acquiescence update strategies introduced in section 4.4.4 and determining the appropriate task ordering the robot team should use to maximize efficiency. This section presents my empirical investigations of these two issues. As a result of these investigations, I discovered a fourth approach to task ordering which, combined with the appropriate impatience/acquiescence update strategy, results in more efficient team performances. I then compare the results of this best approach (called strategy IV) to the optimal solution for those problems in which the optimal solution can be computed.

4.5.1 Effect of Impatience/Acquiescence Update Strategy

As I discussed in section 4.4.4, a number of strategies are possible for updating the robot impatience and acquiescence parameters based on information learned about robot performance. Three in particular that I investigated are Distrust Performance Knowledge (Strategy I), Let the Best Robot Win (Strategy II), and Give Robots a Fighting Chance (Strategy III). In these experiments, I used the Modified Shortest Task First approach to ordering tasks described in section 4.4.3. This approach to task ordering is varied in the next subsection.

To determine the relative merits of these strategies, I ran a large number of test runs in simulation, comparing the results of the strategies in terms of the time and the energy required to complete the mission. In this study, simulation runs offered much more opportunity to study the effects of a number of factors on the performance of the three strategies than would be possible using our laboratory's limited number of physical robots with relatively fixed physical capabilities. In these experimental runs, I investigated the performances of the strategies as functions of the relative task coverage, the relative mission size, the degree of heterogeneity across robots, and the number of robots. As we shall see in the remainder of this subsection, I discovered that each of the three strategies works well in some situations, but not as well in others.

Data Collection Methods

In making observations about the relative performance of the three strategies, it is important to not generalize conclusions based on too few specific examples, since the outcome of any specific example can often be quite different from the average performance of the strategies over a range of similar mission scenarios. Thus, I collected the data by first varying the number of robots on the team (n) from 2 to 20, the number of tasks the team must perform (m) from 1 to 40, the task coverage from 1 to 10, and the heterogeneity from 0 percent to 3200 percent[3]. For this study, I composed the missions of completely independent subtasks involving no ordering constraints, I distributed the capabilities uniformly across the robots based upon the given task coverage, and I assumed the same task coverage for all tasks required by the mission. To obtain a given percent of heterogeneity, x, I first randomly selected a task length, l, between 0 and 500 for a given task, t, and then, for each robot r with the ability to perform that task, the length of time required for r to complete t was defined to be $l \times (1 + \frac{x}{100} \times y)$, where y is a number between 0 and 1, chosen randomly for each robot, for each task. Thus, a given degree of heterogeneity, x, means that any two robots sharing the ability to perform a given task can vary *up to* $x\%$ in the time required for them to complete that task.

In this study, I did not address the issue of robot failure directly. However, the strategies I studied here do not cause robots to distinguish between task failure in other robots and slower completion times in those robots. Thus, since a task failure is treated no differently from a less efficient robot, I can view robot failures as being included in the heterogeneity difference across robots.

For ease of discussion throughout this section, I define a *scenario* as a 4-tuple (n, m, *task_coverage*, *heterogeneity*) of a given run of this simulation. Then, for any scenario defining the number of robots, the size of the mission, the level of task coverage, and the percent of heterogeneity, I ran 200 different test runs, varying the assignment of tasks to robots and the quality of their performance randomly according to the given values of task coverage and heterogeneity. The average over these 200 runs was then considered the characteristic performance of that scenario.

Results and Discussion

In analyzing the data from these test runs, it became apparent that the performance of the strategies was dependent upon all of the factors I studied: the relative task coverage, the relative size of the mission, and the degree of heterogeneity in the robots. (By *relative* mission size and task coverage, I mean the mission size and task coverage normalized by the number of robots on the team.) The performance was

[3]In this context, robot team members can actually be *heterogeneous* in two obvious ways: (1) they can have different behavior sets that give them the ability to perform different tasks, and (2) they can share the ability to perform the same task, but demonstrate different *qualities* of performance of that task (e.g. the time required to complete the task may vary). For this study, the first type of heterogeneity is included in the task coverage of the team. Thus, when I speak of varying heterogeneity in this subsection, I am referring to the degree of difference in the qualities of performance of the same task by the subgroup of robots which can perform that task.

also dependent upon the degree to which condition 3 — Progress When Working (as introduced in chapter 3) — was assumed to hold. If the effect of the robot's actions cannot be sensed through the world until the task is completed, then reassigning the task to another robot requires the second robot to completely re-do the first robot's actions. In this case, the Progress When Working condition does not hold. On the other hand, if the robot's actions *can* be fully sensed through the world, then a robot that takes over another robot's actions only has to complete the remainder of the task, thus avoiding duplication of effort. Here, the Progress When Working condition *does* hold. In between these two extremes lie situations requiring a duplication of some, but not all of the actions taken by the first robot to accomplish the now-acquiesced task. In this study, I consider the effects of the two extremes on the performance of the three impatience/acquiescence strategies.

As I discuss these results, it is helpful to keep in mind two primary differences in the operation of these three strategies. They differ in:

1. The method which determines which robot from a group of idle robots gets to perform a task not yet underway (i.e. the initial action selection choice).

2. The method by which a robot overrides the performance of another robot.

I explain my results in terms of these two primary differences between the strategies.

To analyze the results of a given scenario, I noted the comparative performances of the three impatience/acquiescence update strategies of that scenario at the point ($task\_coverage/n$, m/n) on a plot of relative task coverage versus relative mission size. Repeating this process for all of the scenarios results in the time and energy profiles shown in figures 4-1 and 4-2, respectively. Let us examine these results in more detail.

I first note that the three strategies are equivalent for teams in which the degree of heterogeneity is 0, regardless of any other factors, because I have assumed a uniform distribution of tasks across robots for a given task coverage. Thus, since any robot can perform any of its tasks as well as any other robot, the action selection strategy does not matter as long as robots do indeed select tasks to pursue. Since all of these strategies do cause robots to pursue some incomplete task, we observe no differences when the degree of heterogeneity is 0.

For all other robot teams, however, four distinct areas of relative strategy performances are found in terms of both time and energy usage, as shown in figures 4-1 and 4-2: regions 1, 2, 3, and 4. Each of these regions are defined in terms of the ratio of task coverage to mission size (m), as follows:

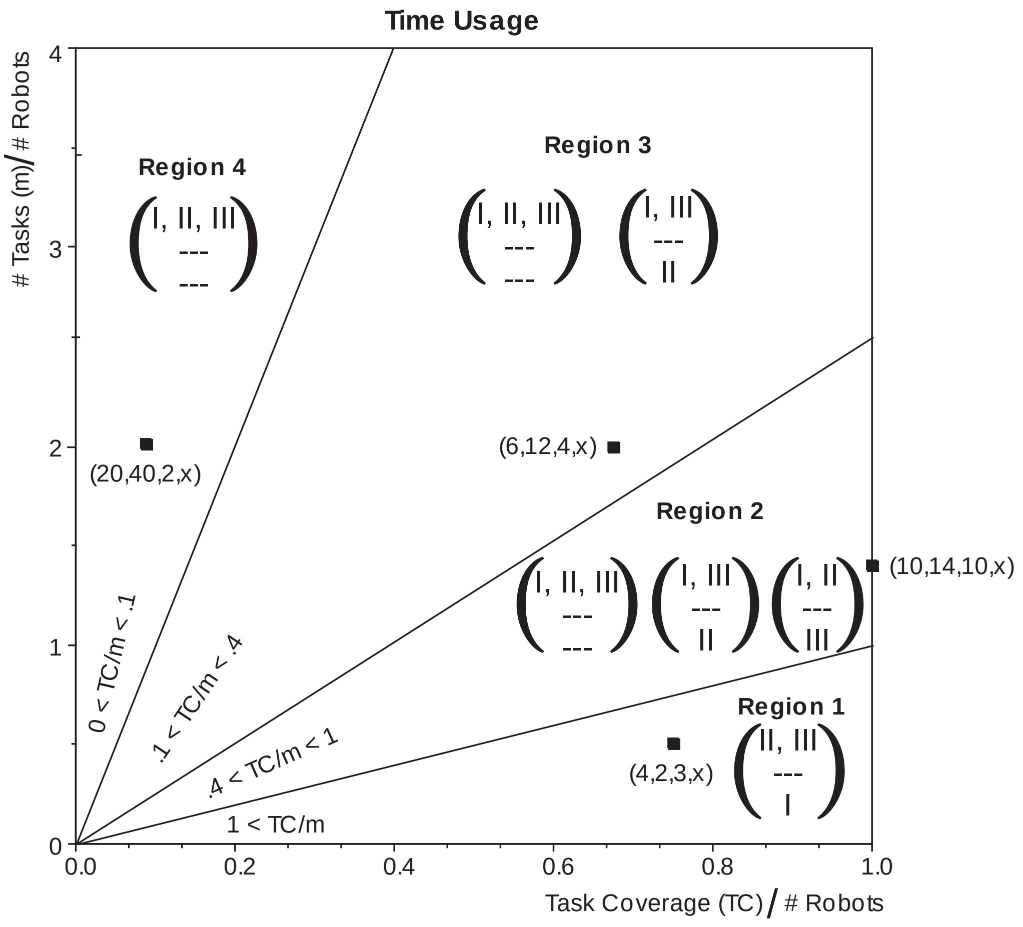

Figure 4-1: Summary of time usage for three impatience/acquiescence strategies. In this and in the next figure, the strategy numbers (I, II, III) in large parentheses indicate the relative performance of the three strategies in each of the regions, where the first row in parentheses indicates the best performer(s). The four points noted with small black squares are exemplar misions of their corresponding regions, whose time and energy usages are shown in figures 4-3 through 4-10. The values in the small parentheses by each of these four points describe the corresponding cooperative scenario by giving the number of robots, the number of tasks, and the task coverage (in that order) used in the exemplar.

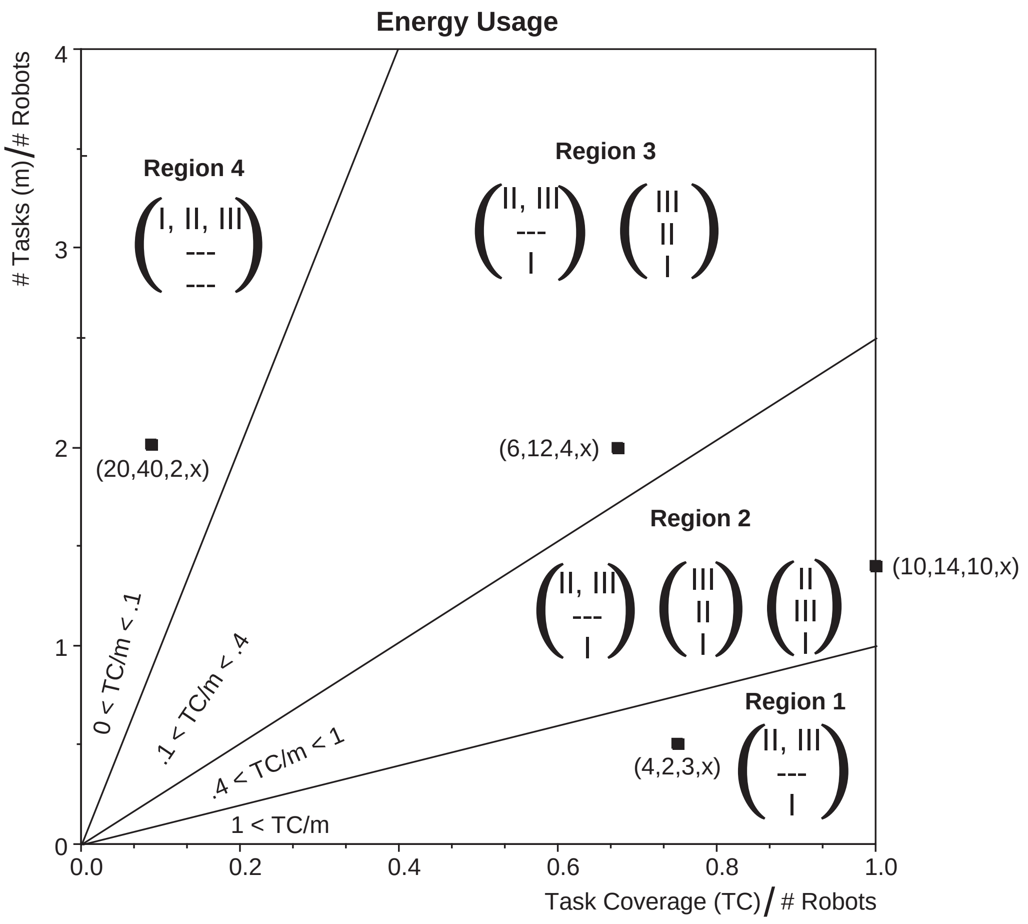

Figure 4-2: Summary of energy usage for three impatience/acquiescence strategies. (Refer to the previous figure for an explanation of the notation.)

Region 1: $1.0 < task\_coverage/m$
Region 2: $0.4 < task\_coverage/m < 1.0$
Region 3: $0.1 < task\_coverage/m < 0.4$
Region 4: $0.0 < task\_coverage/m < 0.1$

Intuitively, region 1 corresponds to those scenarios in which many robots are able to perform a relatively low number of tasks. In this region, not enough work is available to occupy all the robots; thus, the primary issue is determining which robots will be "allowed" to perform which tasks. As we progress to regions 2, 3, and then 4, we encounter scenarios in which progressively fewer robots on average are available to perform any given task in the mission. As we shall see, the average number of robots that "compete" to execute each task plays a large role in the relative performances of the three impatience/acquiescence update strategies. Of course, the boundaries between these regions are not crisp, as the transition from one region to the next is smooth. Nevertheless, they do indicate general trends that are interesting to understand.

First let us consider region 4, which consists of those scenarios involving a very low task coverage to mission size ratio. What we discover in this area is that the choice of impatience/acquiescence update strategy makes no difference to the team performance because, in this region, either robots have virtually no overlap in their abilities, or the mission is large enough that robots need not "compete" for tasks to perform. Low overlap in abilities implies that only one allocation of tasks to robots is possible, and thus the controlling strategy makes no difference. Figures 4-3 and 4-4 show a typical time and energy performance of the strategies for scenarios in region 4.

Now let us consider the relative performances of teams controlled by the three strategies in region 1. Figures 4-5 and 4-6 illustrate a typical performance of the three strategies for scenarios in this region, showing the time and energy results of four robots performing two tasks, in which 75% of the robots have the capability to perform each task. This combination of task coverage and mission size indicates that most of the robots on the team are able to perform most of the tasks required by the mission. However, because there are so few tasks to perform per robot, the overall group performance is very much dependent upon the initial action selection choice of each robot, rather than the method by which the robots elect to override the actions of teammates. Some robots may elect to perform a task, while other robots may elect to remain idle due to the presence of team members that are thought to be able to accomplish the tasks more efficiently. Under strategy I, all robots select the task which they expect to be able to complete the quickest, without the use of knowledge about the capabilities of teammates. If more than one robot selects the same action, the fixed tie-breaking mechanism determines which robot wins, regardless of their

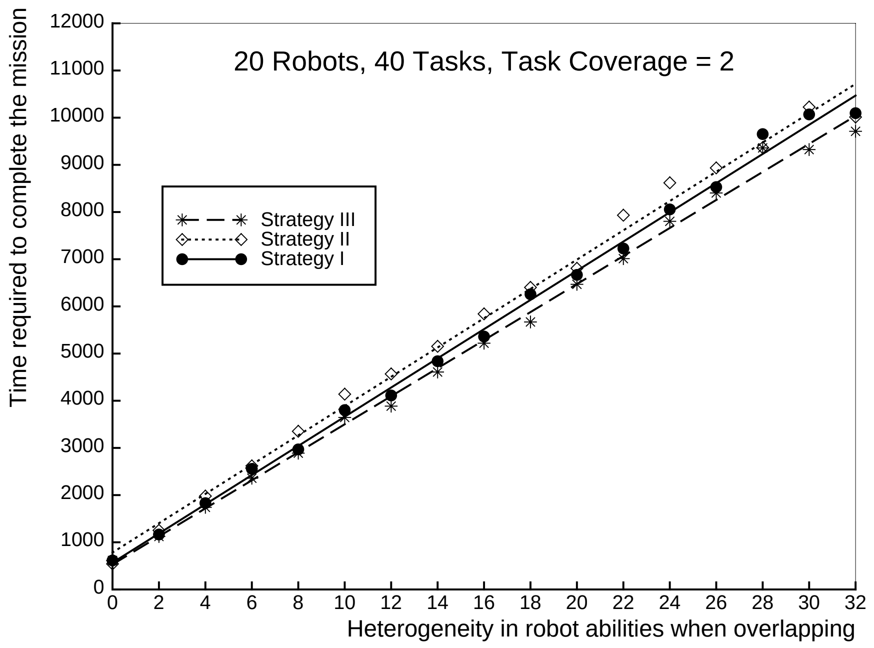

Figure 4-3: An average time performance of the three impatience/acquiescence strategies in region 4. Each data point shown in this and in the next 7 figures is an average value over 200 runs of the corresponding scenario. Refer to the text for more details.

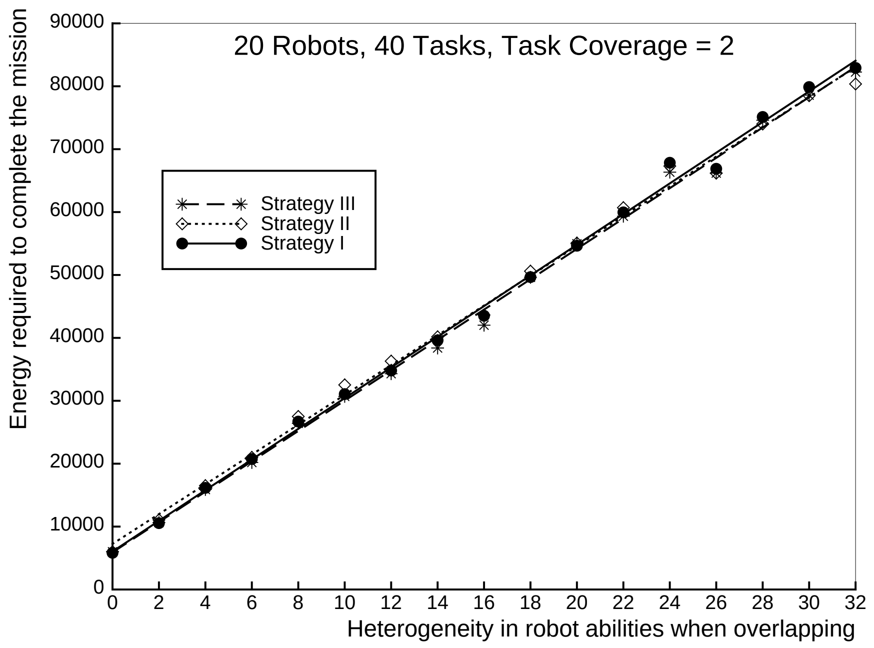

Figure 4-4: An average energy performance of the three impatience/acquiescence strategies in region 4.

relative capabilities. Thus, each task may not be executed by the robot which can perform that task the best.

On the other hand, under strategies II and III, robots select their actions with regard to the expected capabilities of their teammates. Thus, robots are initially motivated to perform only those tasks that they should be able to complete quicker than any other robot team member. Since so few tasks are to be performed relative to the size of the team, it is quite likely that on average, each task will be completed by the robot who can perform that task most efficiently. Thus, strategies II and III perform much better than strategy I in region 1 in terms of both time and energy.

As we move into region 2, an interesting phenomenon occurs with the relative performances of the three strategies. Moving away from region 1 into region 2 means that the relative performances of the strategies will be affected not only by the initial action selection choice which influenced the performance in region 1, but also by the mechanism by which robots override the performances of their teammates. The override mechanism becomes more important in this region because there are fewer available robots per task — on average less than one, but greater than about .4. This in turn means that some of the robots will make more than one action selection, while the remaining idle robots will be watching to override their performance if the successive rounds of action selection do not result in the better robots working on

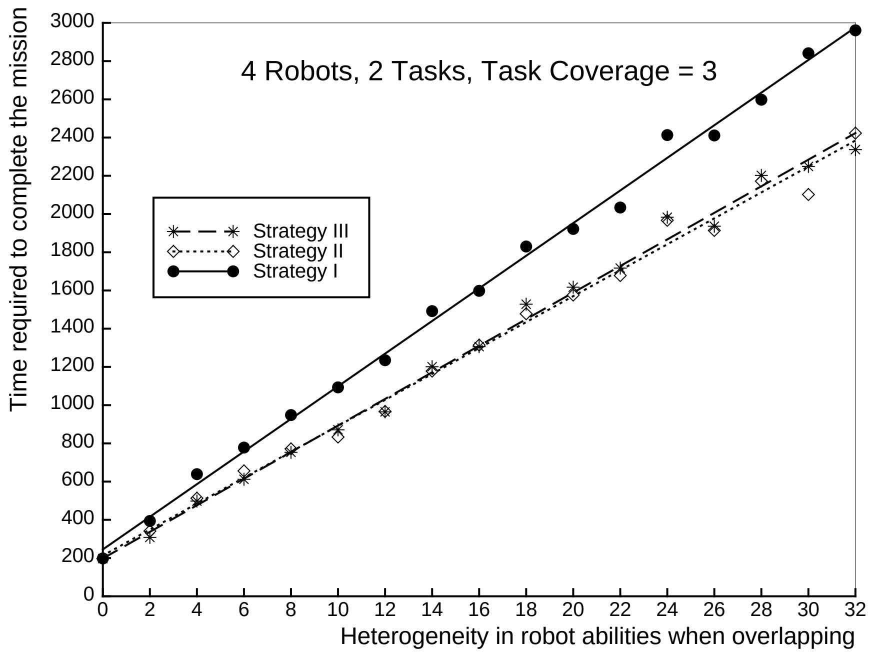

Figure 4-5: An average time performance of the the three impatience/acquiescence strategies in region 1.

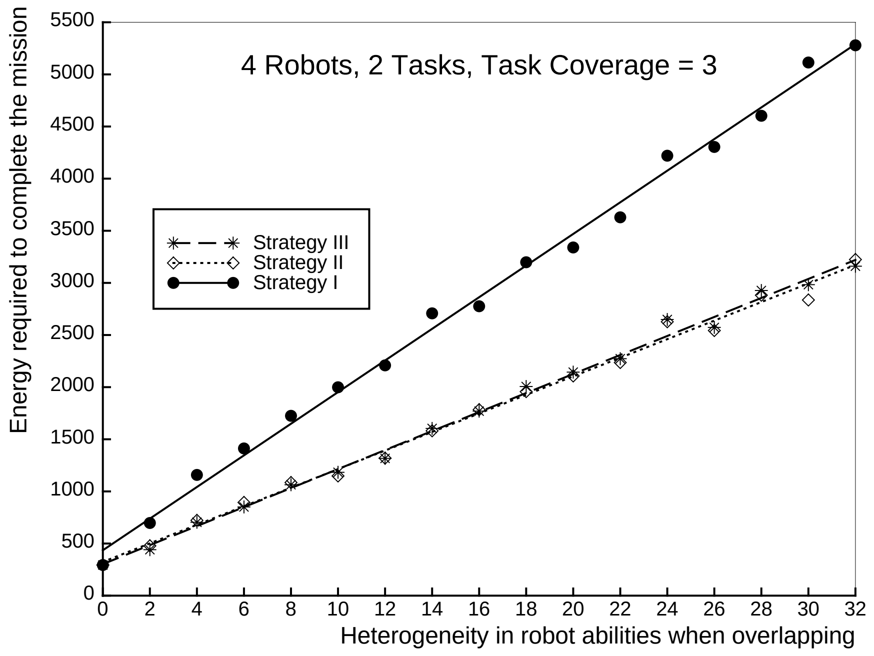

Figure 4-6: An average energy performance of the three impatience/acquiescence strategies in region 1.

the tasks at which they excel. What we discover in this region, then, is that the relative performances of strategies II and III vary based upon the heterogeneity of the robots when they share task capabilities, and the degree to which the Progress When Working condition holds.

An example helps illustrate this point. Let us suppose that a cooperative team is composed of two robots, r_1 and r_2, both of which can perform the three tasks required on the mission — a_1, a_2, and a_3. This scenario of two robots, three tasks, and a task coverage of two lies at point (1, 1.5) on the time and energy profile graphs shown in figures 4-1 and 4-2, and thus is in region 2. Further suppose that r_1 performs its tasks very efficiently, requiring 75 time units to perform each of its tasks, whereas robot r_2 is less efficient, requiring 100 time units to perform each of its tasks. Initially, r_1 selects the action it can perform the quickest, say a_1, and r_2 sits idle for a while, since it knows that it cannot perform its tasks efficiently relative to the other robots on the team. However, while r_2 is idle, its environmental feedback indicates that tasks still need to be performed; thus, r_2 becomes bored, leading it to elect to perform one of the tasks not yet underway — say a_2. In the meantime, assume r_1 completes a_1, goes on to also complete a_3, and is now waiting on r_2 to complete task a_2. Under strategy II ("let the best robot win"), r_1 would become impatient with r_2 after r_2 had attempted the task for 75 time units, at which point r_2 would acquiesce task a_3. Similarly, under strategy I ("distrust performance knowledge"), r_1 would become impatient with r_2 after 75 time units, although r_2 would not give up a_3 until 100 time units had passed, leading to both robots performing a_3. Under strategy III ("give robots a fighting chance"), r_1 would allow r_2 to continue its execution of a_3 for 100 time units.

What affect does this have on the relative time and energy usage of the team in region 2? The answer depends on the degree to which the Progress When Working condition holds. First, let us consider the relative performances of strategies II and III. Figures 4-7 and 4-8 show a typical performance of the strategies in region 2. When the Progress When Working condition holds, robot will be able to fully sense the effect of other robot's actions through the world. Thus, strategy II will outperform strategy III in terms of both time and energy for any degree of heterogeneity (except full homogeneity) because a quicker robot will take over the task from a slower robot without having to duplicate the slower robot's actions. However, when the Progress When Working condition does not hold, robots will not be able to sense the effect of other robot's actions through the world until the task is complete. Thus, overriding the action of another robot leads to the task being performed again in its entirety. This is actually useful when robots are highly heterogeneous, since a more efficient robot will be able to perform the entire task in less time than the slower robot needed to complete the task. But it is not useful when robots are only mildly heterogeneous, since the time required for the faster robot to fully execute the task is longer than the

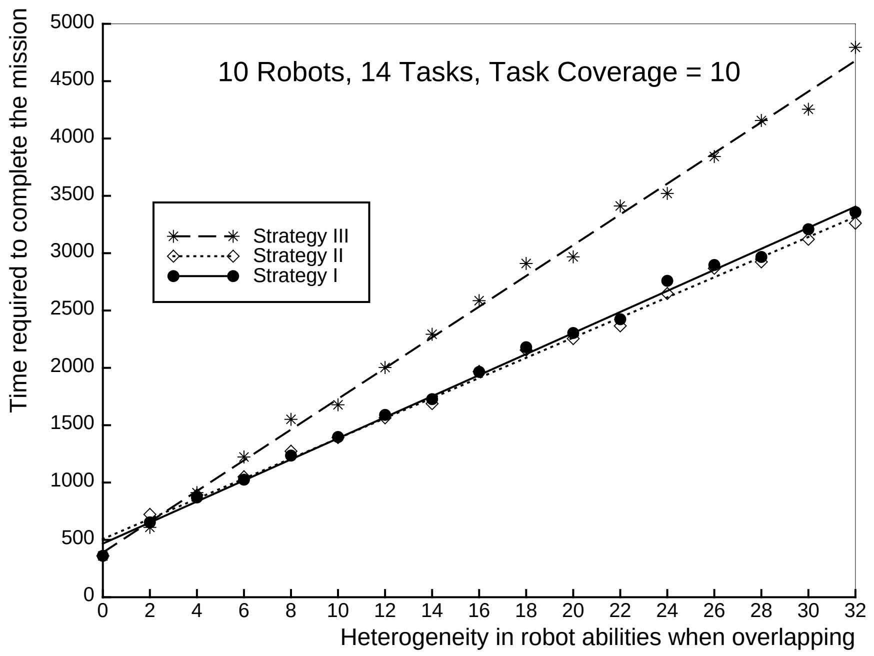

Figure 4-7: An average time performance of the three impatience/acquiescence strategies in region 2.

remaining time required for the slower robot to complete that task. Thus, strategy II outperforms strategy III when the Progress When Working condition holds for any degree of heterogeneity, or when the Progress When Working condition does not hold, but the team is composed of highly heterogeneous robots. On the other hand, strategy III outperforms strategy II for mildly heterogeneous teams when the Progress When Working condition does not hold.

Now let us examine the performance of strategy I relative to that of strategies II and III in region 2. As noted earlier, the override strategy plays a critical role in the relative performances of the three strategies in this region. Since strategy I can often lead to more than one robot attempting the same task at the same time, its relative performance depends upon the degree to which the robots interfere with each other. If we assume no interference, then the strategy I override mechanism is not harmful in terms of time because the two robots merely continue working until one of them has completed the task, regardless of whether or not the Progress When Working condition holds. Strategy I therefore matches the time performance of the better of strategies II and III in region 2 for any given degree of robot heterogeneity. However, this same override mechanism causes strategy I to perform quite badly in this region in terms of energy usage, due to these replicated actions.

Moving into region 3, we are presented with scenarios involving a fairly low ratio

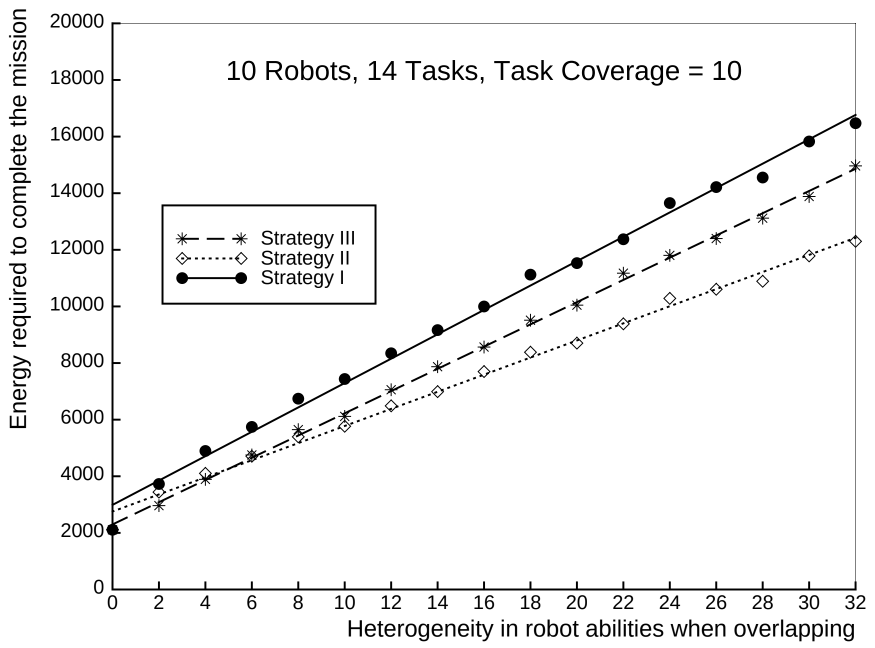

Figure 4-8: An average energy performance of the three impatience/acquiescence strategies in region 2.

of task coverage to mission size. Figures 4-9 and 4-10 illustrate a typical performance for the three strategies in this region. Here, missions involve plenty of work for each robot to perform; thus, it does not make as much sense in this region for a robot to override the performance of a teammate when there is a good deal of unfinished work remaining to be accomplished. However, since strategies I and II do not make a distinction between tasks not yet attempted and tasks being performed by poorer robots, we find that strategies I and II tend to get bogged down trying to improve the performance of other robots even when tasks are available that no robot is pursuing. This turns out not to be a significant problem for the time metric when the Progress When Working condition holds, since the the overriding robot need not repeat the entire task. However, when the Progress When Working condition does not hold, strategy II in particular is penalized, because the overriding robot has to repeat the entire task, whereas with strategy I, both robots continue to work on the task until the quickest robot has completed the task. Of course, this replication does hurt strategy I in terms of energy usage, so it performs the poorest in this region for energy usage, regardless of other factors. Strategy III does not suffer from this problem because it does not cause a robot r_i to override another robot just because r_i thinks it (r_i) can perform the task quicker. Thus, it performs well in terms of both time and energy in region 3.

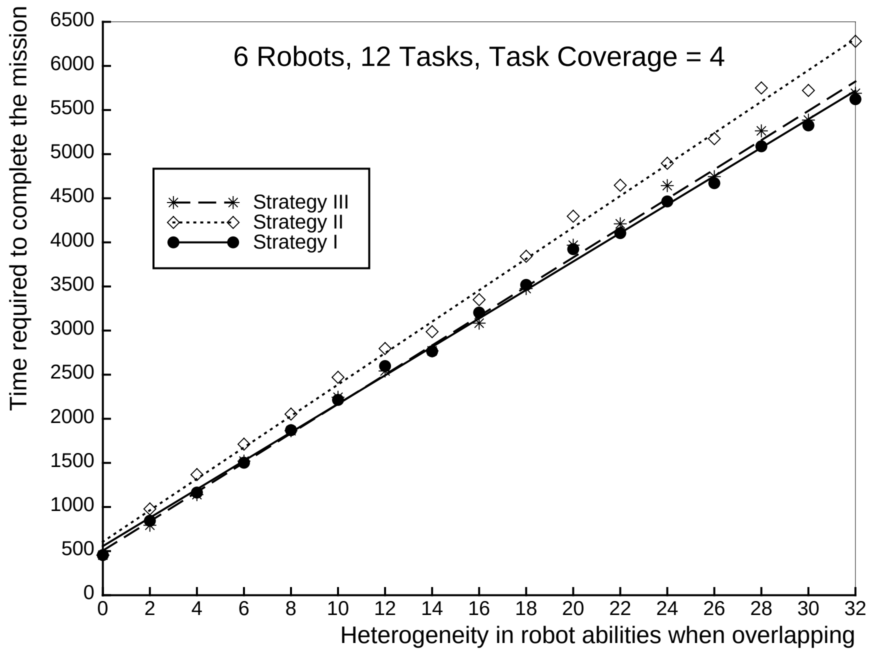

Figure 4-9: An average time performance of the three impatience/acquiescence strategies in region 3, assuming the Progress When Working condition does not hold.

We can also note that heterogeneity does not play much of a role in the relative strategy performances in this region. Again, this is due to the increased amount of work available for each robot to perform, which causes robots to be better off working on tasks that have not yet been started, rather than worrying about efficiency override considerations due to heterogeneity.

In summary, we see that the performance of the three impatience/acquiescence update strategies is dependent upon a number of factors: the relative task coverage, the relative mission size, the degree of robot heterogeneity, and the degree to which the Progress When Working condition holds. Tables 4.2 and 4.3 summarize these results by giving the preferred strategy for each combination of these factors. What we find is that strategies II ("let the best robot win") and III ("give robots a fighting chance") perform well in a large proportion of the scenarios in terms of both time and energy, whereas strategy I performs well in terms of time for many scenarios. As could be expected, the reasons why the strategies perform poorly in some scenarios are the same reasons why they perform well in others. For example, strategy II performs well for highly heterogeneous robot teams because it is quick to override. Yet, it also performs poorly in region 3 because it is *too* quick to override. Likewise, strategy III performs well for mildly heterogeneous robot teams because it does not readily cause overrides. However, when it performs poorly, it is because it causes robots to be too

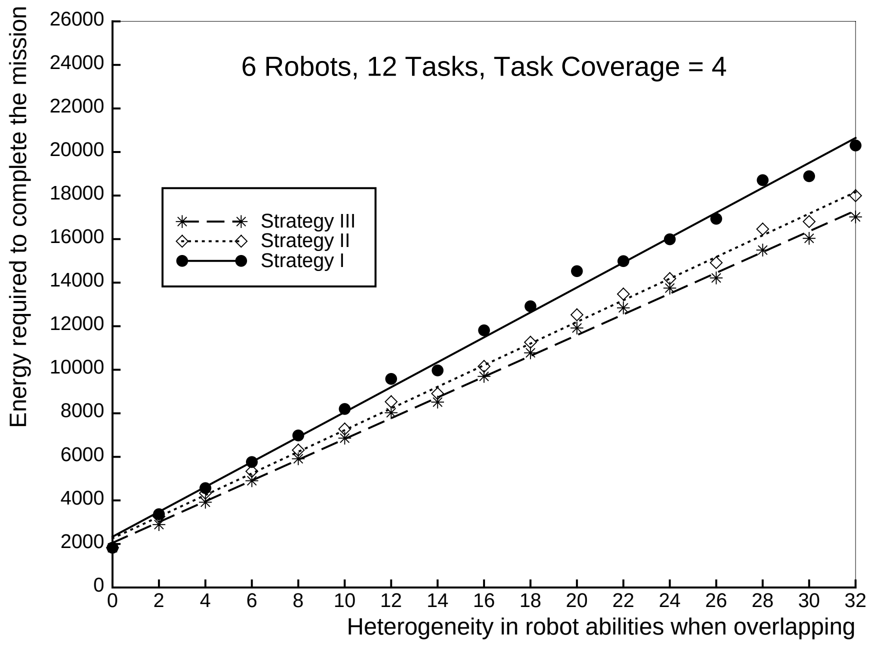

Figure 4-10: An average energy performance of the three impatience/acquiescence strategies in region 3, assuming the Progress When Working condition does not hold.

slow to override.

These results suggest an alternative method as a variation on strategy II which should improve its performance in region 3 when the Progress When Working condition does not hold. Under this alternative approach, robot r_i should effectively divide its tasks into the following two revised categories:

1. Those which robot r_i expects to be able to perform better than all other robots present on the team, *and which no other robot is currently pursuing.*

2. All other tasks r_i can perform.

This division of categories should eliminate strategy II's poor performance in region 3 when the Progress When Working condition does not hold, thus making it more suitable for general use. The remaining area of poor performance for strategy II — mildly heterogeneous robots in situations in which the Progress When Working condition does not old — cannot really be fixed without damaging the ability of the team to respond to robot failures.

Preferred Strategy: Time as Metric

Progress When Working?	Region	Heterogeneity	Strategy
Yes	1	—	II or III
Yes	2	High or Mild	I or II
Yes	3	—	I, II, or III
Yes	4	—	I, II, or III
No	1	—	II or III
No	2	Mild	I or III
No	2	High	I or II
No	3	—	I or III
No	4	—	I, II, or III
—	—	None	I, II, or III

Table 4.2: Summary of preferred impatience/acquiescence strategies for time as the performance metric. The preferred strategy is a function of whether the Progress When Working condition holds, the number of robots (n), the task coverage, the number of tasks (m) and the degree of heterogeneity when robot capabilities overlap. The values of High, Mild, and None for degree of heterogeneity stand roughly for the following: High is greater than 600%, Mild is less than 300%, and None is 0%.

