ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation

Lynne E. Parker
Center for Engineering Systems Advanced Research
Oak Ridge National Laboratory
P. O. Box 2008, Mailstop 6355
Oak Ridge, TN 37831-6355
email: ParkerLE@ornl.gov
phone: (423) 241-4959, fax: (423) 574-0405
URL: http://avalon.epm.ornl.gov/~parkerle/

Abstract

ALLIANCE is a software architecture that facilitates the fault tolerant cooperative control of teams of heterogeneous mobile robots performing missions composed of loosely coupled subtasks that may have ordering dependencies. ALLIANCE allows teams of robots, each of which possesses a variety of high-level functions that it can perform during a mission, to individually select appropriate actions throughout the mission based on the requirements of the mission, the activities of other robots, the current environmental conditions, and the robot's own internal states. ALLIANCE is a fully distributed, behavior-based architecture that incorporates the use of mathematically-modeled motivations (such as impatience and acquiescence) within each robot to achieve adaptive action selection. Since cooperative robotic teams usually work in dynamic and unpredictable environments, this software architecture allows the robot team members to respond robustly, reliably, flexibly, and coherently to unexpected environmental changes and modifications in the robot team that may occur due to mechanical failure, the learning of new skills, or the addition or removal of robots from the team by human intervention. The feasibility of this architecture is demonstrated in an implementation on a team of mobile robots performing a laboratory version of hazardous waste cleanup.

I. Introduction

A. Motivation

A key driving force in the development of mobile robotic systems is their potential for reducing the need for human presence in dangerous applications. Applications such as the cleanup of toxic waste, nuclear power plant decommissioning, planetary exploration, fire fighting, search and rescue missions, security, surveillance, and reconnaissance tasks have elements of danger in which human casualties are possible, or even likely. In all of these applications, it is desirable to reduce the risk to humans through the use of autonomous robot technology. Other applications, such as manufacturing or industrial and/or household maintenance, are of a highly repetitive nature, creating tasks that humans find monotonous or fatiguing. In these cases, the quality of the solution may be increased by employing autonomous agents.

One approach to creating an autonomous robotic solution to a given application is to try to build a single robot to address that application. This robot would be designed to have all of the capabilities necessary to complete the mission on its own, perhaps with some guidance from a human controller. For smaller-scale applications, the single robot approach is often feasible. However, a large number of the human solutions to these real world applications of interest employ the use of multiple humans supporting and complementing each other. Rather than having one human performing the task alone, a team of workers is formed that have a variety of specialized skills. These workers are available to help each other, and to provide individualized expertise when needed in the application. Each human team member is typically assigned a role (or roles) to fulfill during the mission that is based upon that human's skills and experience. The humans will also share some common capabilities that allow them to perform some tasks interchangeably, depending upon the workload of the individual during the mission. Unexpected events may occur during the mission that require a dynamic reallocation of tasks by the humans

to address the new circumstances. Many examples of human teams of this type can be found that are very successful and efficient in performing their mission. Real-world applications that are well-suited for team-based approaches include the U.S. Department of Energy applications of decontamination and decommissioning of legacy manufacturing facilities and hazardous waste cleanup; the U.S. Department of Defense applications of surveillance and reconnaissance and remote warfare; the NASA applications of space exploration; and commercial and private applications of industrial and household maintenance, firefighting, search and rescue, and security. Most of these applications are composed of tasks that are inherently distributed, either in space, time, or functionality, and thus require a distributed solution.

Since realistic human solutions to these types of applications require multiple humans to work together, it is feasible to examine the use of robot teams for automated solutions to these tasks. There are a number of potential advantages to using a distributed mobile robot system. It may be possible to reduce the total cost of the system by constructing multiple simpler robots rather than a monolithic single robot. The complexity of many environments or missions may actually require a mixture of robotic capabilities that is too extensive to design into a single robot. In the general case, a robot team consists of a variety of types of robots, each type of which specializes in (possibly overlapping) areas of capability. Utilizing a team of robots may make it possible to increase the robustness of the system by taking advantage of the parallelism and redundancy of multiple robots. Additionally, time constraints may require the use of multiple robots working simultaneously on different aspects of the mission in order to successfully accomplish the objective. Certainly, a distributed solution is the only viable approach for the application domains mentioned above that are inherently distributed in time, space, and/or functionality. Thus, in the most general case, the robot team would consist of a variety of heterogeneous types of robots working together to accomplish the mission that no individual robot could accomplish alone.

