

**ornl**

ORNL/TM-13000

RECEIVED  
DEC 28 1995  
OSTI

**OAK RIDGE  
NATIONAL  
LABORATORY**

**MARTIN MARIETTA**

**L-ALLIANCE: A Mechanism for  
Adaptive Action Selection in  
Heterogeneous Multi-Robot Teams**

L. E. Parker

MANAGED BY  
MARTIN MARIETTA ENERGY SYSTEMS, INC.  
FOR THE UNITED STATES  
DEPARTMENT OF ENERGY

This report has been reproduced directly from the best available copy.

Available to DOE and DOE contractors from the Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831; prices available from (615) 576-8401, FTS 626-8401.

Available to the public from the National Technical Information Service, U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161.

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

Computer Science and Mathematics Division

**L-ALLIANCE: A MECHANISM FOR ADAPTIVE ACTION SELECTION  
IN HETEROGENEOUS MULTI-ROBOT TEAMS**

L. E. PARKER

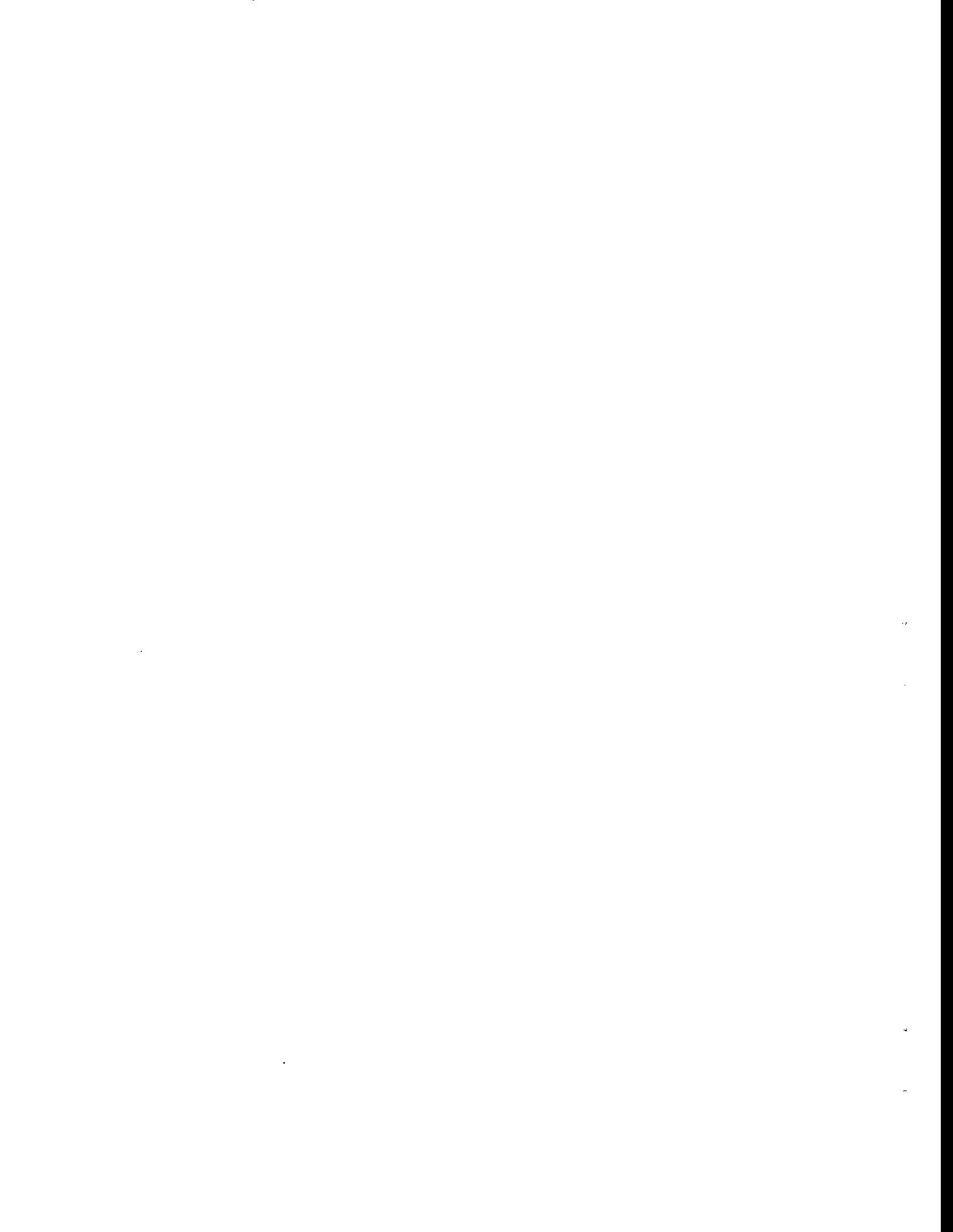
Research sponsored by the Engineering Research Program, Office of Basic Energy  
Sciences and U. S. Department of Energy

Date Published: November 1995

Prepared by the  
Oak Ridge National Laboratory  
Oak Ridge, Tennessee 37831  
managed by  
Lockheed Martin Energy Systems  
for the  
U. S. Department of Energy  
under Contract No. DE-AC05-84OR21400

DISTRIBUTION OF THIS DOCUMENT IS UNLIMITED

**MASTER**



# Contents

<b>1</b>	<b>INTRODUCTION</b>	<b>1</b>
<b>2</b>	<b>RELATED WORK</b>	<b>3</b>
<b>3</b>	<b>THE ALLIANCE ARCHITECTURE</b>	<b>5</b>
<b>4</b>	<b>MOTIVATION FOR EFFICIENCY IMPROVEMENTS</b>	<b>9</b>
<b>5</b>	<b>THE EFFICIENCY PROBLEM</b>	<b>11</b>
<b>6</b>	<b>OVERVIEW OF THE L-ALLIANCE MECHANISM</b>	<b>13</b>
6.1	ASSUMPTIONS MADE IN L-ALLIANCE . . . . .	13
6.2	PERFORMANCE MONITORS . . . . .	14
6.3	TWO L-ALLIANCE CONTROL PHASES . . . . .	14
6.3.1	Active Learning Phase . . . . .	15
6.3.2	Adaptive Learning Phase . . . . .	16
<b>7</b>	<b>EMPIRICAL INVESTIGATIONS OF DYNAMIC PARAMETER UPDATE MECHANISMS</b>	<b>17</b>
7.1	THREE IMPATIENCE/ACQUIESCENCE UPDATE STRATEGIES	18
7.1.1	Strategy I: Distrust Performance Knowledge about Teammates	19
7.1.2	Strategy II: Let the Best Robot Win . . . . .	20
7.1.3	Strategy III: Give Robots a Fighting Chance . . . . .	21
7.2	THREE TASK ORDERING STRATEGIES . . . . .	21
7.2.1	Longest Task First . . . . .	22
7.2.2	Modified Shortest Task First . . . . .	22
7.2.3	Modified Random Task Selection . . . . .	23
7.3	EXPERIMENTAL RESULTS OF L-ALLIANCE CONTROL STRATE- GIES . . . . .	23
7.3.1	Effect of Impatience/Acquiescence Update Strategy . . . . .	24
7.3.2	Effect of Task Ordering Approach . . . . .	30
7.4	THE PREFERRED L-ALLIANCE DISTRIBUTED CONTROL STRAT- EGY FOR EFFICIENCY AND FAULT TOLERANCE . . . . .	34
7.5	COMPARISON TO THE OPTIMAL SOLUTION . . . . .	35
<b>8</b>	<b>L-ALLIANCE FORMAL MODEL</b>	<b>39</b>
8.1	THRESHOLD OF ACTIVATION . . . . .	39
8.2	SENSORY FEEDBACK . . . . .	39
8.3	INTER-ROBOT COMMUNICATION . . . . .	40
8.4	SUPPRESSION FROM ACTIVE BEHAVIOR SETS . . . . .	41
8.5	LEARNED ROBOT INFLUENCE . . . . .	42
8.6	ROBOT IMPATIENCE . . . . .	44
8.7	ROBOT ACQUIESCENCE . . . . .	47

8.8	MOTIVATION CALCULATION . . . . .	48
<b>9</b>	<b>IMPLEMENTATION ON MOBILE ROBOTS</b>	<b>51</b>
9.1	THE POOL OF HETEROGENEOUS ROBOTS . . . . .	51
9.2	THE BOX PUSHING DEMONSTRATION . . . . .	52
9.3	ROBOT SOFTWARE DESIGN . . . . .	54
9.3.1	R-2 Control . . . . .	54
9.3.2	Genghis-II Control . . . . .	55
9.4	ROBOT EXPERIMENTATION RESULTS . . . . .	56
9.4.1	Experiment 1: Robot "failure" . . . . .	57
9.4.2	Experiment 2: Increased heterogeneity . . . . .	58
<b>10</b>	<b>CONCLUSIONS</b>	<b>61</b>
<b>11</b>	<b>ACKNOWLEDGEMENTS</b>	<b>63</b>

## List of Figures

1	The ALLIANCE architecture. . . . .	7
2	The L-ALLIANCE architecture. . . . .	15
3	Summary of time usage for three impatience/acquiescence strategies. . . . .	26
4	An average time performance of the three impatience/acquiescence strategies in region 1. . . . .	27
5	An average time performance of the three impatience/acquiescence strategies in region 2 when the <i>Progress When Working</i> condition is not true. . . . .	29
6	An average time performance of the three impatience/acquiescence strategies in region 3. . . . .	30
7	An average time performance of the three impatience/acquiescence strategies in region 4. . . . .	31
8	Typical change in mission completion time when using random task selection instead of shortest task first selection. . . . .	33
9	Mission scenarios over which the optimal result could be computed. . . . .	35
10	Comparison of the preferred control strategy performance with the optimal performance. . . . .	37
11	The pool of heterogeneous robots — three R-2s and one Genghis-II. . . . .	52
12	The L-ALLIANCE design of the R-2 software for the box pushing demonstration. . . . .	55
13	The L-ALLIANCE design of the Genghis-II software for the box pushing demonstration. . . . .	56
14	The beginning of the box pushing demonstration. . . . .	57
15	Fault tolerant action selection. . . . .	58
16	Adaptivity due to heterogeneity. . . . .	59
17	Response to robot failure. . . . .	60



## List of Tables

1	High level task-achieving functions of various robots. . . . .	6
2	Three impatience/acquiescence update strategies . . . . .	19
3	Preferred impatience/acquiescence strategies when time is the performance metric. . . . .	31



## ABSTRACT

In practical applications of robotics, it is usually quite difficult, if not impossible, for the system designer to fully predict the environmental states in which the robots will operate. The complexity of the problem is further increased when dealing with teams of robots which themselves may be incompletely known and characterized in advance. It is thus highly desirable for robot teams to be able to adapt their performance during the mission due to changes in the environment, or to changes in other robot team members. In previous work [40, 44], we introduced a behavior-based mechanism — called the ALLIANCE architecture — that facilitates the fault tolerant cooperative control of multi-robot teams. However, this previous work did not address the issue of how to dynamically update the control parameters during a mission to adapt to ongoing changes in the environment or in the robot team, and to ensure the efficiency of the collective team actions. In this paper, we address this issue by proposing the L-ALLIANCE mechanism, which defines an automated method whereby robots can use knowledge learned from previous experience to continually improve their collective action selection when working on missions composed of loosely coupled, discrete subtasks. This ability to dynamically update robotic control parameters provides a number of distinct advantages: it alleviates the need for human tuning of control parameters, it facilitates the use of custom-designed multi-robot teams for any given application, it improves the efficiency of the mission performance, and it allows robots to continually adapt their performance over time due to changes in the robot team and/or the environment. We describe the L-ALLIANCE mechanism, present the results of various alternative update strategies we investigated, present the formal model of the L-ALLIANCE mechanism, and present the results of a simple proof of concept implementation on a small team of heterogeneous mobile robots.



# 1 INTRODUCTION

Achieving cooperative robotics is desirable for a number of reasons. First, many robotic applications are inherently distributed in space, time, or functionality, thus requiring a distributed solution. Second, it is quite possible that many applications could be solved much more quickly if the mission could be divided across a number of robots operating in parallel. Third, by duplicating capabilities across robot team members, one has the potential of increasing the robustness and reliability of the automated solution through redundancy. Furthermore, it may actually be much cheaper and more practical in many applications to build a number of less capable robots that can work together at a mission, rather than trying to build one robot which can perform the entire mission with adequate reliability.