Preferred Strategy: Energy as Metric

Progress When Working?	Region	Heterogeneity	Strategy
Yes	1	—	II or III
Yes	2	High or Mild	II
Yes	3	—	II or III
Yes	4	—	I, II, or III
No	1	—	II or III
No	2	Mild	III
No	2	High	II
No	3	—	III
No	4	—	I, II, or III
—	—	None	I, II, or III

Table 4.3: Preferred impatience/acquiescence strategies when energy is the performance metric.

4.5.2 Effect of Task Ordering Approach

As discussed in section 4.4.3, I initially investigated three approaches for allowing a robot to determine which task to select from those tasks that are not already being attempted by any robot — the Longest Task First approach, the Modified Shortest Task First approach, and the Modified Random Task Selection approach. I dismissed the Longest Task First approach quickly, since it gives dismal results for heterogeneous robot teams in which failures occur. I then compared the relative performance of the Shortest Task First approach with the simple Modified Random Task Selection approach. If a simple random selection of the next task performs just as well as the Modified Shortest Task First approach, then the control strategy would be less dependent upon knowledge of other robot capabilities.

To investigate this question, I performed a similar set of simulation experiments as discussed in the previous subsection. These simulation experiments allowed me to vary the number of robots, the size of the mission, and the heterogeneity of the robots in many more ways than would be possible on my available collection of physical robots. In these experiments, I studied the relative effects of the Modified Shortest Task First approach and the Modified Random Task Selection approach when using each of the three impatience/acquiescence strategies discussed in sections 4.4.4 and 4.5.1. As we shall see in the following sections, the Modified Shortest Task First approach performed much better than the Modified Random Task Selection approach for teams controlled using impatience/acquiescence update strategy I (Distrust Performance Knowledge about Teammates), and performed somewhat better than the Modified Random Task Selection approach for teams controlled using impatience/acquiescence update strategy III (Give Robots a Fighting Chance). However, it actually performed worse than the Modified Random Task Selection approach for impatience/acquiescence update strategy II (Let the Best Robot Win). The following sections discuss these results.

Data Collection Methods

The data collected for these experiments was obtained using very similar methods to those described in the previous subsection. The only difference is that in these runs, the motivational behaviors interacted to cause the robots to select their next action from category 1 (those actions the robot expects to be able to perform better than any other team member) or from category 2 (all other actions the robot can perform) using a random selection rather than a shortest task first selection. As in the previous section, 200 runs for a given scenario (number of robots, number of tasks, task coverage, and degree of heterogeneity) were averaged to derive a typical performance for that scenario.

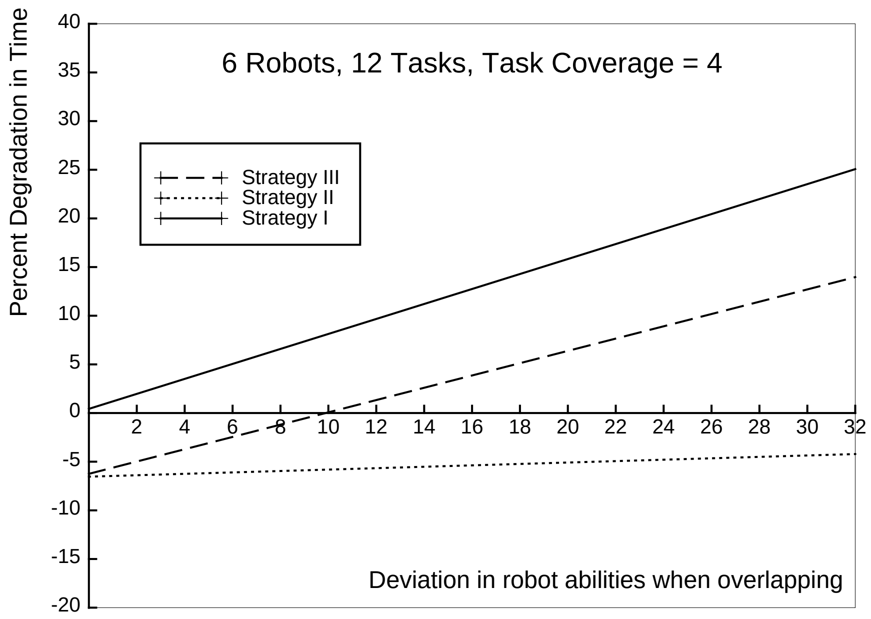

Figure 4-11: Typical performance degradation due to Random Task Selection.

Results and Discussion

The results from this study proved to be quite interesting. One's first reaction might be to assume that the Shortest Task First approach would exhibit improved performance over the Random Task Selection approach regardless of the impatience/acquiescence update strategy. However, this turns out not to be the case. Figure 4-11 shows a typical outcome of this comparison, in terms of time, for a six-robot team with twelve tasks to perform and a task coverage of four. (The energy outcome is similar.) As this figure shows, although the Random Task Selection approach does degrade the performance of teams controlled by strategies I and III, it actually improves the performance of teams controlled with strategy II (Let the Best Robot Win).

The reason for this performance improvement for strategy II concerns the theoretical advantages of using the Longest Task First selection strategy that I dismissed earlier. Recall that the Longest Task First approach should theoretically result in shorter mission completion times for homogeneous robot teams because the longer tasks are pursued first while available robots perform the shorter tasks in parallel. However, I dismissed this approach for heterogeneous robot teams which can perform the same tasks with different qualities. The problem I encountered was that the pure Longest Task First approach caused each task to be pursued by the robot with the longest task completion time. However, if I modify this strategy so that robots first effectively divide their tasks into the two categories I have described (category 1 tasks are those which the robot expects to be able to perform better than any other robot, and category 2 tasks are all the remaining tasks that robot can perform), and then use a Longest Task First mechanism to select among the category 1 tasks and a Shortest Task First mechanism for selecting among the category 2 tasks, the problem of heterogeneity is circumvented.

What I find is that the Random Task Selection approach for impatience/acquiescence update strategy II actually moves the robot control toward a Longest Task First approach, since any random selection of an action must result in a longer task than that chosen with the Shortest Task First approach. However, since the robot only uses the Longest Task First mechanism for tasks in category 1, we do not run into problems due to heterogeneity. Thus, the performance of strategy II actually improves with the Random Task Selection approach.

In fact, robots controlled with strategy III also experience improvement due to this modified Longest Task First approach for the same reasoning. However, this improvement is offset somewhat because robots in strategy III do not override the performances of poorer robots that have selected tasks badly from category 2. Overall, then, robots controlled using strategy III display poorer performance when using the Random Task Selection approach.

Strategy I, however, does not benefit from the move towards the Longest Task First approach, because a robot using this strategy does not have the information required to segment its tasks into the two categories. Although such a robot does have the ability to override a bad action selection of a poorer robot, the override mechanism is not sufficient to overcome the degradation in performance due to poor task selections by all robot team members. Thus, strategy I suffers the worst from the Random Task Selection approach.

4.5.3 The "Winning" Distributed Control Strategy for Efficiency and Fault Tolerance

The results of the previous two subsections lead to an improved robot control strategy, which I call Strategy IV. This strategy is based upon modifications to Strategy II as suggested in the previous two sections. The resulting interaction of the distributed motivational behaviors causes each robot r_i to effectively do the following:

1. Effectively divide the tasks into two categories:
    (a) Those tasks which r_i expects to be able to perform better than any other team member, and which no other robot is currently performing.
    (b) All other tasks r_i can perform.

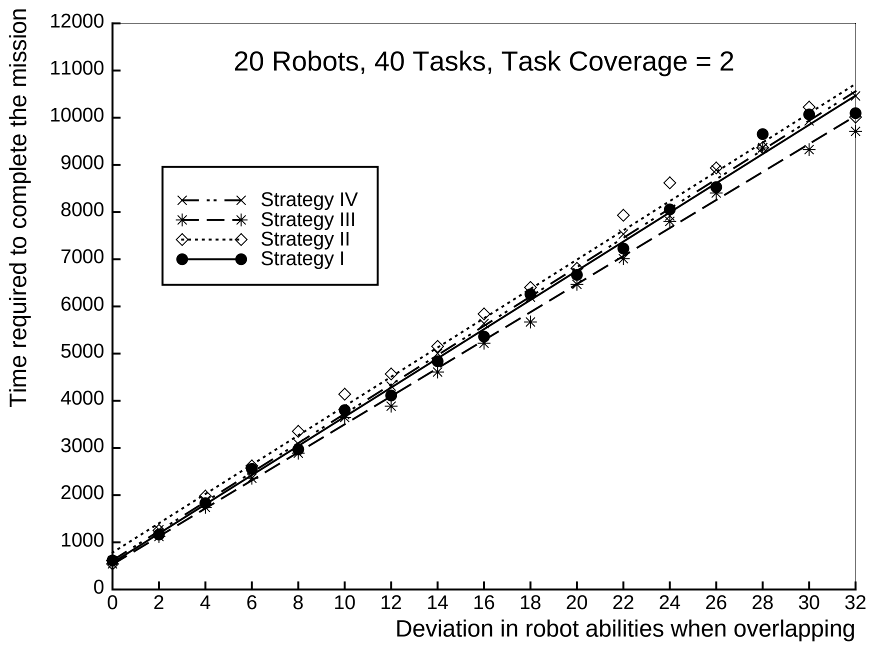

Figure 4-12: Comparative time performance of strategy IV in region 4.

2. Repeat the following until no more tasks are left:

   (a) Select tasks from the first category according to the Longest Task First approach, unless no more tasks remain in category 1.

   (b) Select tasks from the second category according to the Shortest Task First approach.

If a robot has no learned knowledge about team member capabilities, all of its tasks go into the second category.

Figures 4-12 through 4-19 show the results of strategy IV compared to the previous three impatience/acquiescence strategies which used a Modified Shortest Task First ordering of tasks. As these figures show, the new strategy IV performs as well or better than any of the other three strategies in terms of both time and energy, regardless of the size of the robot team, the size of the mission, the relative task coverage, the level of heterogeneity on the robot team, or the degree to which the effects of robot actions can be sensed through the world. The only very minor exception to this statement is in region 2 for mildly heterogeneous robot teams in which the Progress When Working condition does not hold; in this case, strategy III performs slightly better.

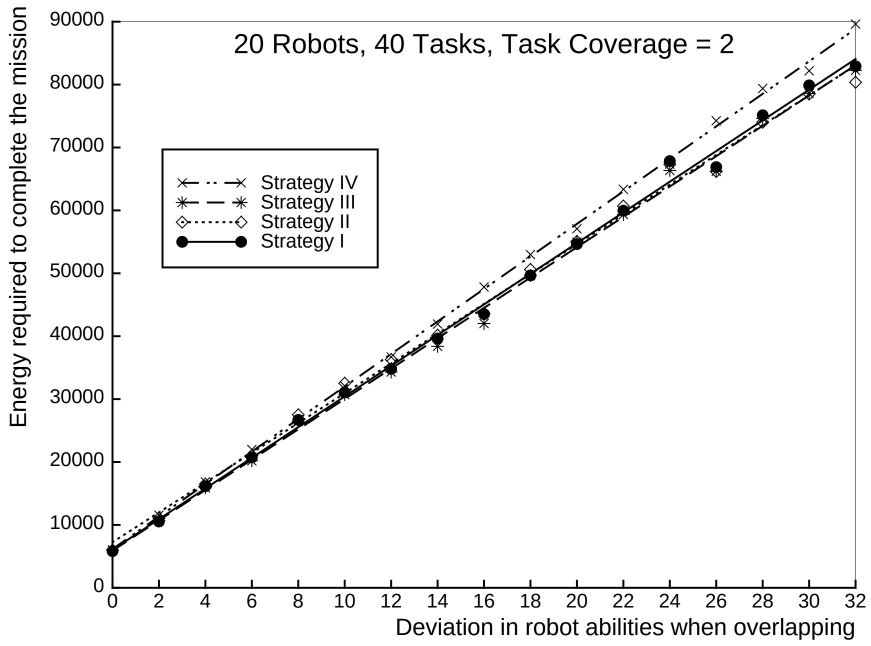

Figure 4-13: Comparative performance of strategy IV in region 4.

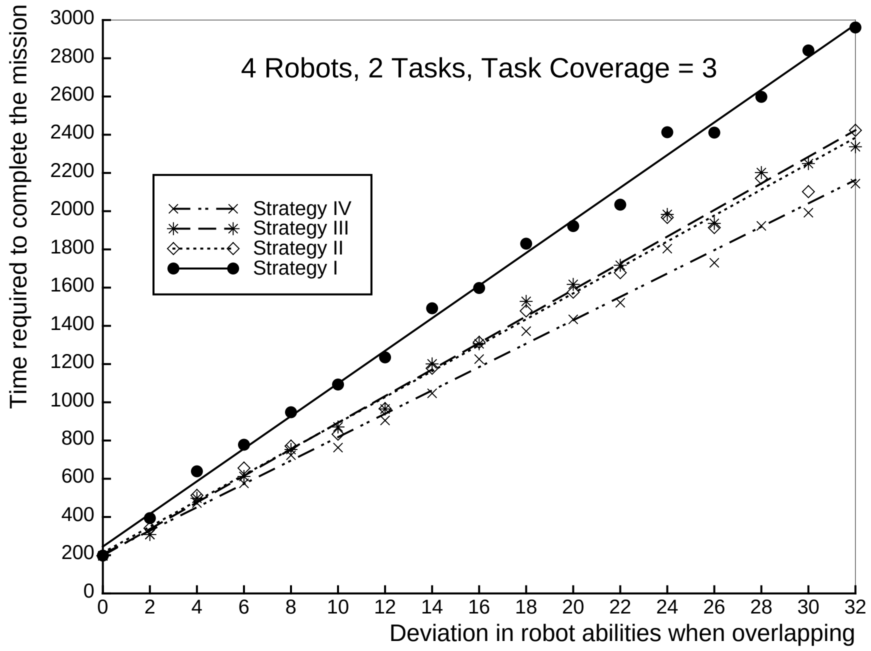

Figure 4-14: Comparative time performance of strategy IV in region 1.

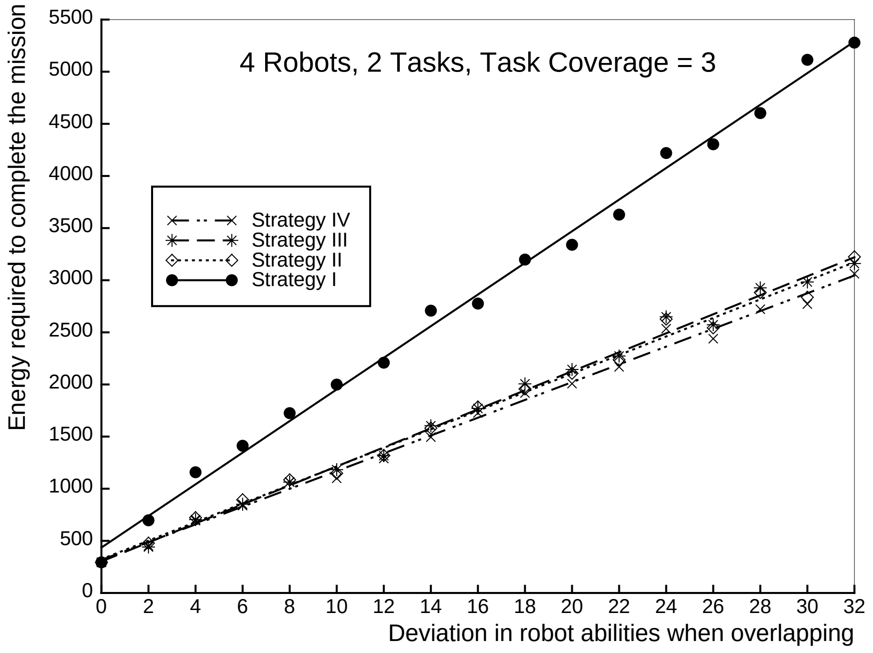

Figure 4-15: Comparative energy performance of strategy IV in region 1.

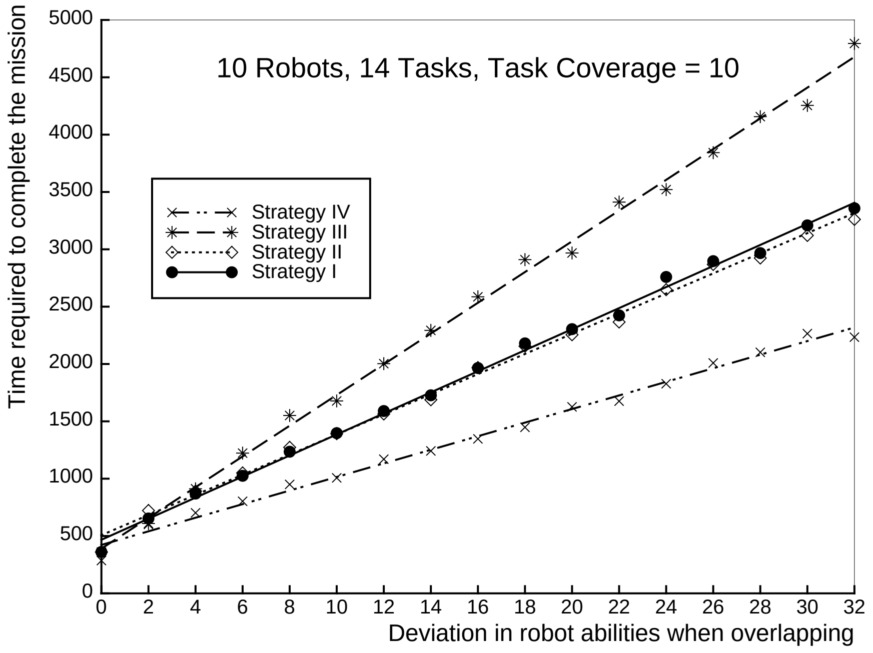

Figure 4-16: Comparative time performance of strategy IV in region 2.

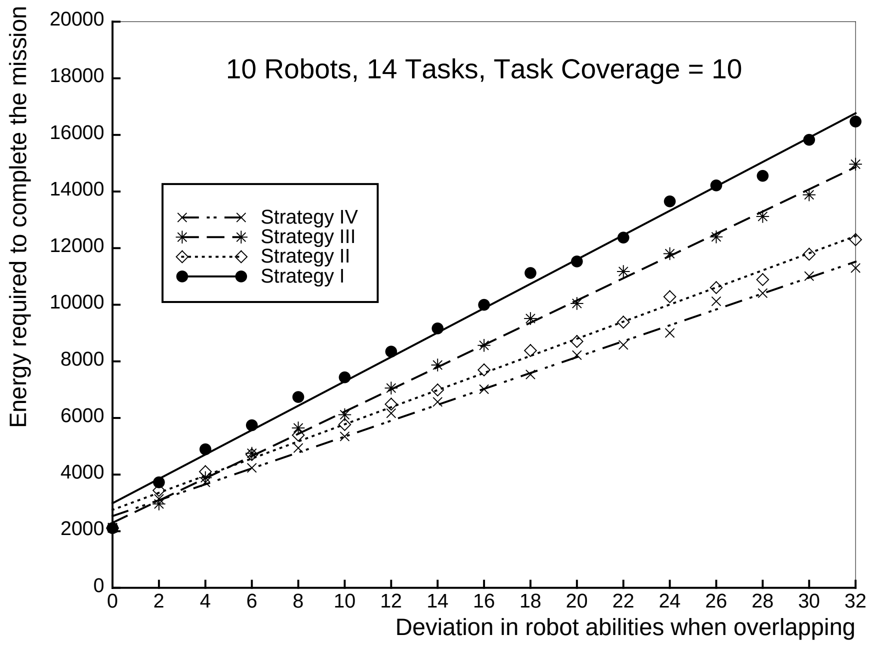

Figure 4-17: Comparative energy performance of strategy IV in region 2.

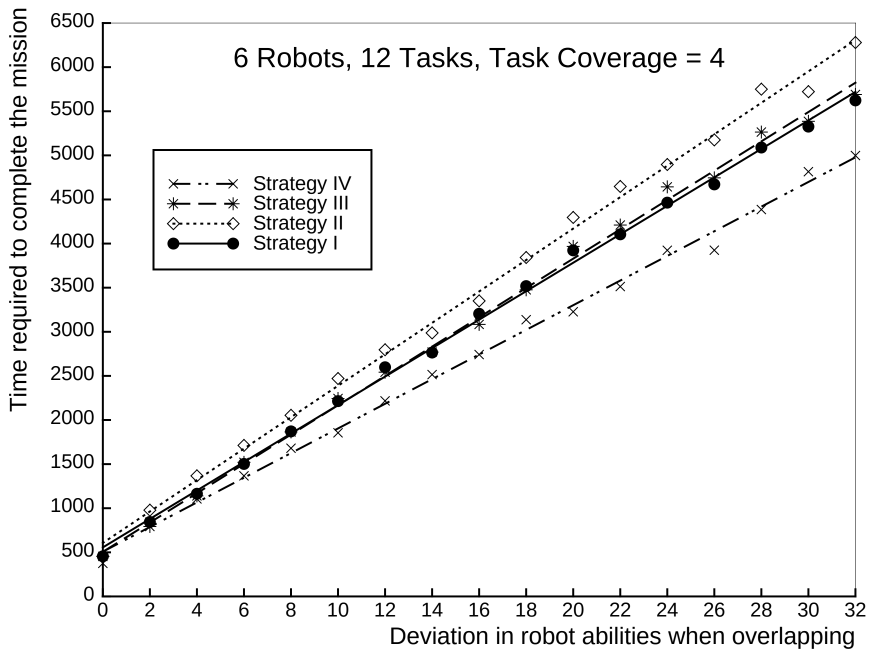

Figure 4-18: Comparative time performance of strategy IV in region 3.

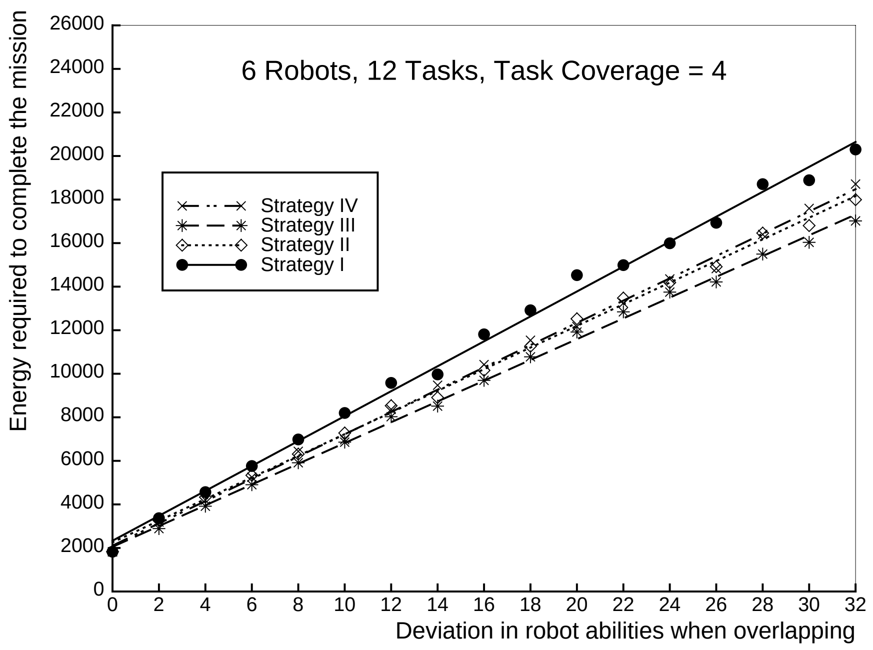

Figure 4-19: Comparative energy performance of strategy IV in region 3.

4.5.4 Comparison to the Optimal Solution

Since the learning problem is NP-hard, it is very difficult to compare the performance of the L-ALLIANCE approach to the optimal result. The problem is exponential in the number of tasks ($O(n^m)$, for n robots and m tasks), and thus the optimal solution becomes virtually impossible to calculate even for fairly small values of n and m. However, the optimal result can be calculated for many small problems in which the value of n^m is reasonable. I can then compare the results of the L-ALLIANCE control strategy IV with the optimal solution. What we find is that L-ALLIANCE performs quite well for these small problems.

This analysis was performed as follows. I composed triples ($n, m, task\_coverage$) of every possible integral combination of values[4] for which n^m was less than or equal to 2^{17}, up to values of n equal to 17. Figure 4-20 plots the triples (496 of them) for which I could calculate the optimal solution; these triples are plotted according to their $task\_coverage/m$ ratio. Because of the exponential nature of the allocation problem, the optimal result for any scenario in region 4 could not be computed. I then performed 200 random runs corresponding to each triple, varying heterogeneity differences from 0 to 3200%, and recording the time and energy usage of the optimal result

[4]The limits of $n \leq 17$ and $n^m < 2^{17}$ were derived in an ad hoc fashion based upon the time required to compute the optimal solution on our fastest available serial computer — a Sparc-10.

and of the solution found by L-ALLIANCE strategy IV. The average of these 200 runs then indicated the typical performance of the optimal and of the L-ALLIANCE allocations for that degree of heterogeneity. I then computed the average percent worse of the L-ALLIANCE solution over the optimal solution for each of the heterogeneity differences of that scenario. This value indicates the relative performance of the L-ALLIANCE strategy IV solution compared to the optimal solution for that triple. Finally, I categorized that triple into the appropriate time/energy profile region (see figures 4-1 and 4-2) according to its $task\_coverage/m$ ratio. I repeated this process for all the triples, and then computed the average percent difference across each of the four regions (actually, three regions, since I had no data from region 4).

Figure 4-21 shows the results, indicating the percent worse than the optimal result for the scenarios in regions 1, 2, and 3 for both time and energy. A total of 331 scenarios make up the region 1 average, 139 scenarios make up the region 2 average, and 26 scenarios make up the region 3 average. These results indicate that L-ALLIANCE performs quite well for these smaller scenarios — less than 20% worse than optimal for any region, for either time or energy, with much better performance in region 1. The worst-case performance in terms of time was for the scenario involving four robots, eight tasks, and a task coverage of three, which was 28% worse than the optimal; this scenario is indicated in figure 4-20 by the large dot at location (.75, 2) in region 3. The worst-case performance in terms of energy was for the scenario involving five robots, six tasks, and a task coverage of five, which was 25% worse than the optimal; this scenario is indicated in figure 4-20 by the large dot at location (1, 1.2) in region 2.

The key question, of course, is how seriously we should expect the performance of L-ALLIANCE to degrade as the size of the problem increases. Strategy IV performs particularly well in region 1 because, although the knowledge is distributed across motivational behaviors, the robots are essentially using global knowledge in their action selection due to the high task coverage and low mission size. However, as the relative number of tasks to perform increases, the purely greedy approach cannot always result in near-optimal performances because it will at times be more efficient to make several less-than-optimal local task selections to arrive at a globally optimal result. Quantifying how much worse L-ALLIANCE performance can be than the optimal is difficult, however, and warrants further study. This is a primary topic of future study.

4.6 Discussion of Formal Model of L-ALLIANCE

Now that the philosophy behind the L-ALLIANCE learning approach has been presented, let us look in detail at how this philosophy is designed into the motivational

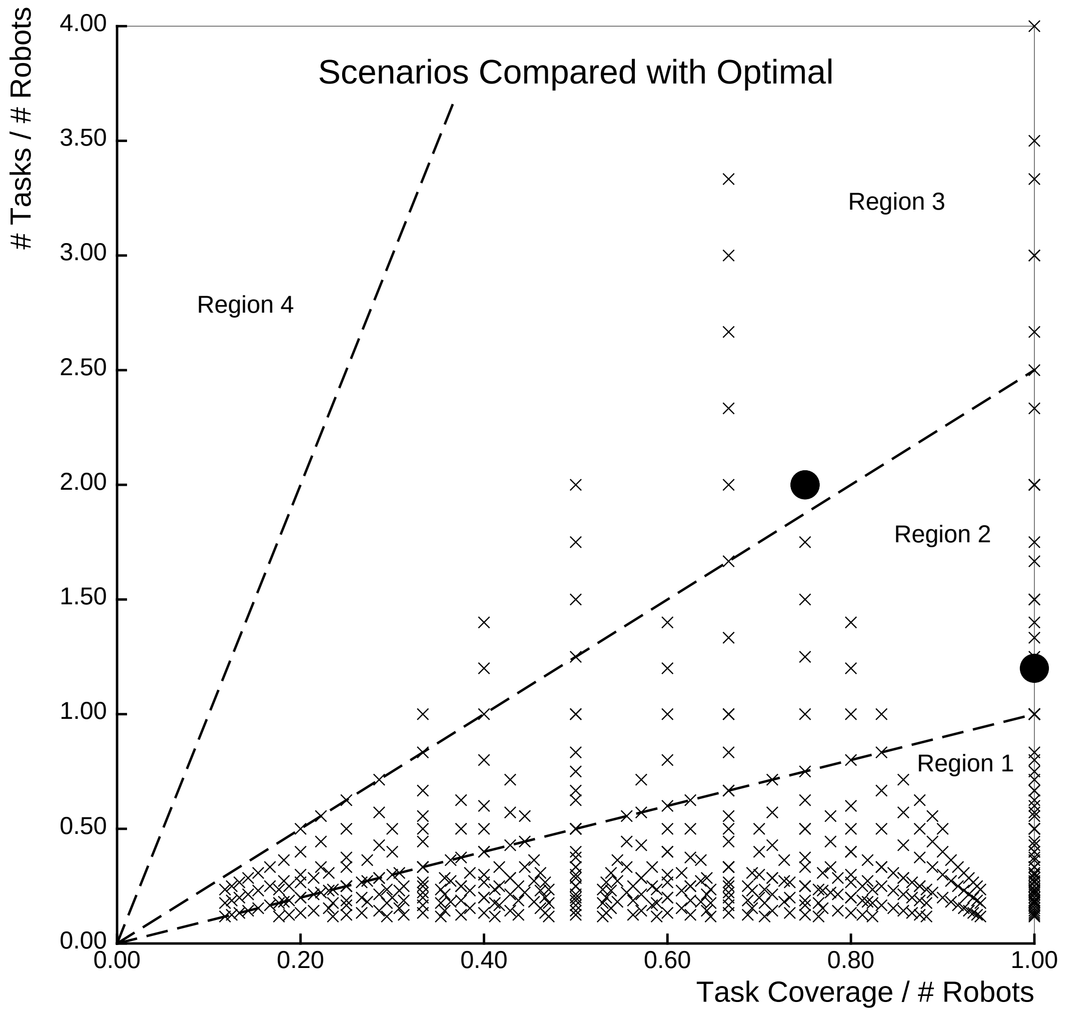

Figure 4-20: The data points shown correspond to scenarios for which the optimal result could be computed. For each of these scenarios, I compared the time and energy usage required for the strategy IV distributed action selection technique against the required time and energy usage for the optimal result. The data points correspond to a total of 496 scenarios. The two heavy dots are the scenarios for which the time or energy performance was the worst. The dashed lines indicate the same four regions as shown in figures 4-1 and 4-2.

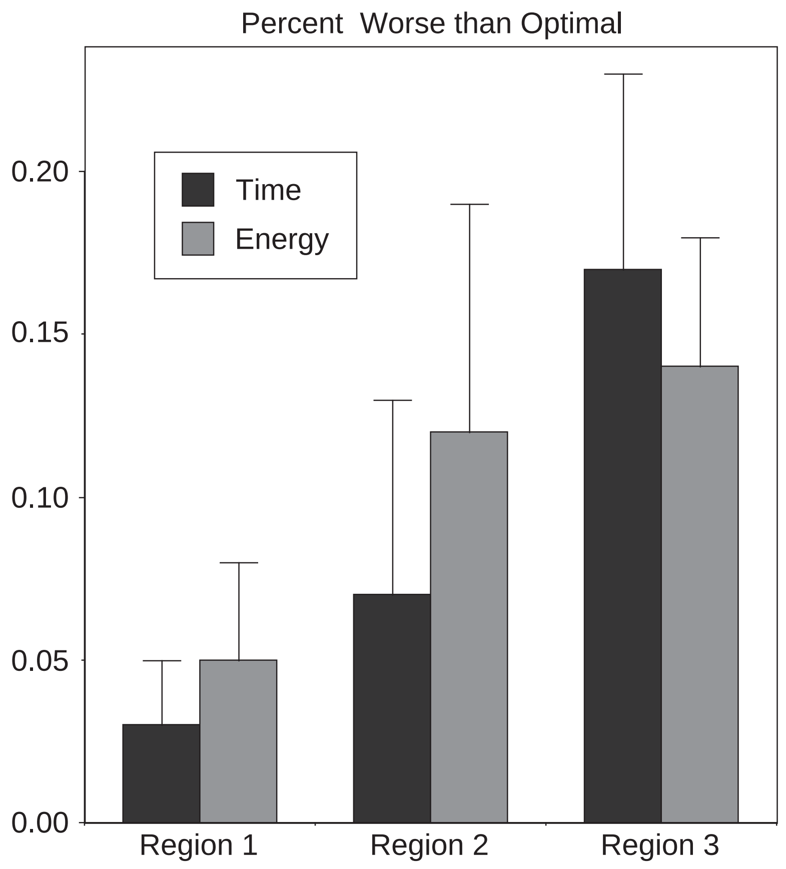

Figure 4-21: Comparison of strategy IV performance with the optimal performance. In region 1, the averages are over 331 scenarios, in region 2 the averages are over 139 scenarios, and in region 3 the averages are over 26 scenarios. Since the efficiency problem is exponential in the number of tasks, the optimal results could not be derived for any scenarios in region 4. The error bars indicate one standard deviation in the performance differences.

behavior mechanism. I organize this subsection by first discussing the threshold of activation of the behavior sets, followed by a discussion of the parameter settings pertinent to each of the sources of input to a robot's motivational behavior: sensory feedback, inter-robot communication, suppression from active behavior sets, learned robot influence, robot impatience, and robot acquiescence. In these sections, I discuss only those parameter issues in L-ALLIANCE that were previously ignored in my description of ALLIANCE. Chapter 3 provides the philosophy behind the basic ALLIANCE mechanism, which remains true for L-ALLIANCE. I conclude this section by showing how the L-ALLIANCE inputs are combined to compute the motivational levels. Appendix B summarizes the L-ALLIANCE formal model for easy reference[5].

Threshold of activation

A parameter of key importance to the efficiency of the robot team is the threshold of activation, θ. This parameter is used not only to determine the motivational level at which a behavior set is activated, but, more importantly, as a way of calibrating the impatience and acquiescence rates across motivational behaviors and across robots. Recall from section 4.5.3 that I want the interaction of motivational behaviors to result in a robot selecting either the task it can perform the quickest or the task that requires the robot the longest time to accomplish, depending upon the task category. Since the L-ALLIANCE mechanism is distributed across several parallel processes, these orderings can be accomplished by setting the $\delta\_slow_{ij}(k,t)$ and $\delta\_fast_{ij}(t)$ impatience rates to values proportional to the expected completion times of their corresponding tasks. However, these rates are meaningless if the behavior sets activate at different levels, since a behavior set with a slower rate of impatience could activate before one with a faster impatience rate if the first behavior set had a low enough threshold of activation. Likewise, I want the robot team member that is superior at a given task to "win" the ability to perform that task by activating it prior to any of its teammates. Yet again, this cannot be accomplished if the robots have different thresholds of activation.

It is therefore important for the sake of efficiency for the value of θ to be uniform across robots and across the motivational behaviors of each robot. This uniformity should be quite easy to achieve: it can be obtained simply by the human designer broadcasting the desired value to all robots at the start of the mission, or by providing the robots with a simple arbitration mechanism that allows the team on its own to come to a consensus on what value of θ to use. Of course, as we saw in chapter 3, having uncalibrated θ's across motivational behaviors or across robots is not a catas-

[5]The model described in this section and in appendix B is a more recent version of that presented in [Parker, 1993b].

trophic problem — the robots will still be able to accomplish their mission, although less efficiently.

Sensory feedback

The use of sensory feedback in L-ALLIANCE is unchanged from its use in ALLIANCE.

Inter-robot communication

The rate at which robots communicate their current actions to their teammates is of central importance in ALLIANCE and L-ALLIANCE to the *awareness* robot team members have of the actions of their teammates. This in turn affects the efficiency of the team's selection of actions, since lack of awareness of the actions of teammates can lead to replication of effort and decreased efficiency. Since this issue is addressed extensively in chapter 5, I will not repeat my conclusions here. Suffice it to say that to ensure maximal efficiency, it is best to set the communication rates, ρ_i, to be fairly frequent relative to the time required to complete each task in the mission. Since the task completion time is usually many orders of magnitude larger than the time required to broadcast a message, it is likely that the communication system capacity easily suffices to meet this requirement.

The second parameter dealing with inter-robot communication is τ_i. This parameter is especially important for allowing a robot to know which other robots are present and to some extent functioning on the team. Although I want robots to adapt their own actions according to the current and expected actions of their teammates, I do not want robots to continue to be influenced by a robot that was on the team, but at some point has ceased to function. Thus, robots must at all times know which other robots are present and functioning on the team. This is implemented in ALLIANCE as follows: at the beginning of the mission, team members are unaware of any other robot on the team. The first message a robot receives from another robot, however, is sufficient to alert the receiving robot to the presence of that team member, since all robot messages are tagged with the unique identification of the sender. The robots then monitor the elapsed time from the most recent broadcast message of any type from each robot team member. If a robot does not hear from a particular teammate for a period of time τ_i, then it must assume that that teammate is no longer available to perform tasks in the mission.

Clearly, the proper value of τ_i is dependent upon each robot team member's ρ_i settings. If team members have different values for these parameters, then they cannot be sure how long to wait on messages from other robots. However, the difficulty should be minor if the τ_i values are set conservatively — say, to several times one's own time delay between messages. Even so, if a robot r_i erroneously assumes a team member r_k is no longer functional, the receipt of just one message from that team member at some point in the future is sufficient to reactivate r_k's influence on r_i's activities.

To refer to the team members that a robot r_i thinks are currently present on the team, I define the following set:

$$robots\_present(i,t) = \{k | \exists j.(comm\_received(i,k,j,t-\tau_i,t) = 1)\}$$

The $robots\_present(i,t)$ set consists simply of those robots r_k from which r_i has received some type of communication message in the last τ_i time units.

Suppression from active behavior sets

The suppression from active behavior sets in L-ALLIANCE is implemented identically to the method utilized in ALLIANCE.