However, the use of multiple robots is not without its disadvantages. If not properly constructed, multiple-robot systems may actually increase the complexity of an automated solution rather than simplifying it. In multi-robot approaches, one has to deal with many challenging issues that do not arise in single-robot systems, such as achieving coherence, determining how to decompose and allocate the problem among a group of robots, determining how to enable the robots to interact, and so forth. In fact, the difficulties in designing a cooperative team are significant. In [5], Bond and Gasser describe the basic problems the field of Distributed Artificial Intelligence must address; those aspects directly related to situated multi-robot systems include the following:

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

These are challenging issues which have been extensively studied, but yet still have many open research issues remaining to be addressed. As described in the following section, while much research in recent years has addressed the issues of autonomous robots and multi-robot cooperation, current robotics technology is still far from achieving many of these real world applications. We believe that one reason for this technology gap is that previous work has not adequately addressed the issues of fault tolerance and adaptivity. Here, by *fault tolerance*, we mean the ability of the robot team to respond to individual robot failures or failures in communication that may occur at any time during the mission. The fault tolerant response of interest in this article is the dynamic re-selection (or re-allocation) of tasks by robot team members due to robot failures or a dynamically changing environment. We want the robot team as a whole to be able to complete its mission to the greatest extent possible in spite of any single-point failure. By *adaptivity*, we mean the ability of the robot team to change its behavior over time in response to a dynamic environment, changes in the team mission, or changes in the team capabilities or composition, to either improve performance or to prevent unnecessary degradation in performance.

The ALLIANCE architecture described in this article offers one solution to multi-robot cooperation that not only addresses the issues inherent in any multi-robot team, but also allows the robotic teams to be fault tolerant, reliable, and adaptable. Requiring fault tolerance in a cooperative architecture emphasizes the need to build teams that minimize their vulnerability to individual robot outages (either full or partial outages). Reliability refers to the dependability of a system, and whether it functions properly and correctly each time it is utilized. This concept differs slightly from fault tolerance in that we want to be assured that a robot team correctly accomplishes its mission even when individual robot failures do not occur. One measure of the reliability of the architecture is its ability to guarantee that the mission will be solved, within certain operating constraints, when applied to any given cooperative robot team. Adaptivity in a cooperative team allows that team to be responsive

to changes in individual robot skills and performance, to dynamic environmental changes, and to changes in the robot team composition as robots dynamically join or leave the cooperative team.

This article describes the control architecture, ALLIANCE, that we have developed which facilitates fault tolerant, reliable, and adaptive cooperation among small- to medium-sized teams of heterogeneous mobile robots, performing (in dynamic environments) missions composed of independent tasks that can have ordering dependencies. We begin by describing the related work in this area, followed by a detailed discussion of the features of ALLIANCE. We then illustrate the viability of this architecture by describing the results of implementing ALLIANCE on a team of robots performing a laboratory version of hazardous waste cleanup, which requires the robots to find the initial locations of two spills, move the two spills to a goal destination, and periodically report the team's progress to a human monitoring the mission. In the appendix, we supply a proof of ALLIANCE's mission termination for a restricted set of cooperative robotic applications.

II. Related Work

The amount of research in the field of cooperative mobile robotics has grown substantially in recent years. This work can be broadly categorized into two groups: swarm-type cooperation and "intentional" cooperation. (This paper addresses the second area of cooperation.) The swarm-type approach to multi-robot cooperation deals with large numbers of homogeneous robots. This approach is useful for non-time-critical applications involving numerous repetitions of the same activity over a relatively large area, such a cleaning a parking lot or collecting rock samples on Mars. The approach to cooperative control taken in these systems is derived from the fields of neurobiology, ethology, psychophysics, and sociology, and is typically 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. Such approaches usually rely on mathematical convergence results (such as the random walk theorem [12]) that indicate the desired outcome over a sufficiently long period of time. A key research issue in this scenario is determining the proper design of the local control laws that will allow the collection of robots to solve a given problem.

A number of researchers have studied the issues of swarm robotics. Deneubourg *et al.* [16] describe simulation results of a distributed sorting algorithm. Theraulaz *et al.* [36] extract cooperative control strategies, such as foraging, from a study of Polistes wasp colonies. Steels [34] 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 [17] describe simulation studies of foraging and chain-making robots. In [25] Mataric describes the results of implementing group behaviors such as dispersion, aggregation, and flocking on a group of physical robots. Beni and Wang [4] describe methods of generating arbitrary patterns in cyclic cellular robotics. Kube and Zhang [22] present the results of implementing an emergent control strategy on a group of five physical robots performing the task of locating and pushing a brightly lit box. Stilwell and Bay [35] 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.* [1] present research concerned with sensing, communication, and social organization for tasks such as foraging. The CEBOT work, described in [19] 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.