Achieving cooperative robotics, however, is quite challenging. Many issues must be addressed in order to develop a working cooperative team, including action selection, coherence, conflict resolution, and communication. Cooperative teams often work in dynamic and unpredictable environments, thus requiring the robot team members to respond robustly, reliably, and adaptively to unexpected environmental changes, failures in the inter-robot communication system, noisy sensors and effectors, 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. Many multi-robot applications also require the use of heterogeneous robots with overlapping capabilities, which must coordinate their selection of tasks in order to efficiently accomplish their overall mission. However, the appropriate selection of tasks during one mission may not be proper for another similar mission, or for a later time in the same mission, due to changes that occur in the robot team or the environment. Since it is virtually impossible for the human designer to predict the robot environment and complete mission characteristics in advance, the robots must be able to autonomously adapt their actions over time based upon knowledge they learn from previous experience.

In previous work [40, 44], we introduced a formalism — called the ALLIANCE architecture — that facilitates the fault tolerant cooperative control of multi-robot teams. This behavior-based, fully distributed framework allows robots to select appropriate actions based upon the requirements of the mission, the activities of other robots, the current environmental conditions, and their own internal states. The ALLIANCE architecture is based upon the interaction of mathematically modeled control parameters that represent motivations of behavior, such as impatience and acquiescence, within each robot. These motivations allow robots to take over tasks from other team members if those team members do not demonstrate their ability — through their effect on the world — to accomplish those tasks. Similarly, it allows a robot to give up its own current task if its sensory feedback indicates that adequate progress is not being made to accomplish that task.

However, the ALLIANCE architecture does not address the issue of how to dynamically update the control parameters during a mission to adapt to ongoing

changes in the environment or in the robot team, and to ensure the efficiency of the collective team actions. Instead, it was assumed that a human designer provided the appropriate control parameters at the beginning of the mission that allow the robots to cooperate effectively. While our previous work illustrated that a high degree of fault tolerance is possible using these fixed control parameters, a much higher level of flexibility, adaptivity, and efficiency can be achieved by dynamically varying the control parameters during the mission. The L-ALLIANCE architecture provides this capability by defining an automated mechanism whereby robots can use knowledge learned from previous experience to continually improve their collective action selection when working on missions composed of loosely-coupled, discrete subtasks. The ability to dynamically update the control parameters provides a number of distinct advantages: it alleviates the need for human tuning of control parameters, it facilitates the use of custom-designed multi-robot teams for any given application, it improves the efficiency of the mission performance, and it allows robots to continually adapt their performance over time due to changes in the robot team and/or the environment.

This paper presents the L-ALLIANCE mechanism for dynamic control parameter updates. We begin with an overview of related cooperative robotics work in the following section. We then present a brief overview of the ALLIANCE architecture in section 6, followed by a discussion of the motivation for efficiency improvements in L-ALLIANCE. Section 7 presents a simplified version of the efficiency problem and shows that it is intractable, thus leading to the need for approximate solutions. Section 8 presents an overview of the L-ALLIANCE mechanism, followed by the discussion of the various control strategies we investigated for use in L-ALLIANCE. The formal model of L-ALLIANCE is presented in section 7.5. In section 8.8, we present the results of a simple proof of concept implementation of this approach on a team of mobile robots performing a box pushing demonstration. Finally, we offer concluding remarks in section 9.4.2.

## 2 RELATED WORK

Research in cooperative robotics can be characterized in many ways. In [20], Dudek proposes a taxonomy of cooperative robotics that distinguishes systems based upon the size of the team, the communication range, topology, and bandwidth, re-configurability, unit processing ability, and team composition (heterogeneous versus homogeneous). Here, we broadly segment the cooperative robotics work into two categories based on team composition: large numbers of homogeneous robots versus smaller numbers of heterogeneous robots.

A significant body of research in cooperative mobile robotics deals with the study of large numbers (or swarms) of homogeneous robots. This approach to multi-robot 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 typically taken in these systems is derived from the fields of neurobiology, ethology, psychophysics, and sociology, and is characterized by teams of large numbers of homogeneous robots, each of which has fairly limited capabilities on its own. However, when many such simple robots are brought together, globally interesting behavior can emerge as a result of the local interactions of the robots. A key research issue in this scenario is determining the proper design of the local control laws that allow the collection of robots to solve a given problem.

A number of researchers have studied the issues of swarm robotics. Deneubourg et al. [17] describe simulation results of a distributed sorting algorithm. Theraulaz et al. [49] extract cooperative control strategies, such as foraging, from a study of *Polistes* wasp colonies. Steels [47] presents simulation studies of the use of several dynamical systems to achieve emergent functionality as applied to the problem of collecting rock samples on a distant planet. Drogoul and Ferber [19] describe simulation studies of foraging and chain-making robots. McFarland [37] describes a robot ecosystem that allows cooperation to emerge in a collective team. In [34] Mataric describes the results of implementing group behaviors such as dispersion, aggregation, and flocking on a group of mobile robots. Beni and Wang [5] describe methods of generating arbitrary patterns in cyclic cellular robotics. Kube and Zhang [31] present the results of implementing an emergent control strategy on a group of five mobile robots performing the task of locating and pushing a brightly lit box. Stilwell and Bay [48] 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. Arkin et al. [2] present research concerned with sensing, communication, and social organization for tasks such as foraging. The CEBOT work, described in [24] and many related papers, has many similar goals to other swarm-type multi-robotic systems; however, the CEBOT robots can be one of a number of robot classes, rather than purely homogeneous.

Another primary area of research in cooperative control deals with achieving "intentional" cooperation among a limited number of typically heterogeneous robots performing several distinct tasks. In this type of cooperative system, the robots often

have to deal with some sort of efficiency constraint that requires a more directed type of cooperation than is found in the swarm approach described above. Although individual robots in this approach are usually able to perform some useful task on their own, groups of such robots are often able to accomplish missions that no individual robot can accomplish on its own. The general research issues of adaptive action selection, communication, and conflict resolution are of particular importance in these types of systems.

Two bodies of previous research are particularly applicable to this second type of cooperation. First, several researchers have directly addressed this cooperative robot problem by developing control algorithms and implementing them either on physical robots or on simulations of physical robots that make reasonable assumptions about robot capabilities. Examples of this research include the work of Noreils [39], who proposes a three-layered control architecture that includes a planner level, a control level, and a functional level; Caloud et al. [12], who describe an architecture that includes a task planner, a task allocator, a motion planner, and an execution monitor; Asama et al [4] who describes an architecture called ACTRESS that utilizes a negotiation framework to allow robots to recruit help when needed; Cohen et al [13], who use a hierarchical division of authority to address the problem of cooperative fire-fighting; and Wang [50], who proposes the use of several distributed mutual exclusion algorithms that use a "sign-board" for inter-robot communication.

The second, significantly larger, body of research related to intentional cooperation comes from the Distributed Artificial Intelligence (DAI) community, which has produced a great deal of work addressing this type of intentional cooperation among generic agents. These agents are typically software systems running as interacting processes to solve a common problem rather than embodied, sensor-based robots. In most of this work, the issue of task allocation has been the driving influence that dictates the design of the architecture for cooperation. Typically, the DAI approaches use a distributed, negotiation-based mechanism to determine the allocation of tasks to agents. See [7] for many of the seminal papers in this field.

Much less work has been done in the area of multi-robot learning, although the topic is gaining increased interest. Asada et al. [3] proposes a method for learning new behaviors by coordinating previously learned behaviors using Q-learning. They have applied their approach to a simulation of robots playing a simplified version of competitive soccer, and are transferring their results to physical robots. Mataric [36] introduces a method for combining basic behaviors into higher-level behaviors through the use of unsupervised reinforcement learning, heterogeneous reward functions, and progress estimators. This mechanism was applied to a team of robots learning to perform a foraging task. Kubo and Kakazu [32] proposed another reinforcement learning mechanism that use a progress value for determining reinforcement, and applied it to simulated ant colonies competing for food.

### 3 THE ALLIANCE ARCHITECTURE

The L-ALLIANCE dynamic parameter learning mechanism is built upon our earlier work — the ALLIANCE architecture. Thus, to provide suitable background information, we first briefly review the ALLIANCE approach to fault tolerant cooperative control in this section.

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. Table 1 gives examples of what we consider to be the high-level task-achieving functions of a number of previously reported robots.

In ALLIANCE, individual robots are designed using a behavior-based approach [8]. Under the behavior-based construction, 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. This approach has been used successfully in a number of robotic applications, several of which are described in [11].

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 1 shows the general architecture of ALLIANCE and illustrates three such behavior sets. The  $j$ th behavior set,  $a_{ij}$ , of a robot  $r_i$  corresponds to those levels of competence required to perform some high-level task-achieving function. When a robot activates a behavior set, we say that it has selected the task corresponding to that behavior set. Since different robots may have different ways of performing the same task, and therefore activate different behavior sets to perform that task, we define the function  $h_i(a_{ij})$ , for all robots  $r_i$  on the team, to refer to the task that robot  $r_i$  is working on when it activates its  $j$ -th behavior set,  $a_{ij}$ .

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. Thus, controlling the activation of each of these behavior sets is a *motivational behavior*. Due to conflicting goals, only one behavior set per robot can be active at any point in time. This restriction is implemented via cross-inhibition of motivational behaviors, represented by the arcs at the top of figure 1, 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

Robot	High-Level Functions
Allen [8]	Wander
Attila/Hannibal [21]	Keep walking
Genghis [9]	Keep walking
George/HARV [1]	Reactively navigate
Herbert [14]	Collect empty soda cans
Hilare [28]	Map office environment
Polly [29]	Give 7th floor AI Lab tours
Rocky III [38, 26]	Search for soft soil; acquire soil sample; return sample to home
Rocky IV [27, 26]	Collect soil sample; chip rocks; deploy instruments; return sample to home
RPV [6]	Reactively navigate underwater
Squirt [23]	Eavesdrop
Toto [35]	Map office environment; go to goal

Table 1: High level task-achieving functions of various robots.

high-level goal the robot is currently pursuing. Examples of this type of continually active competence are shown in figure 1 as layer 0, layer 1, and layer 2.

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 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.

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 activate behavior set  $a_{ij}$  increases at a "fast" rate (which we call  $\delta_{fast_{ij}}(t)$ ) as long as the task corresponding to that behavior set (i.e.  $h_i(a_{ij})$ ) is not completed, as determined from sensory feedback. However, the robots must 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  $h_i(a_{ij})$ , then  $r_i$  is satisfied for some period of time that the task is going to be accomplished even without its own participation, and thus go on to some other applicable action. Its motivation to activate behavior set  $a_{ij}$  still increases, but at a slower rate (which we call  $\delta_{slow_{ij}}(k, t)$ ). 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 (often called *action recognition*) is not a trivial problem, and often requires perceptual abilities that are not yet possible with current sensing technology. As it stands today, the sensory capabilities of even the

## The ALLIANCE Architecture

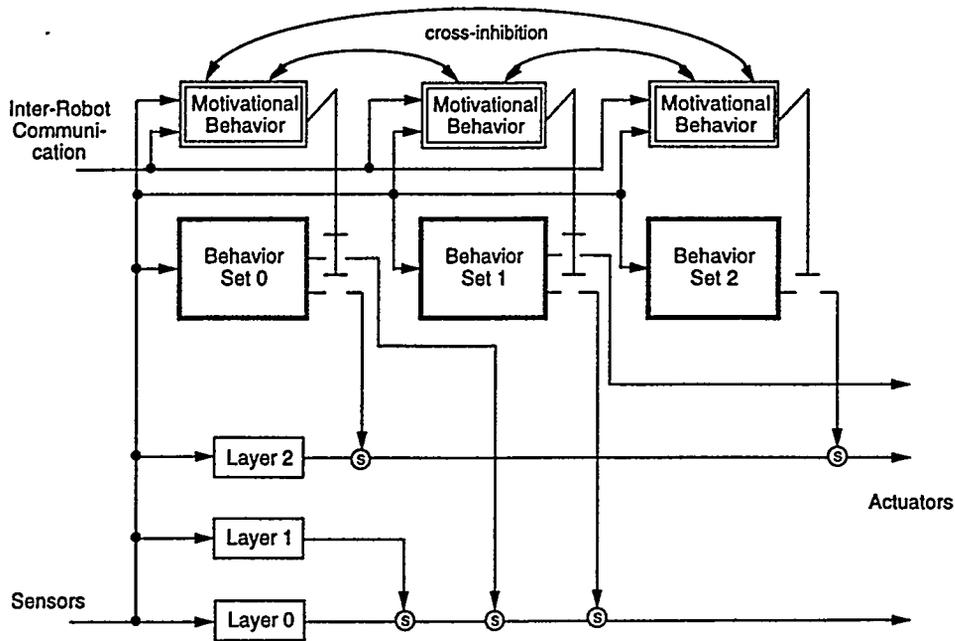


Figure 1: The ALLIANCE architecture. The symbols in this figure that connect the output of each motivational behavior with the output of its corresponding behavior set (vertical lines with short horizontal bars) indicate that a motivational behavior either allows all or none of the outputs of its behavior set to pass through to the robot's actuators. The "S" inside a circle indicates that the output of a higher level behavior is suppressed as needed by lower level behaviors (e.g. to avoid an obstacle).

lower animals far exceed present robotic capabilities. Thus, to enhance the robots' perceptual abilities, ALLIANCE utilizes a simple form of broadcast communication to allow robots to inform other team members of their current activities, rather than relying totally on sensory capabilities. 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.