Learned robot influence

When a robot is operating in the active learning phase as described in section 4.4.2, it selects its next task from among those tasks that are not currently being attempted by any other robot. Thus, a task $h_i(a_{ij})$ that robot r_i will consider selecting in the active learning phase is determined by the following function:

$$learning\_impatience_{ij}(t) = \begin{cases} 0 & \text{if } (\sum_{x \in robots\_present(i,t)} comm\_received(i,x,j,0,t)) \neq 0 \\ 1 & \text{otherwise} \end{cases}$$

This function says that a robot r_i considers activating a task $h_i(a_{ij})$ in the active learning mode only if r_i has not received a communication message from some robot r_x on the team indicating that r_x is pursuing task $h_i(a_{ij})$. On the other hand, when a robot is in the adaptive learning phase as described in section 4.4.2, it selects its actions based upon the knowledge learned about its own and other robot capabilities by using control strategy IV as described in section 4.5.3.

An additional role of the learned robot influence parameters, however, is to overlook the previously demonstrated capabilities of team members if tasks remain to be accomplished. This is implemented by causing the robot to be initially "blinded" to category 1 tasks — i.e. those tasks that other robot team members should be able to perform well — and thus not consider them for activation. However, if no tasks remain in the first category, the robot is idle and begins to become bored. Once robot r_i's boredom has crossed a threshold, it is no longer blinded to the tasks that other robot team members should be able to perform, causing r_i to select a task from the second category.

The resulting group behavior, then, is for the robots which have exclusive capabilities to perform certain tasks to select those tasks immediately. Additionally, for those tasks with a task coverage greater than 1, the robot that is expected to perform the task best across the available robots is more likely to select that task. As we saw in section 4.5, this approach thus yields a very efficient execution of the mission.

In terms of the formal model, I refer to this learned influence by the following definitions. First, I define a value μ that gives the number of trials over which robot maintains task performance averages and standard deviations. As stated earlier, the value of μ is fairly small; in my experiments, maintaining information over about 5 trials provided good results. I then define the function:

$task\_time_i(k, j, t)$ = The average time over the last μ trials of robot r_k's performance of task $h_i(a_{ij})$ plus one standard deviation, as measured by robot r_i

In the case of robot failure, the time attributed to the failed robot is some penalty factor (greater than 1) times the actual attempted time. As we shall see below (in the section describing the robot impatience parameters), this penalty factor in the case of task failure is important for allowing a robot to overcome its failure to achieve one task and go on to perform some other task at which it can succeed. The important point to note is that repeated failures cause the expected completion time of the failed task to monotonically increase, leading to slower rates of impatience for the failed task. If a robot continues to select a task at which it repeatedly fails, the updates to the impatience parameters eventually cause the robot to become more impatient to perform some other task at which it *can* succeed. This, therefore, prevents the robot from getting stuck forever performing a task at which it cannot succeed while it still has some task which it could successfully complete. Of course, the larger the penalty factor, the less likely the robot will repeatedly select a task at which it cannot succeed.

The tasks are then divided into the two categories of strategy IV according to the following function:

$$task\_category_{ij}(t) = \begin{cases} 1 & \text{if } (task\_time_i(i,j,t) = \min\limits_{k \in robots\_present(i,t)} task\_time_i(k,j,t)) \\ & \text{and } ((\sum\limits_{x \in robots\_present(i,t)} comm\_received(i,x,j,t-\tau_i,t)) = 0) \\ 2 & \text{otherwise} \end{cases}$$

This function says that task $h_i(a_{ij})$ belongs to the first category in robot r_i at time t if robot r_i's expected task completion time for task $h_i(a_{ij})$ is the minimum of the robot team members that r_i knows about, and if r_i has not received a message from any other robot on the team, r_x, in the last τ_i time units which indicates that r_x is currently performing task $h_i(a_{ij})$.

Next, I define the function that indicates the level of boredom of robot r_i. Given a boredom threshold, *boredom_threshold*$_i$, and a rate of boredom, *boredom_rate*$_i$, I define the boredom function as follows:

$$boredom_i(t) = \begin{cases} 0 & \text{for } t = 0 \\ (\prod_j activity\_suppression_{ij}(t)) & \text{otherwise} \\ \quad \times(boredom_i(t-1) + boredom\_rate_i) & \end{cases}$$

This function says that robot r_i's level of boredom is 0 at time 0 and whenever some behavior set a_{ij} is active on r_i. Otherwise, the level of boredom increments linearly over time according to the rate *boredom_rate*$_i$.

I now define the function that indicates which tasks a robot considers for activation:

$$learned\_robot\_influence_{ij}(t) = \begin{cases} 0 & \text{if } (boredom_i(t) < boredom\_threshold_i) \text{ and} \\ & (task\_category_{ij}(t) = 2) \\ 1 & \text{otherwise} \end{cases}$$

The function says that robot r_i considers activating a task $h_i(a_{ij})$ at time t only if that task is in category 1, or if the robot is bored.

Robot impatience

The primary robot impatience parameter is $\phi_{ij}(k,t)$, whose value in L-ALLIANCE varies during the mission based on the robot's experience. As discussed in section 4.4.4, the value of $\phi_{ij}(k,t)$ is set according to the selected impatience/acquiescence update strategy. The results presented earlier in this chapter indicate that the most efficient global action selections can be obtained by setting the value of $\phi_{ij}(k,t)$ as follows:

- For mildly heterogeneous teams in which Condition 3 (Progress When Active) does not hold, $\phi_{ij}(k,t)$ should be set to *task_time*$_i(k,j,t)$ (i.e. the time r_i expects robot r_k should need to complete task $h_i(a_{ij})$; this is strategy III).

- Otherwise, $\phi_{ij}(k,t)$ should be set to *task_time*$_i(i,j,t)$ (i.e. r_i's own expected time required to complete task $h_i(a_{ij})$; this is strategy IV).

If robot teams do not know which of these two situations holds in a given mission, the earlier results indicate that strategy IV is the preferred approach. Once the value for $\phi_{ij}(k,t)$ is determined, it is used to update the slow and fast rates of impatience ($\delta\_slow_{ij}(k,t)$ and $\delta\_fast_{ij}(t)$) according to control strategy IV. Recall that $\delta\_slow_{ij}(k,t)$ is the rate at which robot r_i becomes impatient with task $h_i(a_{ij})$ not becoming complete in the presence of robot r_k performing that task, and that $\delta\_fast_{ij}(t)$ is the rate at which r_i becomes impatient with task $h_i(a_{ij})$ not becoming complete either when no other robot is working on task $h_i(a_{ij})$, or when another robot has worked for too long on task $h_i(a_{ij})$. The idea is to set these parameters to cause the motivational behaviors to interact in such a way that each robot selects tasks from the first task category (see again section 4.5.3) according to the longest task first, and to select from the second task category according to the shortest task first. Because of the definition of the two task categories, the $\delta\_slow_{ij}(k,t)$ parameters only affect tasks in the second category, which means that $\delta\_slow_{ij}(k,t)$ should grow faster than $\delta\_slow_{ip}(k,t)$ only if robot r_i expects to perform task $h_i(a_{ij})$ faster than it expects to perform task $h_i(a_{ip})$. The $\delta\_slow_{ij}(k,t)$ parameter is therefore updated according to the following:

$$\delta\_slow_{ij}(k,t) \leftarrow \theta / \phi_{ij}(k,t)$$

This setting ensures that the time required for the behavior set's motivation to increase from 0 until it exceeds the threshold of activation equals the time of r_i's patience with r_k. Since the motivation is reset to 0 when r_k first begins execution of task $h_i(a_{ij})$, but never again, this ensures that r_i does indeed give r_k an opportunity to perform task $h_i(a_{ij})$. However, r_i cannot be fooled by repeated unsuccessful attempts by r_k to perform task $h_i(a_{ij})$; thus r_i will eventually take over this task if r_k does not demonstrate its ability to accomplish it.

Now let us examine the $\delta\_fast_{ij}(t)$ parameters; these parameters can affect the selection of tasks from either task category one or two, which means they must at times cause tasks to be selected according to the shortest first, and at other times according to the longest first. An additional detail concerning robot idle time between task activations must now be addressed. Any $\delta\_fast_{ij}(t)$ parameter corresponding to a task in the second category could be set the same as $\delta\_slow_{ij}(k,t)$ for some k. This would indeed cause the tasks to be selected in ascending order according to the expected task completion time. However, we must note that during the time in which the $\delta\_fast_{ij}(t)$ parameters are below the threshold θ, the robot is idle. Thus, setting a $\delta\_fast_{ij}(t)$ parameter the same as its corresponding $\delta\_slow_{ij}(k,t)$ parameter would cause the robot to wait for a period of time $\phi_{ij}(k,t)$ before activating task $h_i(a_{ij})$, which in turn means that the robot would remain idle about as long as it spends performing tasks. This is clearly unacceptable for the sake of efficiency, so the

$\delta\_fast_{ij}(t)$ parameter must be scaled in some way that reduces robot idle time while maintaining the relative impatience rates across motivational behaviors.

One easy way of scaling the $\delta\_fast_{ij}(t)$ parameters is to multiply them by some constant greater than 1. However, while this approach does reduce the idle time and maintain the relative ordering among the tasks, it still does not place an upper bound on how long a robot might remain idle during its mission. A better way of scaling the idle times is to map them to some acceptable range based upon expected task completion time. To do this, I define the notion of a *minimum allowable delay* and a *maximum allowable delay*, which give the range of times a robot can remain idle while waiting on its next behavior set to be activated. The actual values for these allowable delays should be set by the human designer according to the application. The only restriction is that the minimum delay should be greater than 0. Then, the *ideal* method of scaling the rates to within this range requires the motivational behaviors to ascertain the global minimum and maximum expected task completion times across all tasks of the mission. The reason why the global minimum and maximum times are ideal is because this allows the rates of impatience for a given task to remain calibrated across robots. However, unless outer boundaries for these values are provided by the human designer in advance, this requirement violates the distributed nature of L-ALLIANCE across robots. Although it would be quite possible to provide the robots with the ability to determine these global minimum and maximum task completion times through the broadcast communication system, I will not violate the distributed nature of L-ALLIANCE across robots. Instead, I approximate these global minimum and maximum task completion times with the minimum and maximum task completion times known within a given robot. Although this, too, violates the purely distributed nature of L-ALLIANCE within an individual robot, it can easily be accomplished through message passing between the motivational behaviors or a shared memory location[6]. With these new values, then, the proper settings of the $\delta\_fast_{ij}(t)$ parameters are determined as follows:
Let:

$$\begin{aligned}
min\_delay &= \text{minimum allowed delay} \\
max\_delay &= \text{maximum allowed delay} \\
high &= \max_{k,j} task\_time_i(k,j,t)
\end{aligned}$$

[6]If this, too, is undesirable, then the motivational behaviors can be provided with the expected minimum and maximum task completion times at the beginning of the mission, and then approximate the proper scaling. Small changes in the definition of $\delta\_fast_{ij}(t)$ would then be necessary to ensure that the rates do not drop below zero.

$$\begin{aligned} low &= \min_{k,j} task\_time_i(k,j,t) \\ scale\_factor &= \frac{max\_delay - min\_delay}{high - low} \end{aligned}$$

Then:

$$\delta\_fast_{ij}(t) = \begin{cases} \frac{\theta}{min\_delay + (task\_time_i(i,j,t) - low) \times scale\_factor} & \text{if } task\_category_{ij}(t) = 2 \\ \frac{\theta}{max\_delay - (task\_time_i(i,j,t) - low) \times scale\_factor} & \text{otherwise} \end{cases}$$

Thus, in the case of category 2 tasks, the fast impatience rates grow more quickly for the shorter tasks, whereas category 1 task impatience rates grow more quickly for longer tasks. In either case, the maximum delay before task activation is *max_delay*.

Robot acquiescence

The two robot acquiescence parameters are $\psi_{ij}(t)$ — the time before r_i yields task $h_i(a_{ij})$ to another robot — and $\lambda_{ij}(t)$ — the time before robot r_i gives up on itself to try to find something more useful it can accomplish. As described in section 4.4.4, the first of these parameters is updated according to the current impatience/acquiescence parameter update strategy, as follows:

- For mildly heterogeneous teams in which Condition 3 (Progress When Active) does not hold, $\psi_{ij}(t)$ should be set to $task\_time_i(i,j,t)$ (i.e. the time r_i expects to need to complete task $h_i(a_{ij})$; this is strategy III).

- Otherwise, $\psi_{ij}(t)$ should be set to $\min_{k \in robots\_present(i,t)} task\_time_i(k,j,t)$ (i.e. the minimum time r_i expects any robot would need to perform task $h_i(a_{ij})$; this is strategy IV).

The value of the $\lambda_{ij}(t)$ parameter should be based upon the time robot r_i expects it requires to perform task $h_i(a_{ij})$. This parameter should be conservatively set, however, so that mild underestimates of expected task time do not cause a robot to give up prematurely. Values for $\lambda_{ij}(t)$ set at two or three times the expected task completion time seem to work well in practice.

Motivation calculation

All of these inputs can now be combined into a simple motivational behavior calculation. During the active learning phase, the motivation of robot r_i to perform behavior set a_{ij} at time t is calculated as follows:

DURING ACTIVE LEARNING PHASE:

$$\begin{aligned}
random\_increment &\leftarrow \theta \times (\text{a random number between 0 and 1}) \\
m_{ij}(0) &= 0 \\
m_{ij}(t) &= [m_{ij}(t-1) + random\_increment] \\
&\quad \times sensory\_feedback_{ij}(t) \\
&\quad \times activity\_suppression_{ij}(t) \\
&\quad \times learning\_impatience_{ij}(t)
\end{aligned}$$

The motivation to perform any given task thus increments at some random rate until it crosses the threshold, unless the task becomes complete (*sensory_feedback*), some other behavior set activates first (*activity_suppression*), or some other robot has taken on that task (*learning_impatience*).

When the robots are working on a "live" mission, their motivations to perform their tasks increment according to the robots' learned information. The motivations are thus calculated as follows:

DURING ADAPTIVE PHASE:

$$\begin{aligned}
m_{ij}(0) &= 0 \\
m_{ij}(t) &= [m_{ij}(t-1) + impatience_{ij}(t)] \\
&\quad \times sensory\_feedback_{ij}(t) \\
&\quad \times activity\_suppression_{ij}(t) \\
&\quad \times impatience\_reset_{ij}(t) \\
&\quad \times acquiescence_{ij}(t) \\
&\quad \times learned\_robot\_influence_{ij}(t)
\end{aligned}$$

Robot r_i's motivation to perform any given task during the adaptive phase thus increments at the proper impatience rate (based upon the activities of other robots) until it crosses the threshold, unless the task becomes complete (*sensory_feedback*), some other behavior set activates first (*activity_suppression*), some other robot has taken over that task (*impatience_reset*), the robot decides to acquiesce the task (*acquiescence*), or some other robot is present that should be able to accomplish the task better than r_i (*learned_robot_influence*).

In either the active or the adaptive learning phases, when behavior set a_{ij} is operational in robot r_i, the corresponding motivational behavior broadcasts r_i's current activity to its teammates at a rate of ρ_i.

4.7 Summary and Contributions

This chapter has presented the L-ALLIANCE dynamic parameter update mechanism which allows robot teams to improve their efficiency from trial to trial on missions composed of independent subtasks. After showing that this efficiency problem is NP-hard, I investigated the performance of a number of heuristic control strategies as a function of the number of robots on the team, the size of the mission, the degree of task coverage, the degree of robot heterogeneity when task abilities are shared, and the degree to which the Progress When Working condition holds true. From the results, I derived a control strategy — strategy IV — that performs well for almost any combination of the above factors. I compared the results of this strategy to the optimal results for small problems and found that control strategy IV performs within 20% of the optimal solution for these small problems. Determining L-ALLIANCE's expected performance for much larger problems, however, is challenging, and is thus a primary topic for future study.

I now again review my initial design requirements from chapter 1 and note the advantages L-ALLIANCE offers over ALLIANCE.

4.7.1 Meeting Design Requirements

The primary advances which L-ALLIANCE makes over ALLIANCE are in the areas of adaptivity and coherence. I briefly review these advances here.

Flexibility and Adaptivity

A major advantage of the L-ALLIANCE control mechanism is its ability to allow robots to adapt over a number of trials to the performance of team members. As robot abilities in performing their tasks change over time due to learning, mechanical drift, or mechanical repairs, team members continually adapt their parameters to respond to these changes. Thus, since learning is always activated — even when performing a "live" mission — each robot is able to evolve its response over time to the presence of other team members and their actions. This enhances the ability of the human designer to custom-design teams of robots for given type of mission without the need to perform a great deal of pre-programming.

Coherence

As mentioned in chapter 1, one very common measure of system coherence is the efficiency with which the system accomplishes its mission. As we saw in chapter 3, ALLIANCE addresses one aspect of efficiency by providing mechanisms for robots to minimize replication of tasks. However, ALLIANCE does not provide mechanisms ensuring that the team accomplishes its mission with minimal time or energy expenditure. As we saw in chapter 3, ALLIANCE is guaranteed to allow the robot team to accomplish its mission when the Progress When Working condition is true (barring robot failures). However, we could not be confident that the robot team under ALLIANCE would accomplish the mission efficiently. In fact, if the impatience and acquiescence parameters are set particularly badly, the team members could thrash between tasks or select tasks they perform very inefficiently. As we have seen in this chapter, L-ALLIANCE corrects this problem by providing a control mechanism that minimizes thrashing between tasks.

Another, more difficult, efficiency consideration in heterogeneous robot teams is the ability for each robot to select its own actions in light of the capabilities of its teammates. Since heterogeneous teams are often composed of robots that can perform the same task with quite different performance characteristics, it is important that robots be able to select their actions so as to minimize the total time or energy required for the team to complete its mission. L-ALLIANCE accomplishes this efficiency improvement by incorporating a distributed control mechanism that allows a team of robots to select tasks efficiently based upon their relative abilities. This control mechanism has been shown to result in very close to optimal results for small problems. Additionally, even for those problems for which the optimal solution could not be computed, the final L-ALLIANCE control mechanism has been shown to result in dramatic improvements over more naive approaches.

4.7.2 Contributions

The primary contribution described in this chapter is the L-ALLIANCE extension to ALLIANCE that preserves the fault tolerant features of ALLIANCE while incorporating a dynamic parameter update mechanism that uses learned knowledge to improve robot team performance. This extension to ALLIANCE results in the following desirable characteristics:

- Improves efficiency for cooperative teams applied to missions composed of independent tasks.
- Eliminates the need for human adjustments of parameters.
- Allows human designer to custom-design robot teams for specific missions.
- Requires no advance knowledge of the capabilities of team members.
- Allows robot team members to adapt their performance over time to changes in the environment or in the team member capabilities.

Chapter 5

Robot Awareness and Action Recognition

The ALLIANCE and L-ALLIANCE architectures described in chapters 3 and 4 address the important question of how to allow a distributed team of robots to accomplish its mission while maintaining robustness, reliability, flexibility, and coherence. It is important to note, however, that these architectures rely on the ability of robot team members to determine the current actions of their teammates as well as the effect of those actions. Due to the difficulty of achieving an automated solution to passive action interpretation, the ALLIANCE and L-ALLIANCE architectures utilize a broadcast communication mechanism as a substitute. As described in chapter 3, this communication mechanism requires each robot team member to periodically broadcast its current actions. Other team members hear these broadcasts and alter their own actions accordingly. This communication mechanism, therefore, allows a detour around the important, yet difficult, topic of passive action recognition without the assumption of an unrealistic "black box" that provides "magical" capabilities to the robot team members.

However, one may rightly question whether this substitution nullifies my claims for the robustness of ALLIANCE and L-ALLIANCE, asking whether the failure of one system component — the communication mechanism — could cause the failure of the entire robot team. Since communication is used in ALLIANCE and L-ALLIANCE to substitute for action recognition, the failure of the communication system implies that robot team members are no longer *aware* of the actions of their teammates. An accurate description of our concern here, then, is the issue of *robot awareness*, rather than the broader issue of *communication failure*. This chapter explores this question of robot awareness and action recognition, particularly as it applies to the ALLIANCE and L-ALLIANCE architectures. I first review the related work in the area of action recognition to identify the current state of the art, and then present the results of a detailed study of this issue using physical robots performing the hazardous waste cleanup mission.

5.1 Action and Goal Recognition

Action recognition is the problem of observing the behavior of an agent and interpreting that behavior as a discrete action or actions. Related to this notion is the more well-known artificial intelligence problem of *goal* or *plan* recognition, which is concerned with explaining the behavior of an agent[1]. The two notions are distinct, however, because research on the former topic focuses on the issue of *what* an agent is doing, whereas the latter topic concentrates on *why* an agent is doing what it is doing. These issues are of interest for cooperative mobile robotics because they can provide robots with the ability to respond more appropriately to the actions and intentions of their teammates. Without at least a rudimentary ability to perform action and goal recognition, robot teams will have difficulty achieving coherence in their task selection, as discussed in the following section. In this section, I explore the issues of goal recognition and action recognition, reviewing the related research in these areas and making comparisons to the approach employed in ALLIANCE and L-ALLIANCE.

5.1.1 Goal Recognition

A significant amount of research in artificial intelligence has addressed the topic of goal recognition. An application domain particularly well-studied is that of natural language discourse understanding. This research deals primarily with the role of plans and intentions in understanding dialogues. A broad selection of papers in this area can be found in [Cohen *et al.*, 1990b]; additional work in this area is described in [Carberry, 1990, Charniak and Goldman, 1989, Konolige and Pollack, 1989, Mayfield, 1989]. A second main body of work in goal recognition is the area of intelligent user interfaces; examples of this area of research can be found in [Goodman and Litman, 1990, Raskutti and Zukerman, 1991].

Although the specific approaches to goal recognition vary greatly, nearly all approaches involve the use of a model of agent behavior for use in interpreting that

[1]Since the meanings of *goal* recognition and *plan* recognition are often blurred in AI and Distributed AI research, I henceforth use the term *goal recognition* to encompass both *goal* and *plan* recognition as used by the traditional AI and DAI literature. I do this more for philosophical reasons than for conciseness, because the term *plan* recognition implies that the observed agent actually *has* a plan, in the traditional AI sense (see chapter 8 for a description of what is meant by "traditional AI"). I resist making this assumption as applied to mobile robots for reasons that should be obvious from this report.

agent's actions. Bond and Gasser in [Bond and Gasser, 1988] cite the reasons for modeling agents as follows:

- Models are useful for predicting the requirements and effects of events not directly sensible (e.g. because they will occur in the future).
- Models can reduce communications requirements.
- Models can be useful for evaluating the credibility, usefulness, reliability, or timeliness of data.
- Models may improve efficiency by focusing activity or by directing search.

It is interesting to note that most of the agent models proposed in this body of research are quite elaborate, and stress the manipulation of declarative (as opposed to *procedural*) knowledge of agent actions. The usefulness of these elaborate models for goal recognition within physical robots that must operate in real-time, however, is uncertain. As described in more detail in chapter 8, this type of elaborate modeling used by traditional artificial intelligence approaches for general robot control has not performed well when applied to physical mobile robots. In fact, the most successful mobile robots to date have been built according to lessons learned from ethology, in which a few relatively simple rules of action interact to create the emergent behavior of the robot [Maes, 1990]. Thus, we might expect that the same principle which holds for general robot control also holds for goal recognition in physical robots. Studies involving East African vervet monkeys [Cheney and Seyfarth, 1990] have indeed shown that these animals view the world as things that *act*, not as things that *think* and *feel*. In other words, these monkeys can well understand behaviors of other animals in their society without having a concept of the knowledge or beliefs that may have caused those behaviors. Even without the ability to model the beliefs of other monkeys, however, these animals are able to cooperate to an amazing extent. Thus, we have an existence proof that complex models of intention are unnecessary for cooperation at the level exhibited by most social animals. Indeed, I have shown in this report that for the application domains I have studied, elaborate understanding of robot intentions is not necessary to achieve cooperative control.

Agent modeling issues aside, as noted in [Huber and Durfee, 1993], the existing goal recognition research is of limited use in multi-robot cooperation because the research almost invariably assumes that error-free symbolic descriptions of the current action (and possibly previous actions) taken by an agent and the current state of the world are always available. Since the systems do not deal with the difficult problem of obtaining the symbolic descriptions in the first place, nor with the problem of uncertainty in the observations, they have little relevance to physical robot domains.

One exception to this is the preliminary work described in [Huber and Durfee, 1993], which explicitly addresses this issue for mobile robot cooperation. In this paper, Huber and Durfee describe experiments in which one robot tries to infer the goal destination of another robot by analyzing its movements and taking into account uncertainties in the observations.

5.1.2 Action Recognition

Recognizing the actions of teammates can be performed by a robot in one of two ways:

- Through explicit communication
- Through interpretation of behavior observations

Clearly, the easiest method is the first, which involves having each robot explicitly communicate its current action to teammates according to some protocol or predetermined arrangement. As we have seen, this is the method utilized in ALLIANCE and L-ALLIANCE, whereby robots broadcast their current actions to teammates at some specified rate.

However, this simple method will obviously not work for applications in which the communication medium is not available (e.g. due to a noisy environment or faulty equipment), is costly (e.g. in terms of time or robot safety, in military applications), or is of limited bandwidth. In such applications, the robots must rely on other sensors — primarily visual — to observe and interpret the actions of team members.

Unfortunately, research in action recognition is much more limited than that addressing goal recognition, due to the difficulty of passive action interpretation. The current state of the art in this field provides robots with the ability to interpret simple teleoperated assembly tasks using non-visual sensors [Takahashi *et al.*, 1993, Yang *et al.*, 1993] and the ability to visually recognize human actions in simple blocks-world construction tasks [Kuniyoshi and Inoue, 1993, Ikeuchi *et al.*, 1993]. All four of the projects cited involve a robot first observing the human performance of a task and then interpreting and converting that task into either a symbolic or a kinematic representation which can then be reproduced by the robot. These research projects are notable for advancing the the field of passive action interpretation. However, they also demonstrate the significant difficulty of the action recognition problem, since they currently only work for very simple problems.

5.2 The Importance of Robot Awareness

As stated at the beginning of chapter 3, ALLIANCE assumes that with some probability greater than 0, each robot r_i on the team can detect the actions of other team members for which r_i has redundant capabilities. However, I also stated that I recognize that passive action interpretation is quite difficult and that explicit communications mechanisms substituting for this ability are not infallible. Thus, it is important to investigate the impact on the team's performance of incomplete, or even absent, knowledge about the actions of other team members and their effects on the environment. I use the term *awareness* to refer to this knowledge a robot has about the current actions of other robot team members.

To investigate this issue, I used four experimental setups of the hazardous waste cleanup mission that varied the number of robots on the team and the level of awareness the robots had of the actions of their teammates. The four versions of this experiment are:

I. Two-robot team, full awareness of teammates' actions.

II. Three-robot team, full awareness of teammates' actions.

III. Two-robot team, no awareness of teammates' actions.

IV. Three-robot team, no awareness of teammates' actions.

To achieve versions III and IV — those involving no awareness — the broadcast communications of each robot were turned off. Since these broadcasts are the sole mechanism in these experiments allowing robots to detect the actions of other robots whose effects could not otherwise be sensed through the world, the effect was to cause each robot to "think" it was working alone.

The outcomes of these experiments were evaluated based on their impact on the amount of time and energy required to complete the mission. To measure the energy usage, I made the approximation that a robot that is turned on but is not moving either its wheels or its gripper uses zero energy, whereas a robot that is using any of its four motors (i.e. right wheel, left wheel, grip, or lift) uses a unit quantity of energy per unit time. This approximation is not always correct with these robots, because a robot clearly uses more energy the more motors it activates at once. But since the energy requirements of the R-2 electronics when the R-2 is idle are indeed much less than the R-2's total energy requirements while it is moving, this approximation enables a simplified analysis while still providing a valid comparison of the four experimental setups.

Typical runs of experimental versions I and II were described in section 3.8, with their corresponding action selection traces shown in figures 3-9 and 3-10. Analogous traces of the actions selected by each robot for typical runs of experimental versions III and IV are shown in figures 5-1 and 5-2. Note in these traces that the robots replicate the actions of finding the initial and final spill locations and of reporting the progress. In fact, the replication of effort is the primary effect appearing in ALLIANCE that results from reduced awareness of the actions of other team members. The extent of this effect is the subject of investigation of this section.

It is important to note, however, that as we study the effect of the lack of awareness in ALLIANCE, the robot team is still able to complete its mission using the ALLIANCE architecture even with no knowledge of the actions of other robots. Certainly, their performance is less efficient and coherent than when awareness is possible, but at least they do get the job done. This is an important point in the usefulness of ALLIANCE, in that the robots are able to limp along to accomplish their part of the mission even when they "think" they are working alone. This gives the team increased robustness in the presence of communication failures.

For each of these four experimental setups, I ran 10 missions to completion on the physical robots and collected data on the actions selected by each robot at each point in time and the length of time they required to complete those actions. I considered the mission to be complete when 80% of the spill was moved to the goal location, which in this case meant when 8 of the 10 spill objects were at the goal location. All of the mission runs were measured up to, but not beyond the 80% complete stage for uniformity of measurements. One caveat is that if the robot moving the 8th spill object decided that it was time to report the progress, the mission was not considered complete until that robot had concluded its progress report.

5.2.1 Results and Discussion

As it turns out, the analysis of the performance of this mission by the four experimental robot teams is more complex than may be obvious at first glance. Two phenomena unrelated to cooperative robot issues arose during the experiments that must be factored out of the experimental evaluation; both of these phenomena have to do with events occurring while robots are performing the *move-spill* task, which is described in section 3.8.

First, it frequently happens that when picking up a spill object for transport to the final spill location, a robot actually "scoops up" additional spill objects in the cavity under its gripper and thus transports more than one spill object at a time[2]. An example of this phenomenon is shown in figure 3-15. In fact, on average 1.6 spill objects are moved per transport to the final spill location, which means that an

[2]This is analogous to a real-life hazardous waste cleanup robot which may, at times, be able to pick up larger portions of the spill for transport.

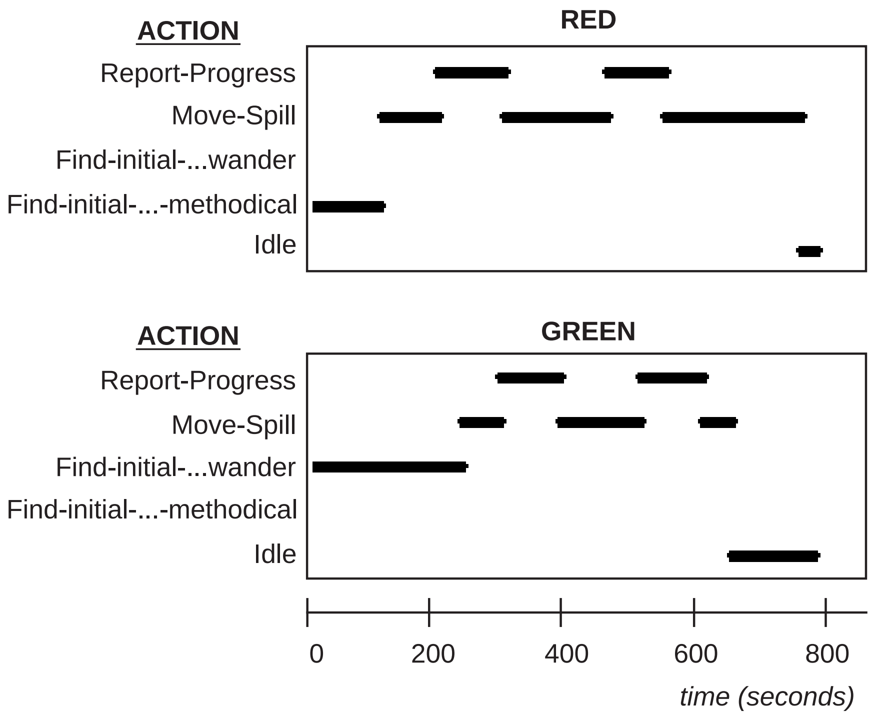

Figure 5-1: Robot actions selected during a typical experimental version III with two robots (RED and GREEN) and no awareness.

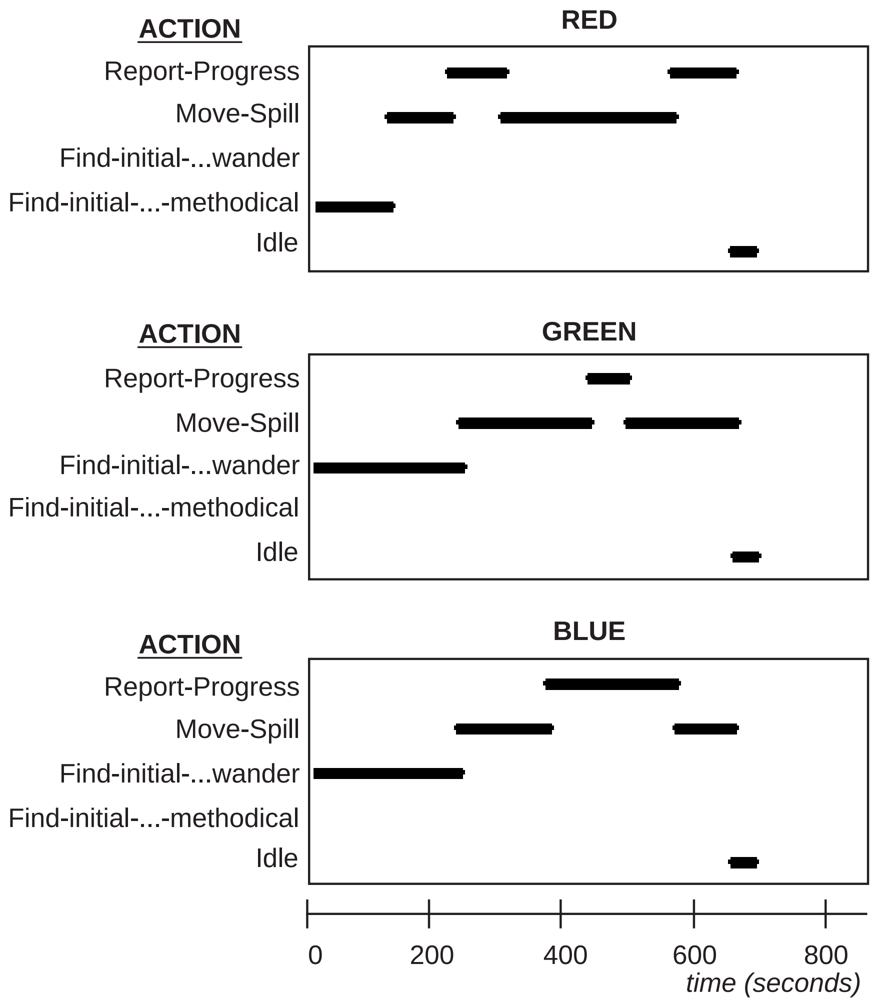

Figure 5-2: Robot actions selected during a typical experimental version IV with three robots (RED, GREEN, and BLUE) and no awareness.

average of 5 spill object transports are required to move 80% of the objects to the goal location. On one remarkable occasion, a robot actually managed to move 6 spill objects to the goal at once — two objects in its gripper and four objects in its front cavity. Of course, this is pure luck and relies on spill objects lying directly in the robot's path as it moves toward the goal when grasping an object and on the robot's not having to back up and turn away from an obstacle on its way to the goal. (Backing up and turning causes the robot to lose the spill objects in its front cavity.) Clearly, when a robot is able to move several objects in one trip, fewer trips are needed in total to accomplish the mission, which in turn means that the time and energy required to complete the mission varies accordingly. Additionally, accomplishing the move of the spill objects more quickly impacts the number of progress reports needed during the mission, as the total number of reports made is a function of the task completion time.

A second phenomenon observed in these experiments is that the time required for any individual robot to locate a spill object and move it to the goal location varies enormously. This is due to a combination of sensor and effector noise, and to the random search pattern the robots use to sweep the initial spill area. Figure 5-3 shows the average time (which, in this case, is the same as the average energy) required for each of the robots BLUE, GREEN, and RED to perform one instance of the three high-level tasks of this mission — finding the locations, making one spill object transport trip, and making one progress report — while on either a two-robot or a three-robot team. As this figure shows, the deviation in the time required to accomplish one transport of a spill object is quite large. Since the *move-spill* task is effectively shared by the robots in all four of these experimental setups, any effects due to robot awareness, or lack thereof, can be lost in the noise if the *move-spill* task comprises a large portion of the total mission time.

Because of these phenomena, the data collected for each of the four versions of the robot experiment must be normalized. To do this, I define functions that give us the average time and energy used by each experimental version.