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. Additionally, even if another robot  $r_k$  has not taken over the task, robot  $r_i$  may give up its task anyway if it 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 a robot is able to adapt its actions to its own failures.

The behavior-based 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. Refer to [40, 44] for more details of the ALLIANCE architecture, including the formal mathematical model of ALLIANCE, proofs of correction which guarantee, under certain conditions, that a mobile robot team will accomplish its missions under the control of ALLIANCE, and results of robot implementations of the ALLIANCE architecture.

## 4 MOTIVATION FOR EFFICIENCY IMPROVEMENTS

As described in the previous section, 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?*

In heterogeneous robot teams, there may often be more than one robot that can accomplish a given task, but with different levels of performance. The heterogeneous members of a robot team may also have different mixtures of capabilities, such that the “best” action for a given robot may vary depending upon which other robots belong to the team. Ideally, the robot individuals optimize their action selections depending upon the other team members that are present, and their capabilities.

- *Can we enable the robot team to increase its performance over time?*

It is desirable that robot teams use knowledge learned from previous experience to improve mission performance from trial to trial. This obviates the need for attempting to hand-code the “optimal” team configuration in advance, thus greatly reducing the programming burden.

- *Does failure at one task imply total robot failure?*

Ideally, a robot should recognize when it has failed at a given task, and continue to another task for which it is better suited. Failure at one task should not preclude the execution of another, unrelated, task.

- *How do we minimize robot idle time?*

As described in the previous section, the ALLIANCE architecture utilizes motivations to achieve appropriate action selection. However, as described, the architecture does not address the issue of robot idle time while the motivations are increasing. For practical applications, we must ensure an acceptably short upper limit on the idle time.

The L-ALLIANCE enhancement to ALLIANCE addresses these issues of efficiency by incorporating a dynamic control 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 control parameter settings, including the following:

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

As described earlier, ALLIANCE requires human programmer tuning of motivational behavior control parameters to achieve desired levels of robot performance. The use of any architecture is much simpler to use if humans are relieved of the responsibility of having to tune numerous parameters.

**2. Improves 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. Facilitates 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 affects its own behavior.

Providing robot team members with the ability to automatically update their 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 pre-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 question is how robots use the performance knowledge to efficiently select their own actions.

## 5 THE EFFICIENCY PROBLEM

To understand the difficulty of this efficiency problem, we first look formally at a simplified version of the problem, showing that even the simplified version is NP-hard. This leads to the need for approximate solutions.

Let  $R = \{r_1, r_2, \dots, r_n\}$  represent the finite set of  $n$  robots on a cooperative team, and the finite set  $T = \{task_1, task_2, \dots, task_m\}$  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 finite set  $A_i = \{a_{i1}, a_{i2}, \dots\}$ . Since different robots may have different ways of performing the same task, we define the set of  $n$  functions  $H$ , where  $H : A_i \rightarrow T$ ,  $H = \{h_1(a_{1k}), h_2(a_{2k}), \dots, h_n(a_{nk})\}$ , and  $h_i(a_{ik})$  returns the task,  $task_j$ , that robot  $r_i$  is working on when it activates behavior set  $a_{ik}$ .

We denote the metric evaluation function as  $q(a_{ij})$ , 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  $q(a_{ij})$  function. Thus, an additional aspect to the robot’s learning problem is actually obtaining the performance quality information required to make the appropriate action selection choice.

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

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

**Condition 1 (Different Robots are Different):** *Different robots may have different collections of capabilities; thus, we 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, we do not assume that if  $h_i(a_{ix}) = h_j(a_{jy})$ , then  $q(a_{ix}) = q(a_{jy})$ .*

Let us assume, for the simplified case, that the performance measurements of the robots performing the tasks for which they are capable are known in advance. Then we define the formal efficiency problem under condition 1 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 the following is minimized, according to the time performance metric:

$$\max_i \left( \sum_{a_{ik} \in U_i} q(a_{ik}) \right)$$

The first two constraints of the efficiency problem ensure that each task in the mission is assigned to some robot that can actually accomplish that task. The final constraint ensures that the total time required to complete the mission is minimized. Since robot team members will 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 the performance metric is time, the maximum amount of time any robot takes to perform its set of actions should be minimized.

Under the assumption that the robots have accurate and complete 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? It can be easily shown that the efficiency problem, AEP, is NP-hard by restriction to the well-known NP-complete problem PARTITION [25]. The PARTITION problem is as follows: given a finite set  $W$  and a “size”  $s(w) \in \mathbb{Z}^+$  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 1** *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  $n = 2$ ,  $A_1 = A_2 = W$ ,  $\forall i. \forall j. (h_1(a_{ij}) = \text{task}_j)$ , 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.  $\square$

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, we cannot expect the robot teams to be able to derive an optimal action selection policy in a reasonable length of time. Thus, we look instead to heuristic approximations to the problem that work well in practice.

## 6 OVERVIEW OF THE L-ALLIANCE MECHANISM

This section provides an overview of the L-ALLIANCE approach to the dynamic update of cooperative team control parameters. This approach was developed primarily to provide an infrastructure that affords a robot team with a high degree of fault tolerance and efficiency when working on missions composed of independent, discrete tasks. The assumptions made in the development of this approach are described, followed by a description of the overall framework under which L-ALLIANCE operates.

### 6.1 ASSUMPTIONS MADE IN L-ALLIANCE

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

- A robot's average performance in executing a specific task over a few recent trials is a reasonable indicator of that robot's expected performance of the same or related tasks 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 is 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 challenge, of course, is determining which aspects of a robot's performance are good predictors of future performance. 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, robots do 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. In L-ALLIANCE, we have used the simple measure of the *time* of task completion, which has served to be a good indicator of future performance.

The second assumption deals with the well-known credit assignment problem, which is concerned with determining which process receives credit (or punishment) for the successful (or unsuccessful) outcome of an action. The assumption made 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 assumes 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, we accept this oversimplification here.

## 6.2 PERFORMANCE MONITORS

Figure 2 illustrates the L-ALLIANCE extensions to the ALLIANCE architecture. These extensions incorporate the use of performance monitors for each motivational behavior within each robot. Each monitor is responsible for observing, evaluating, and cataloging the performance of any robot team member (including itself) whenever it performs the task corresponding to that monitor's respective behavior set. Formally, robot  $r_i$ , programmed with the  $b$  behavior sets  $A = \{a_{i1}, a_{i2}, \dots, a_{ib}\}$ , also has  $b$  monitors  $MON_i = \{mon_{i1}, mon_{i2}, \dots, mon_{ib}\}$ , such that monitor  $mon_{ij}$  observes the performance of any robot performing task  $h_i(a_{ij})$ , keeping track of the time of task completion (or other appropriate performance quality measure) of that robot. As mentioned earlier, since passive action observation is quite difficult to accomplish, the robots use the broadcast communication mechanism in L-ALLIANCE to inform teammates of their current actions, from which robots can derive task completion times. Monitor  $mon_{ij}$  then uses the mechanism described below to update the control parameters of behavior set  $a_{ij}$  based upon this learned knowledge. 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.

## 6.3 TWO L-ALLIANCE CONTROL PHASES

The ability of robots to monitor, evaluate, and catalog the performance of team members in executing certain tasks is of central importance to L-ALLIANCE. Without this ability, a robot must rely on human-supplied performance measurements of robot team members. 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 due to the occurrence of a robot failure.

The degree to which robot team members can obtain 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 can 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 is completed as efficiently as possible. Even so, as they perform the mission, the robots 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

The L-ALLIANCE Architecture

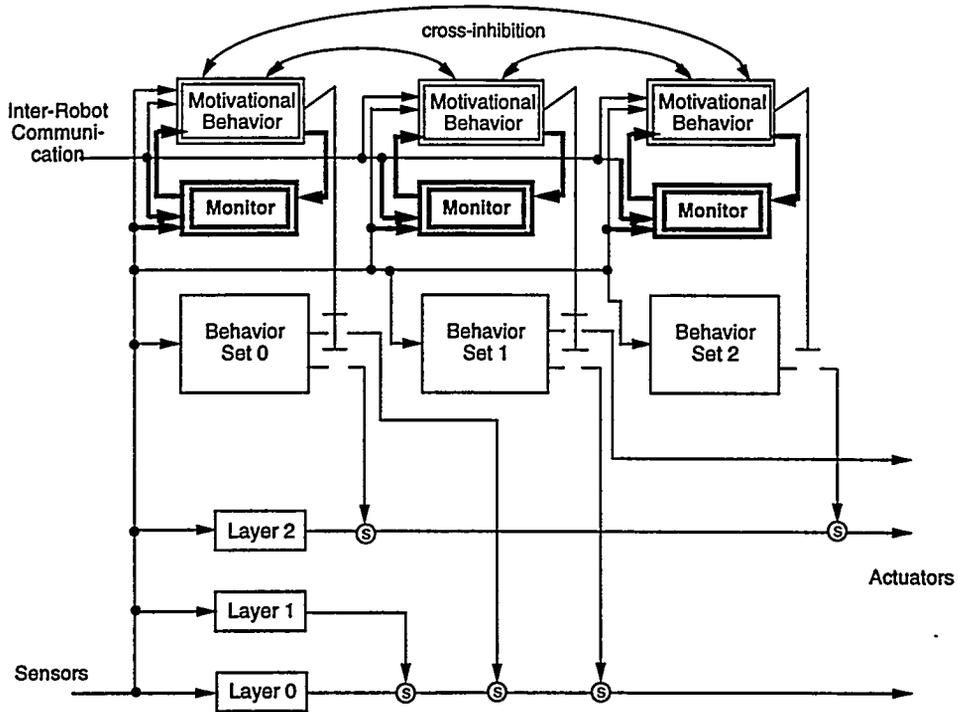


Figure 2: The L-ALLIANCE architecture. The changes from the ALLIANCE architecture are shown in bold. These changes add a monitor corresponding to each motivational behavior within each robot. These monitors are responsible for observing and recording the performance of robot team members, and of adapting the control parameters of the respective motivational behaviors accordingly. The “S” inside a circle indicates that the output of a higher level behavior is suppressed as needed by lower level behaviors (e.g. to avoid an obstacle).

missions, the robots enter the *active learning* phase, whereas during live missions, they enter the *adaptive learning* phase.

### 6.3.1 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 incomplete, 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 the active learning phase, each monitor  $mon_{ij}$  in each robot  $r_i$  monitors the performance of all robots  $r_k$  that are performing task  $h_i(a_{ij})$ . Monitor  $mon_{ij}$  observing robot  $r_k$  then catalogues the average time plus one standard deviation required by robot  $r_k$  to perform task  $h_i(a_{ij})$ , maintaining this information over only the previous  $\mu$  trials of  $r_k$ 's performance of  $h_i(a_{ij})$ . This running average plus one standard deviation is called  $task\_time_i(k, j, t)$ . In the case of robot failure, the actual time attributed to the failed robot is some penalty factor (greater than 1) multiplied by the actual attempted time. The standard deviation is added to the task time to account for environmental variations and sensory and effector noise that will undoubtedly cause performance to differ across task executions. Determining how many trials,  $\mu$ , over which to maintain this data depends upon the desired characteristics of the robot team [46]. 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. The experiments reported in this article have shown that an average over about 5 trials results in good predictive capability, while still allowing the robots to be responsive to failures.

### 6.3.2 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 must accomplish the mission with available knowledge 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 article, rather than being maximally patient and minimally acquiescent as they are in the active learning phase. However, the monitors within each robot continue to monitor and catalog robot performances during this phase, and update the average task completion times and standard deviations for the most recent  $\mu$  trials.

## 7 EMPIRICAL INVESTIGATIONS OF DYNAMIC PARAMETER UPDATE MECHANISMS

Once the quality measurements have been obtained, they are input to a control mechanism that allows the robot team to improve its efficiency over time while not sacrificing the fault tolerant characteristics demonstrated through the ALLIANCE architecture. Since, in the context of L-ALLIANCE, we are interested in the fault tolerant and efficient execution of a mission composed of loosely-coupled subtasks, this control problem translates into two related issues: (1) how an individual robot determines whether to interrupt the task currently being performed by another robot (i.e. become impatient), or whether it should acquiesce its own current task (either to some other team member, or to attempt some other task), and (2) how an individual robot selects from among a number of incomplete tasks that no other team member is currently performing.

The answers to these action selection questions largely determine 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 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. However, as we have already noted, this efficiency problem is NP-hard, and thus we cannot expect the robots to be able to derive an optimal selection of actions even if they did possess complete global information. Thus, we investigated a number of greedy approaches to this problem to find those approaches that work well in practice. The investigation of potential control approaches considered a number of factors that affect performance. The following subsections describe the approaches we investigated as a function of the *task coverage*, the relative mission size, the degree of heterogeneity across robots, the number of robots, and a condition we call *Progress When Working*.

In this context, we define *task coverage* as the measure of the number of capabilities on the robot team that may allow some team member to achieve a given task, given by:

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

The relative mission size is given by the ratio of the total number of tasks required by the mission to the size of the robot team. In this context, robot team members can be *heterogeneous* in two 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, the “degree of heterogeneity” in this section refers to the degree of difference in the qualities of performance of the same task by those robots which can perform that task.

Finally, we define a condition that holds in many multi-robotic applications:

**Condition 2 (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  $\epsilon$  such that  $z$  is reduced by a finite amount which is at least some constant  $\delta$  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 2 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  $\epsilon$ . 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  $w$ . This condition 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 2 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, in a wide variety of applications this is a valid assumption. Of course, this does not preclude dynamic environmental changes from increasing the workload of the robot team. As we shall see, the relative performances of the various control strategies vary depending upon whether or not this condition is true in a given situation.

Although we investigated the various dynamic parameter update strategies as functions of these factors, our goal was to find a single automated technique that could be incorporated into each robot on the team such that, regardless of the specific situation in which the robots find themselves, the robots could select the most appropriate control strategy for their situation. Ideally, the selection criteria should be as simple as possible, so that robots do not have to deliberate extensively to ascertain the proper parameter update technique.

## 7.1 THREE IMPATIENCE/ACQUIESCENCE UPDATE STRATEGIES

As described in section , the motivational behaviors are the foundation of the adaptive action selection facilitated by ALLIANCE. The primary motivations incorporated into

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

ALLIANCE are the impatience and acquiescence motivations, which allow robot team members to dynamically reallocate their actions based upon the effect the robots have on the world. Since these motivations are incorporated into the team members as control parameters, we must address how the appropriate parameter settings of these motivations are obtained. Colloquially, this problem can be stated as “knowing when to give up”, either on one’s own performance, or on the performance of other team members. Specifically, a robot must 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 robot  $r_i$ ’s response to robot  $r_k$ ’s performance of task  $h_i(a_{ij})$  at time  $t$ :  $\phi_{ij}(k, t)$  (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. We studied three impatience/acquiescence update control strategies, which are discussed in the following paragraphs. Table 2 summarizes the parameter settings for these strategies.

#### 7.1.1 Strategy I: Distrust Performance Knowledge about Teammates

The first impatience/acquiescence parameter update strategy takes a minimalist approach to the problem by requiring the robot to use only 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, 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 is not 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.

### 7.1.2 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 informally 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 both (1) they are not successful, and (2) that another robot on the team can do a better job.

### 7.1.3 Strategy III: Give Robots a Fighting Chance

The third strategy for updating the impatience and acquiescence factors results in a 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.

## 7.2 THREE TASK ORDERING STRATEGIES

The second issue in the action selection problem is determining how each robot selects from among a number of incomplete tasks that no other team member is currently performing. We investigated a number of approaches to this task ordering problem. A key concern in evaluating the alternative approaches is the degree of vulnerability of the robot team to any type of component failure — either the failure of robots or the failure of the communication system. If the robots are absolutely dependent upon the communication system to perform anything useful, then efforts to create 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<sup>1</sup>. Thus, to assure a high level of fault tolerance, robots cannot be required to wait to be “awarded a bid”, or to receive permission from some other robot via a communicated message before starting on a task, since a communication failure would cause the team to accomplish nothing.

We therefore investigated three approaches in which each robot's next action selection is either a greedy choice based upon the expected execution time of the tasks it is able to perform, or is a random choice of actions. The following paragraphs describe these three task ordering approaches, called Longest Task First, Modified Shortest Task First, and Modified Random Task Selection.

---

<sup>1</sup>As anecdotal evidence of this problem, at a recent AAAI robot competition [16, 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.

### 7.2.1 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 [25] for identical processors. In this approach, the tasks are assigned to processors in order of non-increasing task length. Thus, we first attempted a distributed version of Descending First Fit to determine its effectiveness for the multi-robot application domain. The distributed version, which we 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 fast and slow impatience parameters of each motivational behavior ( $\delta_{fast_{ij}}(t)$  and  $\delta_{slow_{ij}}(k, t)$ ) 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.

### 7.2.2 Modified Shortest Task First

As a logical next step, we 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; in other words, the average completion time of the tasks in the mission is minimized [15]. However, in our approach, the pure Shortest Task First technique is modified somewhat 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 specific 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, we 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, we 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 quicker than all other

robots present on the team.

2. All other tasks  $r_i$  can perform.

This two-category mechanism is implemented via the *learned\_robot\_influence* function defined in the formal L-ALLIANCE model in section 7.5, which initially “blinds” the robot to those tasks in the second category. This causes the robot to first select from among those actions that it feels it can perform quicker 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, the robot does not idle indefinitely just because other team members might possibly be able to accomplish the tasks in the second category. Instead, each robot is 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 the settings of two parameters in L-ALLIANCE:  $\delta_{slow_{ij}}(k, t)$  and  $\delta_{fast_{ij}}(t)$ . To cause a robot to select the task it expects to perform the quickest, these rates of impatience for each behavior set grow at a rate inversely proportional to the expected task completion time. Section 7.5 discusses the details of how this is implemented.

### 7.2.3 Modified Random Task Selection

As a baseline against which to compare the other approaches, a random selection of tasks was also studied. 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.

## 7.3 EXPERIMENTAL RESULTS OF L-ALLIANCE CONTROL STRATEGIES

To determine the relative merits of the three strategies presented in the previous section, we executed large number of test runs in simulation<sup>2</sup>, comparing the results of the strategies in terms of the time required to complete the mission. In making observations about the relative performances of the strategies, it is important to not generalize conclusions based on too few examples, since the outcome of any specific

---

<sup>2</sup>In this study, simulation runs offered much more opportunity to study the effects of a large 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. We then validated the results on physical robots, as described in section 8.8.

example can often be quite different from the average performance of the strategies over a range of similar mission scenarios. We thus 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 degree of heterogeneity from 0 percent to 3200 percent. For this study, the missions were composed of completely independent subtasks involving no ordering constraints, the capabilities were distributed uniformly across the robots based upon the given task coverage, and the same task coverage was assumed for all tasks required by the mission. Note that although this study did not address the issue of robot failure explicitly, the strategies studied here do not distinguish between task failure in robots and slower completion times in those robots. Thus, since task failure is treated no differently from gross inefficiency, robot failures are implicitly included in the heterogeneity difference across robots.

For ease of discussion in this subsection, we define a *scenario* as a 4-tuple ( $n$ ,  $m$ , *task\_coverage*, *heterogeneity*) of a given run of the simulation. For each scenario defining the number of robots, the size of the mission, the level of task coverage, and the percent of heterogeneity, two hundred different test runs were executed, 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.

To clarify the discussion, we discuss separately the effects of the impatience/acquiescence update strategy and the effects of the task ordering approach. The following two subsections present and discuss the results of these studies.

### 7.3.1 Effect of Impatience/Acquiescence Update Strategy

In order to separate the results of the impatience/acquiescence update strategy from those of the task ordering approach, we assume in this subsection that the robot team uses the Modified Shortest Task First approach to task ordering, and present the comparison of the impatience/acquiescence results under this assumption. The following subsection discusses the results when we relax this assumption.

We first note that the three impatience/acquiescence update strategies are equivalent for teams in which the degree of heterogeneity is 0, regardless of any other factors, because we 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 in L-ALLIANCE 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 time usage, as shown in figure 3: 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:

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

In figure 3, 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 indicates the best performer(s). As described in the previous section, strategy I (section 7.1.1) is “Distrust Performance Knowledge about Teammates”, strategy II (section 7.1.2) is “Let the Best Robot Win”, and strategy III (section 7.1.3) is “Give Robots a Fighting Chance”. In figure 3, when more than one set of values is given (in regions 2 and 3), the relative performances depend upon the *Progress When Working* condition. The four points noted with small black squares are exemplar missions of their corresponding regions, whose time usages are shown in later figures. 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 used in the exemplar.

Intuitively, region 1 corresponds to those scenarios in which many robots are available 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 perform tasks. As we progress to regions 2, 3, and 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, consider the relative performances of teams controlled by the three strategies in region 1. Figure 4 illustrates a typical performance of the three strategies for scenarios in this region, showing the time 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, a fixed tie-breaking mechanism determines which robot wins, regardless of their 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

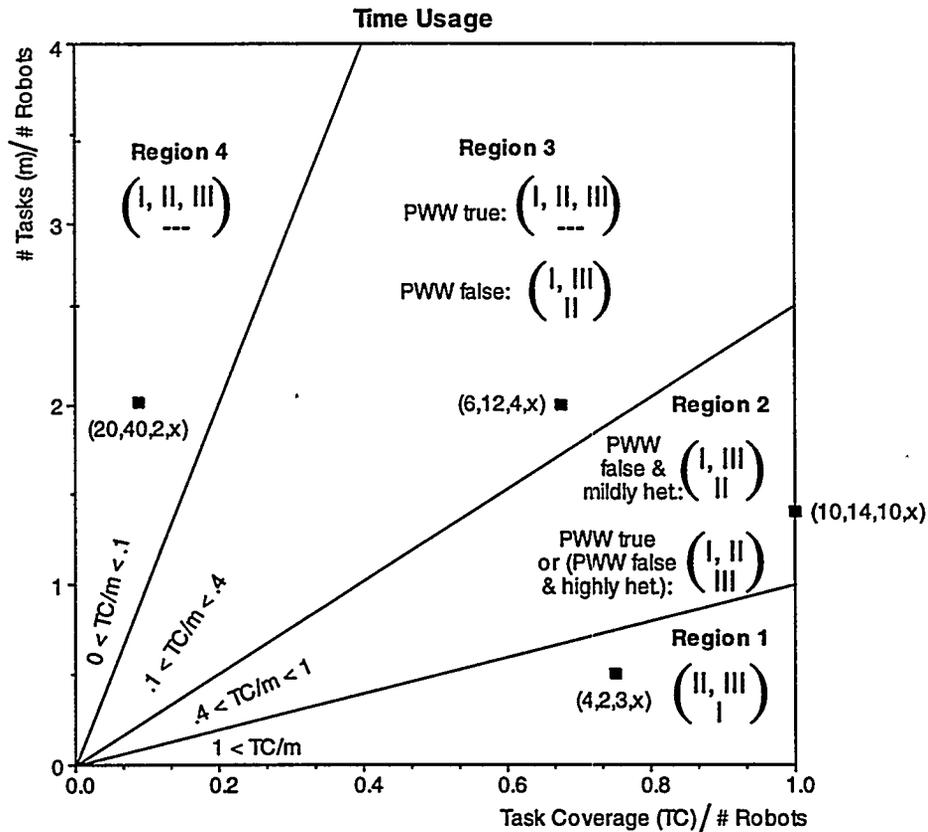


Figure 3: Summary of time usage for three impatience/acquiescence strategies. In this figure, the strategy numbers (I, II, III) in large parentheses indicate the relative performance of the three strategies in each of the regions, where the first row indicates the best performer(s). When more than one set of values are given (in regions 2 and 3), the relative performances depend upon the *Progress When Working* (PWW) condition. The four points noted with small black squares are exemplar missions of their corresponding regions, whose time usages are shown in later figures. 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 used in the exemplar.

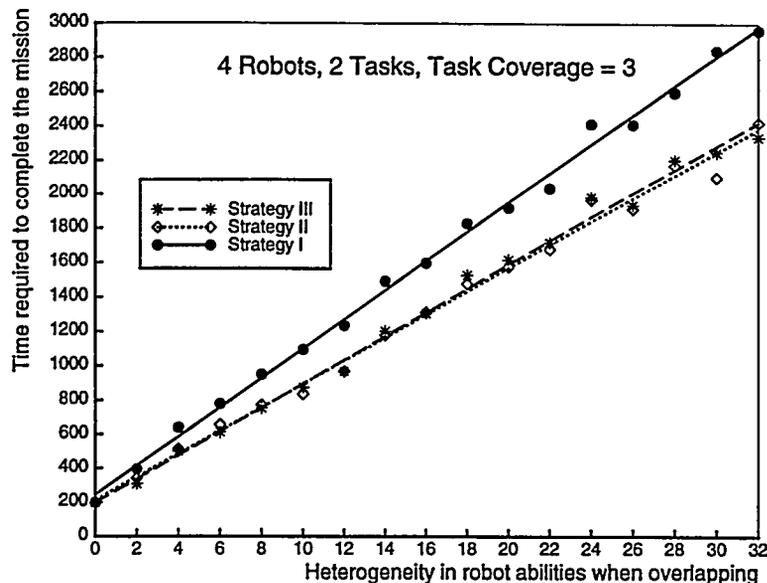


Figure 4: An average time performance of the the three impatience/acquiescence strategies in region 1. Each data point shown in this and in the next three figures is an average value over two hundred runs of the corresponding scenario. Refer to the text for more details.