Let:

$$\begin{aligned}
n &= \text{number of robots on the team} \\
energy(task) &= \text{the average energy required by a robot team} \\
& \text{member to perform one instance of } task \\
time(task) &= \text{the average time required by a robot team} \\
& \text{member to perform one instance of } task \\
num\_moves &= \text{the average number of spill object moves} \\
& \text{required to complete the mission}
\end{aligned}$$

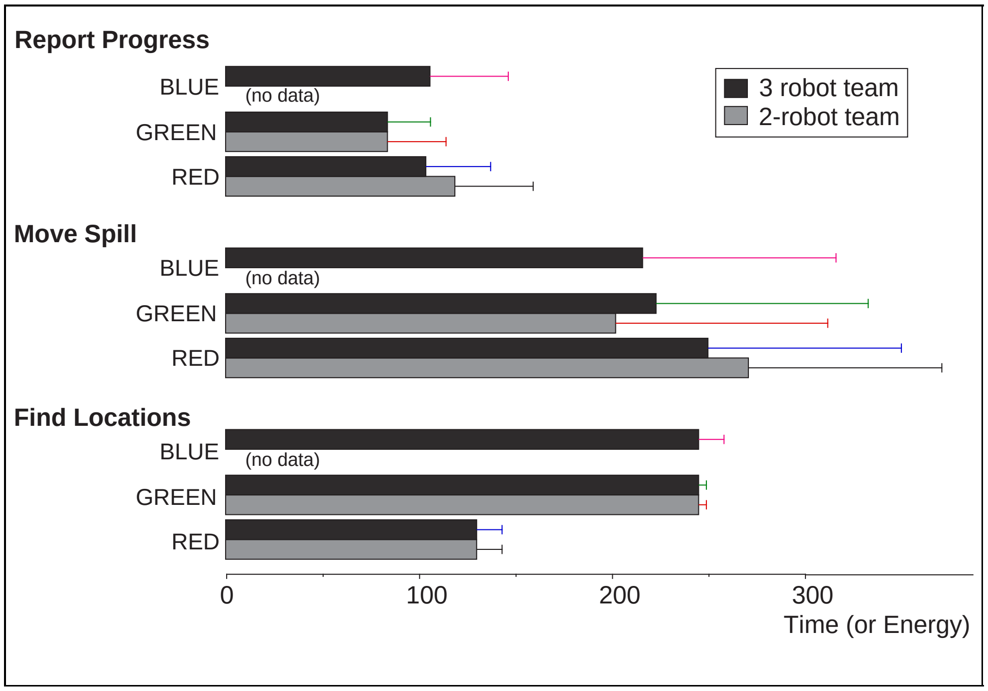

Figure 5-3: Average time (or energy) used by each of the 3 robots (BLUE, GREEN, and RED) to perform one instance of the tasks in the hazardous waste cleanup mission (*report-progress*, *move-pucks*, and *find-locations*), either when working on a two-robot team or on a three-robot team. No data was collected for BLUE on a 2-robot team, since it was not used in the 2-robot experiments. Error bars indicate one standard deviation in the measured times (energies).

$$\begin{aligned} &= \text{5 (from the discussion above)} \\ \mathit{num\_RPs\_per\_move}(\mathit{version}) &= \text{the average number of progress reports} \\ &\quad\ \text{performed per spill object move for} \\ &\quad\ \text{experimental version } \mathit{version} \end{aligned}$$

Then the average energy required to perform a mission for a given experimental version is:

For experimental versions I and II:

$$\begin{aligned} &\mathit{mission\_energy}(\mathit{version}) = \\ &\quad \mathit{energy}(\text{find-locations}) \\ &\qquad + \mathit{num\_moves} \times (\mathit{energy}(\text{move-spill}) + \mathit{num\_RPs\_per\_move}(\mathit{version}) \\ &\qquad\qquad \times \mathit{energy}(\text{report-progress})) \end{aligned} \tag{5.1}$$

For experimental versions III and IV:

$$\begin{aligned} &\mathit{mission\_energy}(\mathit{version}) = \\ &\quad n \times \mathit{energy}(\text{find-locations}) \\ &\qquad + \mathit{num\_moves} \times (\mathit{energy}(\text{move-spill}) + \mathit{num\_RPs\_per\_move}(\mathit{version}) \\ &\qquad\qquad \times \mathit{energy}(\text{report-progress})) \end{aligned} \tag{5.2}$$

The average time required to perform a mission is then defined as:

$$\begin{aligned} &\mathit{mission\_time}(\mathit{version}) = \\ &\quad \mathit{time}(\text{find-locations}) + \\ &\qquad [\mathit{num\_moves} \times (\mathit{time}(\text{move-spill}) + \mathit{num\_RPs\_per\_move}(\mathit{version}) \\ &\qquad\qquad \times \mathit{time}(\text{report-progress}))]/n \end{aligned} \tag{5.3}$$

The value of *num_RPs_per_move(version)* for each of the four experimental setups was obtained from the experimental data by averaging the ratio of the number of progress reports performed in a mission to the number of puck moves required by that mission. These values are given in figure 5-4. The values of *energy*(find-locations), *energy*(move-spill), *energy*(report-progress), *time*(find-locations), *time*(move-spill), and *time*(report-progress) were derived from the experimental data shown in figure 5-3 by averaging the time (or energy) values for all the robots for each of the three tasks. Since the time (or energy) required for each robot to perform each task did not vary significantly whether the robot was on a two-robot team or on a three-robot team, averaging across the team size does not skew the data. Additionally, since going from a two-robot team to a three-robot team did not increase interference, we can obtain the time requirements of a given experimental version by dividing by n the total accumulated time required for n robots to perform the *move-spill* and *report-progress* tasks for that version. Of course, this approximation cannot be expected to hold for any

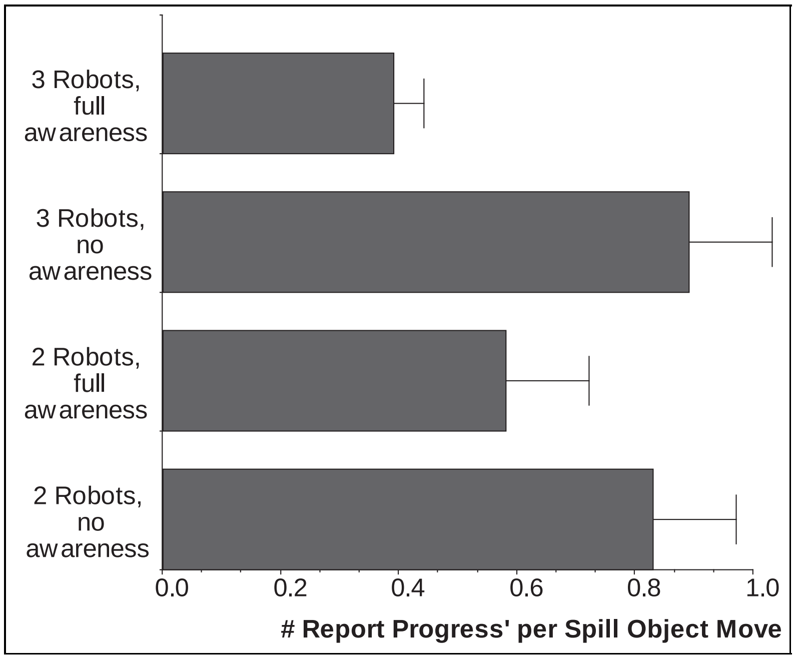

Figure 5-4: Average number of progress reports required per move of a spill object for each of the four experimental versions.

arbitrarily large robot team size, since at some point interference among the robots becomes a significant factor. One disadvantage of this averaging, however, is that it eliminates the differences in robot performance due to heterogeneity, primarily in the *find-locations* task. Again, I accept these averages for now to allow other interesting generalizations to be made.

Studying the group behavior of these four experimental setups brings to light a key issue in the analysis of the effect of awareness. Since the primary result of the lack of awareness is the replication of effort on those tasks for which robots have overlapping capabilities, the effect of this lack of knowledge varies depending upon the extent of overlap in robot capabilities and on the extent to which the effect of the actions of other robots can be sensed "through the world".[3] If robots have high overlap in their capabilities, along with a great difficulty in sensing the effects of the actions of other robots through the world, then the effect of the lack of awareness is clearly much greater than if the robots have no overlapping capabilities and do

[3]This ability to sense the effects of actions through the world can also be viewed in terms of the Progress When Working condition introduced in chapter 3. If action effects *can* be sensed, then the Progress When Working condition is true; otherwise, the Progress When Working condition does not hold.

not rely on the ability to sense the effects of other robots' actions through the world to select their own actions. On the other hand, if the total execution cost of the redundant actions is trivial compared to other actions that are not replicated, then the lack of awareness does not have an appreciable effect. The proper way to analyze the effect of awareness, then, is to compute its impact as a function of (1) the degree of redundancy in the capabilities of the robot team, (2) the ability of the robots to sense the effects of actions through the world, and (3) the cost of the replicated actions, relative to the cost of the entire mission.

Let us therefore examine this issue in more detail. In the cleanup mission introduced earlier, each instance of the *move-spill* task is an action whose effects can be sensed through the world; robots do not try to move spill objects that are no longer at the initial spill site. On the other hand, the *find-locations* task and each instance of the *report-progress* task are all information gathering or information broadcasting types of actions whose effects cannot be sensed through the world by this robot team. Thus, we find that the lack of awareness of the actions of other robots causes a replication of the *find-locations* task and the *report-progress* task, but does not cause a replication of effort in the *move-spill* task. Comparing the traces of figures 3-9 and 3-10 with figures 5-1 and 5-2, respectively, illustrates this replication.

To analyze how serious the replication of effort due to limited awareness can be, I define relative cost measures, $g_{energy}(task_k)$ and $g_{time}(task_k)$, for a specific task $task_k$ as follows:

$$g_{energy}(task_k) = \frac{energy(task_k)}{\sum_{j \in \text{all tasks}} energy(task_j)}$$

where $energy(task)$ is the energy required to perform one instance of the task $task$.

$$g_{time}(task_k) = \frac{time(task_k)}{\sum_{j \in \text{all tasks}} time(task_j)}$$

where $time(task)$ is the time required to perform one instance of the task $task$.

I use these relative cost measures in the next two subsections to quantify the cost of a replicated task. I can then vary the relative costs of these tasks to determine their impact on the teams' performance.

The Effect of Awareness on Energy Usage

Figures 5-5 and 5-6 plot the effect of varying the relative costs of the two replicated tasks, *find-locations* and *report-progress*, on the average energy required to perform the mission for each of the four experimental setups, using the energy function, *mission_energy(version)*, defined in equations 5.1 and 5.2. In the case of *find-locations*,

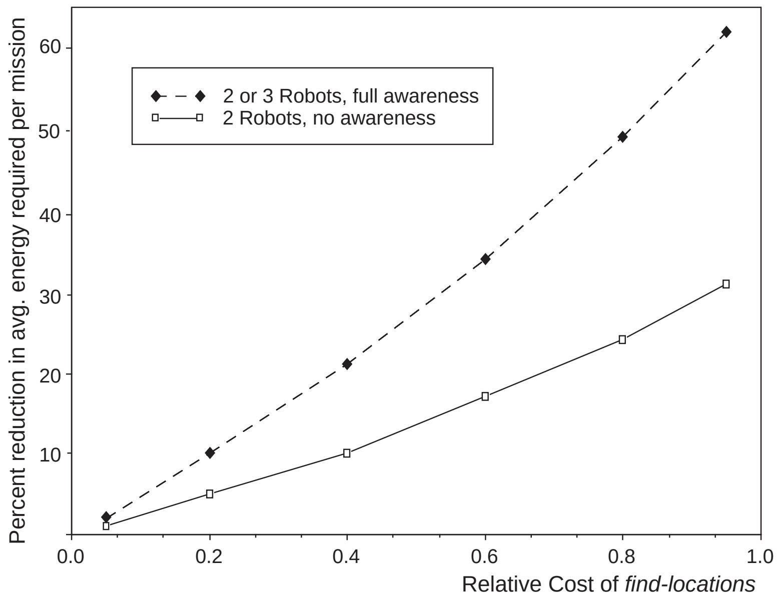

Figure 5-5: This graph shows the percentage reduction in average energy required as a function of the relative *find-locations* cost. Here, the worst energy performance is the baseline case with three robots and no awareness. The percent reduction in average energy required to complete the mission from this baseline case for the remaining three experimental versions is shown.

the worst version is the three-robot team with no awareness of the actions of the other robots. The two-robot team with no awareness performs from 1% to 31% better than the three-robot/no-awareness version (depending upon the relative cost of *find-locations*), while both the two-robot team and the three-robot team with full awareness performed from 2% to 62% better than the worst case.

In the case of *report-progress*, the worst versions were both the two- and three-robot teams with no awareness. Performing from 2% to 29% better was the two-robot team with full awareness, while the three-robot team with awareness performed from 3% to 51% better (again, depending upon the relative cost of *report-progress*).

As expected, the team performance improves with awareness, regardless of the task coverage afforded by the robot team — that is, regardless of the redundancy in robot capabilities for each task — because replication of actions is prevented. We also observe that for any level of team redundancy, the degree of improvement with awareness increases as the relative cost of the redundant action increases. This, too, is expected, since the energy saved with awareness is a direct function of the energy required to perform the redundant action.

Two additional points are interesting to note:

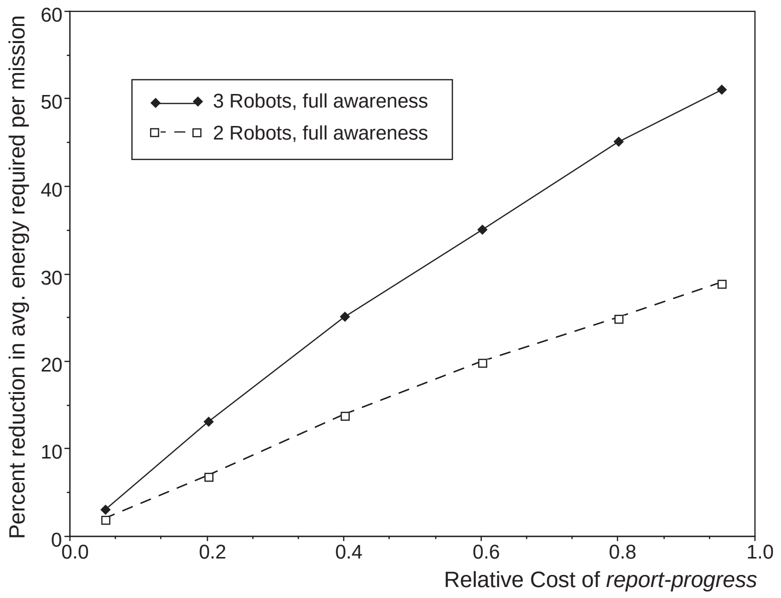

Figure 5-6: Of the 4 experimental versions for the hazardous waste cleanup mission, the ones requiring the most average energy to complete the mission are the situations involving no awareness of the actions of the other robots; both of these baseline cases require the same average energy to complete the mission. The percentage reduction in average energy required to complete the mission from these baseline cases is shown as a function of the relative energy cost of the *report-progress* task.

- In the case of the *find-locations* task, the energy performance of the two-robot team without awareness (version III) was better than the three-robot team without awareness (version IV), whereas for the *report-progress* task, the energy performances of these two teams were identical. In other words, the energy required to perform the *find-locations* task without awareness is multiplied by a factor of n (the number of robots on the team), whereas the energy required to perform the *report-progress* task without awareness is proportional only to the number of spill object moves of the mission.

- In the case of the *report-progress* task, the energy performance of the three-robot team with awareness (version II) was better than the two-robot team with awareness (version I), whereas in the *find-locations* task, the energy performances of these two teams were identical. In other words, the energy requirement of the *find-locations* task with awareness is fixed, whereas the energy requirement of the *report-progress* task with awareness is proportional to the number of spill objects moves divided by n (the number of robots on the team).

The first situation occurs because of differences in the extent to which the effect of the actions of other robots can be sensed through the world. In this situation, both teams lack awareness of the actions of other robots. The tasks that they might replicate due to this lack of awareness are the *find-locations* task and several instances of the *report-progress* task. Although neither of these task effects can be sensed through the world by this team, the *report-progress* task is closely tied to the *move-spill* task which *is* detectable through the world. Since the only time a robot tries to initiate the *report-progress* task is after it has completed the transport of a spill object to the goal, and since there are a fixed number of spill objects to be moved, the question of whether to report the progress only arises a fixed number of times for the team as a whole. Also, since the time required for one robot to find and move a spill object (see figure 5-3) is approximately the same as the time allowed between progress reports, robots in both versions III and IV are motivated to report their progress almost every time that they deliver a spill object[4]. Thus, regardless of whether the team consists of two or of three robots without awareness, the *report-progress* behavior set is activated a fixed number of times, which means that the energy requirements remain the same.

On the other hand, the *find-locations* task, whose effects cannot be sensed through the world, is replicated by each robot on the team having the ability to perform that task, which means that as redundancy across robots increases, so does the energy usage. The lesson from this first situation, then, is that in the absence of robot

[4]For experimental versions III and IV, the actual percentage of progress reports per move of a spill object is about 86% (see figure 5-4).

awareness, redundancy across robots is detrimental for those redundant robot tasks whose effects cannot be sensed through the world.

The second situation observed from figures 5-5 and 5-6 arises due to differences in the required number of instances of tasks and how well those instances can be distributed across the robot team. In this situation, both robot teams in question (versions I and II) have full awareness of the actions of other robots. The differences lie in the number of instances required by the hazardous waste cleanup mission of the *find-locations* task versus the *report-progress* task and how well they can be distributed across robots. While only a single instance of the *find-locations* task is required, up to 8 instances of the *report-progress* task are necessary to complete the mission. Although each instance of these tasks can be distributed to any robot team member with the required capabilities, no single instance can be broken down into parts to be shared by more than one robot. Thus, in the case of the one required instance of the *find-locations* task, once one of the robots has selected that action, the other robots have to just wait patiently for the first robot to finish that action (of course, another robot may take over the task due to failure by the first robot, but that is another issue). In this case, an increased degree of redundancy across the team for this action does not provide any advantage, and so the performance of the two-robot team is not different from that of the three-robot team when both have awareness.

The benefit provided by the three-robot team over the two-robot team (both with full awareness) in the case of the *report-progress* task is obtained via a reduction in the required number of instances of the task. Since the three-robot team can complete the fixed amount of spill-moving required by the mission faster than the two-robot team, the time required to complete the mission is shortened. This is turn leads to a reduction in the number of progress reports required by the mission, which leads to less work for each robot to obtain the proper number of reports. Thus, the lesson learned here is that increased redundancy of robot capabilities in the presence of full robot awareness helps when the mission requires several instances of tasks that can be distributed across the robot team; it does not help, however, for single instances of tasks that cannot be shared.

The Effect of Awareness on Time Requirements

Figures 5-7 and 5-8 show the effect of varying the relative costs of the *find-locations* and *report-progress* tasks on the average time required to complete the mission for each experimental version. The function plotted is the time function $mission\_time(version)$ defined in equation 5.3, while varying the *find-locations* or *report-progress* costs, respectively. For both of these tasks, the worst time performance occurred with version III — two robots without awareness. For the *find-locations* task, the two-robot team with awareness performed from 9% down to 1% better than the

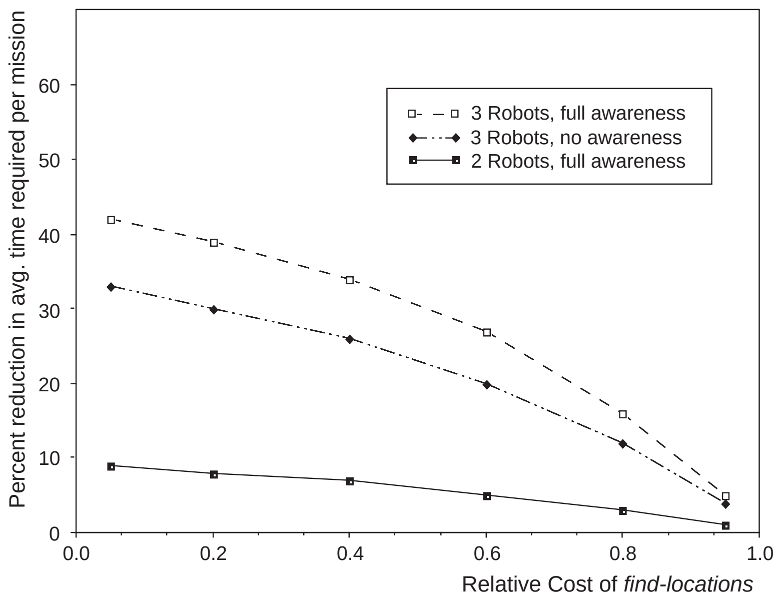

Figure 5-7: Of the 4 experimental versions for the hazardous waste cleanup mission, the one requiring the most average time to complete the mission is the situation involving two robots no awareness of their teammates' actions. The percentage reduction in average time required to complete the mission from this baseline case is shown as a function of the relative time cost of the *find-locations* task.

baseline case, the three-robot team without awareness performed from 33% down to 4% better, and the three-robot team with awareness performed from 42% down to 5% better.

For the *report-progress* task, the two-robot team with awareness performed from 2% to 31% better, the three-robot team without awareness performed from 25% to 33% better, and the three-robot team with full awareness performed from 27% to 68% better.

As we saw with the energy requirements, the presence of awareness on the robot team improves the time performance of this mission regardless of the relative cost of the redundant tasks or the level of redundancy on the team. However, the time curves do have a noticeably different character than the energy curves, and are worth understanding. A couple of observations can be made:

- Three-robot teams always give a better time performance than two-robot teams for this mission, regardless of the presence or absence of awareness.

- As the relative cost of the *find-locations* task increases, the benefits of awareness and team redundancy decrease.

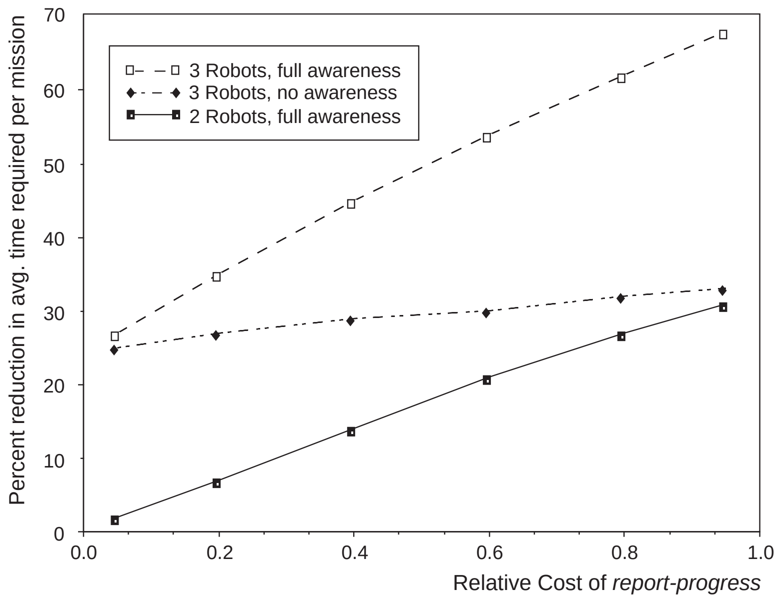

Figure 5-8: Of the 4 experimental versions for the hazardous waste cleanup mission, the one requiring the most average time to complete the mission is the situation involving two robots no awareness of their teammates' actions. The percentage reduction in average time required to complete the mission from this baseline case is shown as a function of the relative time cost of the *report-progress* task.

The first observation is easily understood for the three-robot team with awareness, since it is quite sensible that dividing up a given amount of work across more robots without replicating any tasks would result in the mission being completed more quickly than for a two-robot team either with or without awareness. However, why would a three-robot team *without* awareness perform more quickly than a two-robot team *with* awareness? The answer lies in the tradeoff between the beneficial effects of redundancy for tasks whose effects can be sensed through the world and the adverse effects of redundancy for tasks whose effects cannot be sensed through the world. When the cost of the tasks replicated due to lack of awareness is relatively low, the redundancy in task coverage for those actions whose effects *can* be sensed through the world is the dominating factor, and thus a noticeable improvement occurs. In this specific example, then, we see that when the *report-progress* relative cost is low, the three-robot team, even without awareness, provides a decided advantage because it moves the spill more quickly without incurring much of a penalty for repetitive progress reports. As the cost of *report-progress* increases, however, this advantage dwindles. It is quite interesting to note, however, that although we observe in figure 5-8 the near convergence of the performances of the three-robot team without awareness and the two-robot team with awareness, a two-robot team with awareness can actually never perform the mission more quickly than a team with a larger *report-progress* and *move-spill* coverage, even when the larger team has no awareness (that is, until we get into interference effects in larger teams). The reason is that the progress report rate per spill object move for each team member will never be greater than about .86 (see figure 5-4). The amount of *report-progress* time required per spill object move is thus $.58/2$ for the two-robot team with awareness, and $.86/n$ for any team without awareness. Since $.86/n < .58/2$ for all $n > 2$, the two-robot team with awareness can never require less total time to perform the progress reports. In addition, since larger teams move the spill objects by a time reduction factor of $1/n$ as opposed to $1/2$ for a two-robot team, the two-robot team can never accomplish the mission more quickly. The lesson learned here is that although awareness is helpful for a fixed-sized robot team, a larger team without awareness may actually be able to perform the mission more quickly if a significant proportion of the mission consists of tasks whose effects can be sensed through the world.

The second observation — that as the relative cost of the *find-locations* task increases, the benefits of awareness and team redundancy decrease — is due to a matter of proportions. In the case of the *find-locations* task, lack of awareness causes no time penalty — it simply leads all the robots to perform the *find-locations* task once at the beginning of the mission. Thus, when the *find-locations* cost is low, a significant proportional benefit in preventing the repetitive execution of the tasks *move-spill* and *report-progress* can be drawn from awareness and team redundancy. On the other hand, if the fixed, non-shareable startup cost of the mission is large compared to those portions of the mission that can be shared across the team, then it makes little time difference, proportionally, whether the team has awareness or task redundancy. The lesson learned here is that awareness and task redundancy can help with the time requirements of the mission only if the mission includes a fair percentage of shareable tasks, especially those whose effects cannot otherwise be sensed through the world.

5.3 Summary of Awareness Findings

In this chapter, I have examined the impact of the loss of communication in ALLIANCE on the performance of two-robot and three-robot teams performing the hazardous waste cleanup mission. This loss of the broadcast communication capability leads to the inability of robots to be aware of the actions of their teammates, which in turn leads robots to select their actions based purely on feedback from the world through their remaining sensors and on their own internal motivations and priorities. Since robots cannot always detect the effects of the actions of their teammates through the world, the lack of awareness in ALLIANCE can lead to the redundant execution of certain tasks required by the mission. I studied the extent of this effect on mission performance as functions of (1) the degree of redundancy in robot capabilities, (2) the ability of the robots to detect the actions of other robots through the world, and (3) the cost of the redundant tasks.

The findings can be summarized as follows:

- For robot actions whose effects can be fully sensed through the world, the lack of awareness causes no change in the time or energy required to complete the mission, for a given level of robot redundancy.

- For robot actions whose effects cannot be sensed through the world, the lack of awareness causes an increase in the energy requirements of the mission. This increased energy requirement worsens as the level of robot redundancy increases and as the cost of the redundant actions increases.

- For robot actions whose effects cannot be fully sensed through the world, the lack of awareness causes an increase in the time requirements of the mission, unless the redundant actions are taken when the robot(s) would otherwise have been idle.

- For a given mission to be completed by a robot team with full awareness, increasing the level of robot redundancy reduces the time requirements for those tasks which can be shared by the team, but it has no effect on the energy requirements for these shareable tasks.

- Increasing the level of team redundancy with full awareness does not improve the time or energy requirements of tasks that cannot be distributed across more than one robot.

- A team without awareness may be able to perform a mission more quickly than a team with a lower level of redundancy if a significant proportion of the mission consists of tasks whose effects can be sensed through the world.

Chapter 6

Additional Implementations of ALLIANCE

The ALLIANCE architecture has been successfully implemented in a variety of proof of concept applications on both physical and simulated mobile robots. The applications implemented on physical robots include the hazardous waste cleanup mission introduced in chapter 1 (also described in [Parker, 1994]) and a cooperative box pushing demonstration. The applications using simulated mobile robots include a janitorial service mission and a bounding overwatch mission (reminiscent of military surveillance). In this chapter, I describe the results of the box pushing demonstration and the simulated robot missions. Refer to chapters 3 and 5 for details on the hazardous waste cleanup mission.

All of these missions using the ALLIANCE architecture have been well-tested. Over 50 logged physical robot runs of the hazardous waste cleanup mission and over 30 physical robot runs of the box pushing demonstration were completed to elucidate the important issues in heterogeneous robot cooperation. Many runs of each of these physical robot applications are available on videotape. The missions implemented on simulated robots encompass dozens of runs each, most of which were logged in the study of the action selection mechanism.

Section 6.1 describes the physical robot experiments involving the box pushing demonstration. Section 6.2 describes the simulated janitorial service mission, and section 6.3 describes the bounding overwatch mission.

6.1 The Box Pushing Demonstration

In this report, the cooperative box pushing demonstration offers a simple and straightforward illustration of a key characteristic of the ALLIANCE architecture: fault tolerant and adaptive control due to dynamic changes in the robot team. I refer to this example as a *demonstration* rather than a *mission* to emphasize the smaller scope of the box pushing application compared to the other implementations of the ALLIANCE architecture. This demonstration was implemented for the primary purpose of providing a concise video which illustrates the key points of this report.

This box pushing demonstration requires a long box to be pushed across a room; the box is sufficiently heavy and long that one robot cannot push in the middle of the box to move it across the room. Thus, the box must be pushed at both ends in order to accomplish this demonstration. To synchronize the pushing at the two ends, the demonstration is defined in terms of two recurring tasks — (1) push a little on the left end, and (2) push a little on the right end — neither of which can be activated (except for the first time) unless the opposite side has just been pushed. I implemented this demonstration using a heterogeneous robot team of two R-2s and Genghis-II. I use this demonstration to illustrate how the ALLIANCE architecture endows robot team members with fault tolerant action selection due to the failure of robot team members, and with adaptive action selection due to the heterogeneity of the robot team. Note that my emphasis in these experiments is on issues of fault tolerant cooperation rather than the design of the ultimate box pusher. Thus, I am not concerned at present with issues such as robots pushing the box into a corner, obstacles interfering with the robots, how robots detect box alignment, and so forth.

Cooperative box pushing is a popular task for multi-robot system researchers, perhaps because of its minimal requirements for sensors and effectors. Of course, no two box pushing scenarios are defined the same, as researchers (including myself) naturally define the task to illustrate the advantages of their research. Donald *et al.* [Donald *et al.*, 1993] use a box pushing demonstration to investigate more general issues of information complexity and information invariants. They define three alternative control strategies for two-robot cooperative box pushing which vary in the communication and sensing requirements. Their third control strategy (which they call Protocol II) is of particular interest to the goals of the box pushing demonstration I have defined here, since it can accomplish one type of fault tolerant cooperation that ALLIANCE allows below in experiment 1 — namely, the ability to recover from a failed team member.[1] Protocol II uses no explicit communication, but rather assumes the presence of a sensor that allows a robot to detect the orientation of the box with respect to itself. By using orientation information, a robot can detect the effects of the actions of its teammates, and adjust its own actions accordingly by moving either left or right along the box. If a robot's teammate fails, then that robot can adjust its position right or left as it pushes to maintain alignment of the box. However, the

[1]This type of fault tolerance can only be obtained with the "uniform" version of Donald's protocol II, rather than the "almost uniform" version.

Donald control strategy is specific to box pushing, and does not address the general fault tolerant action selection problem that is addressed with ALLIANCE.

In [Noreils, 1993], Noreils describes a cooperative box pushing experiment in which one physical robot acts as the *pusher* to push a box against the wall, and a second physical robot acts as a *supervisor* to ensure that the box actually reaches the wall. If an obstacle is in the way which prevents this task from being completed, the two robots adjust their positions so that they can push the obstacle out of the way, and then the original pushing task is continued. The control architecture of these robots consists of a planner level (for planning the task), a control level (for supervising and monitoring the execution), and a functional level (for controlling the sensors and effectors). In general, recovery from errors during cooperation is performed by "leader" robots, which are designed to interact with other leader robots and "worker" robots to ensure consistency of a re-planned solution. Although this research recognizes the need for fault tolerant control, most issues of fault tolerance have not yet been well-studied for this architecture, as admitted by Noreils in [Noreils, 1993]. For instance, it is unclear in this architecture (1) how robots detect failed robots, (2) how the team recovers from the failure of a leader, and (3) how the team handles communication failures.

Kube and Zhang [Kube and Zhang, 1992] report on experiments in which robot teams utilize only simple reflex behaviors and no explicit communication to gather around a box (sensed as a bright light) and push it. Experiments are reported using both simulated and physical robot teams. Under this approach, robots have only implicit knowledge of the presence of other robot team members. Fault tolerance is achieved in this architecture by ensuring the presence of an adequate number of robots that can push anywhere along the box and still move the box. However, if the number of robots were to fall below some critical threshold, the remaining robots would not have the "know how" to compensate for the shortage, and would thus fail at their mission.

In [Asama *et al.*, 1992], Asama *et al.* report on simulation experiments in which two robots work to push objects to the sides of the room. Some of the objects can be pushed by individual robots, while other objects require the cooperation of two robots because of the weight of the object. When cooperation is required, one robot communicates a request for cooperation, to which the second robot responds when it is available. Their system also includes a path planning process to determine the desired path over which the current object should be pushed. Issues of fault tolerant control are not addressed in their approach.

In the next subsections, I describe the design of the R-2 and Genghis-II ALLIANCE software for the box pushing demonstration. I then describe the experiments using these robots and the results.

6.1.1 Robot Software Design

Since the capabilities of the R-2 and Genghis-II robots differ, the software design of the box pushing demonstration for these robots varies somewhat. I therefore describe the ALLIANCE box pushing software of these robots separately.

R-2 Control

Figure 6-1 shows the ALLIANCE implementation of the box pushing demonstration for the R-2 robots. As shown in this figure, the R-2 is controlled by two behavior sets — one for pushing a little on the left end of the box (called *push-left*), and one for pushing a little on the right end of the box (called *push-right*). As specified by the ALLIANCE architecture, the activation of each of these behavior sets is controlled by a motivational behavior. Let us now examine the design of the *push-left* motivational behavior and the *push-left* behavior set of a robot r_i in more detail; the *push-right* design is symmetric to that of *push-left*.

The sensory feedback required before the *push-left* motivational behavior within r_i can activate its behavior set is an indication that the right end of the box has just been pushed. This requirement is indicated in figure 6-1 by the *pushed-at-right* arrow entering the *push-left* motivational behavior. The right end of the box can be pushed either by some robot other than r_i, or it can be pushed by r_i itself. If r_i is the robot doing the pushing, then the *pushed-at-right* feedback comes from an internal message from r_i's *push-right* motivational behavior. However, if some robot other than r_i is pushing, then r_i must detect when that other robot has completed its push. Since this detection is impossible for the R-2s with their current sensory suites, the robots are provided with this capability by having the team members broadcast a message after each push that indicates the completion of their current push. The pushing is initiated at the beginning of the demonstration by programming the control code so that each robot "thinks" that the opposite end of the box has just been pushed.

When the sensory feedback is satisfied, the *push-left* motivational behavior grows impatient at either a rate δ_fast_R (the R subscript stands for any R-2 robot) if no other robot is performing the *push-left* task, or at a rate $\delta\_slow_R(robot\text{-}id)$ when robot *robot-id* is performing the *push-left* task.[2] When the *push-left* motivation grows above threshold, the *push-left* behavior set is activated. The *push-left* behavior set involves first acquiring the left end of the box and then pushing a little on that end. If the robot

[2]To simplify the notation, I omit the j subscript of the fast and slow impatience rates (see appendix A) since the fast rates of impatience are the same for all behavior sets in all R-2s, and the slow rates of impatience are the same functions of *robot-id* for all R-2s. I also omit the dependence upon t of these impatience rates, since I do not deal here with updating these parameters during the demonstration.

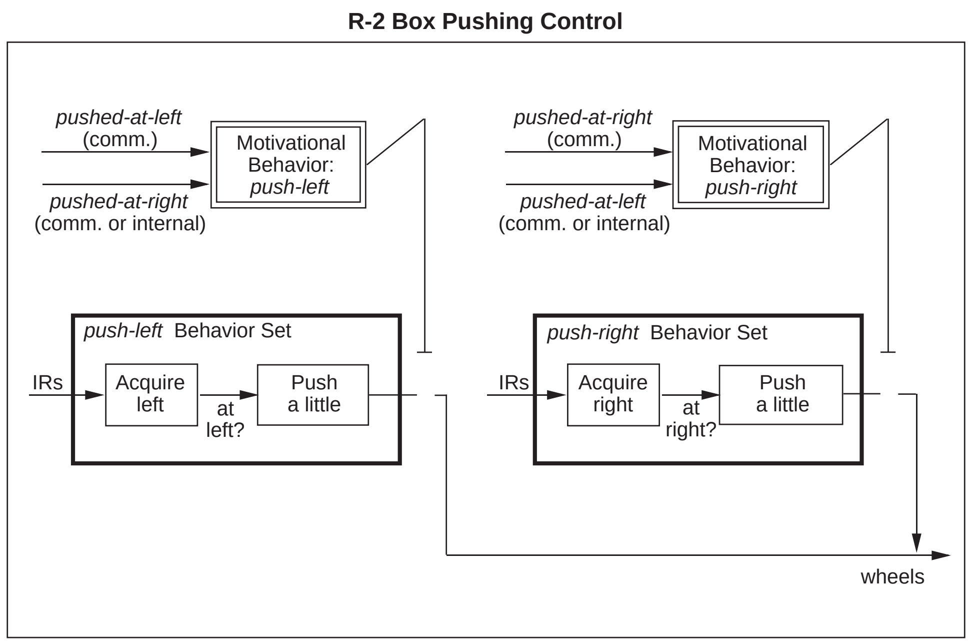

Figure 6-1: The ALLIANCE design of the R-2 software for the box pushing demonstration.

is already at the left end of the box, then no acquiring has to take place. Otherwise, the R-2 assumes it is at the right end of the box, and moves to the left end of the box by using the infrared sensors on its right side to follow the box to the end, and then backing up and turning into the box. As we shall see below, this ability to acquire the opposite end of the box during the demonstration is important in achieving fault tolerant cooperative control. At the beginning of the demonstration, I would ideally like the R-2 to be able to locate one end of the box on its own. However, since this is beyond the scope of these proof of concept experiments, an implicit assumption is made in the R-2 control that at the beginning of the demonstration, the R-2 is facing into a known end of the box.

As the R-2 pushes, it uses the infrared sensors at the ends of its gripper fingers to remain in contact with the box. The current push is considered to be complete when the R-2 has pushed for a prescribed period of time. After the *push-left* task is completed, the motivation to perform that task temporarily returns to 0. However, the motivation begins growing again as soon as the sensory feedback indicates the task is needed.

Genghis-II Control

Genghis-II and the R-2s are different in two primary ways. First, Genghis-II cannot acquire the opposite end of the box, due to a lack of sensory capabilities, and second, Genghis-II cannot push the box as quickly as an R-2, due to less powerful effectors. The first difference means that Genghis-II can only push at its current location. Thus, implicit in the control of Genghis-II is the assumption that it is located at a known end of the box at the beginning of the demonstration. The second difference with the R-2s implies that if an R-2 pushes with the same duration, speed, and frequency when teamed with Genghis-II as it does when teamed with another R-2, the robot team will have problems accomplishing its demonstration due to severe box misalignment.

Figure 6-2 shows the organization of Genghis-II's box pushing software. As this figure shows, Genghis-II is controlled by two behavior sets, each of which is under the control of a motivational behavior. Genghis-II's pushing at its current location is controlled by the *push* behavior set. The only sensory feedback which satisfies the *push* motivational behavior is that which indicates that some other robot is pushing the opposite end of the box. This requirement is shown in figure 6-2 as the *pushed-at-left/right* arrow going into the *push* motivational behavior. Once the sensory feedback is satisfied, Genghis-II becomes impatient to perform the *push* behavior at a rate δ_fast_{GP} (the G subscript refers to *Genghis-II*; the P subscript refers to the *push* behavior set). Once the motivation crosses the threshold of activation, the *push* behavior set is activated, causing Genghis-II to push the box by walking into it while using its whiskers to maintain contact with the box. Once Genghis-II has pushed a given length of time, the motivation to perform *push* returns to 0, growing again whenever the sensory feedback is satisfied.