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 is 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 time.

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 is 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. This in turn means that at times idle robots will be watching to override their teammates' performances. We discover 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  $a_3$ . 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  $a_2$ . 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 effect does this have on the relative time 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. When the *Progress When Working* condition holds, a robot is able to fully sense the effect of other robot’s actions through the world. Thus, strategy II outperforms strategy III in terms of time for any degree of heterogeneity (except full homogeneity) because a quicker robot takes 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 are not 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 is able to perform the entire task in less time than the slower robot needs 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 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. Figure 5 shows a typical performance of the strategies in region 2 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.

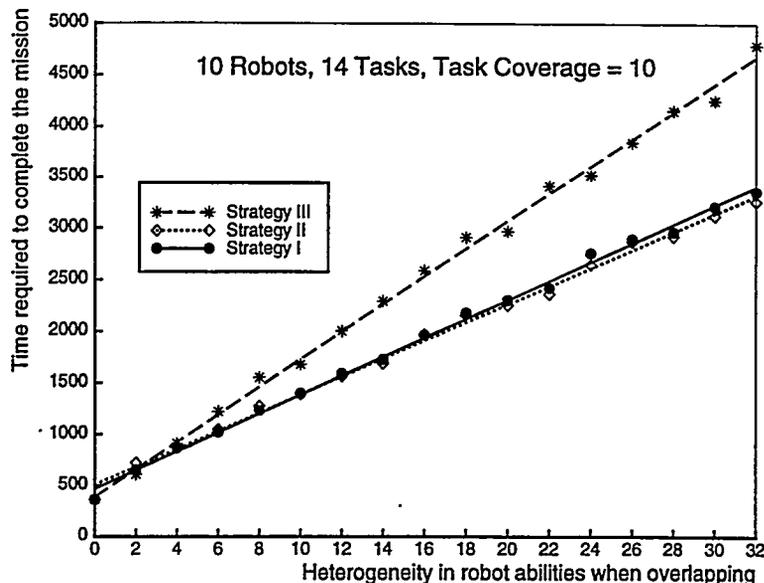


Figure 5: An average time performance of the three impatience/acquiescence strategies in region 2 when the *Progress When Working* condition is not true.

Moving into region 3 results in scenarios involving a fairly low ratio of task coverage to mission size. Figure 6 illustrates 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 significant amount of unfinished work remaining to be accomplished. However, since strategies I and II under the Modified Shortest Task First approach 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 overriding robot need not repeat the entire task. However, when the *Progress When Working* condition is false, strategy II is somewhat 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. 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. We 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.

Finally, we 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

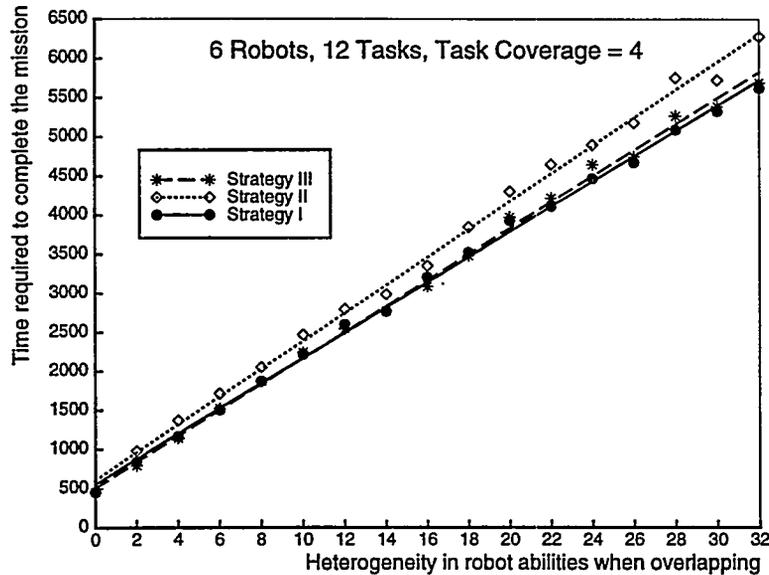


Figure 6: An average time performance of the three impatience/acquiescence strategies in region 3, assuming the *Progress When Working* condition does not hold.

choice of impatience/acquiescence update strategy makes little 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. Figure 7 shows a typical time performance of the strategies for scenarios in region 4.

In summary, we see that the performance of the three impatience/acquiescence update strategies under the Modified Shortest Task First approach 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. Table 3 summarizes these results by giving the preferred strategy for each combination of these factors. What we find is that all three strategies perform well in certain scenarios, while they perform poorly in others. 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 worse 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 slow to override.

### 7.3.2 Effect of Task Ordering Approach

As discussed in section 7.2, we investigated three approaches for allowing a robot to determine which task to select from those tasks that are not already being attempted

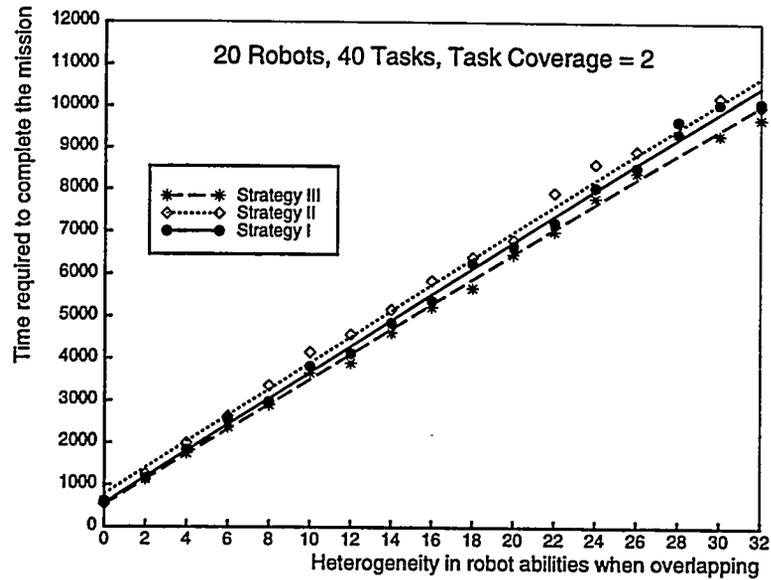


Figure 7: An average time performance of the three impatience/acquiescence strategies in region 4.

### Preferred Strategy: Time as Metric

Progress When Working?	Region	Heterogeneity	Strategy
Yes	1	—	II or III
Yes	2	—	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 3: 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%.

by any robot — the Longest Task First approach, the Modified Shortest Task First approach, and the Modified Random Task Selection approach. We quickly dismissed the distributed Longest Task First approach, because it 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 1 (see section ), 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. Clearly, this does not result in collectively efficient task execution.

We then compared the relative performance of the Modified Shortest Task First approach with the Modified Random Task Selection approach. If a simple random selection of the next task performs just as well as the shortest task first approach, then the control strategy would be less dependent upon knowledge of other robot capabilities.

To investigate these alternatives, we performed a similar set of simulation experiments as discussed in the previous subsection, varying the number of robots, the size of the mission, and the heterogeneity of the robots. In these experiments, we studied the relative effects of the Modified Random Task Selection approach under each of the three impatience/acquiescence strategies as compared to the results of the the impatience/acquiescence update strategies under the Modified Shortest Task First approach.

Figure 8 shows a typical outcome of this comparison in terms of time; this example shows the percent change in mission completion time when using a random task choice instead of shortest task first for a six-robot team with twelve tasks and a task coverage of four. 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 discussed earlier. The Longest Task First approach theoretically results 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, we dismissed this approach for heterogeneous robot teams which can perform the same tasks with different qualities, since it caused each task to be pursued by the robot with the longest task completion time. However, if we allow robots to effectively divide their tasks into two categories — (1) those which the robot expects to be able to perform quicker than any other robot, and (2) all remaining tasks that robot can perform — and then use a longest task first mechanism to select among the first category and a shortest task first mechanism for selecting among the second category tasks, the problem of heterogeneity is circumvented.

What we find is that the Random Task Selection approach for impa-

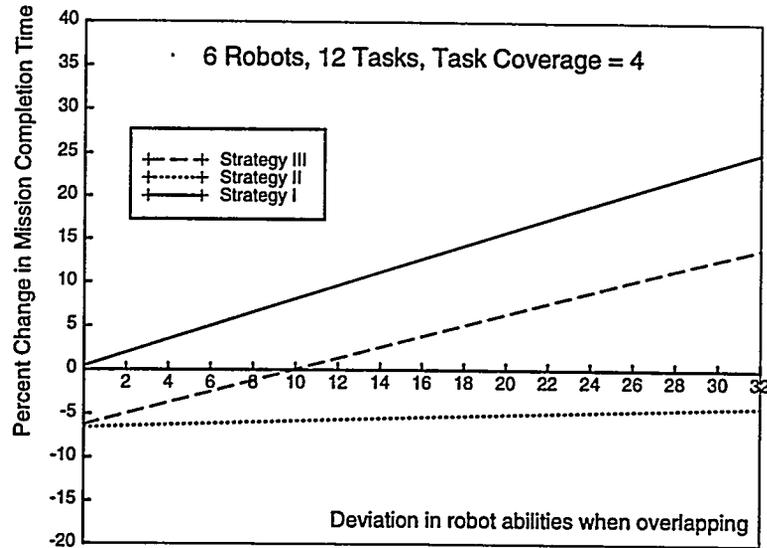


Figure 8: Typical change in mission completion time when using random task selection instead of shortest task first selection.

tience/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, robots controlled using strategy III display poorer performance when using the Random Task Selection approach, compared to the Shortest Task First 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.

## 7.4 THE PREFERRED L-ALLIANCE DISTRIBUTED CONTROL STRATEGY FOR EFFICIENCY AND FAULT TOLERANCE

The results of the control strategy investigations lead to a preferred strategy under which each robot  $r_i$  effectively does the following:

1. 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.
2. Repeat the following until sensory feedback indicates that 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 the first category.
  - (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.

Note here that the actual implementation of this control strategy in L-ALLIANCE is distributed within each robot across the monitors and motivational behaviors. Although the L-ALLIANCE approach could be implemented on each robot as a centralized controlling behavior, doing so would compromise the robustness and fault tolerant design goals of the architecture. A centralized process responsible for obtaining all of the performance quality measurements of robot team members 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, the centralized decision-maker would have too much work to do to effectively control the robot, leading to the classical problems of centralized *sense-plan-act* robot control architectures [8]. Distributing the control mechanism in ALLIANCE and L-ALLIANCE makes it rather straightforward 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 a single software module growing arbitrarily large to handle increased mission sizes.

Of course, distributing the knowledge across many motivational behaviors does 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

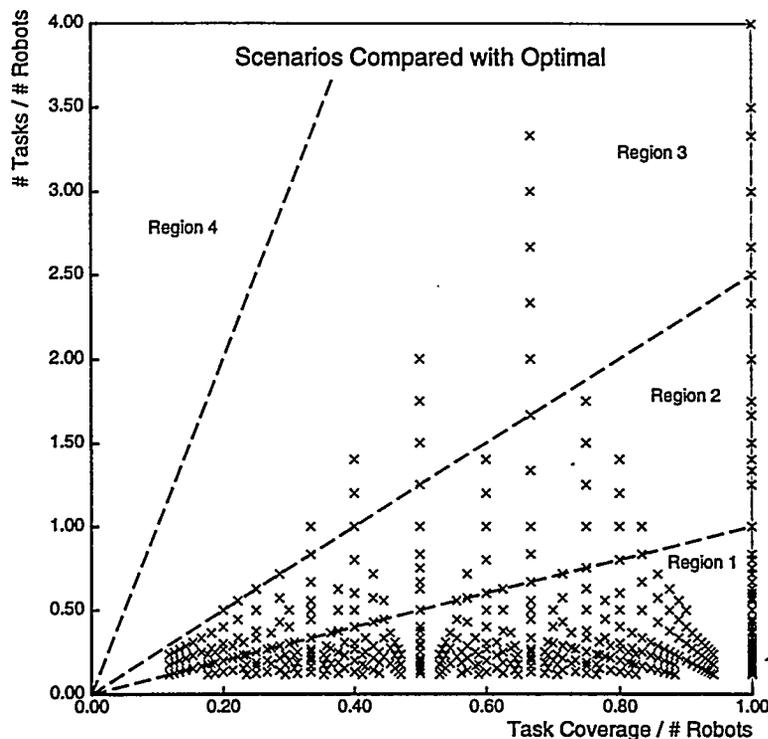


Figure 9: The data points shown correspond to scenarios for which the optimal result could be computed. For each of these scenarios, we compared the time usage required for recommended distributed action selection technique against the required time usage for the optimal result. The data points correspond to a total of 496 scenarios. The dashed lines indicate the same four regions as shown in figure 3.

so that all tasks become complete? The interaction of the motivational behaviors on an individual robot must be designed to allow the collective interaction of the team members' actions to result in the most efficient execution of the mission possible. The formal model providing the details of this distribution is described in section 7.5.