The sensory feedback required for the *go-home* behavior set to be activated is the opposite of that required for the *push* behavior set — namely, that no other robot is pushing at the opposite end of the box. When the sensory feedback for *go-home* is satisfied, the motivation to activate *go-home* grows at the rate δ_fast_{GH} (the H subscript refers to the *go-home* behavior set), with the behavior set being activated as soon as the motivation crosses the threshold. The *go-home* behavior set causes Genghis-II to walk away from the box.

6.1.2 Experiments and Results

To demonstrate the fault tolerant, adaptive nature of the ALLIANCE architecture due to changes in the robot team capabilities, I undertook two basic experiments using the box pushing demonstration. Both of these experiments began with two R-2s pushing the box — one at each end of the box — as illustrated in figure 6-3. I note that the fast rates of impatience were set such that the delay between individual

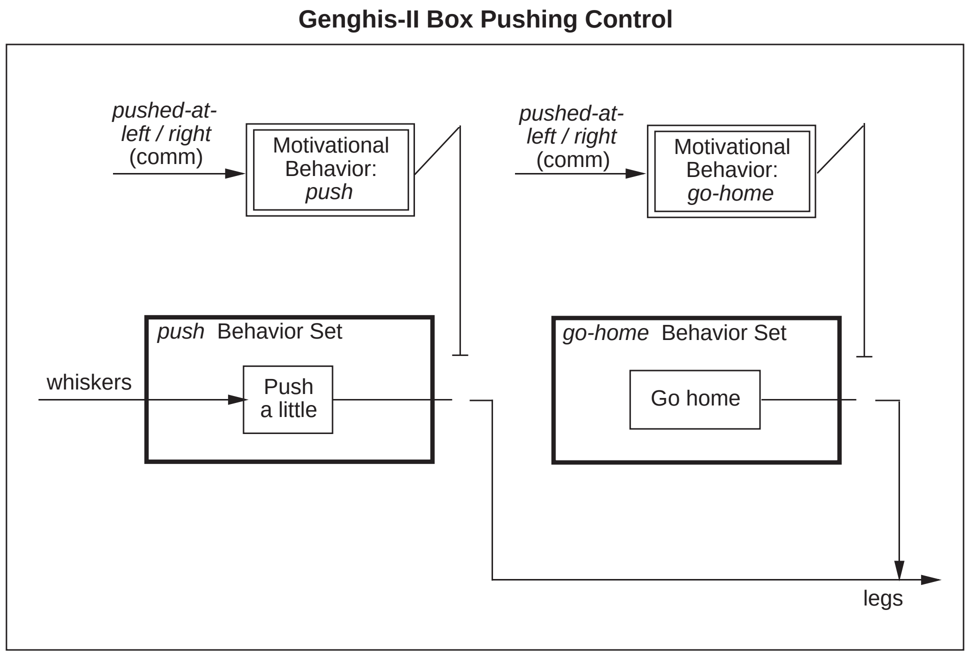
Genghis-II Box Pushing Control
pushed-at-left / right (comm)
Motivational Behavior: push
pushed-at-left / right (comm)
Motivational Behavior: go-home
push Behavior Set
whiskers
Push a little
go-home Behavior Set
Go home
legs

Figure 6-2: The ALLIANCE design of the Genghis-II software for the box pushing demonstration.

pushes by each robot is quite small — from imperceptible to about 2 to 3 seconds, depending upon when the .3 Hz communication messages actually get transmitted.

After the two R-2s push the box for a while I dynamically altered the capabilities of the robot team in two ways. In the first experiment, I altered the team by seizing one of the R-2 robots during the demonstration and turning it off, mimicking a robot failure; I then later added it back into the team. In the second experiment, I again seized one of the R-2 robots, but this time I replaced it with Genghis-II, thus making the team much more heterogeneous; I then later seized the remaining R-2 robot, leaving Genghis-II as the sole team member. The following subsections describe the results of these two experiments.

Experiment 1: Robot "failure"

As I have emphasized, a primary goal of the ALLIANCE architecture is to allow robots to recover from failures of robot team members. Thus, by seizing an R-2 and turning it off, I test the ability of the remaining R-2 to respond to that "failure" and adapt its action selection accordingly. In this experiment, what we observe after the seizure is that after a brief pause of about 5 to 8 seconds (which is dependent upon the setting of the δ_slow_R(R-2) parameter), the remaining R-2 begins acquiring

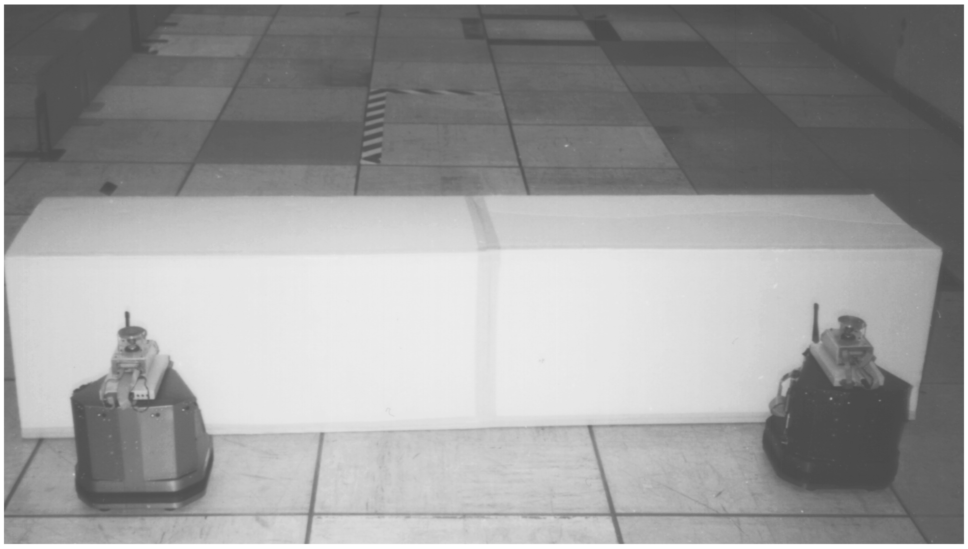

Figure 6-3: The beginning of the box pushing demonstration. Two R-2s are pushing the box across the room.

the opposite end of the box, as shown in figure 6-4, and then pushes at its new end of the box. This R-2 continues its back and forth pushing, executing both tasks of pushing the left end of the box and pushing the right end of the box as long as it fails to "hear" through the broadcast communication mechanism that another robot is performing the push at the opposite end of the box. When I add back in the second R-2, however, the still-working robot adapts its actions again, now just pushing one side of the box, since it is satisfied that the other end of the box is also getting pushed. Thus, the robot team demonstrates its ability to recover from the failure of a robot team member.

Experiment 2: Increased heterogeneity

Another goal of the ALLIANCE architecture is to allow *heterogeneous* robot teams to work together efficiently. Robots can be heterogeneous in two obvious ways. First, robots may differ in which tasks they are able to accomplish, and second, robots may differ in how well they perform the same task. In this experiment, I deal primarily with the second type of heterogeneity, in which Genghis-II and the R-2 use quite different mechanisms for pushing the box. By substituting robots during the middle of a demonstration, I test the ability of the remaining team member to respond to the dynamic change in the heterogeneity of the team.

What we observe in this experiment is that the remaining R-2 begins pushing much less frequently as soon as it "hears" that Genghis-II, rather than an R-2, is

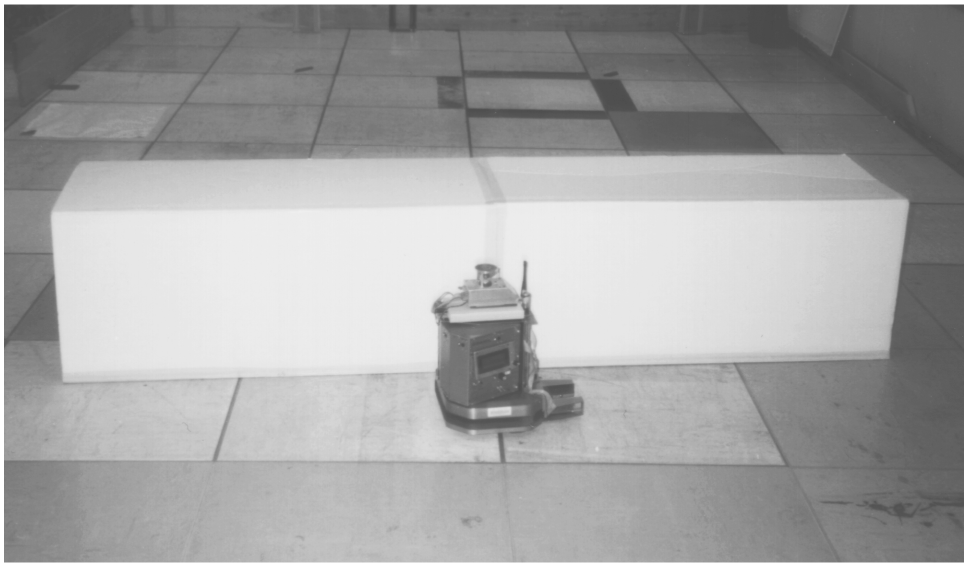

Figure 6-4: Fault tolerant action selection. In this first experiment, I seize one of the R-2 robots and turn it off. This causes the remaining R-2 robot to have to perform both tasks of the box pushing demonstration: pushing at the right end of the box, and pushing at the left end of the box.

the robot pushing the opposite end of the box. Thus, the robots remain more or less aligned during their pushing. Figure 6-5 illustrates the R-2 and Genghis-II pushing together.

The reduced rate of pushing in the R-2 when Genghis-II is added is caused by the following. First of all, the R-2's δ_slow_R(R-2) and δ_slow_R(Genghis-II) parameters differ quite a bit since Genghis-II is much slower at pushing the box than the R-2. Note that these parameter differences are easily learned by these robots using the features of the L-ALLIANCE architecture which allow the robots to the monitor the performance of robot team members. In this case, the R-2s learn parameters in which δ_slow_R(Genghis-II) is less than δ_slow_R(R-2). With this in mind, let us assume that the R-2 is pushing on the left of the box, and that Genghis-II is swapped into the team on the right end of the box. Since Genghis-II takes longer to complete its pushing than the old R-2 did, the sensory feedback of the remaining R-2's *push-left* motivational behavior is not satisfied as frequently, and thus the R-2's *push-left* behavior set cannot be activated as frequently. In the meantime, the *push-right* motivational behavior of the remaining R-2 is becoming more impatient to activate the *push-right* behavior set since it is not "hearing" that any other robot is accomplishing that task. However, since the *push-right* motivation is now growing at a reduced rate of impatience, δ_slow_R(Genghis-II), the motivation to activate the *push-right*

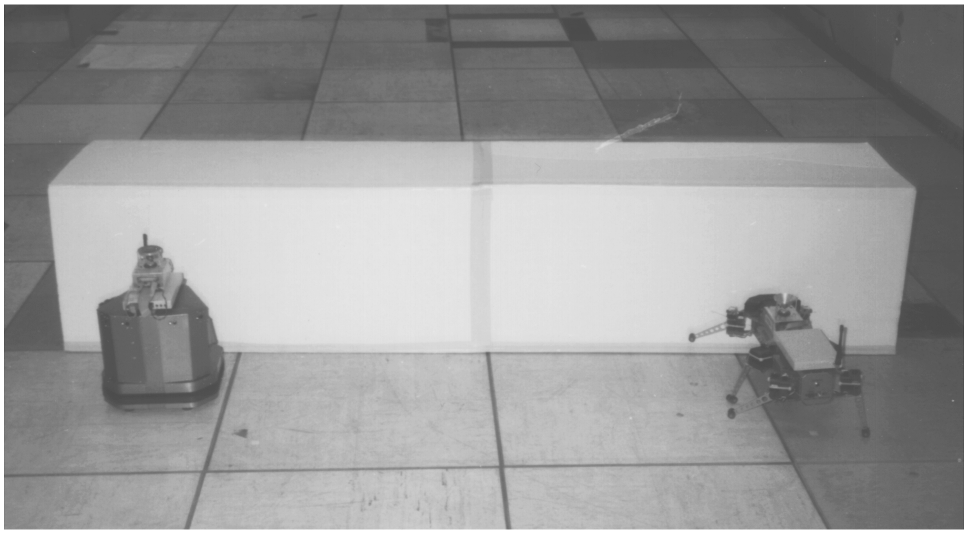

Figure 6-5: Adaptivity due to heterogeneity. In this second experiment, I again seize one of the R-2 robots, but this time I replace it with Genghis-II. Since Genghis-II cannot push as powerfully as an R-2, the remaining R-2 robot adapts its actions by pushing less frequently.

behavior set does not cross the threshold of activation before Genghis-II announces its completion of the task. This in turn prevents the remaining R-2 from taking over the push of the right side of the box as long as Genghis-II continues to push. In this manner, the R-2 demonstrates its ability to adapt to a dynamic change in team heterogeneity.

I complete this experiment by removing the remaining R-2 from the team. This causes Genghis-II to activate its *go-home* behavior, as shown in figure 6-6. Thus, Genghis-II also demonstrates its adaptive action selection due to the actions and failures of robot team members.

6.1.3 Discussion

One last point I want to stress with the box pushing demonstration is the ease with which this demonstration was implemented using ALLIANCE. The time required for me to implement this demonstration on the robots from when the box was built until I had the videotaped results was exactly 1 week. Granted, my definition of the box pushing demonstration does not include the very difficult problems of maintaining box alignment, preventing the robots from pushing into corners, and so forth. However, as stated earlier, the point of these experiments is to demonstrate the adaptive action selection features of ALLIANCE rather than the ultimate box pusher. Another

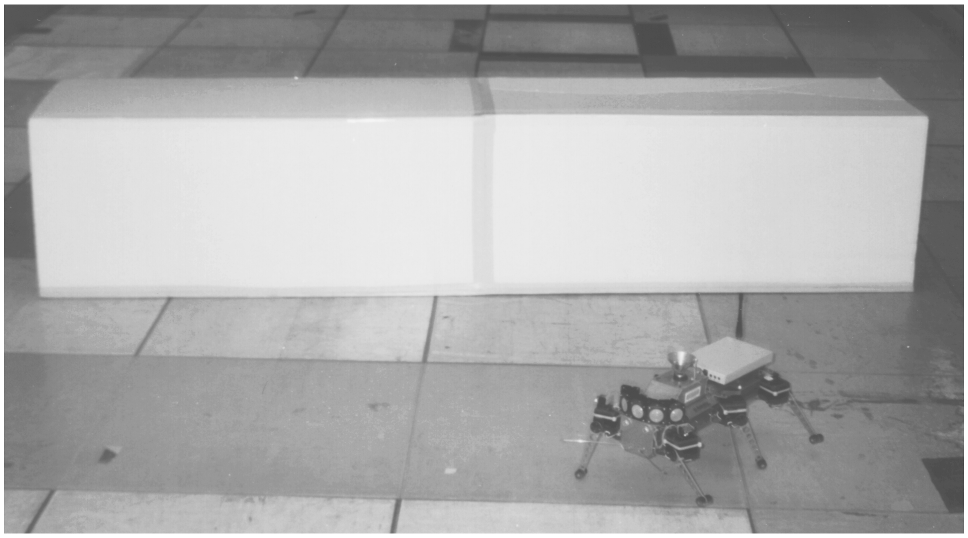
Figure 6-6: Response to robot failure. At the end of the second experiment, I seize the remaining R-2 robot, leaving Genghis-II alone to perform the demonstration. Since Genghis-II cannot complete the demonstration on its own, it activates its *go-home* behavior set.

way of stating this is that my emphasis is on the development of the *motivational behaviors* rather than on the behavior sets, since the interaction of the motivational behaviors is what specifies the action selection characteristics of the robot team. In fact, the implementation of the motivational behaviors for this demonstration was quite straight-forward. The main problems I had in this implementation concerned weaknesses in robot sensing capabilities, which made the development of the behavior sets challenging. The primary difficulty was in overcoming noisy infrared sensors and inconsistent wheel servo control loops to allow the robot to reacquire the opposite end of the box without losing the box. Nevertheless, this problem was overcome well enough to obtain numerous videotaped runs of successfully completed demonstrations.

6.2 The Janitorial Service Mission

As another illustrative example of the type of cooperation we would like a collection of robots to be able to accomplish, consider a janitorial service team of robots. This type of robot team is required to clean a room in an office building that is unfamiliar to the robots at the beginning of the task, and may change dynamically due to people occupying the room. The overall mission consists of three high-level tasks: emptying the garbage, dusting the furniture, and cleaning the floors. The robots assigned to this mission are not familiar with the capabilities of the other robots in the team, which may change dynamically due to mechanical failure or environmental change. Each robot has a different mixture of capabilities which allows it to complete a portion of the mission on its own. However, since no single robot has all the capabilities necessary to accomplish the mission alone, the robots must work as a team. Due to limited time and energy, we would also like the robots to accomplish their mission efficiently, minimizing redundant actions as much as possible. The efficiency requirements and the dynamic environment of this mission require the robots to adapt their activities over time due to the current and previous actions of other robots, and to the sensory feedback received from the environment.

This mission offers the opportunity to illustrate methods of implementing the ALLIANCE architecture for missions involving numerous independent repetitions of the same subtask. For example, although this mission is composed of three high-level tasks — emptying the garbage, dusting the furniture, and cleaning the floor — each of these tasks involves a number of separate subtasks which could be performed independently by more than one robot, such as having different robots dust different pieces of furniture simultaneously. Of course, we would not want to have a separate motivational behavior and behavior set for each piece of furniture to dust, or for each garbage can to be emptied. Instead, we illustrate some techniques for allowing one motivational behavior to control which subtask a given robot elects to perform. A slightly different method of this task subdivision is used in this mission for each of the three high-level tasks.

In the remainder of this section, I describe the results of implementing this janitorial service mission using the robot simulator described in chapter 2. First, the software control of the robots is presented, followed by the results of the team's performance.

6.2.1 Robot Software Design

In my experiments with the janitorial service team, I varied the capabilities of the robots so that different robots would be able to perform different subsets of the required tasks. For example, one robot may be able to dust the furniture and clean the floor, while another robot may be able to empty the garbage and clean the floor. Rather than illustrating all possible combinations, however, this subsection describes the capabilities of a generic robot that is able to perform all of the tasks of this mission. Since the behavior sets are independent, those behavior sets not appropriate for a robot without all of these capabilities can be easily removed from that robot's software control. Figure 6-7 shows the software control of a robot which can perform all of the tasks required in the janitorial service mission. In this design, three behavior

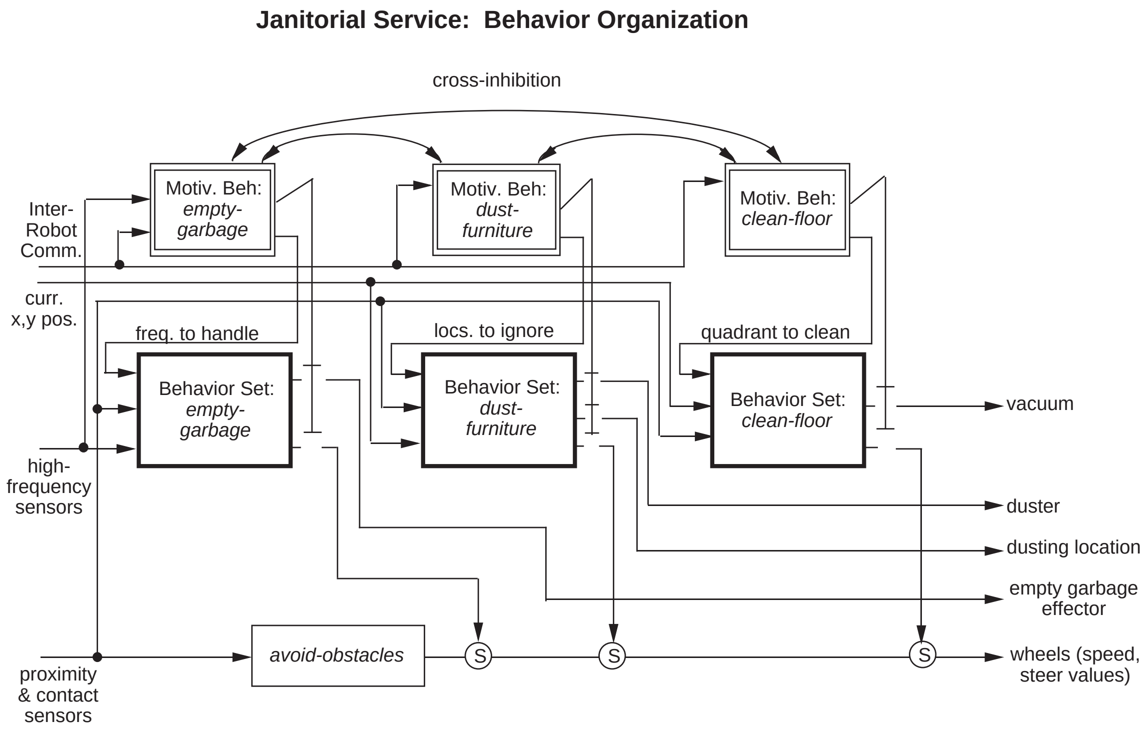
Janitorial Service: Behavior Organization
cross-inhibition
Motiv. Beh: empty-garbage
Motiv. Beh: dust-furniture
Motiv. Beh: clean-floor
Inter-Robot Comm.
curr. x,y pos.
freq. to handle
locs. to ignore
quadrant to clean
Behavior Set: empty-garbage
Behavior Set: dust-furniture
Behavior Set: clean-floor
high-frequency sensors
vacuum
duster
dusting location
empty garbage effector
avoid-obstacles
S
S
S
wheels (speed, steer values)
proximity & contact sensors

Figure 6-7: The ALLIANCE-based control for the janitorial service mission. Not all inputs and outputs are shown. Refer to figures 6-8 through 6-10 for more detail.

sets — *empty-garbage*, *dust-furniture*, and *clean-floor* — are each controlled by a motivational behavior.

A number of assumptions were made in this application to make this mission tractable. As with the box pushing demonstration, the purpose of this implementation is to illustrate the ability of ALLIANCE to generate fault tolerant, adaptive cooperative control; the purpose is not to generate the ideal garbage emptying behavior, furniture duster, or floor cleaner. Thus, I experimented with various approaches to the design of the janitorial service behavior sets, and made several assumptions about the capabilities of the simulated robots and the structure of the robot team's environment. These assumptions are described in the following subsections as the control for each of the three behavior sets is discussed.

Empty-garbage behavior set

The software design of the *empty-garbage* behavior set is shown in figure 6-8. To simplify a robot's detection of the garbage cans, I assume that a high frequency emitter is attached to each can. To allow the garbage emptying task to be divided

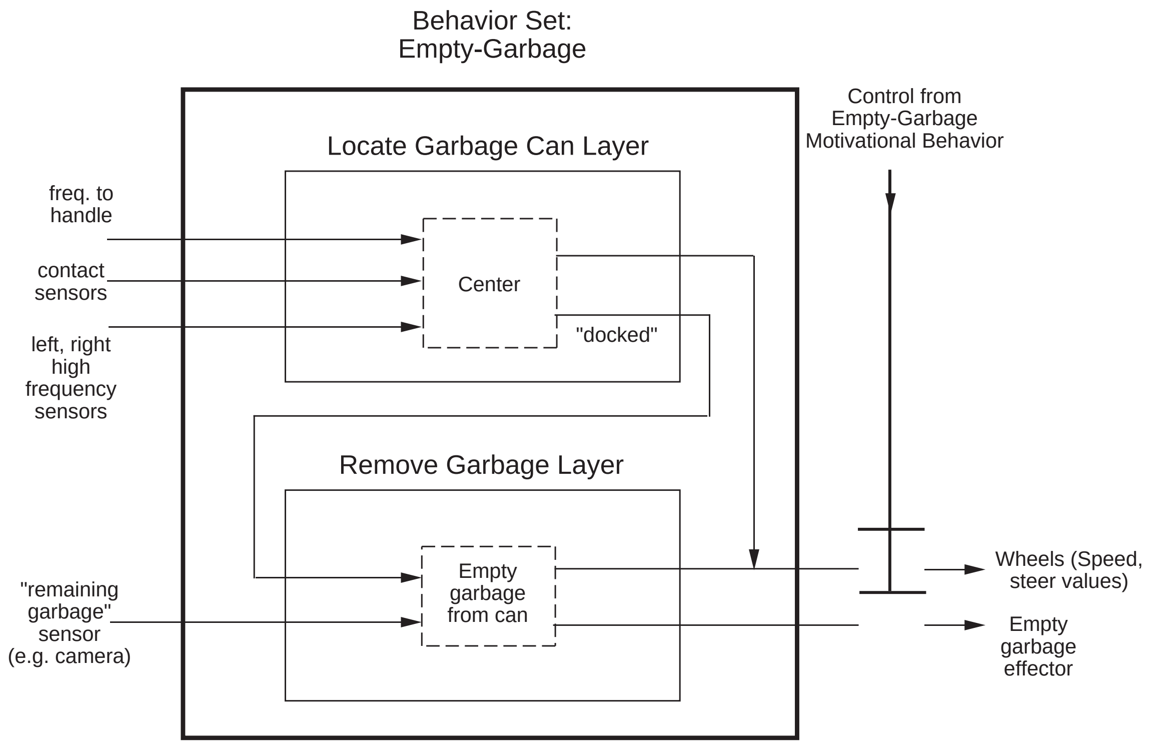

Figure 6-8: The robot control organization within the *empty-garbage* behavior set.

across more than one robot, I assume that a small number of different frequency emitters are available, and are distributed somewhat uniformly across the garbage cans in the room. I further assume that each garbage emptying robot has two sensors for detection of each frequency used in the mission, for use in localization. Using more than one frequency allows one robot to concentrate on emptying the cans emitting a given frequency while some other robot empties those of a different frequency. Finally, I assume that once the garbage is emptied out of a particular can, that garbage can's high frequency emitter is turned off.

The sensory feedback required before the *empty-garbage* behavior set can be activated is the detection of some high frequency sound which indicates the need to empty a garbage can. Once this sensory feedback is received, the *empty-garbage* motivational behavior grows impatient to activate its behavior set for that high frequency sound at either a fast or a slow impatience rate, depending upon the current activities of the team members. Note that the motivational behavior monitors more than one rate of impatience for this behavior set — one for each pair of frequency detectors of the given robot. If some other robot is currently working on emptying the garbage cans emitting a given frequency (as indicated through its broadcast communication message of the type "r_i emptying garbage frequency f"), then the corresponding motivation grows at a slow rate; otherwise, it grows at a fast rate. Once the motivation to activate the behavior set for a given frequency value has grown above threshold, the motivational behavior sends a message to the behavior set indicating the garbage can frequency that behavior set should listen to, and the behavior set becomes active. The behavior set then proceeds to follow the gradient of the given frequency until it reaches the garbage can, at which point the effector for emptying the garbage (which could be a powerful vacuum, a manipulator, etc.) is activated until the can is emptied.

Dust-furniture behavior set

The software design of the *dust-furniture* behavior set is shown in figure 6-9. In this mission, a *dustable* piece of furniture is defined as any object less than a given height. This is implemented on the robots by having two rings of proximity sensors — one higher than the given dustable object height and one lower than the given height. Thus, any object which triggers the second ring of sensors, but not the first ring, should be dusted. To reduce the difficulty of the dusting robot's task, I assume that each furniture dusting robot is told the number of dustable objects in the room at the beginning of the mission. However, the robots do not know the locations of the furniture, and must thus search for them. I further assume that some global positioning system, such as that described in chapter 2, is available to the robots for uniquely identifying a dustable piece of furniture by location. Finally, I assume that when a robot activates the *dust-furniture* behavior set, it broadcasts one of three messages: (1) if the robot is currently searching for a dustable object, it broadcasts a generic "r_i dusting furniture" message; (2) if the robot has actually located a yet-to-be-dusted dustable object, it broadcasts a message such as "r_i dusting furniture at location x, y", which indicates the global position of the piece of furniture it is dusting; or (3) if the robot has concluded the dusting of a piece of furniture, it broadcasts a message such as "r_i completed dusting of object at location x, y".

The sensory feedback required for the *dust-furniture* motivational behavior to consider activating its behavior set in robot r_i is that fewer than the given number of dustable objects have actually been dusted either by r_i or by some other robot team member (as indicated through the broadcast communication mechanism). Thus, the *dust-furniture* motivational behavior becomes impatient at a fast rate of impatience as long as dustable objects remain to be dusted. However, the *dust-furniture* task is considered to be a task that can be shared by more that one robot team member. Thus, we would like more than one robot to be able to search for objects to dust, but we do not want them to attempt to dust the same object at the same time, or to dust an already-dusted piece of furniture. Thus, the motivational behavior also keeps

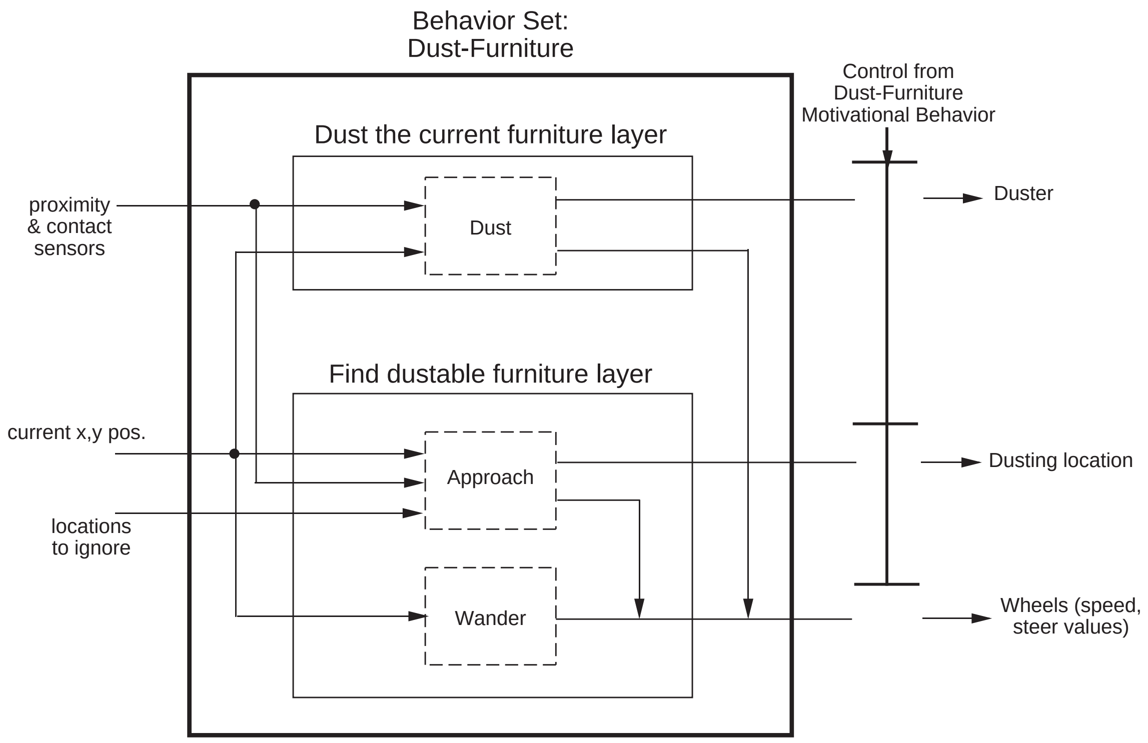

Figure 6-9: The robot control organization within the *dust-furniture* behavior set.

track of the locations of furniture objects that have been dusted or are being dusted by some other robot. If another robot, r_k, is currently dusting some object, then the motivational behavior of robot r_i allows r_k some period of time (according to a slow rate of impatience) to complete the dusting of that object. However, if that object remains undusted for a period of time (as indicated by the communication feedback), r_i's *dust-furniture* motivational behavior becomes impatient with r_k, and thus the object that r_k is dusting also becomes fair game for r_i to dust. The information about which dustable objects to ignore is conveyed to the behavior set by the motivational behavior. Thus, whenever the motivation to activate the *dust-furniture* behavior set crosses the threshold of activation, the motivational behavior also sends the locations to the *dust-furniture* behavior set of furniture objects to ignore. The *dust-furniture* behavior set then searches for objects to dust, but ignores those dustable objects which are located at one of the locations on the "ignore list". In this manner, the robots can share the task of dusting the furniture without getting in each other's way.

The *dust-furniture* behavior set involves a simple wandering technique to search the room for objects yet to be dusted. A more methodical method of searching the room, such as that used in the *clean-floor* behavior set (see below), could certainly be used instead of a random walk. However, it was interesting to use a variety of techniques in this application to investigate the characteristics of the various approaches. Once a dustable object not on the "ignore list" is located, it is approached, and the robot uses some dust effector to dust the object as it circles around the accessible sides of the object. Note that I did not model the kinematics and dynamics of a physical dust effector here — a major simplifying assumption. Again, this type of modeling is outside the scope of this demonstration of adaptive cooperative control.

Clean-floor behavior set

The third task of the janitorial service team is to clean the floor. The organization of the *clean-floor* behavior set is given in figure 6-10. To ensure that the entire floor gets cleaned efficiently, this behavior set utilizes a coarse grid map to keep track of the areas already cleaned, those yet to be cleaned, and those that are inaccessible due to obstacles. An assumption for this behavior set is that the room to be cleaned is rectangular, but its dimensions are unknown. A cleaning robot is then first required to circle the perimeter of the room to determine its dimensions (not unlike the approach to locating the spill in the hazardous waste cleanup mission — see section 3.8), cleaning as it goes, and then to begin cleaning the remainder of the room. As with the *empty-garbage* and *dust-furniture* tasks, we would like the *clean-floor* task to be potentially divided across more than one robot. Thus, this behavior set is designed to clean the floor by quadrants after the initial room dimensions are determined, with different robots potentially cleaning different quadrants of the room. The messages communicated by the *clean-floor* motivational behavior are of three types: (1) "r_i finding room dimensions", (2) "r_i cleaning quadrant q", and (3) "r_i completed cleaning of quadrant q".

The sensory feedback required before the *clean-floor* behavior set can be activated in robot r_i is that some quadrant remains to be cleaned either by robot r_i or by some other robot, as determined from the broadcast communication messages. The motivation to activate the *clean-floor* behavior set thus grows at a fast rate of impatience as long as some quadrant remains to be cleaned. However, as with the *dust-furniture* and *empty-garbage* behavior sets, we do not want robots to interfere with each other's efforts by having all the cleaning robots selecting the same quadrant to clean, to the exclusion of the remaining quadrants. Thus, the motivational behavior keeps track of the quadrants currently being cleaned by other robots, and becomes impatient to clean those quadrants at a slow rate of impatience. When the motivation to activate *clean-floor* crosses the threshold of activation, the motivational behavior informs the *clean-floor* behavior set of the quadrant it should clean, based upon these rates of impatience.

When the *clean-floor* behavior set is activated in robot r_i, r_i first determines the dimensions of the room as described above (if it has not already found the dimensions), and then begins cleaning the quadrant specified by the motivational behavior. A more efficient method of implementing this behavior set would be to separate this task into two separate tasks — (1) finding the room dimensions, and (2) cleaning a quadrant — as done in the hazardous waste cleanup mission (see section 3.8). This would allow one robot to find the room dimensions and communicate the information to other floor cleaning robots.

Cleaning a quadrant requires the robot to visit every square within a coarse grid of that quadrant, or determining through contact and proximity sensors that a grid area is inaccessible due to obstacles. The simplistic algorithm used for covering the quadrant is as follows:

```
If (grid cell to my right is unvisited), then turn right.
Else, if (grid cell to my front is unvisited), then go straight.
    Else, if (grid cell to my left is unvisited), then go left.
        Else, search for an unvisited, unoccupied cell.
            If found, head toward it.
            Else, declare completion.
```

As the robot traverses the quadrant, it uses its proximity and contact sensors to mark grid cells occupied by obstacles. It also marks its current grid location as visited. The coarseness of the grid, the location of obstacles, and the simplistic nature of this traversal algorithm leads to an interesting "snaking" pattern of coverage, as shown in the later snapshots of figure 6-12, which is explained in more detail below.

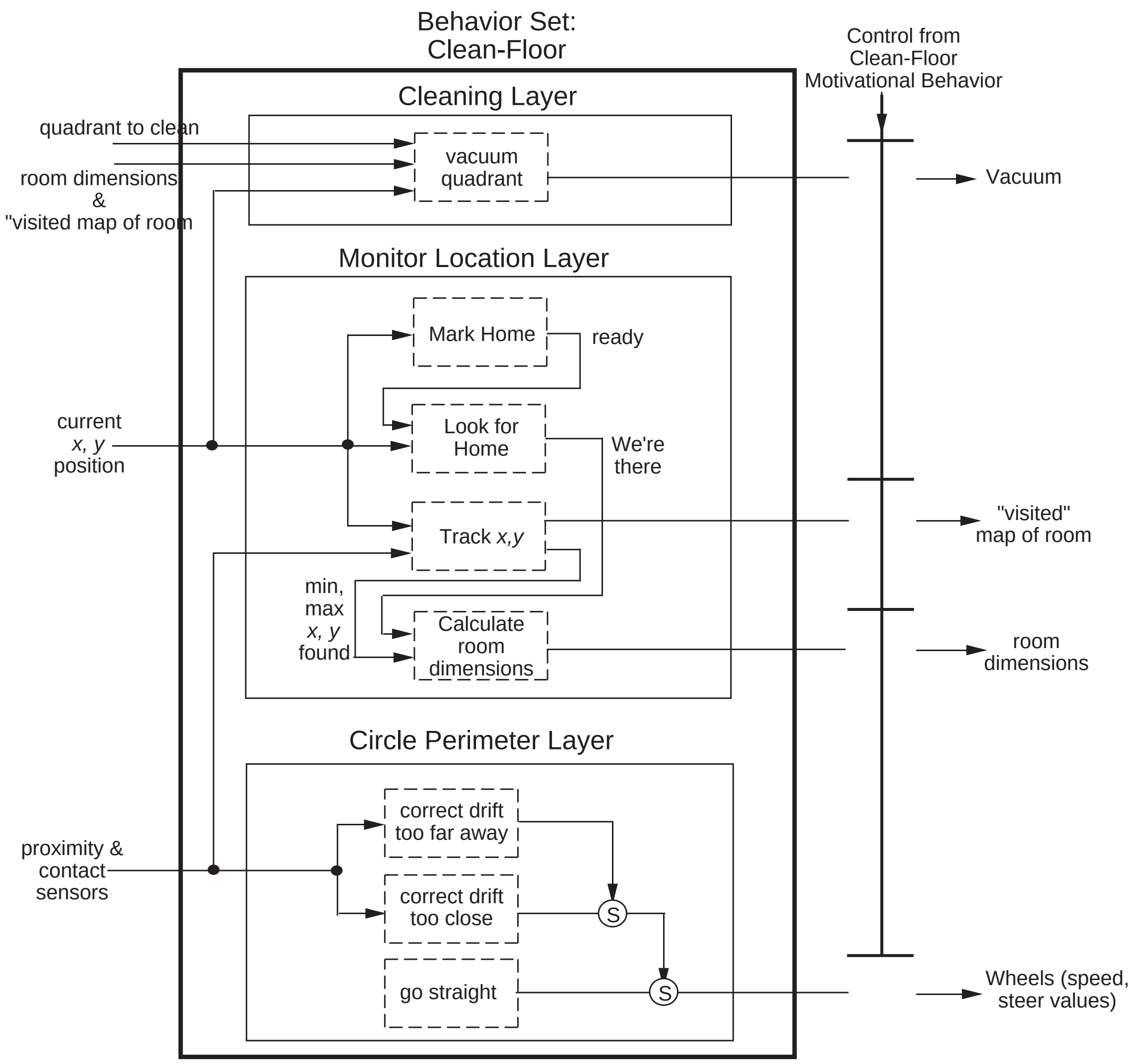

Figure 6-10: The robot control organization within the *clean-floor* behavior set.

Confirmation of completed tasks

An additional comment is in order here on how robots confirm the completion of tasks performed by other robots. As I have emphasized throughout this report, mechanical problems may lead a robot to "think" that it has successfully completed a task when, in fact, it has not. Thus, it is important for achieving truly fault tolerant cooperative control that robot team members verify the success of the actions of its teammates. However, we do not want robots to spend all of their time checking up on their neighbors when known incomplete tasks remain to be performed. A compromise, then, is to provide robots with additional low-priority behavior sets (that is, behavior sets with very slow rates of impatience) that are responsible for verifying the results of previously completed tasks. These behavior sets would be activated towards the end of the mission after the primary tasks are complete, and would involve re-visiting dustable objects, garbage cans, and/or quadrants of the room to ensure the tasks were successfully completed. If the sensory feedback indicates that a task was not successfully completed, this feedback would again trigger the appropriate behavior set in that robot, and thus cause that task to be re-executed. Although I did implement this type of verification behavior set in my simulation experiments, the details are omitted from figure 6-7.

6.2.2 Results

Snapshots of a typical run of the janitorial service robot team simulation are shown in figure 6-12. In this run, the team is composed of three robots, which are shown in the lower left corner of the first frame of figure 6-12. Each of these robots can perform two of the tasks of the janitorial service mission: robot r_1 (the leftmost robot) can perform *empty-garbage* and *dust-furniture*, robot r_2 (the center robot) can perform *empty-garbage* and *clean-floor*, and robot r_3 (the rightmost robot) can perform *dust-furniture* and *clean-floor*. In this example, the mission involves three garbage cans and three dustable objects; figure 6-11 indicates the identity of the various objects in these simulation snapshots.

Under the ALLIANCE control, these robots were able to successfully divide the tasks among themselves in a reactive and dynamic fashion to accomplish their mission without duplicating the actions of other robots. As an example of the adaptation of action selection due to the actions of other robots, consider the first 4 frames of figure 6-12. Initially, robots r_1 and r_2 both select the action of emptying the garbage, and both head up toward the closest garbage can, while robot r_3 elects to dust the furniture. However, upon hearing that robot r_2 is also headed to empty the same garbage can, robot r_1 is satisfied that that garbage can will be emptied, and thus selects another garbage can to empty (in this case, the one to the lower right, as

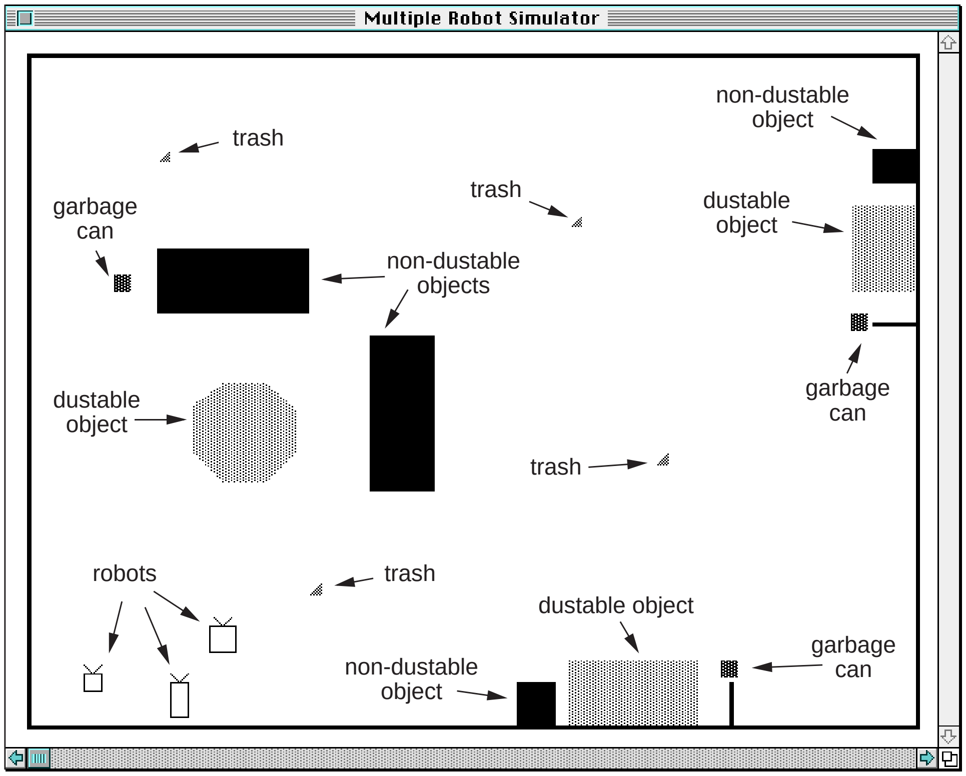

Figure 6-11: Labeled objects in a typical janitorial service experiment.

shown in the second snapshot). After emptying its garbage can, robot r_2 then heads toward the third garbage can to the right center of the room (as seen in the third and fourth snapshots), bypassing the garbage can that robot r_1 has emptied. Robot r_1 is then satisfied that all garbage cans will be emptied, even though robot r_2 has not yet reached the third garbage can, and proceeds to select another action — that of dusting furniture — as seen in the fourth snapshot.

Also shown in the fourth snapshot is robot r_2 completing the emptying of the last garbage can. Robot r_2, in snapshot five, then selects to clean the floor, which requires it to first circle the perimeter of the room, as shown in snapshots five through nine. In the meantime, back in the fifth snapshot, r_3 completes its dusting of the circular object, and proceeds to search for another object to dust. It wanders the room until the sixth snapshot, at which time it hears that r_1 has completed dusting the last piece of furniture. This causes r_3 to go on to another task, namely that of cleaning the floor. To clean the floor, robot r_3 first circles the perimeter of the room, as shown in snapshots six through ten. It then proceeds to clean the upper left quadrant.

After r_1 completes its furniture dusting in snapshot six, all of its tasks — emptying the garbage and dusting the furniture — are complete. Thus, it has nothing left to do (this example does not include the verification behavior sets), causing it to wait out the rest of the mission in its current location.

Once r_2 completes its circle of the perimeter in snapshot nine, it begins cleaning the lower right quadrant in snapshots ten and eleven while robot r_3 continues its cleaning of the upper left quadrant. Robot r_2 then goes on to clean the floor of the upper right quadrant in snapshot 12, and then to the lower left quadrant in snapshots 13 through fifteen. In the meantime, the obstacles in the upper left quadrant have caused r_3 difficulties in cleaning the upper left quadrant. It finally completes its cleaning of that quadrant in the final snapshot, at which time the mission is complete.

In other experiments with this mission, many unexpected events have been modeled to illustrate the adaptiveness of the architecture to the dynamic environment and the actions of other robots. For example, in the discussion above, if either of the robots r_1 or r_2 were unsuccessful in emptying one of the garbage cans, the other robot would become impatient with the lack of progress and proceed to empty that can regardless of the fact that the other robot had selected that action earlier. If additional garbage cans are added, the robots react fluidly to the environmental change and empty the new cans as if they were always present. If an existing garbage can is suddenly empty, the robots again react immediately by pursuing some other task. New team members can be added to the group and are allowed by the existing team members to help with the mission. Many other such changes to the environment and the abilities of the robots were simulated and handled successfully by this architecture.

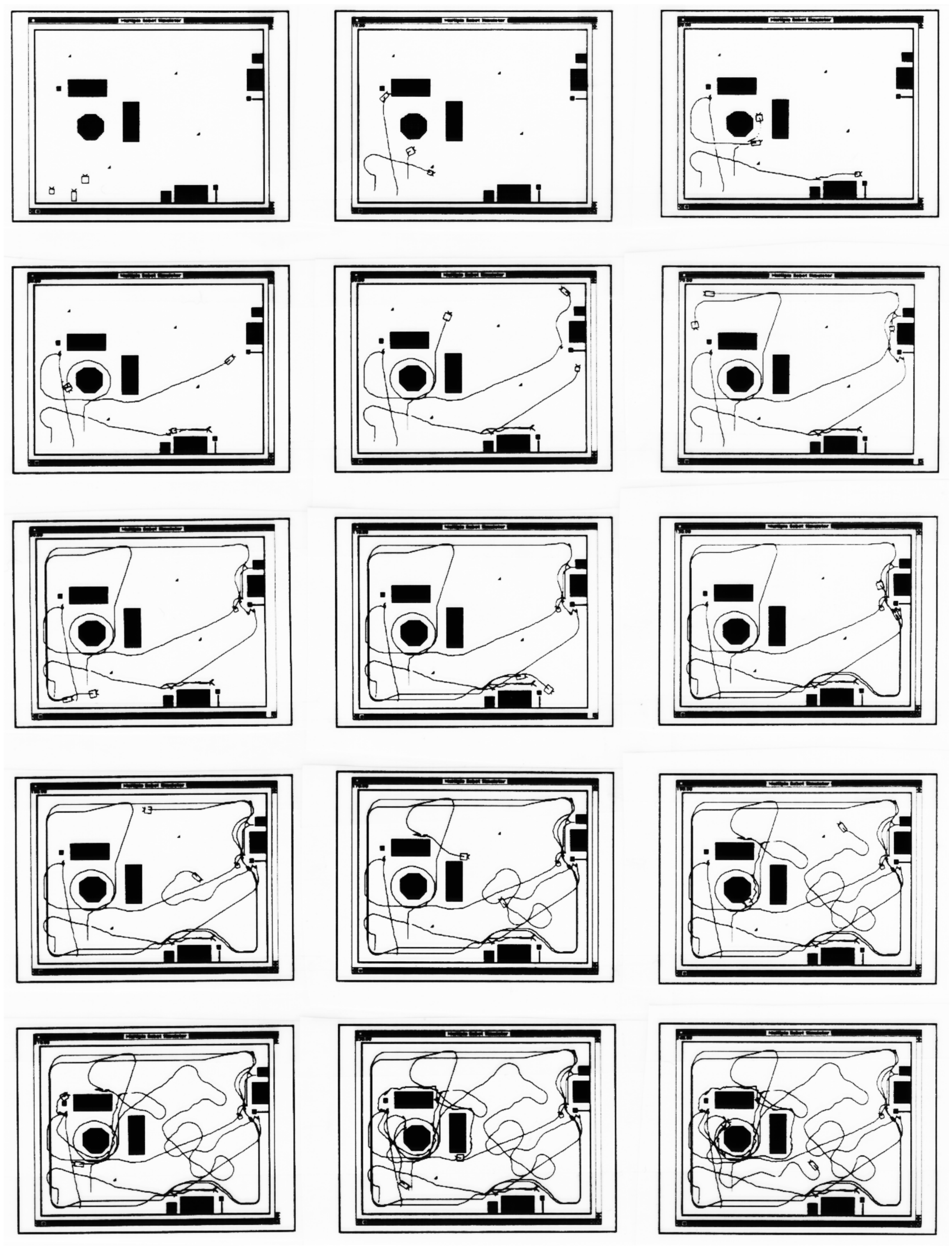

Figure 6-12: A typical run of the janitorial service simulation (read the snapshots from left to right, top to bottom).

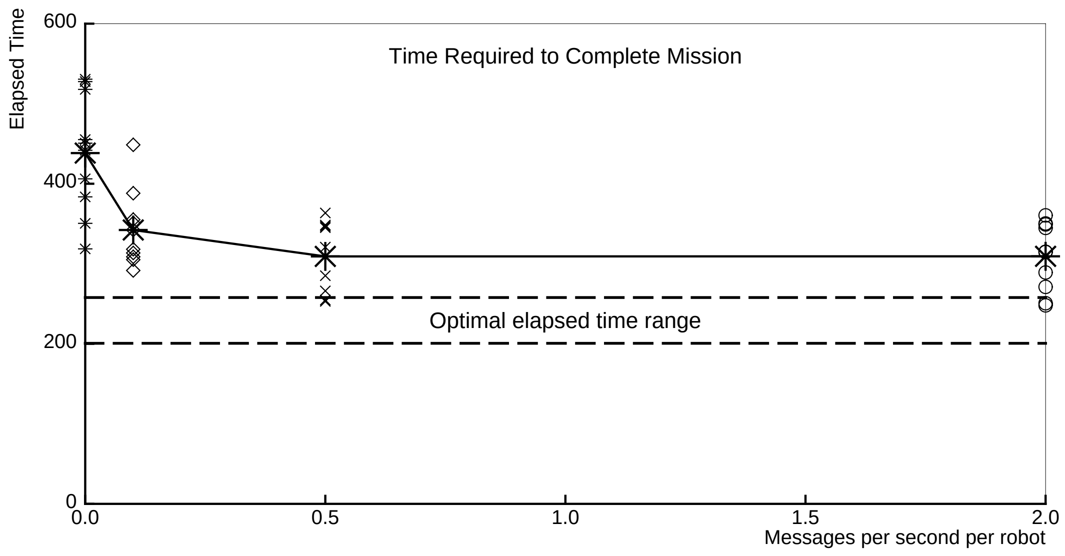

Figure 6-13: Elapsed time results of 10 runs of 4 values of ρ_i. (Asterisks denote the mean values.)

6.2.3 Effect of Robot Awareness

As described in chapter 5, the degree to which robots are aware of the actions of their teammates affects the efficiency of the mission execution. The results in chapter 5 were derived from experimentation with the hazardous waste cleanup mission. In this subsection, I present the results of a related study of robot awareness for the janitorial service mission. In this study, I present the results of varying the rate of broadcast communication, ρ_i, to determine its effect on the resulting group behavior. In these experiments, the performance measure utilized to provide quantitative comparisons between the cooperative team behaviors across experiments was the *mission completion time*, which is simply the elapsed time from the beginning of the mission until the last task of the mission is completed. Figure 6-13 shows the results of four experiments that were conducted which varied the rate of communication for each robot and compares these results with the optimal time expenditure that is possible[3]. The measurements for each of 10 runs for each communication rate are shown, along with the mean values. The variance in performance for a given communication rate is due to sensory and effector noise.

In the first experiment, ρ_i equals 0 for all i, which means that no communication at all took place, which in turn means that no robot knew anything about the

[3]The optimal time measure was obtained by human engineering all the parameters to achieve the minimum time possible for this task with the given robots.

current actions of other robots. We discover that even though the robots required much more time than the optimal, they were still able to accomplish their mission successfully, completing the mission in an average of 438 time units. This result is important because it illustrates the robustness of the architecture even amidst complete communication breakdown.

In the second experiment, ρ_i equals 0.1 for all i, which means that each robot broadcasts a message of its current activities at a rate of 1 every 10 seconds. We see that this small amount of communication significantly improved performance over no communication at all, with a mean reduction of 22% in required time.

In the third experiment, ρ_i equals 0.5 for all i, so that robots communicated a message every 2 seconds. Again, we see a performance improvement over the previous experiment of 10% in time.

However, we see from the fourth experiment, in which ρ_i equals 2 for all i, that we are not getting closer to the optimal solution. As might be expected, one cannot necessarily achieve the optimal solution by just increasing the communications rates. Instead, achieving the optimal solution involves varying other parameters through the L-ALLIANCE learning mechanism to improve the efficiency of the team performance. The janitorial service mission is a good example of the type of mission in which the L-ALLIANCE efficiency mechanisms would be particularly useful. This mission involves a number of independent tasks which can be executed by differing subsets of robot team members at possibly differing levels of performance. By using the techniques described in chapter 4, the robot team members could use learned knowledge about the capabilities of their teammates to select their tasks to achieve very efficient mission completion times. Unfortunately, time did not allow experimentation of the L-ALLIANCE mechanisms in this particular application. However, the generic results presented in chapter 4 apply directly to this type of application.

Thus, the findings from these experiments support those of chapter 5: although useful work can be accomplished without robots knowing about other team members' activities, improved performance can be achieved when robots are aware of the current actions of their teammates. However, these experiments also show that knowledge of current activities alone is not sufficient to achieve the optimal performance, and thus additional learning mechanisms are required to achieve optimal performances.

6.3 The Bounding Overwatch Mission

An additional, quite different, simulation application also implemented using ALLIANCE is the bounding overwatch problem. The point of this mission is to illustrate how ALLIANCE can be used for applications that involve a lot of ordering dependencies among tasks, rather than being composed of several independent tasks.

This mission requires a team of two types of robots to dynamically divide themselves into two subgroups having equal distribution of robot types, and then to travel to the initial assembly points of their respective subgroups and determine a subgroup leader. Next, one team must head out for the next waypoint (i.e., they *bound*) while the other team monitors their progress and remains alert for danger (i.e., they *overwatch*). Once the first team reaches its waypoint, the roles of the teams switch, so that the first team overwatches while the second team bounds. As the reader may suspect, this mission is motivated by a military surveillance scenario, in which a team of autonomous vehicles (such as tanks) must safely traverse an area thought to be occupied by enemy forces.

6.3.1 Robot Software Design and Results

Figure 6-14 shows the ALLIANCE-based control of the robots under the bounding overwatch mission. At the beginning of the mission, the only behavior set whose sensory feedback is satisfied is the *join-group* behavior set. Since this task must be performed by all robot team members, the motivational behaviors in all the robots activate this behavior set at the beginning of the mission This behavior set is important because it allows the team of robots to divide themselves into two equal subgroups. This division is accomplished using the following simple rule in each robot:

```
Wait a random length of time t (between 0 and some prespecified
    maximum time).
While waiting, monitor the messages of robot team members,
    keeping track of the number and type of robots in each
    subgroup so far.
After the random wait period is over, do the following:
   1.  Select the subgroup with the minimum number of my type so far.
   2.  If the two subgroups have equal distributions of my type,
         Then:  Select the group with the fewest members,
                  breaking ties arbitrarily.
   3  Broadcast the group I have joined and my robot type.
```

The prespecified maximum time of a wait should be long enough to reduce the likelihood of interference between robot messages to an acceptable level. Once a robot has joined a group, it moves to the prespecified gathering location for its group. The first snapshot of figure 6-15 shows the initial location of a group of eight robots — four of each of two types. The second snapshot shows the robots dynamically dividing into the two groups and moving to the specified gathering locations (indicated in figure 6-15 by the two small triangles closest to the robot starting locations).

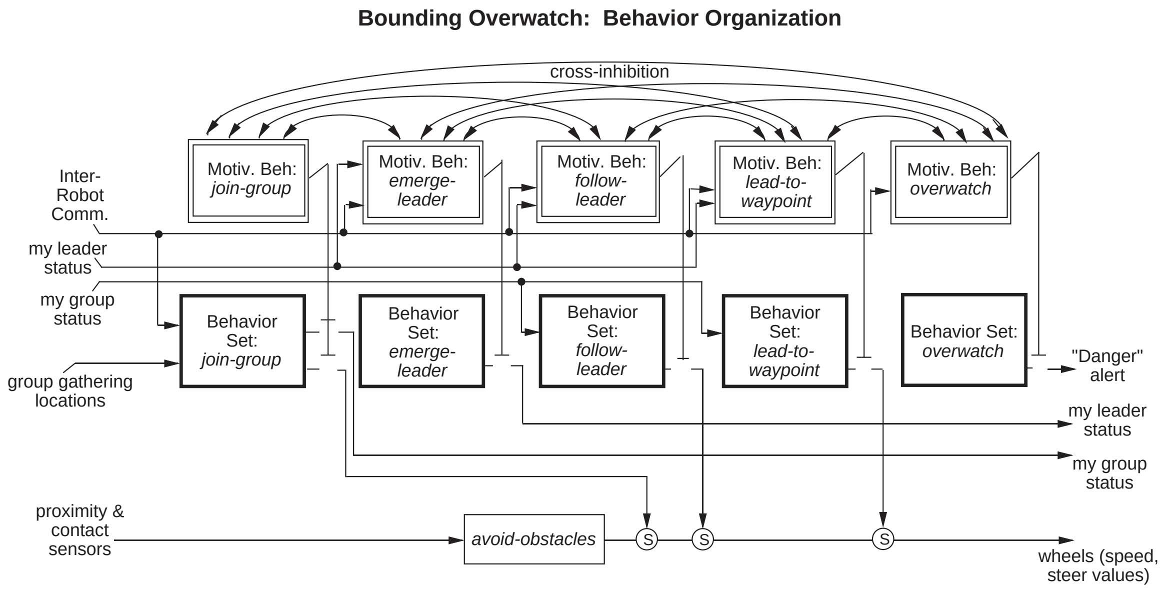
Bounding Overwatch: Behavior Organization
cross-inhibition
Motiv. Beh: join-group
Motiv. Beh: emerge-leader
Motiv. Beh: follow-leader
Motiv. Beh: lead-to-waypoint
Motiv. Beh: overwatch
Inter-Robot Comm.
my leader status
my group status
group gathering locations
Behavior Set: join-group
Behavior Set: emerge-leader
Behavior Set: follow-leader
Behavior Set: lead-to-waypoint
Behavior Set: overwatch
"Danger" alert
my leader status
my group status
proximity & contact sensors
avoid-obstacles
S
S
S
wheels (speed, steer values)

Figure 6-14: The ALLIANCE-based control for the bounding overwatch mission.

The preconditions for the *emerge-leader* behavior set to activate in robot r_i are that (1) r_i has arrived at (more accurately, visited) its group's initial gathering point, and (2) r_i's group does not yet have a group leader. If these conditions are met, then the *emerge-leader* behavior set is activated. The result is that the first robot to arrive at its group's gathering point becomes that group's leader. An interesting side-effect of this definition is that if, at any point in the future, robot r_i's group loses its leader, then r_i will become motivated to emerge as the team's leader. Since many other team members will also have this motivation, the relative rates of impatience across robots will determine which robot actually *does* emerge as the leader. Ideally, the rates of impatience are set such that the robots which make better leaders become motivated more quickly to become a leader. If there is ever a tie in which more than one robot decides to become a leader at the same time, a fixed priority among the robots breaks the tie.

Once all the robots have gathered at their starting locations and the leaders have emerged, the preconditions for the *overwatch* behavior set are satisfied in all of the robots, causing all the robots to begin watching out for some sort of danger, such as the presence of an enemy agent. (In this simulation, however, no sources of danger were modeled.) The precondition for the *lead-to-waypoint* behavior set in a leader robot is that the previous team has just bounded to its next waypoint. In order to initiate the bounding at the beginning of the mission, this condition is hardcoded into the leader of the first team as soon as its team has collected at the initial gathering

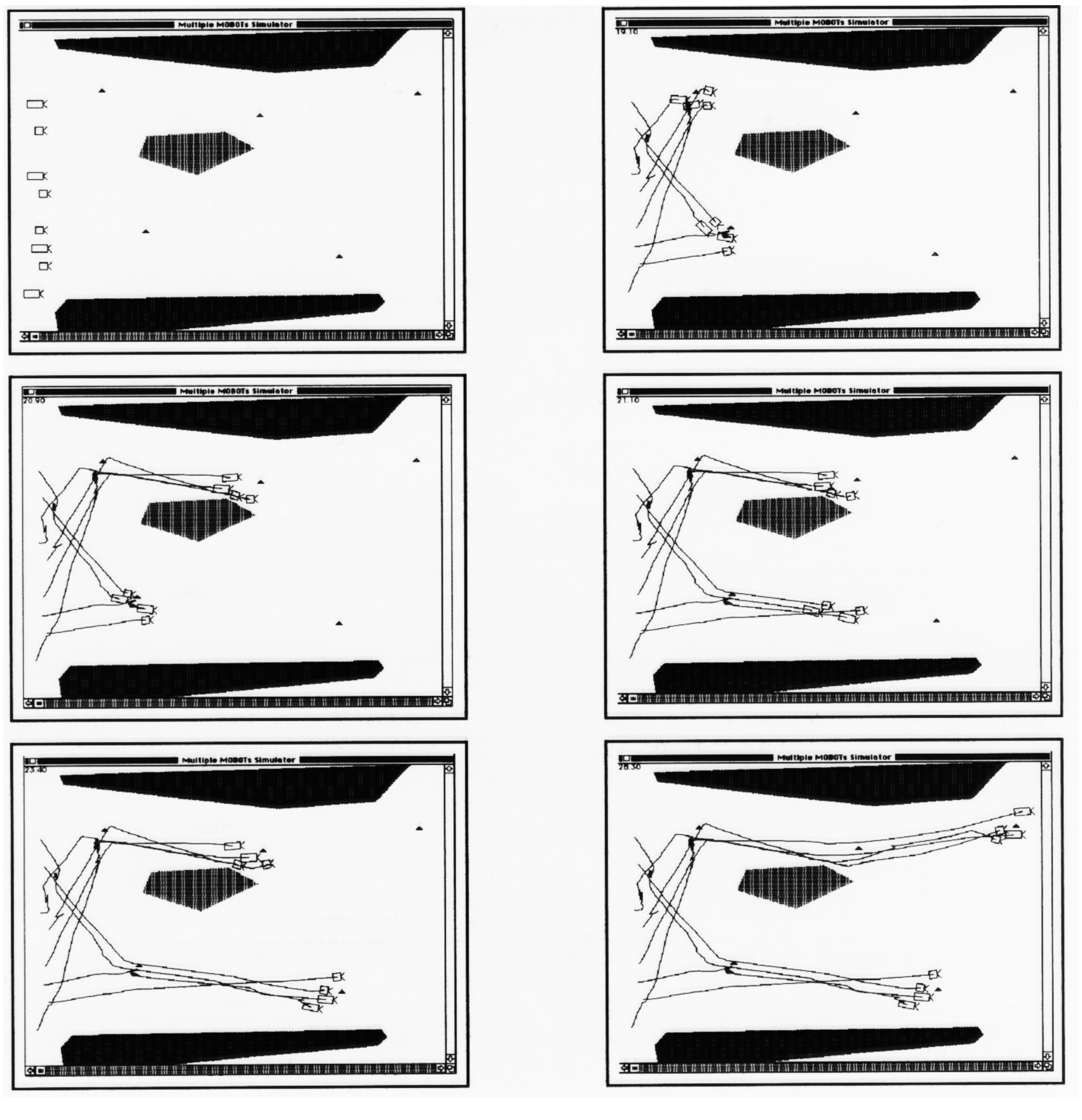

Figure 6-15: A typical run of the bounding overwatch mission (read the snapshots from left to right, top to bottom).

location. Thus, the leader of the first team initiates the bounding to the next waypoint. This, in turn, satisfies the preconditions of the *follow-leader* behavior set in the remaining robots on the first team. The result is that the leader robot leads the team to the next waypoint while the rest of its team follows along. The members of the second team, in the meantime, have activated their *overwatch* behavior sets, and are overwatching the first team's progress. This scenario is shown in the third frame of figure 6-15.

Once the first team's leader has arrived at its next waypoint, the completion of the *lead-to-waypoint* is broadcast. Upon hearing this, the leader of the second team's preconditions for *lead-to-waypoint* are satisfied, causing it to lead its team to its next waypoint while the first team overwatches. This continues, as shown in figure 6-15, until the teams reach some prespecified destination.

This illustration describes the basic idea behind the fault tolerant execution of the bounding overwatch mission. One can imagine much more complex versions of this mission that involve numerous roles (such as clearing paths, monitoring the rear of the group, searching for good waypoints, and so forth) that must be carried out by various team members. These roles can be easily and dynamically reallocated among team members with the ALLIANCE architecture when needed due to the failure of robot team members or due to increased requirements of the mission (perhaps due to an attack of enemy forces) in the same way as the leader role is dynamically reallocated in this example.

6.4 Summary

In this chapter, three additional proof of concept implementations of the ALLIANCE architecture have been presented: the box pushing demonstration on physical robots, and the janitorial service and bounding overwatch missions on simulated robots. The box pushing demonstration offers a short, easily understood example of key characteristics of the ALLIANCE architecture — fault tolerant control amidst the failure of robot team members, and adaptive control due to changing capabilities of the robot team. The janitorial service application offers an example of a mission involving numerous independent tasks that must be carried out, and the ability of ALLIANCE to allow robot team members to select their actions to eliminate duplication of effort. It also illustrates an example in which the L-ALLIANCE architecture can be of particular benefit in improving the efficiency of the team performance. Finally, the bounding overwatch mission offers an illustration of how several tasks with fixed ordering constraints can be solved using ALLIANCE. This architecture offers an easy way to achieve dynamic role transferral in missions involving changes in the environment or in robot team. These applications, along with the hazardous waste cleanup mission described in chapter 3, illustrate the wide variety of applications for which the ALLIANCE architecture is suited.

Chapter 7

Designing Control Laws

As we have seen, the ALLIANCE and L-ALLIANCE architectures allow robot teams to accomplish missions of loosely coupled, largely independent subtasks with a significant degree of coherence. However, the extent of coherence attainable by these teams has been shown to be dependent upon the knowledge individual robots possess concerning the current actions and previous performance of their teammates. This knowledge can actually be viewed as partial global information about the current state and intentions of the robot team. The more limited this global knowledge becomes, the more each robot depends upon its own local knowledge for action selection, which may in turn decrease the coherence of the team. However, due to the design of ALLIANCE and L-ALLIANCE, the use of the global knowledge is fortunately not detrimental to the processing requirements of the individual robots. Thus, the use of global knowledge can be incorporated into ALLIANCE and L-ALLIANCE without impacting the team performance.

It is interesting to step back for a moment and consider whether this principle of "increased global knowledge implies increased coherence" holds for a different type of cooperative robot mission — namely, one requiring spatial coordination among robot team members. It is appealing to be able to develop control laws that utilize strictly local information such that the desired group coordination emerges from the interaction of the local control laws. Indeed, research has shown that certain types of spatial coordination missions can be achieved using local control knowledge alone [Deneubourg *et al.*, 1992, Drogoul and Ferber, 1992, Franklin and Harmon, 1987, Goss and Deneubourg, 1992, Kube and Zhang, 1992, Miller, 1990, Steels, 1990, Stilwell and Bay, 1993, Theraulaz *et al.*, 1990]. However, the question that remains is determining the degree to which group coordination and coherence can be achieved for a given application with purely local control knowledge, and when more global knowledge is needed to obtain the desired results. While I do not attempt to answer this question thoroughly here, I do describe the results of one case study — keeping formation —

which, at first glance, appears to be an application that can be solved using local control alone. However, upon further investigation, we discover that local control rules are not sufficient to obtain the desired level of performance for the mission.

The following sections first distinguish between the notions of global control and local control and then examine the tradeoffs between the two types of control laws. I present the "Keeping Formation" case study which stimulated my thoughts on the local versus global control issues, discussing the design and implementation of several alternative control strategies and the results. This chapter concludes with a summary of the general principles and guidelines derived through this case study. (See [Parker, 1993a] for a related discussion of this issue.)

7.1 Descriptions of Global and Local Control

In practice a continuum exists between strictly global and strictly local control laws. Thus, the control laws guiding a robot will probably use a mixture of local and global knowledge, rather than adhering strictly to one type alone. To simplify the discussion, however, these types are considered separately in this section, which compares and contrasts these two types of control.

7.1.1 Global Control

Global control laws utilize the global goals of the cooperative team and/or global knowledge about the team's current or upcoming actions to direct an individual robot's actions. With these laws, a robot is able to influence its own actions toward team-level goals that cannot be sensed in its own local world. To better understand the implications of the use of global control laws, let us look individually at the two types of information utilized by these laws: *global goals* and *global knowledge*. The *global goals* of a team indicate the overall mission that the team is required to accomplish. These goals are typically imposed upon the team by a centralized controller, such as a human or another autonomous robot. Often this controller is a robot from outside the cooperative team rather than from within, although it is not uncommon to have a leading robot within the team specifying these goals.

Of particular impact on the design of cooperative teams is the time at which the global goals become known[Payton, 1991]. If the goals are known and fixed at design-time, then it may be possible to incorporate these goals implicitly into the control laws of each robot. Whether this can be done depends on the proper match between the sensing capabilities of the robots and the sensing requirements of the global goals. If all the information required for a robot to act consistently with the global goals can be sensed locally by that robot at run-time, then the global goals can be designed into the robot. On the other hand, if the goals are not fixed or known at design-time, then they obviously cannot be designed into the robots. In this case, the robots must possess the capability to obtain and appropriately act upon the goals provided at run-time.

The second type of information used by global control laws, *global knowledge*, refers to the additional information that may be necessary for the cooperative team to achieve the global goals. This information typically indicates what other robots in the team are doing or are going to do, or what the environment looks like in relation to the current cooperative task. By definition, all such information is normally not available to the individual robots through their sensors (other than their communication channels); if it were, then I would consider it to be local information.

How does a robot obtain this global knowledge? Several methods are possible. Perhaps the most obvious manner is for a centralized informant (either a human or an autonomous robot either inside or outside of the robot team) to explicitly communicate the information directly to the team as it becomes available. The robots can then utilize this explicitly communicated information as advice, along with locally sensed data, to undertake appropriate actions which are consistent with the global goals. A second method of obtaining global knowledge, albeit in an approximate form, is for robots to passively observe and interpret the actions of another robot as described in the earlier chapter on action recognition. Combined with some goal recognition, this method would allow a robot not only to interpret a teammate's current actions, but also to predict that robot's future actions. In a sense, this method utilizes implicit communication, since the observing robot receives information from the actions of the observed robot.

The use of global goals and information is not without its shortcomings, however. Adequate global information may not be available to achieve the desired global goal. Even with global knowledge, a robot may still not exhibit optimal global behavior unless it utilizes all of the global knowledge available. Processing this global information requires time and resources, both of which are usually limited in real-world applications. If the global goals or information is changing often enough, the robot may not be able to act upon the global knowledge before it becomes out-of-date. Indeed, in some situations, global control of any kind will be impossible, thus mandating the use of local control.

7.1.2 Local Control

Local control laws, on the other hand, guide a robot's actions based on the proximate environment of that robot. Such information is derived from the robot's sensory capabilities, and thus reflects the state of the world near the robot. Local control laws allow robots to react to dynamic changes in their environment without relying on preconceived plans or expectations of the world. As I have noted, careful design of the control laws can allow global functionality to emerge from the interaction of the local control laws of the individual robots. For example, Franklin and Harmon [Franklin and Harmon, 1987] have shown that a global cooperative hunting behavior emerges from the use of three local cooperative control laws: cooperative pursuit, triangulation, and encirclement. These control laws are appealing because of their simplicity and power to generate globally emergent functionality.

However, local control laws also have their limitations — certain global goals cannot be attained through the use of local control laws alone. In some cases, it may be possible to utilize local control laws to achieve an approximation to the optimal results, which may be totally acceptable for many applications. However, since local control relies strictly on features of the environment that can be sensed, those aspects of global goals that have no physical manifestation in the world cannot be acted upon by local control laws.

7.2 Keeping Formation Case Study

Let us now look at the keeping formation case study to see what we can learn about the level of control attainable with various combinations of local and global control. I have implemented and evaluated several control strategies along the local versus global spectrum by performing a wide range of experiments in simulation. For each of the control strategies, I measured the results quantitatively by collecting data on the mission completion time and amount of robot error in performing the mission. This section describes these results, first defining the mission performed by the robots, briefly reviewing the related work in this area, and then discussing the results of experiments with four control strategies that vary in the amount of global and local information.

7.2.1 Task Description

The *keep formation* task requires a group of robots to stay in formation with one another (i.e. remain aligned side to side) while the leader of the group follows a prespecified route and while all robots avoid obstacles as they appear (see figure 7-1). Each of these robots has the ability to sense the location of its neighboring robots relative to itself (local knowledge).

The global goal of this task is twofold: first, the robots should reach their destination as quickly as possible, and, second, they must maintain the specified formation in a manner that appears to a casual human observer to be human-driven, meaning that the robots should not allow huge or "unnatural" (an admittedly subjective measure)

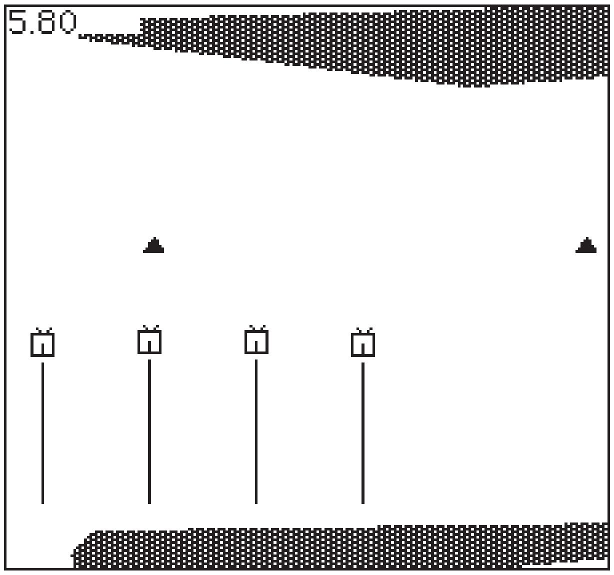

Figure 7-1: Four robots keeping formation side by side.

perturbations in the desired group formation[1]. This subjective measure is quantified by defining the notion of *normalized cumulative formation error*, which is calculated as follows: at a given time t, the formation error, fe_t, is given by

$$fe_t = \sum_{i \neq leader} d_i$$

where d_i is the distance between the current position of robot i and the proper formation position of robot i, based on the leader's current location. The cumulative formation error, cum_fe is then given by:

$$cum\_fe = \sum_{t=0}^{t_{max}} fe_t$$

for integral values of t, meaning that the formation error is sampled and accumulated at discrete points in time up to t_{max}, which is the mission completion time. Since this cumulative formation error is dependent on the total time of mission completion, it is divided by the total mission time to result in the normalized cumulative formation error, which is used as a basis of comparison between the control strategies.