## 7.5 COMPARISON TO THE OPTIMAL SOLUTION

In future work, we plan to analytically study the preferred control strategy described above to determine its best-case, worst-case, and average-case performance expectations relative to the theoretical optimal solution. However, we note that the emphasis in this article is not on deriving the ultimate distributed task scheduler; rather, the goal is the development of an infrastructure that allows distributed robots to demonstrate a high level of fault tolerance while improving their collective efficiency, and to illustrate an adaptive mechanism that achieves this aim. It should be clear that other greedy approaches to dynamic control parameter updates could also be incorporated into the L-ALLIANCE infrastructure if they are shown to be superior to the preferred L-ALLIANCE approach developed above.

Nevertheless, it is important to confirm experimentally that the preferred up-

date mechanism works well in practice when compared to the optimal result. Of course, 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 impractical 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. We thus experimentally compared the results of the preferred L-ALLIANCE control strategy with the optimal solution for those problems in which we could derive the optimal result.

Figure 9 plots the mission scenarios for which we could calculate the optimal solution; these scenarios are plotted according to their *task\_coverage/m* ratio. A large number of experimental runs were then executed in simulation using the same technique as described in the previous section, computing the average percent worse of the L-ALLIANCE solution over the optimal solution for each mission scenario. Figure 10 shows the results, indicating the percent worse than the optimal result for the scenarios in regions 1, 2, and 3. 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, with much better performance in region 1.

The issue, of course, is how seriously we should expect the performance of L-ALLIANCE to degrade as the size of the problem increases. The preferred L-ALLIANCE strategy 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. As we mentioned earlier, quantifying how much worse this preferred control strategy can be than the optimal is a primary topic of future study.

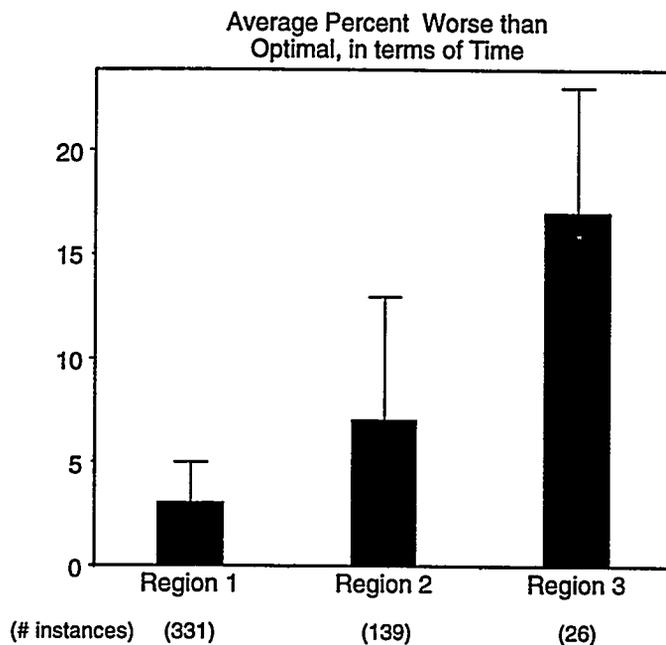


Figure 10: Comparison of the preferred control strategy performance with the optimal performance. In region 1, the averages are over 331 scenarios, in region 2 the averages are over 139 scenarios, and in region 3 the averages are over 26 scenarios. Since the efficiency problem is exponential in the number of tasks, the optimal results could not be readily derived for any scenarios in region 4. The error bars indicate one standard deviation in the performance differences.



## 8 L-ALLIANCE FORMAL MODEL

This section presents the formal model defining how the L-ALLIANCE dynamic parameter update mechanism is incorporated into the motivational behaviors such that the fault tolerant characteristics of ALLIANCE are retained while adding efficiency improvements. We first discuss 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. We conclude this section by showing how the L-ALLIANCE inputs are combined to compute the motivational levels.

### 8.1 THRESHOLD OF ACTIVATION

A parameter of key importance to the efficiency of the robot team is the threshold of activation,  $\theta$ . 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 7.4 that the interaction of motivational behaviors must 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 fast and slow impatience rates ( $\delta_{fast_{ij}}(t)$  and  $\delta_{slow_{ij}}(k, t)$ ) 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, the robot team member that is superior at a given task should "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 that the value of  $\theta$  to be uniform across robots and across the motivational behaviors of each robot.

### 8.2 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 its sensors. Often, tasks are information-gathering types of activities whose need is indicated by the values of programmed state variables. The use of stored state in memory, therefore, can serve 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 in-

formation to the robot (e.g. performing a task in inclement weather outdoors). 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 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 mechanical locomotion errors can lead to the team’s failure to accomplish its mission.

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

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

Note that this use of sensory feedback serves the same purpose as “precondition lists” in traditional planning systems, such as STRIPS [22], or in situated agent planning systems, such as Maes’ spreading activation networks [33]. 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 L-ALLIANCE to make the environmental requirements of behavior set activation more explicit.

### 8.3 INTER-ROBOT COMMUNICATION

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

Two parameters are utilized in L-ALLIANCE to control the broadcast communication among robots:  $\rho_i$  and  $\tau_i$ . The first parameter,  $\rho_i$ , gives the rate at which robot  $r_i$  broadcasts its current activity. The second parameter,  $\tau_i$ , 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 monitor  $mon_{ij}$  of robot  $r_i$  must also note when a team member is pursuing task  $h_i(a_{ij})$ . To refer to this type of monitoring in the formal model, the function *comm\_received* is defined as follows:

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

The rate at which robots communicate their current actions to their teammates is of central importance in 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. (See [45] for a further discussion of this issue.) Thus, to ensure maximal efficiency, the communication rates,  $\rho_i$ , are 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  $\tau_i$ . 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 robots should adapt their own actions according to the current and expected actions of their teammates, they should not 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 L-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  $\tau_i$ , then it must assume that that teammate is no longer available to perform tasks in the mission.

The proper value of  $\tau_i$  is dependent upon each robot team member's  $\rho_i$  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 is minor if the  $\tau_i$  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, we 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  $\tau_i$  time units.

## 8.4 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<sup>3</sup>. 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. We ignore this implementation detail here, however, and simply refer to the cross-behavior set suppression with the following function:

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

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  $t$ .

## 8.5 LEARNED ROBOT INFLUENCE

When a robot is operating in the active learning phase, 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  $r_i$  considers selecting in the active learning phase is determined by the following function:

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

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, it selects its actions based upon the

---

<sup>3</sup>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.

knowledge learned about its own and other robot capabilities by using the control strategy described in section 7.4.

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.

In terms of the formal model, we refer to this learned influence by the following definitions. First, we define  $\mu$  to be the number of trials over which robot maintains task performance averages and standard deviations. As stated earlier, the value of  $\mu$  is fairly small; in these experiments, maintaining information over about 5 trials provided good results. We then define the function:

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

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 in the next subsection, 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 divided into the two categories described in subsection 7.4 according to the following function:

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

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  $\tau_i$  time units which indicates that  $r_x$  is currently performing task  $h_i(a_{ij})$ . Otherwise, the task belongs to category 2.

Next, we 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,  $boredom\_rate_i$ , the boredom function is defined as follows:

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

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  $boredom\_rate_i$ .

We 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.

## 8.6 ROBOT IMPATIENCE

The primary robot impatience parameter is  $\phi_{ij}(k, t)$ , which gives the time that robot  $r_i$  is willing to allow  $r_k$ 's communication message to affect the motivation of behavior set  $a_{ij}$ . This value in L-ALLIANCE varies during the mission based on the robot's experience. The value of  $\phi_{ij}(k, t)$  is set according to the selected impatience/acquiescence update strategy. The results presented earlier indicate that the most efficient global action selections can be obtained by dynamically updating the value of  $\phi_{ij}(k, t)$  as follows:

- For mildly heterogeneous teams in which Condition 2 (*Progress When Working*) does not hold,  $\phi_{ij}(k, t)$  is 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 impatience/acquiescence update strategy III — “Give Robots a Fighting Chance”).
- 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 impatience/acquiescence update strategy II — “Let the Best Robot Win”).

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  $\delta\_fast_{ij}(t)$ ). The slow rate of impatience,  $\delta\_slow_{ij}(k, t)$ , 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, while the fast rate of impatience,  $\delta\_fast_{ij}(t)$ , 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  $h_i(a_{ij})$ . These parameters are set to cause the motivational behaviors to interact in such a way that each robot selects tasks from the first task category (see again section 7.4) 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  $\delta\_slow_{ij}(k, t)$  grows 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 automatically updated during the mission 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 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, note that during the time in which the  $\delta\_fast_{ij}(t)$  parameters are below the threshold  $\theta$ , 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 nearly 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 reduces the idle time and maintains the relative ordering among the tasks, it does not place an upper bound on how long a robot might remain idle during its mission. A preferred way of scaling

the idle times is to map them to some acceptable range based upon expected task completion time. To do this, we 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. The ideal method of scaling the rates within this range requires the motivational behaviors to ascertain the global minimum and maximum expected task completion times across all tasks of the mission, since this allows the rates of impatience for a given task to remain calibrated across robots. We approximate these global minimum and maximum task completion times with the minimum and maximum task completion times known within a given robot. With these values, the proper settings of the  $\delta_{fast_{ij}}(t)$  parameters are given as follows:

Let:

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

Then:

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

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*.

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

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

Thus, the impatience rate is 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  $\tau_i$  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:

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

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.

## 8.7 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 7.1, 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 2 (*Progress When Working*) does not hold,  $\psi_{ij}(t)$  is 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 impatience/acquiescence update strategy III — “Give Robots a Fighting Chance”).
- Otherwise,  $\psi_{ij}(t)$  is 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 impatience/acquiescence update strategy II — “Let the Best Robot Win”).

The value of the  $\lambda_{ij}(t)$  parameter is 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.

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

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

This function says that a robot  $r_i$  does 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  $\lambda_{ij}(t)$

## 8.8 MOTIVATION CALCULATION

All of the robot inputs are 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  $\theta$ , 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\_impactience*).

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

DURING ADAPTIVE PHASE:

$$\begin{aligned} m_{ij}(0) &= 0 \\ m_{ij}(t) &= [m_{ij}(t-1) + impatience_{ij}(t)] \end{aligned}$$

- ×  $sensory\_feedback_{ij}(t)$
- ×  $activity\_suppression_{ij}(t)$
- ×  $impatience\_reset_{ij}(t)$
- ×  $acquiescence_{ij}(t)$
- ×  $learned\_robot\_influence_{ij}(t)$

Robot  $r_i$ 's motivation to perform any given task during the adaptive phase thus increments at a fast or slow impatience rate (based upon the activities of other robots) until it crosses the threshold  $\theta$ , 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  $\rho_i$ .



## 9 IMPLEMENTATION ON MOBILE ROBOTS

The ALLIANCE and L-ALLIANCE architectures have 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 a mockup hazardous waste cleanup mission 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 article, we present the results of the box pushing demonstration, which provides a very simple example of the adaptive, fault tolerant, and efficient characteristics that can be achieved using the L-ALLIANCE architecture. Refer to [43, 41] for details on the other applications.

All of these missions using the ALLIANCE and L-ALLIANCE architectures 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 (see [42] for a sampling of these videotaped experiments). The missions implemented on simulated robots encompass thousands of runs each, most of which were logged in the study of the adaptive action selection mechanism (see section 7.3).

### 9.1 THE POOL OF HETEROGENEOUS ROBOTS

The proof of concept experiments on physical robots were conducted using the robots shown in figure 11. This pool of heterogeneous robots consisted of two types of mobile robots — three R-2s and one Genghis-II — all of which were designed and built by IS Robotics Corporation.