The robots in this mission, designed using a behavior-based approach, are pro-

[1]Of course, I am not requiring that the Turing test be passed by these robots. The point is not to fool humans, but to display human-like strategies toward staying in formation.

vided with competences to avoid obstacles, to follow a specified route, and to keep in formation. In these experiments, I varied the design of the third competence — KEEP_FORMATION — to determine the level of performance we are able to achieve with different levels of local versus global control.

7.2.2 Related Work

Little related work has been done in the area of cooperative robots with the specific aim of determining the tradeoffs between local and global control. Of course, as referenced earlier, many researchers have built systems using local control alone which have been shown to achieve the stated application. However, very rarely does this work attempt to determine the limit of the usefulness of local control laws.

One paper of particular note to this selected keep formation case study, however, is the work done by Wang in [Wang, 1991]. In this paper, Wang considers the problem of a small number of simulated robots remaining in formation. He studies several simple navigation strategies based upon nearest neighbor tracking and develops analytical derivations of the various control strategies. The differences between Wang's studies and those presented in this chapter are twofold: (1) Wang considers continuous motions only, whereas this chapter examines navigation between discrete waypoints, and (2) Wang defines the robot formations in terms of absolute x and y distances only, whereas this chapter also includes an orientation constraint (that is, in the version of the problem presented in this chapter, a robot must maintain the same neighbors to its own left and right, respectively; this constraint is not required by Wang). The consequences of these differences are as follows: (1) continuous motions imply that a robot's current motion trajectory completely defines where the robot will be at the next instant, whereas a robot moving between discrete waypoints may change course abruptly, and (2) lack of orientation constraints implies that similar velocity profiles across robots are possible, whereas orientation constraints may require some robots to accelerate or decelerate relative to other robots in order to maintain formation. Determining which method of defining motions and formation constraints is preferable depends upon the requirements of a given application.

7.2.3 Implementation

The simulation data and snapshots illustrated in this section were obtained using the simulator described in chapter 2. The experiments varied in the route the robots were instructed to follow, the character of the route (i.e., sharp versus smooth turns, following a road or traveling through open terrain, etc.), the number of robots in the team, the formation the robots were to maintain, and the presence of static or dynamic obstacles in the paths of the robots. Typical experiments involved from

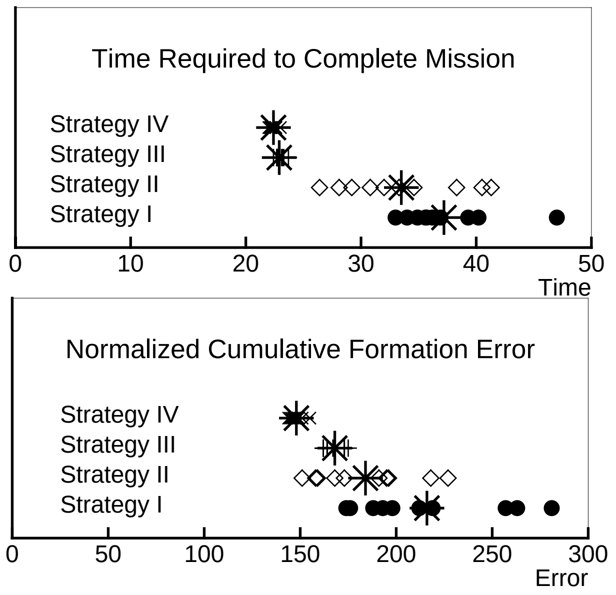

Figure 7-2: Time and error results of 10 runs of each control strategy. (Asterisks denote the mean values.)

1 to 14 robots instructed to follow a specified route while staying in a side-by-side formation. Often, an additional team of robots simultaneously performed a similar task along an intersecting route, requiring the robots in both teams to avoid dynamic obstacles (other robots) as they maintained their formation.

Each of the control strategies described below was implemented and tested separately to determine the group behavior that resulted from each of the strategies. These strategies were evaluated based on the quantitative measures of mission completion time and normalized cumulative formation error described earlier. To collect this data, each experiment for each control strategy was run ten times. Figure 7-2 plots the results, which are discussed in the next subsections[2].

7.2.4 Strategy I: Using local control alone

At first glance, it would appear that KEEP_FORMATION could be achieved using local control laws alone. Each robot could be assigned a leader and then use a simple control law that directs it toward a prespecified offset and direction from its leading robot. As the group leader moves forward along the path (which is known only to the group leader), the other robots follow along to stay in formation. Indeed, in

[2]The variation in results for control strategies I and II is due to unpredictable interference among robots when they stray significantly out of formation.

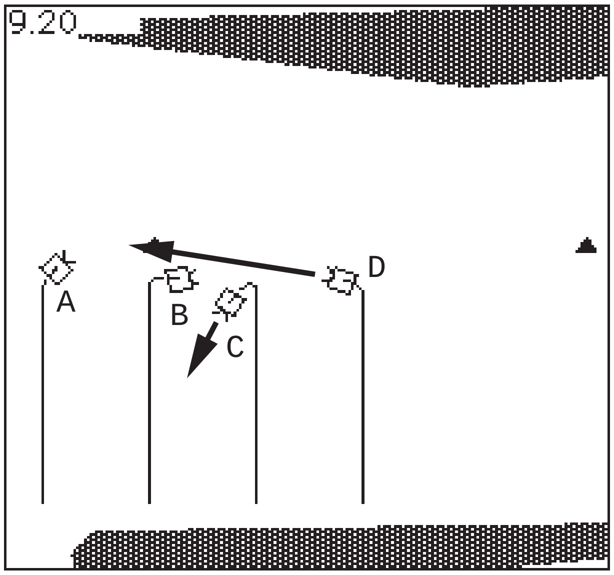

Figure 7-3: Team behavior using Strategy I.

experiments involving relatively few robots traversing smooth routes in the absence of obstacles, I found that this law would perform adequately well. However, a problem arises if the group leader makes a sharp turn along the path, as illustrated in figure 7-3. (In figures 7-3 through 7-6, the bold arrows, when present, indicate the intended direction of travel of the robots, the thin lines show the paths already traversed by the robots, and the leader's path goes from its starting location to the small triangle directly in front of it, and then to the small triangle on the right.) In this snapshot of the simulation, robot B is the overall leader, robots A and C are following robot B, and robot D is following robot C. In following its leader, robot A seeks to always locate itself a preset distance to the left of B, while robots C and D strive to be located the same distance to the right of their respective leaders. In this figure, the group leader, B, is making a right-hand turn. Since the followers are using strictly local information in this case, they continue to follow the same rules as before, maintaining a specified distance and offset from their respective leaders. Robot A performs satisfactorily, aiming toward the location the appropriate distance to the left of B. However, robot C finds itself well out of formation, and thus it turns around and aims toward a location to the right of B. Now, however, we have a problem with robot D. It aims as usual toward the right of C, but this position is out of formation with the rest of the group. Here we see that local control information is not sufficient to achieve the desired global goals. Figure 7-2 shows that this strategy resulted in the worst quantitative performance of all the control strategies studied.

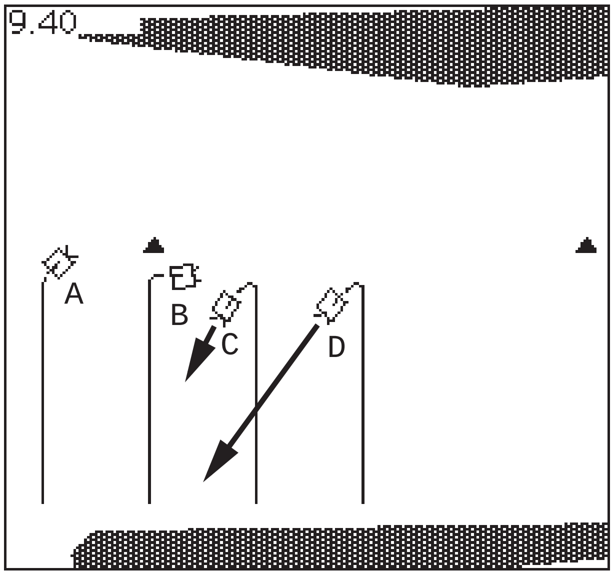

Figure 7-4: Team behavior using Strategy II.

7.2.5 Strategy II: Using local control augmented by a global goal

An improvement on the situation provides the robots with knowledge of the global goal of the group. Now, since the robots are aware that they should achieve a global linear formation, they select their positions after robot B's right-hand turn based on the global formation, while still remaining responsive to the local dynamics of the robots adjacent to them. With this information, robots A and C aim toward the same positions as in the previous case, but robot D now heads toward a more globally appropriate location, as shown in figure 7-4. Unfortunately, these movements could still be inappropriate if the leader is just avoiding an obstacle, rather than making a turn along the path. In spite of this, it is clear that knowledge and use of the global goal can yield improved group coordination. In figure 7-2 we see that this strategy resulted in an average 10% reduction in mission completion time and an average 15% reduction in normalized formation error.

7.2.6 Strategy III: Using local control augmented by a global goal and partial global information

Yet another improvement can be attained by providing the team with partial global knowledge about the path the group is to take. In the previous two cases, the right-hand turn by robot B prompted the other robots to change their alignments. However,

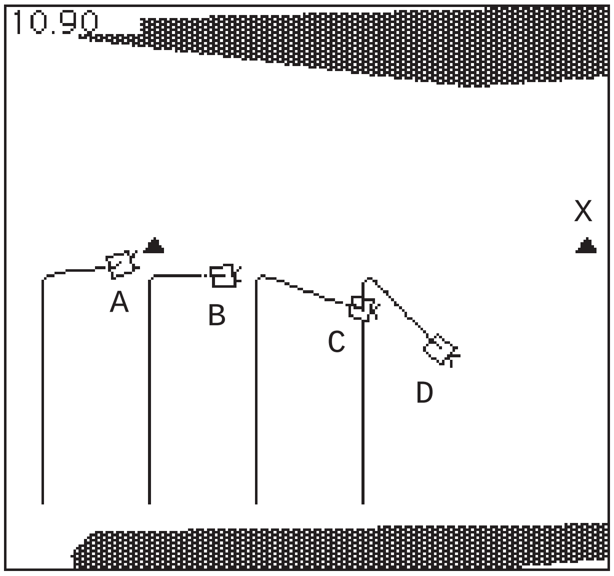

Figure 7-5: Team behavior using Strategy III.

B could have just been avoiding an obstacle, and thus the other robots should have continued along their present path without realignments. Without knowing anything about the route that the leader is following, the robots cannot always react properly to B's actions. Now, however, at the time of robot B's right-hand turn, let us assume that all the robots are told that the group should be headed toward waypoint X, as shown in figure 7-5. With this partial global information, robots C and D can avoid the needless backtracking present in the previous case, and instead aim forward along the route toward the upcoming waypoint, as shown in figure 7-5, moderating their speeds as required to remain in alignment with their neighbors. In this manner, the robots achieve a much more more efficient cooperation, in which we attain average improvements of 38% in time and 22% in error over local control alone, and 32% and 9% average time and error improvements, respectively, over strategy II.

7.2.7 Strategy IV: Using local control augmented by a global goal and more complete global information

Yet another improvement can be achieved with the use of additional global information. Global knowledge of the route the group leader is tracking allows the robot followers to accurately predict future actions of the team members. In this example, knowledge of the global path being followed allows the robots to anticipate the right-hand turn, thus enabling the robots to the right of the leader to stop earlier in preparation for this turn, as shown in figure 7-6. With such predictions, each robot

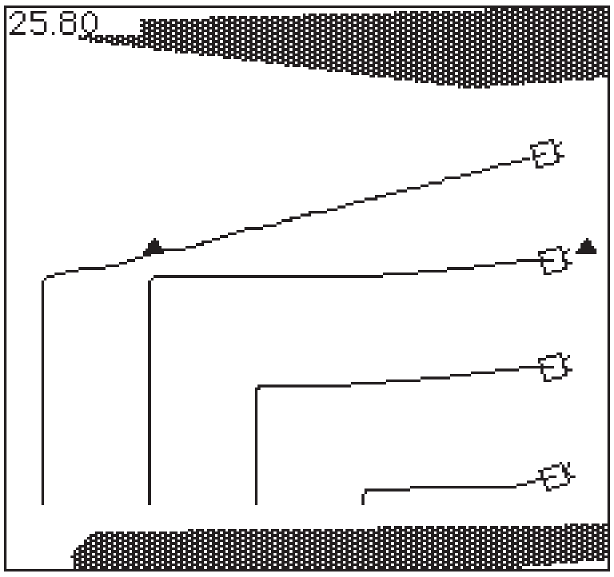

Figure 7-6: Team behavior using Strategy IV.

can modify its actions to better maintain the formation. Using this strategy, we find an additional average error improvement of 12% over strategy III, which is an overall average improvement of 32% in normalized cumulative formation error over local control alone. However, we see little improvement in the mission completion time over strategy III, which is due to the fact that the robots making an error in formation in strategy III have time to correct their errors before the leader reaches the goal, thus not impacting the overall mission completion time.

7.3 The Proper Balance

Having examined the results of this case study, let us generalize these results, to the extent possible, to derive some general principles for the design of cooperative control laws. Selecting the proper balance between the use of local and global control laws is not an easy task, and varies from application to application. Of central importance is determining the acceptable level of cooperation and performance of the autonomous robot team in a particular application. Some applications may be considered successfully accomplished if the team finishes the task at all, regardless of how they do it or how long it takes. Several questions arise when considering the design of cooperative control laws. What are the tradeoffs between global versus local control? Will global and local information conflict, and, if so, how does one arbitrate between them? These issues and others are discussed in the following subsections.

7.3.1 Tradeoffs Between Global and Local Control

Assuming the availability of global goals and/or global knowledge which can be used by the cooperative team, the designer must decide whether to incorporate the use of this global information into the team, or to approach the problem with more local control. In doing this, the designer must weigh the costs of using global information with those of doing without. Several questions must be addressed. First, how static is the global knowledge? The knowledge could be known and fixed at the start of the task, thus making it an excellent candidate for use in a global control law. In general, the more static the global knowledge is, the more practical its use by a global control law.

An additional issue concerns how difficult it is to approximate global knowledge by comparing observations of a robot's actions with a model of that robot's behavior. This type of approximation can be quite challenging, depending upon the complexity of the autonomous robots and the environment. When possible, behavioral observation is more robust and dynamic than the use of global knowledge that may change unexpectedly. As global knowledge becomes more unreliable, a robot team must increase its dependence on behavioral observation and interpretation. Good results with behavior observation and interpretation should be expected particularly for teams of robots possessing a fixed set of discernible actions. One of the primary difficulties with behavior observation, however, lies in the limited ability of robots to sense the current actions of other robots. In cases where the sensing capabilities are not sufficiently extensive, the team can utilize communication to inform other robots of their current actions.

Other issues that must be addressed include: How badly will the performance degrade without the use of global knowledge? How difficult is it to use global knowledge? How costly is it to violate the global goals? How accessible is the global knowledge? How much local information can be sensed? Answers to these questions must be application-dependent, and considered in light of the capabilities of the specific robots to be used, the environment they will be operating in, and the scope of the application. In general, the more unknown the global information is, the more dependence a team must have on local control, perhaps combined with approximations to global knowledge based on behavioral and environmental observation and interpretation.

7.3.2 Conflicts Between Global and Local Control Information

A combination of local and global control in the same robot may lead to conflicts if the control laws are designed to compete with one another by having the global control laws utilize strictly global information, while the local control laws utilize strictly local information. A better way to design the system is to view the global information as providing general guidance for the longer-term actions of a robot, whereas the local information indicates the more short-term, reactive actions the robot should take within the scope of the longer-term goals. This can often be achieved by combining the use of local and global information into a composite control law that more intelligently interprets the local information in the context of the global knowledge.

Problems may still arise if a robot using global knowledge is also trying to react appropriately to a robot that is not using global knowledge. In this case, the designer must provide the robots with the ability to arbitrate between certain aspects of global or local information when the need arises. Perhaps the best way to achieve the interaction of the two types of knowledge is by using local control information to ground global knowledge in the current situation. In this manner, the robots are able to remain focused on the overall goal of their group while reacting to the dynamics of their present contexts.

7.4 Designing Control Laws: Summary and Conclusions

The design of the control laws governing the behavior of individual robots is crucial for the successful development of cooperative robot teams. These control laws may utilize a combination of local and/or global knowledge to achieve the resulting group behavior. A key difficulty in this development is deciding the proper balance between local and global control to achieve the desired emergent group behavior. This chapter has addressed this issue by presenting some general guidelines and principles for determining the appropriate level of global versus local control, developed from quantitative studies of the keep formation case study. To summarize, the basic general principles and guidelines proposed in this chapter are as follows:

- Global goals: If the global goals are known at design-time and all the information required for a robot to act consistently with the global goals can be sensed locally by the robot at run-time, these goals can be designed into the robots.

- Global knowledge: The more static, reliable, completely known, and easy-to-use the global knowledge is, the more practical its use in a global control law. The more unknown the global information, the more dependence the team will have on local control, perhaps combined action recognition to approximate global knowledge.

- Action recognition: Action recognition may provide a suitable approximation to global knowledge, and can thus be utilized to improve group cooperation. This method should be particularly useful when the robots possess a fixed set of discernible or communicable actions.

- Local knowledge: In many applications, particularly those in which *accomplishing* the task is more important than *how* the robots accomplish the task, local control may provide a suitable approximation to the optimal group behavior, thus eliminating the need for the use of global knowledge.

- Proper balance: Global knowledge should be used to provide general guidance for the longer-term actions of a robot, whereas local knowledge indicates the more short-term, reactive actions the robot should perform within the scope of the longer-term goals. This leads to the following basic principle:

  > *Local control information should be used to ground global knowledge in the current situation. This allows the robots to remain focused on the overall goals of their group while reacting to the dynamics of their current situations.*

Chapter 8

Related Cooperative Mobile Robot Work

In recent years, interest in cooperative mobile robot control has grown significantly. Most major artificial intelligence and robotics conferences today have several sessions dealing with multi-robot systems or multi-agent systems. In this chapter, I review this work and relate it to the research I present in this report. As I do this, it is quite interesting to note the analogy between two major directions in current cooperative mobile robot research and a classification of animal societies proposed by Niko Tinbergen in the 1950's. Thus, I first describe this analogy, and then review the existing work in each of two broad classes of cooperative mobile robotics research.

8.1 Analogy: Animal Societies vs. Cooperative Robotics

The behavioristic approach to autonomous robot control that has gained popularity in recent years has its roots in the observations of animal behavior. Animals, particularly the lower animals, are existence proofs that interesting results can be achieved without the need for a complex, human-level architecture. Many animals appear to be "hard-wired" for certain behaviors, producing very stereotypical reactions to particular stimuli. For instance, a robin begins defending its territory when it sees the red breast of another robin, or even a bunch of red feathers [Etkin, 1964]. A pregnant Three-spined Stickleback fish approaches a male Stickleback with a red belly, or even a crude model of a Stickleback, as long as it is painted red underneath [Tinbergen, 1953]. A male grayling butterfly flies up to mate rather large, dark, close, dancing objects, which could include not only female graylings, but also birds, falling leaves, and shadows [Tinbergen, 1965].

Applying animal observations to the realm of autonomous robotics, interesting and seemingly intelligent activities can be obtained by layering behaviors which react to stimuli from the world according to the robot's current internal state [Brooks, 1986]. Rather than decomposing the robot control system based on information processing functions, the behavioristic approach decomposes the control into task achieving behaviors, such as obstacle avoidance, exploration, and map building. The result has been a series of autonomous robots that can survive in a dynamic world, avoiding obstacles, exploring the environment, following walls, building maps, climbing over uneven terrain, and so forth [Brooks, 1990a].

But this same approach — the observation of animal behavior — that has been used for inspiration in the development of individual robots is just as easily used to gain insight into the creation of groups of robots that cooperate toward attaining some goal. By learning how various species of animals function as groups, we can develop ideas for building a cooperating team of autonomous mobile robots.

8.1.1 Broad Classification of Animal Societies

Since there are so many varieties of social behavior in the animal kingdom, a classification of animal societies is useful. One such classification, proposed by Tinbergen [Tinbergen, 1953], is of particular interest for current robotics research in cooperative systems, as it parallels two possible approaches to cooperating mobile robot development. According to Tinbergen, animal societies can be grouped into two broad categories: those that differentiate, and those that integrate.

Societies that differentiate are realized in a dramatic way in the social insect colonies [Wilson, 1971]. These colonies arise due to an innate differentiation of blood relatives that creates a strict division of work and a system of social interactions among the members. Members are formed within the group according to the needs of the society. In this case, the individual exists for the good of the society, and is totally dependent upon the society for its existence. As a group, accomplishments are made that are impossible to achieve except as a whole.

On the other hand, societies that integrate depend upon the attraction of individual, independent animals to each other. Such groups do not consist of blood relatives that "stay together", but instead consist of individuals of the same species that "come together" by integrating ways of behavior [Portmann, 1961]. These individuals are driven by a selfish motivation which leads them to seek group life because it is in their own best interests. Interesting examples of this type of society are wolves and the breeding colonies of many species of birds, in which hundreds or even thousands of birds congregate to find nesting partners. Such birds do not come together due to any blood relationship; instead, the individuals forming this type of society thrive on the support provided by the group. Rather than the individual existing for the good of the society, we find that the society exists for the good of the individual.

8.1.2 Parallels in Cooperative Robotics

In analyzing the research underway in cooperative autonomous mobile robots, a parallel can be drawn with the classifications of animal societies discussed above. A large body of work in robotics involves the study of emergent cooperation in colonies, or swarms, of robots — an approach comparable to differentiating animal societies. This research emphasizes the use of large numbers of identical robots that individually have very little capability, but when combined with others can generate seemingly intelligent cooperative behavior. Cooperation is achieved as a side-effect of the individual robot behaviors.

A second approach parallels the integrative societies in the animal kingdom. This research aims to achieve higher-level, "intentional"[1] cooperation amongst robots. Rather than beginning with robots having very low-level behaviors, individual robots that have a higher degree of "intelligence" and capabilities are combined to achieve purposeful cooperation. The goal is to use robots that can accomplish meaningful tasks individually, and yet can be combined with other robots with additional skills to complement one another in solving tasks that no single robot can perform alone. To be purely analogous to the integrative animal societies, robots in this type of cooperation would have individual, selfish, motivations which lead them to seek cooperation [McFarland, 1991]. Such cooperation would be sought because it is in the best interests of each robot to do so to achieve its mission. Of course, the possession of a selfish motivation to cooperate does not necessarily imply consciousness on the part of the robot. It is doubtful that we would attribute consciousness to all the integrative societies in the animal kingdom; thus, some mechanism must exist for achieving this cooperation without the need for higher-level cognition.

The type of approach one should use for the cooperative robot solution is dependent upon the applications envisioned for the robot team. The differentiating cooperation approach is useful for tasks requiring numerous repetitions of the same activity over a relatively large area (relative to the robot size), such as waxing a floor, agricultural harvesting, cleaning barnacles off of ships, collecting rock samples on a distant planet, and so forth. Such applications would require the availability of an appropriate number of robots to effectively cover the work area while continuing to maintain the critical distance separation.

[1]I place the term *intentional* in quotes because I do not mean to imply that these robots have the power to choose to cooperate. Rather, this term reflects the philosophy of the human designer of the robots, who builds the control architecture so that robots explicitly communicate and/or coordinate actions with their teammates.

On the other hand, the intentional cooperation approach would be required in applications requiring several distinct tasks to be performed, perhaps in synchrony with one another. Throwing more robots at such problems would be useless, since the individual tasks to be performed cannot be broken into smaller, independent subtasks. Examples of this type of application include automated manufacturing, industrial/household maintenance, search and rescue, and security, surveillance, or reconnaissance tasks,

Of course, there is overlap in the relevance of these approaches to various applications, and in some instances the differences are a matter of degree. For instance, if large numbers of robots are too expensive or are not available to be applied to, say, planet exploration, then more purposive cooperation is required to achieve the goal of the mission. Combinations of the approaches are also possible by using intentionally cooperating robots to guide the activities of smaller groups of swarm robots in a coordinated way.

The research presented in this report addresses the development of autonomous robot teams that parallel the integrative type of social animals. This type of cooperation requires achieving coordinated and coherent solutions to problems involving a few robots, each of which is able to perform meaningful tasks alone, but which requires the presence of other agents to fully complete its mission. The agents operate in dynamic, unstructured environments, and must respond appropriately to their sensory feedback, the actions of other agents, and the priorities of the tasks in the mission, adapting their actions as necessary as these inputs change.

8.2 Approaches to Multi-Robot Cooperative Control

In this section, I review the previous work in the cooperative control of teams of mobile robots, grouping the work into the two broad approaches to mobile robotics introduced in the last section.

8.2.1 "Swarm" Cooperation

The predominant body of research in cooperative mobile robotics deals with the study of large numbers (often called *swarms*) of homogeneous robots. As I noted earlier, this approach to robotic cooperation is useful for non-time-critical applications involving numerous repetitions of the same activity over a relatively large area, such as cleaning a parking lot or collecting rock samples on Mars. The approach to cooperative control taken in these systems is derived from the fields of neurobiology, ethology, psychophysics, and sociology, and involves the interaction of a number of simple rules of control within the individual robot. These behavior-based approaches eschew the use of world models and the distinct separation of the system into perceptual, central control, and actuation systems, and instead rely extensively on interactions with the real world to produce the desired global results. This distributed approach has the advantages of removing the bottleneck present in centralized controllers, reducing team susceptibility to individual robot failures, and increasing the reactivity of the team to a dynamic environment, although at the expense of increased difficulty in maintaining global coherence. The difficult problem addressed in these systems is predicting the global behavior of the collective from the design of the control laws in the individual agent. Thus, a typical methodology used in many of these research projects involves first hypothesizing a possible local control law (or laws) that may allow the collection of robots to solve a given problem, and then studying the resulting group behavior using either simulated or physical mobile robots. Such approaches usually rely on mathematical convergence results (such as the random walk theorem [Chung, 1974]) that indicate the desired outcome over a sufficiently long period of time.

A number of papers describe distributed algorithms for harvesting, collecting, and foraging tasks, as well as other group behaviors often found in social animal colonies such as flocking. In [Deneubourg *et al.*, 1990], Deneubourg *et al.* present simulation studies of several strategies for the collection and transport of objects in which the cooperative behavior emerges by either explicit or implicit communication. In [Deneubourg *et al.*, 1990], Deneubourg *et al.* describe simulation results of a distributed sorting algorithm. Theraulaz *et al.* [Theraulaz *et al.*, 1990] extract cooperative control strategies, including foraging strategies, from a study of Polistes wasp colonies. Steels, in [Steels, 1990], presents simulation studies of the use of a several dynamical systems (partially random movement, a gradient field, and a dissipative structure) to achieve emergent functionality. This work is applied to the problem of collecting rock samples on a distant planet. Drogoul and Ferber [Drogoul and Ferber, 1992] describe simulation studies of foraging and chain-making robots. In [Mataric, 1992a], Mataric describes the results of implementing group behaviors such as dispersion, aggregation, and flocking on a group of up to 20 physical robots. Beni and Wang [Beni and Wang, 1990] describe methods of generating arbitrary patterns in cyclic cellular robotics.

Other swarm cooperation work addresses similar tasks while emphasizing the engineering advantages of this type of cooperation. In [Miller, 1990], Miller conjectures on ways numerous small robots can be applied to tasks in planetary exploration, such as site surveys, instrument deployment, and construction and mining. Brooks *et al.* [Brooks *et al.*, 1990] speculate on control strategies for teams of 20 soil-moving robots deployed to a lunar base. In [Kube and Zhang, 1992], Kube and Zhang present the results of implementing an emergent control strategy on a group of five physical robots performing the task of locating and pushing a brightly lit box. Stilwell and

Bay [Stilwell and Bay, 1993] present a method for controlling a swarm of robots using local force sensors to solve the problem of the collective transport of a palletized load. In [Fukuda *et al.*, 1988] and many related papers, Fukuda and others describe the concept of CEBOT — a collection of robots which dynamically constructs various physical formations based upon the current task.

The primary difference between these approaches and the problem addressed in this report is that the above approaches are designed strictly for homogeneous robot teams, in which each robot has the same capabilities. Additionally, issues of efficiency are largely ignored. However, in heterogeneous robot teams, not all tasks can be performed by all team members, and even if more than one robot can perform a given task, they may perform that task quite differently. Thus the proper mapping of subtasks to robots is dependent upon the capabilities and performance of each robot team member. This additional constraint brings many complications to a workable architecture for robot cooperation, and must be addressed explicitly to achieve the desirable level of cooperation.

8.2.2 "Intentional" Cooperation

Although the swarm cooperation approach is useful for many types of real-world tasks, many other real-world tasks require a more directed type of cooperation, perhaps due to time or efficiency constraints that are placed on the mission. Furthermore, this second type of mobile robotic mission usually requires that several distinct tasks be performed. These missions thus usually require a much smaller number of possibly heterogeneous mobile robots involved in more purposeful cooperation. Key issues in these systems include robustly determining which robot should perform which task so as to maximize the efficiency of the team and ensuring the proper coordination among team members to allow them to successfully complete their mission.

Two bodies of previous research are particularly applicable to this type of cooperation. First, several researchers have directly addressed this cooperative robot problem by developing control algorithms and implementing them on physical robots, or at least on simulations of physical robots that make reasonable assumptions about the capabilities of real mobile robots. The second, significantly larger, body of research comes from the Distributed Artificial Intelligence (DAI) community, which has produced a great deal of work addressing this type of "intentional" cooperation among more generic *agents*. These agents are typically software systems running as separate processes; in some models, these agents share memory resources, while in others, each agent has only local memory.

In the following two subsections, I review the work from these two communities as applied to "intentional" cooperation between mobile robots.

Mobile Robot Research

Nearly all of the existing work on heterogeneous physical robots uses a traditional artificial intelligence approach, which breaks the robot controller into modules for sensing, world modeling, planning, and acting (hence, the *sense-model-plan-act* paradigm), rather than the functional decomposition of behavior-based approaches. Noreils [Noreils, 1993] describes one such *sense-model-plan-act* control architecture which includes three layers of control: the planner level, which manages coordinated protocols, decomposes tasks into smaller subunits, and assigns the subtasks to a network of robots; the control level, which organizes and executes a robot's tasks; and the functional level, which provides controlled reactivity. He reports on the implementation of this architecture on two physical mobile robots performing convoying and box pushing. In both of these examples, one of the robots acts as a leader, and the other acts as a follower.

Caloud *et al.* [Caloud *et al.*, 1990] describe another *sense-model-plan-act* architecture which includes a task planner, a task allocator, a motion planner, and an execution monitor. Each robot obtains goals to achieve either based on its own current situation, or via a request by another team member. They use Petri Nets for interpretation of the plan decomposition and execution monitoring. In this paper they report on plans to implement their architecture on three physical robots.

In [Asama *et al.*, 1992] and elsewhere, Asama *et al.* describe their decentralized robot system called ACTRESS, addressing the issues of communication, task assignment, and path planning among heterogeneous robotic agents. Their approach revolves primarily around a negotiation framework which allows robots to recruit help when needed. They have demonstrated their architecture on mobile robots performing a box pushing task.

Wang [Wang, 1993] addresses a similar issue to that addressed in this report — namely, dynamic, distributed task allocation when more than one robot can perform a given task. He proposes the use of several distributed mutual exclusion algorithms that use a "sign-board" for inter-robot communication. These algorithms are used to solve problems including distributed leader finding, the N-way intersection problem, and robot ordering. However, this paper does not address issues of dynamic reallocation due to robot failure and efficiency issues due to robot heterogeneity.

Cohen *et al.* [Cohen *et al.*, 1990a] propose a hierarchical subdivision of authority to address the problem of cooperative fire-fighting. They describe their Phoenix system, which includes a generic simulation environment and a real-time, adaptive planner. The main controller in this architecture is called the Fireboss, which maintains a global view of the environment, forms global plans, and sends instructions to agents to activate their own local planning.

Ohko *et al.* [Ohko *et al.*, 1993] describe a learning system, called LEMMING, which learns knowledge quite similar to that learned in L-ALLIANCE. In their system, however, this knowledge is used by a case-based reasoner for reducing the communication flow between distributed agents. These distributed agents use the Contract Net Protocol [Smith, 1980] to negotiate the allocation of tasks. With LEMMING, the agents can often use point-to-point communication rather than broadcast communication to recruit help directly from those agents known to have the capabilities to perform a given task, thus reducing the overall communication traffic. They present results from a simulation application involving the movement of objects from one location to another by a team of distributed agents.

However, although the need for fault tolerance is noted in these architectures, they typically either make no serious effort at achieving fault tolerant, adaptive control or they assume the presence of unrealistic "black boxes" that continually monitor the environment and provide recovery strategies (usually involving unspecified replanning mechanisms) for handling various types of unexpected events. Thus, in actuality, if one or more of the robots or the communication system fails under these approaches, the entire team is subject to catastrophic failure. Experience with physical mobile robots has shown, however, that robot failure is very common, not only due to the complexity of the robots themselves, but also due to the complexity of the environment in which these robots must be able to operate. Thus, control architectures must explicitly address the dynamic nature of the cooperative team and its environment to be truly useful in real-world applications. Indeed, the approach to cooperative control developed in this report has been designed specifically with the view toward achieving fault tolerance and adaptivity.

Additionally, as I have noted, these existing approaches break the problem into a traditional AI *sense-model-plan-act* decomposition rather than the functional decomposition used in behavior-based approaches. The traditional approach has likely been favored because it presents a clean subdivision between the job planning, task decomposition, and task allocation portions of the mission to be accomplished — a segmentation that may, at first, appear to simplify the cooperative team design. However, the problems with applying these traditional approaches to physical robot teams are the same problems that currently plague these approaches when they are applied to individual situated robots. As argued by Brooks in [Brooks, 1991b] and elsewhere, approaches using a *sense-model-plan-act* framework have been unable to deliver real-time performance in a dynamic world because of their failure to adequately address the situatedness and embodiment of physical robots. Thus, a behavior-based approach to cooperation was utilized in ALLIANCE to increase the robustness and adaptivity of the cooperative team.

Distributed Artificial Intelligence Research

Much theoretical work has been accomplished for intentional agent control by the Distributed Artificial Intelligence (DAI) community ([Bond and Gasser, 1988] contains many examples). In most of this work, the issue of task allocation has been the driving influence that dictates the design of the architecture for cooperation, since the selected approach to task allocation invariably restricts the potential solutions to other issues of cooperation, such as conflict resolution.

Typically, the DAI approaches use a distributed, negotiation-based mechanism to determine the allocation of tasks to agents. One popular negotiation protocol is the contract-net protocol [Davis and Smith, 1983, Smith and Davis, 1981]; other negotiation schemes are described in [Durfee and Montgomery, 1990, Kreifelts and von Martial, 1990, Rosenschein and Genesereth, 1985, Zlotkin and Rosenschein, 1990]. Under these negotiation schemes, no centralized agent has full control over which tasks individual team members should perform. Instead, many agents know which subtasks are required for various portions of the mission to be performed, along with the skills required to achieve those subtasks. These agents then broadcast a request for bids to perform these subtasks, which other agents may respond to if they are available and want to perform these tasks. The broadcasting agent then selects an agent from those that respond and awards the task to the winning agent, who then goes on to perform that task, recruiting yet other agents to help if required. We saw above that these types of negotiation schemes are also popular for work applied directly to physical mobile robots.

However, although DAI work has demonstrated success in a number of domains (e.g. distributed vehicle monitoring [Lesser and Corkill, 1983] and distributed air traffic control [Cammarata *et al.*, 1983]), the proposed solutions have rarely been demonstrated as directly applicable to *situated* agent (i.e. robotic) teams, which have to live in, and react to, a dynamic and uncertain environment using noisy sensors and effectors, and a limited bandwidth, noisy communication mechanism. They typically rely on unrealistic "black boxes" to provide high-level, perfect sensing and action capabilities. Furthermore, as with the approaches of the previous subsection, these DAI approaches typically ignore or only give brief treatment to the issues of robot performance of those tasks after they have been allocated. Such approaches usually assume the robots will eventually accomplish the task they have been assigned, or that some external monitor will provide information to the robots on dynamic changes in the environment or in robot performance. However, to realistically design a cooperative approach to robotics, we must include mechanisms within the software control of each robot that allow the team members to recover from dynamic changes in their environment or in the robot team. Thus, it is unlikely that the current DAI approaches can successfully address the unique aspects present in situated systems.

Chapter 9

Summary and Conclusions

9.1 Summary of Contributions

This report makes several contributions to methods of fault tolerant, adaptive cooperative control and to our understanding of heterogeneous mobile robot cooperation. Foremost is the development of ALLIANCE — a novel, fault tolerant cooperative architecture for small- to medium-sized heterogeneous mobile robot teams applied to missions involving loosely-coupled, largely independent tasks. This architecture has been shown to have the following characteristics:

- Fully distributed at both the individual robot level and at the team level.
- Applicable to robot teams having any degree of heterogeneity.
- Uses no negotiation or two-way conversations.
- Recovers from failures in individual robots or in the communication system.
- Allows new robots to be added to the team at any time.
- Allows adaptive action selection in dynamic environments.
- Eliminates replication of effort when communication is available.
- Provably terminates for a large class of applications.
- Scales easily to large missions.

The next major contribution is an extension to ALLIANCE, called L-ALLIANCE, that preserves the fault tolerant characteristics of ALLIANCE while adding efficiency mechanisms. These mechanisms were shown to improve robot team performance by incorporating learned knowledge into a dynamic parameter update mechanism. This extension to ALLIANCE results in the following advantages:

- Improves efficiency for cooperative teams applied to missions composed of independent tasks.
- Eliminates the need for human parameter adjustments.
- Allows human designer to custom-design robot teams for specific missions.
- Requires no advance knowledge of the capabilities of team members.
- Allows robot team members to adapt their performance over time to changes in the environment or in the team member capabilities.

ALLIANCE and L-ALLIANCE have been implemented on both simulated and physical robot teams performing a variety of missions. These demonstrations validated the architecture and allowed me to study a number of important issues in cooperative control. The missions to which this architecture has been applied and which were described in this report, are:

- Hazardous waste cleanup mission
- Box pushing demonstration
- Janitorial service mission
- Bounding overwatch mission
- Numerous generic missions

I am not aware of any other cooperative control architecture that has exhibited the combination of fault tolerance, reliability, adaptivity, and efficiency possible with ALLIANCE and L-ALLIANCE, and which has been successfully demonstrated on physical mobile robot teams.