The R-2 robot has two drive wheels arranged as a differential pair, and a two-degree-of-freedom gripper for grasping objects. Its sensor suite includes eight infrared sensors and seven bump sensors evenly distributed around the front, sides, and back of the robot. In addition, a break-beam infrared sensor between the gripper and a bump sensor lining the inside of the fingers facilitate the grasping of small objects. The second type of robot, Genghis-II, is a legged robot with six two-degree-of-freedom legs. Its sensor suite includes two whiskers, force detectors on each leg, a passive array of infrared heat sensors, three tactile sensors along the robot belly, four near-infrared sensors, and an inclinometer for measuring the pitch of the robot.

A radio communication system [30] was used in our physical robot implementations to allow the robots to communicate their current actions to each other. This system consists of a radio modem attached to each robot, plus a base station that is responsible for preventing message interference by time-slicing the radio channel among robots. The design of the radio system limits the frequency of messages between robots to only one message every three seconds. All of these implementations, therefore, involve communication between robots at no more than about  $\frac{1}{3}$  Hertz. All of the robots used in these experiments were programmed using the Behavior Language [10].

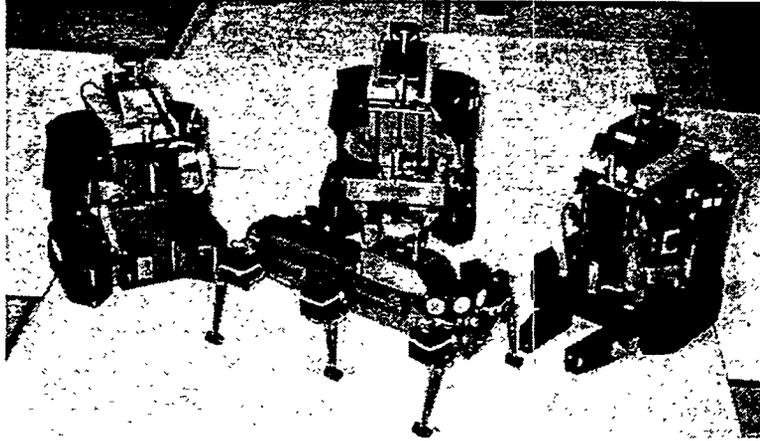


Figure 11: The pool of heterogeneous robots — three R-2s and one Genghis-II.

## 9.2 THE BOX PUSHING DEMONSTRATION

The cooperative box pushing demonstration offers a simple and straight-forward illustration of a key characteristic of the L-ALLIANCE architecture: fault-tolerant and adaptive control due to dynamic changes in the robot team. 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. This demonstration was implemented using a heterogeneous robot team of two R-2s and Genghis-II, and illustrates how the L-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 to increase team efficiency. Note that the emphasis in these experiments is on issues of fault-tolerant and efficient cooperation rather than the design of the ultimate box pusher. Thus, we are 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. Donald et al. [18] use a box pushing demonstration to investigate 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 defined here, since it can accomplish one type of fault-tolerant cooperation that L-ALLIANCE allows

below in experiment 1 — namely, the ability to recover from a failed team member<sup>4</sup>. 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. The Protocol II control strategy, however, is specific to box pushing, and does not address the general fault tolerant and efficient action selection problem that is addressed with L-ALLIANCE.

In [39], Noreils describes a cooperative box pushing experiment in which one robot acts as the *pusher* to push a box against the wall, and a second 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 mentioned by Noreils in [39]. For instance, it is unclear in their 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 [31] 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 their 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 [4], 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 and efficiency are not addressed in their approach.

---

<sup>4</sup>This type of fault tolerance can only be obtained with the "uniform" version of Donald's protocol II, rather than the "almost uniform" version.

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

## 9.3 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. Thus, the L-ALLIANCE box pushing software of these robots is discussed separately.

### 9.3.1 R-2 Control

Figure 12 shows the L-ALLIANCE implementation of the box pushing demonstration for the R-2 robots. (For the sake of clarity, the monitors are not shown in this figure.) 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 L-ALLIANCE architecture, the activation of each of these behavior sets is controlled by a motivational behavior.

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 12 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.

The *push-right* design is symmetric to that of *push-left*.

When the sensory feedback is satisfied, the *push-left* motivational behavior grows impatient at either a rate  $\delta\_fast_R(t)$  (the  $R$  subscript stands for an R-2 robot) if no other robot is performing the *push-left* task, or at a rate  $\delta\_slow_R(robot - id, t)$  when robot  $robot-id$  is performing the *push-left* task. 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 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

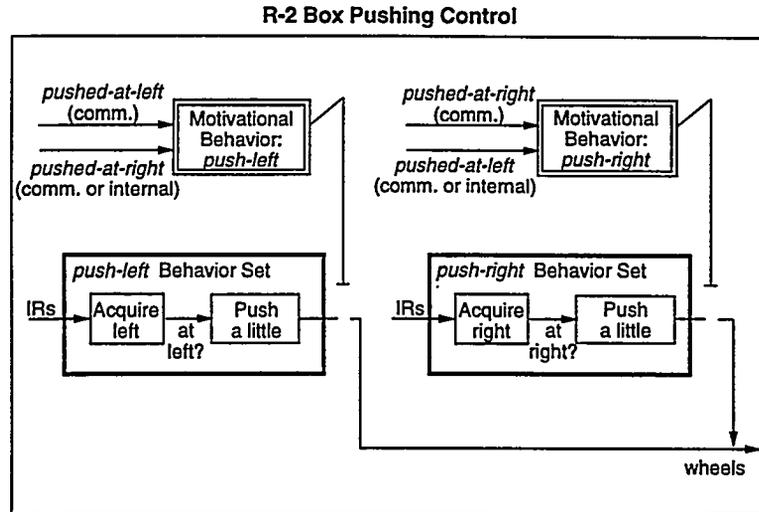


Figure 12: The L-ALLIANCE design of the R-2 software for the box pushing demonstration. (The two L-ALLIANCE monitors are not shown here for the sake of clarity.)

is important in achieving fault tolerant cooperative control. At the beginning of the demonstration, we 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.

### 9.3.2 Genghis-II Control

Genghis-II and the R-2s differ 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. 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 13 shows the organization of Genghis-II's box pushing software. (Again, for the sake of clarity, the two monitors are not shown in this figure.) 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

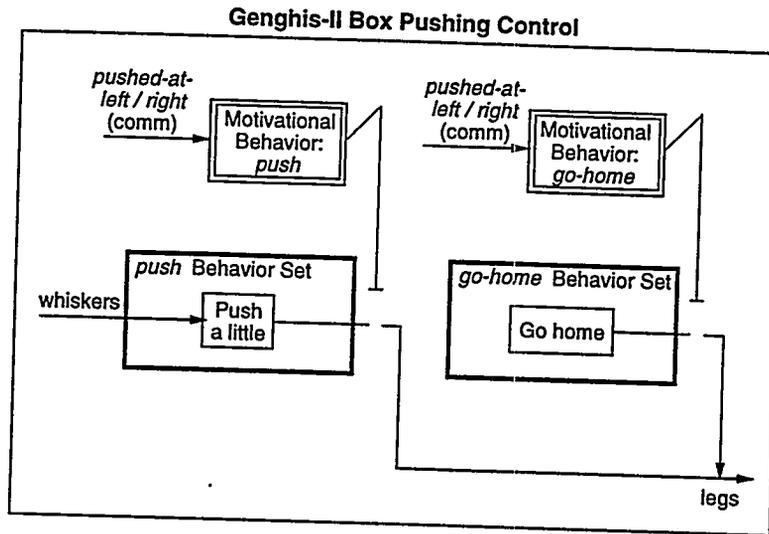


Figure 13: The L-ALLIANCE design of the Genghis-II software for the box pushing demonstration. (The two L-ALLIANCE monitors are not shown here for the sake of clarity.)

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 13 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  $\delta\_fast_{GP}$  (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  $\delta\_fast_{GH}$  (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.

## 9.4 ROBOT EXPERIMENTATION RESULTS

During the active learning phase of the box pushing demonstration, we allowed each robot to “practice” pushing an end of the box with different teammates. Each R-2 robot was teamed either with one other R-2 robot, or with the Genghis-II robot. The Genghis-II robot was teamed with each of two R-2 robots during separate trial .

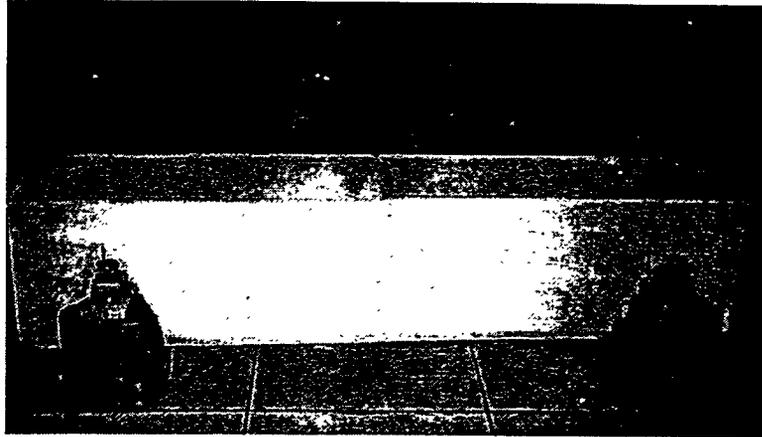


Figure 14: The beginning of the box pushing demonstration. Two R-2s are pushing the box across the room.

periods. During these active learning phases, each robot learned the period of time required for its teammate to push its end of the box by monitoring the broadcast radio communication messages, in which each robot announced when it began the “push a little” task, and when it finished it. These task completion times were automatically fed into the dynamic parameter tuning mechanism described in the previous section to update the  $\delta_{slow}(robot - id, t)$  parameters within each robot.

We then placed the robots in adaptive learning mode and undertook two basic experiments to provide a simple illustration of the fault tolerant, adaptive nature of the robot team under the L-ALLIANCE architecture. Both of these experiments began with two R-2s pushing the box — one at each end of the box — as illustrated in figure 14.

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

#### 9.4.1 Experiment 1: Robot “failure”

As emphasized earlier, a primary goal of the L-ALLIANCE architecture is to allow robots to recover from failures of robot team members. Thus, by seizing an R-2 and turning it off, we 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, the remaining R-2 begins acquiring the opposite end of the box, as shown in figure 15, 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

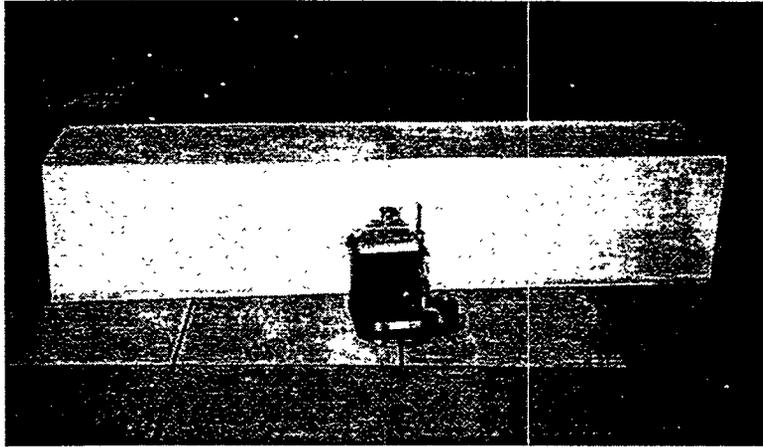


Figure 15: Fault tolerant action selection. In this first experiment, one of the R-2 robots is seized and turned off. This causes the remaining R-2 robot to have to perform both tasks of the box pushing demonstration: pushing at the right end of the box, and pushing at the left end of the box.

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 the second R-2 is returned to the team, 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.

#### 9.4.2 Experiment 2: Increased heterogeneity

Another goal of the L-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, we 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, we 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, due to the simple learning phase of L-ALLIANCE, the remaining R-2 begins pushing much less frequently as soon as it “hears” that Genghis-II, rather than an R-2, is the robot pushing the opposite end of the box. Thus, the robots remain more or less aligned during their pushing. Figure 16 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 learned  $\delta_{slow_R}(R-2)$  and  $\delta_{slow_R}(Genghis-II)$  parameters differ quite a bit since Genghis-II is much slower at pushing the box than

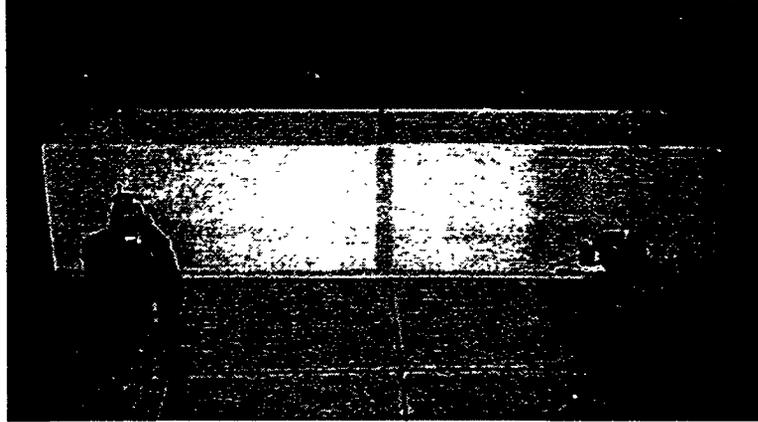


Figure 16: Adaptivity due to heterogeneity. In this second experiment, we again seize one of the R-2 robots, but this time we replace it with Genghis-II. Since Genghis-II cannot push as powerfully as an R-2, the remaining R-2 robot adapts its actions by pushing less frequently.

the R-2. These parameter differences were easily learned by these robots monitoring the performance of their teammates. In this case, the R-2s learn parameters in which  $\delta_{slow_R}(\text{Genghis-II})$  is less than  $\delta_{slow_R}(\text{R-2})$ .

While the R-2 was pushing on the left of the box, Genghis-II was 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,  $\delta_{slow_R}(\text{Genghis-II})$ , the motivation to activate the *push-right* 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.

We complete these experiments by removing the remaining R-2 from the team. This causes Genghis-II to activate its *go-home* behavior, as shown in figure 17. Thus, Genghis-II also demonstrates its adaptive action selection due to the actions and failures of robot team members.

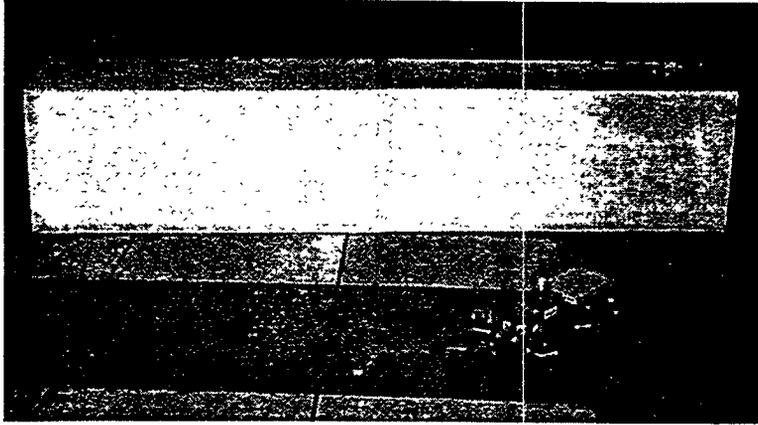


Figure 17: Response to robot failure. At the end of the second experiment, we seize the remaining R-2 robot, leaving Genghis-II alone to perform the demonstration. Since Genghis-II cannot complete the demonstration on its own, it activates its *go-home* behavior set.

## 10 CONCLUSIONS

This article has presented the L-ALLIANCE adaptive cooperative control mechanism that provides the ability for a team of robots to dynamically update control parameters during a mission to respond to changes in the environment or in the robot team. We presented a brief overview of the ALLIANCE architecture upon which L-ALLIANCE is built, along with the motivations for needing efficiency improvements in cooperative team performance. After showing that the efficiency problem is intractable, we discussed a number of alternative control approaches to the dynamic update of control parameters, and discussed the results of a comparison of the strategies in simulation. From these studies, we developed a preferred control strategy that was shown to work well in practice. A formal model of this update strategy was presented. We then presented the results of a simple proof of concept demonstration of L-ALLIANCE using a team of two types of robots performing a box pushing task, illustrating the ability of L-ALLIANCE to achieve efficient control while maintaining the fault tolerant characteristics of ALLIANCE.

The L-ALLIANCE adaptive control parameter update mechanism is important for a number of reasons: it alleviates the need for human tuning of robot control parameters, it facilitates the use of custom-designed multi-robot teams for any given application, it improves the efficiency of the mission performance, and it allows robots to continually adapt their performance over time due to changes in the robot team and/or the environment. In future work, we plan to analytically study the preferred L-ALLIANCE control approach to determine its theoretical performance relative to the optimum result.



## 11 ACKNOWLEDGEMENTS

The author wishes to thank Prof. Rodney A. Brooks of the Massachusetts Institute of Technology's Artificial Intelligence Laboratory, who supervised this research, and the IS Robotics Corporation for building the robots.

Support for this research was provided in part by the University Research Initiative under Office of Naval Research contract N00014-86-K-0685, in part by the Advanced Research Projects Agency under Office of Naval Research contract N00014-85-K-0124, and in part by the Mazda Corporation. Additional support has been provided by the Office of Engineering Research Program, Basic Energy Sciences, of the U.S. Department of Energy, under contract No. DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.



## References

- [1] Ronald C. Arkin. Integrating behavioral, perceptual, and world knowledge in reactive navigation. *Robotics and Autonomous Systems*, 6:105–122, 1990.
- [2] 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.
- [3] M. Asada, E. Uchibe, S. Noda, S. Tawaratsumida, and K. Hosoda. Coordination of multiple behaviors acquired by a vision-based reinforcement learning. In *Proceedings of IEEE RSJGI International Conference on Intelligent Robots and Systems*, pages 917–924, Munich, Germany, 1994.
- [4] 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.
- [5] Gerardo Beni and Jing Wang. On cyclic cellular robotic systems. In *Japan – USA Symposium on Flexible Automation*, pages 1077–1083, Kyoto, Japan, 1990.
- [6] 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.
- [7] Alan Bond and Less Gasser. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann, 1988.
- [8] 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.
- [9] Rodney A. Brooks. A robot that walks: Emergent behavior from a carefully evolved network. *Neural Computation*, 1(2):253–262, 1989.
- [10] Rodney A. Brooks. The behavior language: User’s guide. Memo 1227, MIT A.I. Lab, Cambridge, MA, April 1990.
- [11] Rodney A. Brooks. Elephants don’t play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.
- [12] 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.
- [13] 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.

- [14] Jonathan Connell. A colony architecture for an artificial creature. Technical Report AI-TR-1151, MIT, Cambridge, MA, 1989.
- [15] R. W. Conway, W. L. Maxwell, and L. W. Miller. *Theory of Scheduling*. Addison-Wesley, Reading, Massachusetts, 1967.
- [16] Thomas Dean and R. Peter Bonasso. The 1992 AAAI robot exhibition and competition. *AI Magazine*, 14(1):34–48, Spring 1993.
- [17] 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.
- [18] 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.
- [19] 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.
- [20] Gregory Dudek et al. A taxonomy for swarm robots. In *Proceedings of 1993 IEEE International Conference on Intelligent Robots and Systems (IROS '93)*, pages 441–447, 1993.
- [21] Cynthia Ferrell. Robust agent control of an autonomous robot with many sensors and actuators. Master's thesis, Massachusetts Institute of Technology, 1993.
- [22] 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.
- [23] 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.
- [24] 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.
- [25] Michael R. Garey and David S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman and Company, 1979.
- [26] E. Gat, R. Desai, R. Ivlev, J. Loch, and D. Miller. Behavior control for robotic exploration of planetary surfaces. *IEEE Transactions on Robotics and Automation*, 10(4):490–503, August 1994.

- [27] 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.
- [28] 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.
- [29] Ian Horswill. *Specialization of Perceptual Processes*. PhD thesis, Massachusetts Institute of Technology, 1993.
- [30] IS Robotics, Inc., Somerville, Massachusetts. *ISR Radio Communication and Positioning System*, October 1993.
- [31] 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.
- [32] Masao Kubo and Yukinori Kakazu. Learning coordinated motions in a competition for food between ant colonies. In D. Cliff, P. Husbands, J.-A. Meyer, and S. Wilson, editors, *Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 487–492. MIT Press, 1994.
- [33] Pattie Maes. How to do the right thing. *Connection Science*, 1(3):291–323, 1989.
- [34] 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.
- [35] Maja Mataric. Integration of representation into goal-driven behavior-based robots. *IEEE Transactions on Robotics and Automation*, 8(3):304–312, June 1992.
- [36] Maja Mataric. Learning to behave socially. In D. Cliff, P. Husbands, J.-A. Meyer, and S. Wilson, editors, *Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 453–462. MIT Press, 1994.
- [37] David McFarland. Towards robot cooperation. In D. Cliff, P. Husbands, J.-A. Meyer, and S. Wilson, editors, *Proceedings of the Third International Conference on Simulation of Adaptive Behavior*, pages 440–444. MIT Press, 1994.
- [38] 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.

- [39] 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.
- [40] Lynne E. Parker. ALLIANCE: An architecture for fault tolerant, cooperative control of heterogeneous mobile robots. In *Proceedings of the 1994 IEEE/RSJ/GI International Conference on Intelligent Robots and Systems (IROS '94)*, pages 776–783, Munich, Germany, September 1994.
- [41] 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.
- [42] Lynne E. Parker. Fault tolerant multi-robot cooperation. MIT Artificial Intelligence Lab Videotape AIV-9, December 1994.
- [43] Lynne E. Parker. *Heterogeneous Multi-Robot Cooperation*. PhD thesis, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Cambridge, MA, February 1994. MIT-AI-TR 1465 (1994).
- [44] Lynne E. Parker. ALLIANCE: An architecture for fault tolerant multi-robot cooperation. Technical Report ORNL/TM-12920, Oak Ridge National Laboratory, January 1995.
- [45] Lynne E. Parker. The effect of action recognition and robot awareness in cooperative robotic teams. In *Proceedings of the 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '95)*, Pittsburgh, PA, August 1995.
- [46] 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.
- [47] Luc Steels. Cooperation between distributed agents through self-organization. In Yves Demazeau and Jean-Pierre Muller, editors, *Decentralized A.I.* Elsevier Science, 1990.
- [48] 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.
- [49] 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.

- [50] 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.

## INTERNAL DISTRIBUTION

- |  |  |
|--|--|
| <ul style="list-style-type: none"> <li>1-5. J. Barhen</li> <li>6. M. Beckerman</li> <li>7. C. W. Glover</li> <li>8-12. W. C. Grimmell</li> <li>13. J. P. Jones</li> <li>14. H. E. Knee</li> <li>15-19. R. C. Mann</li> <li>20. M. D. Morris</li> <li>21. E. M. Oblow</li> <li>22-26. C. E. Oliver</li> <li>27-31. L. E. Parker</li> <li>32. V. Protopopescu</li> </ul> | <ul style="list-style-type: none"> <li>33. N. S. Rao</li> <li>34. D. B. Reister</li> <li>35. S. Shekhar</li> <li>36-40. R. F. Sincovec</li> <li>41. E. C. Uberbacher</li> <li>42. M. A. Unseren</li> <li>43. CSMD Reports Office</li> <li>44-45. Laboratory Records Department</li> <li>46. Laboratory Records, ORNL-RC</li> <li>47. Document Reference Section</li> <li>48. Central Research Library</li> <li>49. ORNL Patent Office</li> </ul> |
|--|--|

## EXTERNAL DISTRIBUTION

- 50. Dr. Fred Aminzadeh, 401 Paseo Estrella, Anaheim Hills, CA 92807
- 51. Dr. John Baillieul, Aerospace and Mechanical Engineering Department, Boston University, 110 Cummington St., Boston, MA 02215
- 52. Dr. John Blair, JBX Technologies, 25 Moore Road, Wayland, MA 01778
- 53. Mr. Steve Holland, Robotics, B/MD-63, General Motors Corporation, NAO Manufacturing Center, 30300 Mound Rd., Warren, MI 48090-9040
- 54. Dr. Oscar P. Manley, Division of Engineering, Mathematical, and Geosciences, Office of Basic Energy Sciences, ER-15, U.S. Department of Energy, Germantown, Washington, DC 20545
- 55. Dr. K. S. Narendra, Yale University, Center for Systems Science, Department of Electrical Engineering, P.O. Box 208267, New Haven, CT 06520-8267
- 56. Dr. Wes Snyder, Department of Radiology, Bowman Gray School of Medicine, N.C. Baptist Hospital School of Medicine, 300 S. Hawthorne Dr., Winston-Salem, NC 27103
- 57. Professor Mary F. Wheeler, Department of Mathematical Sciences, Rice University, P.O. Box 1892, Houston, TX 77251

58. Office of Assistant Manager for Energy Research and Development, U.S.  
Department of Energy, Oak Ridge Operations Office, P.O. Box 2001,  
Oak Ridge, TN 37831-8600
- 59-60. Office of Scientific & Technical Information, P. O. Box 62, Oak Ridge, TN  
37830