9.2 Fault Tolerant Cooperative Control

A common question I am asked concerning cooperative robotics research is: "Why work on multiple robots when we can't even make one robot work?". Anyone who has worked on physical robots knows the blood, sweat, and tears that are are required to accomplish what, to an outsider, may seem to be trivial. I cannot count the hours I spent trying to get a robot to use infrared sensors with a 12-inch range to perform simple wall-following without losing the wall, getting the robot to find a puck (spill object), pick it up, carry it to some location, and drop it, without dropping the puck too early or squeezing the puck so hard that it breaks the gripper; preventing the robot from initializing its gripper in an "up" position which prevents it from actually reaching the pucks on the floor; or getting the (legged) robot to back away from an obstacle without "marching in place" first. If the IRs were working, then the gear train would break. If the radio was working, then the gripper motors would break. If the gripper motors worked, then the break beam between them would fail. Perhaps the break beam and the gripper motors would work, but the touch sensor between the fingers would break. And, just as you get everything working, the battery supply runs out, so you have to stop for 30 minutes to recharge everything. The robot then loses its program, so you have to redownload it, but the robot's microprocessor refuses to talk to the Macintosh via the serial port. On and on the problems go, not just for our robots, but for nearly all mobile robots in existence today. Multiply these problems across every robot on the team, and you are asking for some major headaches.

Or *are* you? The beauty of the ALLIANCE and L-ALLIANCE architectures is that they allow error-prone robots operating in a dynamic world to overcome their individual failures as a group and succeed at their mission through the efficient use of redundancy. If one robot fails at a task, some other robot is likely to be able to accomplish that task instead (although possibly at a reduced level of efficiency). Additionally, a robot which may have a faulty sensor or a faulty effector preventing it from succeeding at some of its tasks still has the ability, through L-ALLIANCE, of executing those tasks which it is still able to perform. ALLIANCE and L-ALLIANCE also allow "spare" robots to monitor the team from an out-of-the-way location, yet join in easily when made necessary by robot failures or a dynamic environment. Furthermore, the ability of heterogeneous robots to work together allows systems designers to distribute the capabilities required by the mission across a number of robots, thus reducing the complexity of any individual robot and reducing the likelihood of mission failure. Even when all robots are working properly, L-ALLIANCE provides mechanisms allowing the robots to dynamically select their actions so that they efficiently accomplish their mission.

My experiments with the physical robots demonstrated the fault tolerant nature of ALLIANCE and L-ALLIANCE countless times, in both the hazardous waste cleanup mission and in the box pushing demonstration. Although I usually would intentionally inflict a robot with errors (e.g. by covering up its IRs or turning the robot off) to test the architectures, on numerous occasions spontaneous robot failures would occur which caused the robots to dynamically reallocate their tasks to recover from these "unplanned" failures. For example, in the case of the box pushing demonstration, I was occasionally distracted from the robots, only to look back and find that one robot was dutifully carrying out the entire mission on its own in response to a failure of one of the second robot's processor boards which rendered that robot temporarily useless.

This is not to say that I do not expect reliable mobile robots to ever be built. Two recent theses out of the MIT Mobot Laboratory contribute significantly to the goal of building reliable mobile robots. Horswill [Horswill, 1993] built a robot, Polly, which has operated reliably for hundreds of hours, giving tours of the seventh floor of the MIT Artificial Intelligence Laboratory. Ferrell [Ferrell, 1993] built a distributed, fault tolerant system that allows a six-legged robot, Hannibal, to detect and gracefully compensate for failures in its own hardware components. Nevertheless, these robots operate in the relative friendliness of a research laboratory. When we attempt to use physical robots in environments as dynamic as the Chernobyl situation described in chapter 1, the issues of fault tolerant control become compounded not only by the reliability of the individual robot, but also by the dangerous environment in which the robots operate. The robustness issues of cooperative robot teams will therefore not go away once we have built individually fault tolerant robots; these robots must still be able to operate in challenging or dangerous working environments. Thus, even as individual robots become more reliable, fault tolerant cooperative architectures such as ALLIANCE and L-ALLIANCE will continue to contribute to the design of robust, reliable, flexible, and coherent teams of heterogeneous mobile robots.

9.3 Future Work

Of course, ALLIANCE does have its limitations. Chief among these limitations is the restriction of the cooperative teams to missions involving loosely coupled subtasks whose interdependencies at most involve ordering constraints that are known at design time. Although I have demonstrated a number of quite different types of cooperative robot missions that fall into this category, I have not attempted a disciplined, formal description of the types of problems that can and cannot be formulated as ALLIANCE problems. The types of cooperative robot applications that immediately come to mind as difficult for ALLIANCE are cooperative assembly or construction tasks that are typically solved (in traditional AI systems) using planners. Indeed, these types of missions may actually require planning systems. However, most current research in heterogeneous cooperative mobile robotics *which is actually implemented on physical robots* includes the use of general planners to solve fairly straight-forward coordination and cooperation tasks. It is not at all clear that such elaborate planning systems are needed for the majority of applications, or extensions to those applications, in which they are actually used. The problem is that such planning systems use up valuable computation time deciding what a robot should do next, when in fact much simpler approaches could solve the problem and allow the robot to use its computational resources to deal with the dynamic nature of its environment and its teammates. Horswill addresses a similar issue extensively in his thesis [Horswill, 1993], writing that "if the vast majority of actual *instances* of a problem which are encountered by an agent in its daily life are of one or another very simple variety then the agent may thrive by using simple "hacks" when possible and saving its cognitive resources for the truly hard instances." Thus, an interesting area of future work is to determine the extent to which more tightly coupled tasks can be incorporated into ALLIANCE, and determining any extensions necessary to ALLIANCE to allow for more tightly constrained missions. In a related matter, the efficiency improvements I investigated in L-ALLIANCE were made with the assumption of strictly independent tasks with no ordering constraints. An area of future work, then, is to study and develop efficiency improvements in L-ALLIANCE for missions which do involve ordering constraints.

In chapter 4, I studied the performance of the distributed L-ALLIANCE efficiency strategy by varying a number of factors (the robot team size, the mission size, the level of task coverage, the degree of heterogeneity of the team, and the degree to which the Progress When Working condition (see section 3.7.2) holds) and then comparing the results to the optimal solution for those problems in which the optimum could actually be computed. However, I did not attempt to derive analytical results of the best-case, worst-case, and average-case performance of the distributed L-ALLIANCE control strategy. Thus, an additional area of future work is to try to prove performance bounds on the approximation to the optimal solution, especially for applications involving the possibility of robot failures.

A final interesting, and yet very challenging, area of future work is in the area of action recognition. Supplying robots using the ALLIANCE architecture with the ability to passively interpret and evaluate the actions of its teammates would further contribute to the goals of this architecture, and would largely eliminate the need for the broadcast communication mechanism now utilized in ALLIANCE. As noted in chapter 5, this is a very difficult problem, and yet serves a highly important function for achieving fault tolerant, yet efficient, cooperative control.

Appendix A

Formal Model of ALLIANCE

The formal definition of the motivational behaviors — the primary mechanism for action selection in ALLIANCE — is provided below. Refer to chapter 3 for a full explanation of this model. Chapter 4 describes the extension to ALLIANCE, called L-ALLIANCE, that allows the robots to learn the proper settings of the parameters in this model based on experience.

Recall from chapter 3 that the behavior sets possessed by robot r_i in ALLIANCE are given by the set $A_i = \{a_{i1}, a_{i2}, ...\}$. Since different robots may have different ways of performing the same task, we need a way of referring to the task a robot is working on when it activates a behavior set. Thus, I define the set of n functions $\{h_1(a_{1k}), h_2(a_{2k}), ..., h_n(a_{nk})\}$, where $h_i(a_{ik})$ returns the task of the mission that robot r_i is working on when it activates behavior set a_{ik}.

Given:

$$\theta \;=\; \text{Threshold of activity of a behavior set}$$

Five sources of input affect the motivation to perform a particular behavior set. These inputs are defined as:

Sensory feedback:

$$\mathit{sensory\_feedback}_{ij}(t) = \begin{cases} 1 & \text{if the sensory feedback in robot } r_i \text{ at time } t \\ & \quad \text{indicates that behavior set } a_{ij} \text{ is applicable} \\ 0 & \text{otherwise} \end{cases}$$

Inter-robot communication:

ρ_i = Rate of messages per unit time that robot r_i sends concerning its current activity

$$comm\_received(i,k,j,t_1,t_2) = \begin{cases} 1 & \text{if robot } r_i \text{ has received message from robot } r_k \\ & \text{concerning task } h_i(a_{ij}) \text{ in the time span} \\ & (t_1,t_2)\text{, where } t_1 < t_2 \\ 0 & \text{otherwise} \end{cases}$$

τ_i = Maximum time robot r_i allows to pass without hearing from a particular robot before assuming that that robot has ceased to function

Suppression from active behavior sets:

$$activity\_suppression_{ij}(t) = \begin{cases} 0 & \text{if another behavior set } a_{ik} \text{ is active, } k \neq j\text{, on} \\ & \text{robot } r_i \text{ at time } t \\ 1 & \text{otherwise} \end{cases}$$

Robot impatience:

$\phi_{ij}(k,t)$ = Time during which robot r_i is willing to allow robot r_k's communication message to affect the motivation of behavior set a_{ij}

$\delta\_slow_{ij}(k,t)$ = Rate of impatience of robot r_i concerning behavior set a_{ij} after discovering robot r_k performing task $h_i(a_{ij})$

$\delta\_fast_{ij}(t)$ = Rate of impatience of robot r_i concerning behavior set a_{ij} in the absence of other robots performing task $h_i(a_{ij})$

$$impatience_{ij}(t) = \begin{cases} \min_k(\delta\_slow_{ij}(k,t)) & \text{if } (comm\_received(i,k,j,t-\tau_i,t) = 1) \\ & \text{and} \\ & (comm\_received(i,k,j,0,t-\phi_{ij}(k,t)) = 0) \\ \delta\_fast_{ij}(t) & \text{otherwise} \end{cases}$$

$$impatience\_reset_{ij}(t) = \begin{cases} 0 & \text{if } \exists k.((comm\_received(i,k,j,t-\delta t,t) = 1) \text{ and} \\ & (comm\_received(i,k,j,0,t-\delta t) = 0)), \\ & \text{where } \delta t = \text{time since last communication check} \\ 1 & \text{otherwise} \end{cases}$$

Robot acquiescence:

$\psi_{ij}(t)$ = Time robot r_i wants to maintain behavior set a_{ij}'s activity before yielding to another robot

$\lambda_{ij}(t)$ = Time robot r_i wants to maintain behavior set a_{ij}'s activity before giving up to try another behavior set

$$acquiescence_{ij}(t) = \begin{cases} 0 & \text{if [(behavior set } a_{ij} \text{ of robot } r_i \text{ has been active for more} \\ & \quad \text{than } \psi_{ij}(t) \text{ time units at time } t) \text{ and} \\ & \quad (\exists x.comm\_received(i, x, j, t - \tau_i, t) = 1)] \\ & \text{or} \\ & \quad \text{(behavior set } a_{ij} \text{ of robot } r_i \text{ has been active for more} \\ & \quad \text{than } \lambda_{ij}(t) \text{ time units at time } t) \\ 1 & \text{otherwise} \end{cases}$$

Motivation calculation:

The motivation of robot r_i to perform behavior set a_{ij} at time t is calculated as follows:

$$\begin{aligned} m_{ij}(0) &= 0 \\ m_{ij}(t) &= [m_{ij}(t-1) + impatience_{ij}(t)] \\ &\quad \times sensory\_feedback_{ij}(t) \\ &\quad \times activity\_suppression_{ij}(t) \\ &\quad \times impatience\_reset_{ij}(t) \\ &\quad \times acquiescence_{ij}(t) \end{aligned}$$

Initially, the motivation to perform behavior set a_{ij} in robot r_i is set to 0. This motivation then increases at some positive rate $impatience_{ij}(t)$ unless one of four situations occurs: (1) the sensory feedback indicates that the behavior set is no longer needed, (2) another behavior set in r_i activates, (3) some other robot has just taken over task $h_i(a_{ij})$ for the first time, or (4) the robot has decided to acquiesce the task. In any of these four situations, the motivation returns to 0. Otherwise, the motivation grows until it crosses the threshold θ, at which time the behavior set is activated and the robot can be said to have selected an action. Whenever some behavior set a_{ij} is active in robot r_i, r_i broadcasts its current activity to other robots at a rate of ρ_i.

Appendix B

Formal Model of L-ALLIANCE

The formal definition of the L-ALLIANCE cooperative robot architecture, including mechanisms for learning and efficiency considerations, is provided below. Refer to chapter 4 for a full explanation of this model.

Recall from chapter 3 that the behavior sets possessed by robot r_i in ALLIANCE are given by the set $A_i = \{a_{i1}, a_{i2}, ...\}$. Since different robots may have different ways of performing the same task, we need a way of referring to the task a robot is working on when it activates a behavior set. Thus, I define the set of n functions $\{h_1(a_{1k}), h_2(a_{2k}), ..., h_n(a_{nk})\}$, where $h_i(a_{ik})$ returns the task of the mission that robot r_i is working on when it activates behavior set a_{ik}.

Given:

$$\begin{aligned} \theta &= \text{Threshold of activity of a behavior set} \\ \textit{strategy} &= \text{Current impatience/acquiescence update strategy} \end{aligned}$$

Seven sources of input affect the motivation to perform a particular behavior set. These inputs are defined as:

Sensory feedback:

$$\textit{sensory\_feedback}_{ij}(t) = \begin{cases} 1 & \text{if the sensory feedback in robot } r_i \text{ at time } t \\ & \quad \text{indicates that behavior } a_{ij} \text{ is applicable} \\ 0 & \text{otherwise} \end{cases}$$

Inter-robot communication:

ρ_i = Rate of messages per unit time that robot r_i sends concerning its current activity

$$comm\_received(i,k,j,t_1,t_2) = \begin{cases} 1 & \text{if robot } r_i \text{ has received message from robot } r_k \\ & \quad \text{concerning task } h_i(a_{ij}) \text{ in the time span} \\ & \quad (t_1,t_2)\text{, where } t_1 < t_2 \\ 0 & \text{otherwise} \end{cases}$$

τ_i = Maximum time robot r_i allows to pass without hearing from a particular robot before assuming that that robot has ceased to function

$$robots\_present(i,t) = \{k \mid \exists j.(comm\_received(i,k,j,t-\tau_i,t) = 1)\}$$

Suppression from active behavior sets:

$$activity\_suppression_{ij}(t) = \begin{cases} 0 & \text{if another behavior set } a_{ik} \text{ is active, } k \neq j\text{, on} \\ & \quad \text{robot } r_i \text{ at time } t \\ 1 & \text{otherwise} \end{cases}$$

Learned robot influence:

$$learning\_impatience_{ij}(t) = \begin{cases} 0 & \text{if } (\sum_{x \in robots\_present(i,t)} comm\_received(i,x,j,0,t)) \neq 0 \\ 1 & \text{otherwise} \end{cases}$$

μ = Number of trials over which task performance averages and standard deviations are maintained

$task\_time_i(k,j,t)$ = (average time over last μ trials of r_k's performance of task $h_i(a_{ij})$) + (one standard deviation of these μ attempts), as measured by r_i

$$task\_category_{ij}(t) = \begin{cases} 1 & \text{if } (task\_time(i,j,t) = \min_{k \in robots\_present(i,t)} task\_time(k,j,t)) \\ & \quad \text{and } ((\sum_{x \in robots\_present(i,t)} comm\_received(i,x,j,t-\tau_i,t)) = 0) \\ 2 & \text{otherwise} \end{cases}$$

$boredom\_threshold_i$ = level of boredom at which robot r_i ignores the presence of other robots able to perform some task not currently being executed

$boredom\_rate_i$ = Rate of boredom of robot r_i

$$boredom_i(t) = \begin{cases} 0 & \text{for } t = 0 \\ (\prod_j activity\_suppression_{ij}(t)) & \text{otherwise} \\ \quad \times (boredom_i(t-1) + boredom\_rate_i) & \end{cases}$$

$$learned\_robot\_influence_{ij}(t) = \begin{cases} 0 & \text{if } (boredom_i(t) < boredom\_threshold_i) \text{ and} \\ & (task\_category_{ij}(t) = 2) \\ 1 & \text{otherwise} \end{cases}$$

Robot impatience:

$$\begin{aligned} \phi_{ij}(k,t) &= \text{Time during which robot } r_i \text{ is willing to allow robot } r_k\text{'s} \\ &\quad \text{communication message to affect the motivation of behavior set } a_{ij}. \\ &= \begin{cases} task\_time_i(k,j,t) & \text{if } (strategy = \text{III}) \\ task\_time_i(i,j,t) & \text{if } (strategy = \text{IV}) \end{cases} \end{aligned}$$

$$\begin{aligned} \delta\_slow_{ij}(k,t) &= \text{Rate of impatience of robot } r_i \text{ concerning behavior set } a_{ij} \text{ after} \\ &\quad \text{discovering robot } r_k \text{ performing the task corresponding} \\ &\quad \text{to this behavior set} \\ &= \frac{\theta}{\phi_{ij}(k,t)} \end{aligned}$$

$$\begin{aligned} min\_delay &= \text{minimum allowed delay} \\ max\_delay &= \text{maximum allowed delay} \\ high &= \max_{k,j} task\_time_i(k,j,t) \\ low &= \min_{k,j} task\_time_i(k,j,t) \\ scale\_factor &= \frac{max\_delay - min\_delay}{high - low} \end{aligned}$$

$$\begin{aligned} \delta\_fast_{ij}(t) &= \text{Rate of impatience of robot } r_i \text{ concerning behavior set } a_{ij} \text{ in the} \\ &\quad \text{absence of other robots performing a similar behavior set} \\ &= \begin{cases} \frac{\theta}{min\_delay + (task\_time_i(i,j,t) - low) \times scale\_factor} & \text{if } task\_category_{ij}(t) = 2 \\ \frac{\theta}{max\_delay - (task\_time_i(i,j,t) - low) \times scale\_factor} & \text{otherwise} \end{cases} \end{aligned}$$

$$impatience_{ij}(t) = \begin{cases} \min_k(\delta\_slow_{ij}(k,t)) & \text{if } (comm\_received(i,k,j,t-\tau_i,t) = 1) \\ & \quad \text{and} \\ & \quad (comm\_received(i,k,j,0,t-\phi_{ij}(k,t)) = 0) \\ \delta\_fast_{ij}(t) & \text{otherwise} \end{cases}$$

$$impatience\_reset_{ij}(t) = \begin{cases} 0 & \text{if } \exists k.((comm\_received(i,k,j,t-\delta t,t) = 1) \text{ and} \\ & \quad (comm\_received(i,k,j,0,t-\delta t) = 0)), \\ & \quad \text{where } \delta t = \text{time since last communication check} \\ 1 & \text{otherwise} \end{cases}$$

Robot acquiescence:

$$\begin{aligned}\psi_{ij}(t) &= \text{Time robot } r_i \text{ wants to maintain behavior set } a_{ij}\text{'s activity before yielding to another robot.}\\ &= \begin{cases} task\_time_i(i,j,t) & \text{if } (strategy = \text{III})\\ \min\limits_{k \in robots\_present(i,t)} task\_time_i(k,j,t) & \text{if } (strategy = \text{IV}) \end{cases}\end{aligned}$$

$$\lambda_{ij}(t) = \text{Time robot } r_i \text{ wants to maintain behavior set } a_{ij}\text{'s activity before giving up to try another behavior set}$$

$$acquiescence_{ij}(t) = \begin{cases} 0 & \text{if [(behavior set } a_{ij} \text{ of robot } r_i \text{ has been active for more}\\ & \quad \text{than } \psi_{ij}(t) \text{ time units at time } t) \text{ and}\\ & \quad (\exists x. comm\_received(i,x,j,t-\tau_i,t) = 1)]\\ & \text{or}\\ & \quad \text{(behavior set } a_{ij} \text{ of robot } r_i \text{ has been active for more}\\ & \quad \text{than } \lambda_{ij}(t) \text{ time units at time } t)\\ 1 & \text{otherwise} \end{cases}$$

Motivation calculation:

The motivation of robot r_i to perform behavior set a_{ij} at time t is calculated as follows:

DURING ACTIVE LEARNING PHASE:

$$\begin{aligned} random\_increment &\leftarrow \theta \times \text{(a random number between 0 and 1)}\\ m_{ij}(0) &= 0\\ m_{ij}(t) &= [m_{ij}(t-1) + random\_increment]\\ &\quad \times sensory\_feedback_{ij}(t)\\ &\quad \times activity\_suppression_{ij}(t)\\ &\quad \times learning\_impatience_{ij}(t) \end{aligned}$$

The motivation to perform any given task thus increments at some random rate until it crosses the threshold, unless the task becomes complete (*sensory_feedback*), some other behavior set activates first (*activity_suppression*), or some other robot has taken on that task (*learning_impatience*).

When the robots are working on a "live" mission, their motivations to perform their tasks increment according to the robots' learned information. The motivations are thus calculated as follows:

DURING ADAPTIVE PHASE:

$$\begin{aligned}
m_{ij}(0) &= 0 \\
m_{ij}(t) &= [m_{ij}(t-1) + \mathit{impatience}_{ij}(t)] \\
&\quad \times \mathit{sensory\_feedback}_{ij}(t) \\
&\quad \times \mathit{activity\_suppression}_{ij}(t) \\
&\quad \times \mathit{impatience\_reset}_{ij}(t) \\
&\quad \times \mathit{acquiescence}_{ij}(t) \\
&\quad \times \mathit{learned\_robot\_influence}_{ij}(t)
\end{aligned}$$

In either the active or the adaptive learning phases, when behavior set a_{ij} is operational in robot r_i, the corresponding motivational behavior broadcasts r_i's current activity to its teammates at a rate of ρ_i.

Bibliography

[Angle, 1991] Colin Angle. Design of an artificial creature. Master's thesis, Massachusetts Institute of Technology, 1991.

[Arkin *et al.*, 1993] Ronald C. Arkin, Tucker Balch, and Elizabeth Nitz. Communication of behavioral state in multi-agent retrieval tasks. In *Proceedings of the 1993 International Conference on Robotics and Automation*, pages 588–594, 1993.

[Arkin, 1990] Ronald C. Arkin. Integrating behavioral, perceptual, and world knowledge in reactive navigation. *Robotics and Autonomous Systems*, 6:105–122, 1990.

[Asama *et al.*, 1992] H. Asama, K. Ozaki, A. Matsumoto, Y. Ishida, and I. Endo. Development of task assignment system using communication for multiple autonomous robots. *Journal of Robotics and Mechatronics*, 4(2):122–127, 1992.

[Beni and Wang, 1990] Gerardo Beni and Jing Wang. On cyclic cellular robotic systems. In *Japan – USA Symposium on Flexible Automation*, pages 1077–1083, Kyoto, Japan, 1990.

[Bonasso, 1991] R. Peter Bonasso. Underwater experiments using a reactive system for autonomous vehicles. In *Proceedings of the Eight National Conference on Artificial Intelligence*, pages 794–800, 1991.

[Bond and Gasser, 1988] Alan Bond and Less Gasser. *Readings in Distributed Artificial Intelligence.* Morgan Kaufmann, 1988.

[Braitenberg, 1984] Valentino Braitenberg. *Vehicles: Experiments in Synthetic Psychology.* MIT Press, 1984.

[Brooks *et al.*, 1990] Rodney A. Brooks, Pattie Maes, Maja Mataric, and Grinnell Moore. Lunar base construction robots. In *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems*, pages 389–392, Tsuchiura, Japan, 1990.

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

[Brooks, 1989] Rodney A. Brooks. A robot that walks: Emergent behavior from a carefully evolved network. *Neural Computation*, 1(2):253–262, 1989.

[Brooks, 1990a] Rodney A. Brooks. The behavior language: User's guide. Memo 1227, MIT A.I. Lab, Cambridge, MA, April 1990.

[Brooks, 1990b] Rodney A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

[Brooks, 1991a] Rodney A. Brooks. Artificial life and real robots. In Francisco J. Varela and Paul Bourgine, editors, *Proceedings of the First European Conference on Artificial Life*, pages 3–10, Paris, France, 1991.

[Brooks, 1991b] Rodney A. Brooks. New approaches to robotics. *Science*, 253:1227–1232, September 1991.

[Caloud *et al.*, 1990] Philippe Caloud, Wonyun Choi, Jean-Claude Latombe, Claude Le Pape, and Mark Yim. Indoor automation with many mobile robots. In *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems*, pages 67–72, Tsuchiura, Japan, 1990.

[Cammarata *et al.*, 1983] Stephanie Cammarata, David McArthur, and Randall Steeb. Strategies of cooperation in distributed problem solving. In *Proceedings of 8th International Joint Conference on Artificial Intelligence*, pages 767–770, 1983.

[Carberry, 1990] Mary Sandra Carberry. Incorporating default inferences into plan recognition. In *Proceedings of the National Conference on Artificial Intelligence*, pages 471–478, 1990.

[Charniak and Goldman, 1989] Eugene Charniak and Robert Goldman. A probabilistic model of plan recognition. In *Proceedings of the Eleventh International Conference on Artificial Intelligence*, pages 160–165, 1989.

[Cheney and Seyfarth, 1990] Dorothy Cheney and Robert Seyfarth. *How Monkeys See the World.* University of Chicago Press, 1990.

[Chung, 1974] K. L. Chung. *Elementary Probability Theory with Stochastic Processes.* Springer-Verlag, New York, 1974.

[Cohen *et al.*, 1990a] Paul Cohen, Michael Greenberg, David Hart, and Adele Howe. Real-time problem solving in the Phoenix environment. COINS Technical Report 90-28, University of Massachusetts at Amherst, 1990.

[Cohen *et al.*, 1990b] Philip R. Cohen, Jerry Morgan, and Martha Pollack, editors. *Intentions in Communication.* MIT Press, 1990.

[Connell, 1989] Jonathan Connell. A colony architecture for an artificial creature. Technical Report AI-TR-1151, MIT, Cambridge, MA, 1989.

[Conway *et al.*, 1967] R. W. Conway, W. L. Maxwell, and L. W. Miller. *Theory of Scheduling.* Addison-Wesley, Reading, Massachusetts, 1967.

[Davis and Smith, 1983] Randall Davis and Reid Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20(1):63–109, 1983.

[Dean and Bonasso, 1993] Thomas Dean and R. Peter Bonasso. The 1992 AAAI robot exhibition and competition. *AI Magazine*, 14(1):34–48, Spring 1993.

[Deneubourg *et al.*, 1990] J. Deneubourg, S. Goss, G. Sandini, F. Ferrari, and P. Dario. Self-organizing collection and transport of objects in unpredictable environments. In *Japan-U.S.A. Symposium on Flexible Automation*, pages 1093–1098, 1990.

[Deneubourg *et al.*, 1992] J. Deneubourg, Guy Theraulaz, and Ralph Beckers. Swarm-made architectures. In *Proceedings of the First European Conference on Artificial Life*, pages 123–133, 1992.

[Donald *et al.*, 1993] Bruce Randall Donald, James Jennings, and Daniela Rus. Towards a theory of information invariants for cooperating autonomous mobile robots. In *Proceedings of the International Symposium of Robotics Research*, Hidden Valley, PA, October 1993.

[Drogoul and Ferber, 1992] Alexis Drogoul and Jacques Ferber. From Tom Thumb to the Dockers: Some experiments with foraging robots. In *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 451–459, 1992.

[Durfee and Montgomery, 1990] Edmund Durfee and Thomas Montgomery. A hierarchical protocol for coordinating multiagent behaviors. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 86–93, 1990.

[Etkin, 1964] W. Etkin. *Social Behavior from Fish to Man.* The University of Chicago Press, Chicago, 1964.

[Ferrell, 1993] Cynthia Ferrell. Robust agent control of an autonomous robot with many sensors and actuators. Master's thesis, Massachusetts Institute of Technology, 1993.

[Fikes and Nilsson, 1971] R. E. Fikes and N. J. Nilsson. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence*, 2(3/4):189–208, 1971.

[Flynn *et al.*, 1989] Anita Flynn, R. Brooks, R. Wells, and D. Barrett. Squirt: The prototypical mobile robot for autonomous graduate students. A.I. Memo 1220, Massachusetts Institute of Technology, 1989.

[Franklin and Harmon, 1987] R. F. Franklin and L. A. Harmon. Elements of cooperative behavior. Internal Research and Development Final Report 655404-1-F, ERIM, August 1987.

[Fukuda *et al.*, 1988] T. Fukuda, S. Nakagawa, Y. Kawauchi, and M. Buss. Self organizing robots based on cell structures — CEBOT. In *Proceedings of 1988 IEEE International Workshop on Intelligent Robots and Systems (IROS '88)*, pages 145–150, 1988.

[Gage, 1993] Douglas Gage. Randomized search strategies with imperfect sensors. In *Proceedings of SPIE Mobile Robots VIII*, Boston, September 1993.

[Garey and Johnson, 1979] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman and Company, 1979.

[Gat *et al.*, 1993] Erann Gat, Albert Behar, Rajiv Desai, Robert Ivlev, John Loch, and David Miller. Behavior control for planetary exploration: Interim report. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 567–571, 1993.

[Giralt *et al.*, 1983] G. Giralt, R. Chatila, and M. Vaisset. An integrated navigation and motion control system for autonomous multisensory mobile robots. In M. Brady and R. Paul, editors, *First International Symposium on Robotics Research.* MIT Press, 1983.

[Goodman and Litman, 1990] Bradley A. Goodman and Diane J. Litman. Plan recognition for intelligent interfaces. In *Proceedings of teh 6th Conference on Artificial Intelligence Applications*, pages 297–303, 1990.

[Goss and Deneubourg, 1992] S. Goss and J. Deneubourg. Harvesting by a group of robots. In *Proceedings of the First European Conference on Artificial Life*, pages 195–204, 1992.

[Horswill, 1993] Ian Horswill. *Specialization of Perceptual Processes.* PhD thesis, Massachusetts Institute of Technology, 1993.

[Huber and Durfee, 1993] Marcus Huber and Edmund Durfee. Observational uncertainty in plan recognition among interacting robots. In *Proceedings of the 1993 IJCAI Workshop on Dynamically Interacting Robots*, pages 68–75, 1993.

[Ikeuchi *et al.*, 1993] Katsushi Ikeuchi, Masato Kawade, and Takashi Suehiro. Assembly task recognition with planar, curved, and mechanical contacts. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 688–694, 1993.

[IS Robotics, 1992a] IS Robotics, Inc., Somerville, Massachusetts. *The Genghis II Legged Robot Manual*, November 1992.

[IS Robotics, 1992b] IS Robotics, Inc., Somerville, Massachusetts. *The R-2 Wheeled Robot Manual*, July 1992.

[IS Robotics, 1993a] IS Robotics, Inc., Somerville, Massachusetts. *68332-Based Software Environment: R2E and Nero*, February 1993.

[IS Robotics, 1993b] IS Robotics, Inc., Somerville, Massachusetts. *ISR Genghis II Radio Communication and Positioning System*, October 1993.

[IS Robotics, 1993c] IS Robotics, Inc., Somerville, Massachusetts. *ISR Radio Communication and Positioning System*, October 1993.

[Konolige and Pollack, 1989] Kurt Konolige and Martha Pollack. Ascribing plans to agents. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 924–930, 1989.

[Kreifelts and von Martial, 1990] Thomas Kreifelts and Frank von Martial. A negotiation framework for autonomous agents. In *Proceedings of the Second European Workshop on Modeling Autonomous Agents and Multi-Agent Worlds*, pages 169–182, 1990.

[Kube and Zhang, 1992] C. Ronald Kube and Hong Zhang. Collective robotic intelligence. In *Proceedings of the Second International Workshop on Simulation of Adaptive Behavior*, pages 460–468, 1992.

[Kuniyoshi and Inoue, 1993] Yasuo Kuniyoshi and Hirochika Inoue. Qualitative recognition of ongoing human action sequences. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, pages 1600–1609, 1993.

[Lander and Waterman, 1988] Eric S. Lander and Michael S. Waterman. Genomic mapping by fingerprinting random clones: A mathematical analysis. *Genomics*, 2:231–239, 1988.

[Lesser and Corkill, 1983] Victor R. Lesser and Daniel D. Corkill. The distributed vehicle monitoring testbed: A tool for investigating distributed problem solving networks. *AI Magazine*, pages 15–33, Fall 1983.

[Maes, 1989] Pattie Maes. How to do the right thing. *Connection Science*, 1(3):291–323, 1989.

[Maes, 1990] Pattie Maes, editor. *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back.* MIT Press, 1990.

[Mataric, 1992a] Maja Mataric. Designing emergent behaviors: From local interactions to collective intelligence. In J. Meyer, H. Roitblat, and S. Wilson, editors, *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 432–441. MIT Press, 1992.

[Mataric, 1992b] Maja Mataric. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, June 1992.

[Mayfield, 1989] James Mayfield. Goal analysis: Plan recognition in dialogue systems. Computer Science Division Report UCB 89/521, University of California, Berkeley, August 1989.

[McFarland, 1991] David McFarland. Personal communication, March 1991.

[Miller *et al.*, 1992] David Miller, Rajiv Desai, Erann Gat, Robert Ivlev, and John Loch. Reactive navigation through rough terrain: Experimental results. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, pages 823–828, 1992.

[Miller, 1990] David Miller. Multiple behavior-controlled micro-robots for planetary surface missions. In *Proceedings IEEE Systems, Man, and Cybernetics Conference*, Studio City, CA, 1990.

[Noreils, 1993] Fabrice R. Noreils. Toward a robot architecture integrating cooperation between mobile robots: Application to indoor environment. *The International Journal of Robotics Research*, 12(1):79–98, February 1993.

[Ohko *et al.*, 1993] Takuya Ohko, Kazuo Hiraki, and Yuichiro Anzai. Lemming: A learning system for multi-robot environments. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1141–1146, Yokohama, Japan, 1993.

[Parker, 1992] Lynne E. Parker. Adaptive action selection for cooperative agent teams. In Jean-Arcady Meyer, Herbert Roitblat, and Stewart Wilson, editors, *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 442–450. MIT Press, 1992.

[Parker, 1993a] Lynne E. Parker. Designing control laws for cooperative agent teams. In *Proceedings of the IEEE Robotics and Automation Conference*, pages 582–587, Atlanta, GA, 1993.

[Parker, 1993b] Lynne E. Parker. Learning in cooperative robot teams. In *Working Notes of the IJCAI Workshop on Dynamically Interacting Robots*, Chambery, France, August 1993.

[Parker, 1994] Lynne E. Parker. An experiment in mobile robotic cooperation. In *Proceedings of the ASCE Specialty Conference on Robotics for Challenging Environments*, Albuquerque, NM, February 1994.

[Payton, 1991] David W. Payton. Intelligent real-time control of robotic vehicles. *Communications of the ACM*, 1991.

[Pin *et al.*, 1991] Francois G. Pin, Philippe F. R. Belmans, Susan I. Hruska, Carl W. Steidley, and Lynne E. Parker. Robotic learning from distributed sensory sources. *IEEE Transactions on Systems, Man, and Cybernetics*, 21(5):1216–1223, September/October 1991.

[Portmann, 1961] A. Portmann. *Animals as Social Beings.* The Viking Press, New York, 1961.

[Press, 1986] Associated Press. Mechanical assistant. *New York Times*, page A19, Wednesday, May 7 1986.

[Raskutti and Zukerman, 1991] Bhavani Raskutti and Ingrid Zukerman. Handling uncertainty during plan recognition in task-oriented consultation systems. In *Proceedings of the Seventh Conference on Uncertainty in Artificial Intelligence*, pages 308–315, 1991.

[Rosenschein and Genesereth, 1985] Jeffery Rosenschein and M. R. Genesereth. Deals among rational agents. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 91–99, 1985.

[Smith and Davis, 1981] Reid G. Smith and Randall Davis. Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1):61–70, January 1981.

[Smith, 1980] Reid G. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12):1104–1113, 1980.

[Steels, 1990] Luc Steels. Cooperation between distributed agents through self-organization. In Yves Demazeau and Jean-Pierre Muller, editors, *Decentralized A.I.* Elsevier Science, 1990.

[Stilwell and Bay, 1993] Daniel Stilwell and John Bay. Toward the development of a material transport system using swarms of ant-like robots. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 766–771, 1993.

[Sussman, 1973] Gerald J. Sussman. *A Computer Model of Skill Acquisition.* PhD thesis, Massachusetts Institute of Technology, 1973.

[Takahashi *et al.*, 1993] Tomoichi Takahashi, Hiroyuki Ogata, and Shin yo Muto. A method for analyzing human assembly operations for use in automatically generating robot commands. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 695–700, 1993.

[Theraulaz *et al.*, 1990] Guy Theraulaz, Simon Goss, Jacques Gervet, and Jean-Louis Deneubourg. Task differentiation in Polistes wasp colonies: a model for self-organizing groups of robots. In *Proceedings of the First International Conference on Simulation of Adaptive Behavior*, pages 346–355, 1990.

[Tinbergen, 1953] N. Tinbergen. *Social Behavior in Animals.* Chapman and Hall LTD, Great Britain, 1953.

[Tinbergen, 1965] N. Tinbergen. *Animal Behavior.* Time-Life Books, New York, 1965.

[Wang, 1991] P. K. C. Wang. Navigation strategies for multiple autonomous mobile robots moving in formation. *Journal of Robotic Systems*, 8(2):177–195, 1991.

[Wang, 1993] Jing Wang. DRS operating primitives based on distributed mutual exclusion. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1085–1090, Yokohama, Japan, 1993.

[Wilson, 1971] E. Wilson. *The Insect Societies*. The Belknap Press, Cambridge, 1971.

[Woodruff, 1988] Madeline Gary Woodruff. Information policy and radiation reporting during the Chernobyl and Three Mile Island nuclear accidents. Master's thesis, Massachusetts Institute of Technology, 1988.

[Yamamoto, 1993] Masaki Yamamoto. Sozzy: A hormone-driven autonomous vacuum cleaner. In *Proceedings of the SPIE Mobile Robots VIII Conference*, Boston, Massachusetts, September 1993.

[Yang *et al.*, 1993] Jie Yang, Yangsheng Xu, and C. S. Chen. Hidden markov model approach to skill learning and its application in telerobotics. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 396–402, 1993.

[Zlotkin and Rosenschein, 1990] Gilad Zlotkin and Jeffery Rosenschein. Negotiation and conflict resolution in non-cooperative domains. In *Proceedings of the Eighth National Conference on AI*, pages 100–105, 1990.