The primary difference between these approaches and the problem addressed in this article is that the above approaches are designed strictly for homogeneous robot teams, in which each robot has the same capabilities and control algorithm. Additionally, issues of efficiency are largely ignored. However, in heterogeneous robot teams such as those addressed in this article, not all tasks can be performed by all team members, and even if more than one robot can perform a given task, they may perform that task quite differently. Thus the proper mapping of subtasks to robots is dependent upon the capabilities and performance of each robot team member. This additional constraint brings many complications to a workable architecture for robot cooperation, and must be addressed explicitly to achieve the desirable level of cooperation.

The second 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. Furthermore, this second type of mobile robotic mission usually requires that several distinct tasks be performed. These missions thus

usually require a smaller number of possibly heterogeneous mobile robots involved in more purposeful cooperation. Although individual robots in this approach are typically 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. Key issues in these systems include robustly determining which robot should perform which task so as to maximize the efficiency of the team and ensuring the proper coordination among team members to allow them to successfully complete their mission.

Two bodies of previous research are particularly applicable to this 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 work include Noreils [26], who describes a *sense-model-plan-act* control architecture which includes three layers of control: the planner level, which manages coordinated protocols, decomposes tasks into smaller subunits, and assigns the subtasks to a network of robots; the control level, which organizes and executes a robot's tasks; and the functional level, which provides controlled reactivity. He reports on the implementation of this architecture on two physical mobile robots performing convoying and box pushing. In both of these examples, one of the robots acts as a leader, and the other acts as a follower.

Caloud *et al.* [9] describe another *sense-model-plan-act* architecture which includes a task planner, a task allocator, a motion planner, and an execution monitor. Each robot obtains goals to achieve either based on its own current situation, or via a request by another team member. They use Petri Nets for interpretation of the plan decomposition and execution monitoring. In this paper they report on plans to implement their architecture on three physical robots.

In [2] and elsewhere, Asama *et al.* describe their decentralized robot system called ACTRESS, addressing the issues of communication, task assignment, and path planning among heterogeneous robotic agents. Their approach revolves primarily around a negotiation framework which allows robots to recruit help when needed. They have demonstrated their architecture on mobile robots performing a box pushing task.

Wang [37] addresses a similar issue to that addressed in this article — namely, dynamic, distributed task allocation when more than one robot can perform a given task. He proposes the use of several distributed mutual exclusion algorithms that use a "sign-board" for inter-robot communication. These algorithms are used to solve problems including distributed leader finding, the N-way intersection problem, and robot ordering. However, this earlier paper does not address issues of dynamic reallocation due to robot failure and efficiency issues due to robot heterogeneity.

Cohen *et al.* [13] propose a hierarchical subdivision of authority to address the problem of cooperative fire-fighting. They describe their Phoenix system, which includes a generic simulation environment and a real-time, adaptive planner. The main controller in this architecture is called the Fireboss, which maintains a global view of the environment, forms global plans, and sends instructions to agents to activate their own local planning.

A huge amount of research in optimal task allocation and scheduling has been accomplished previously (e.g. [14]). However, these approaches alone are not directly applicable to multi-robot missions, since they do not address multi-robot performance in a dynamic world involving robot heterogeneity, sensing uncertainty, and the nondeterminism of robot actions.

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 "intentional" cooperation among generic agents. However, 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 [5] for many of the seminal papers in this field.

Our approach is distinct from these earlier DAI approaches in that we embed task-oriented missions in behavior-based systems and directly address solutions to fault tolerant, adaptive control. Although the need for fault tolerance is noted in the earlier architectures, they typically either make no serious effort at achieving fault tolerant, adaptive control or they assume the presence of unrealistic "black boxes" that continually monitor the environment and provide recovery strategies (usually involving unspecified replanning mechanisms) for handling various types of unexpected events. Thus, in actuality, if one or more of the robots or the communication system fails under these approaches, the entire team is subject to catastrophic failure. Experience with physical mobile robots has shown, however, that robot failure is very common, not only due to the complexity of the

robots themselves, but also due to the complexity of the environment in which these robots must be able to operate. Thus, control architectures must explicitly address the dynamic nature of the cooperative team and its environment to be truly useful in real-world applications. Indeed, the approach to cooperative control developed in this article has been designed specifically with the view toward achieving fault tolerance and adaptivity.

Additionally, the earlier approaches break the problem into a traditional AI *sense-model-plan-act* decomposition rather than the functional decomposition used in behavior-based approaches. The traditional approach has likely been favored because it presents a clean subdivision between the job planning, task decomposition, and task allocation portions of the mission to be accomplished — a segmentation that may, at first, appear to simplify the cooperative team design. However, the problems with applying these traditional approaches to physical robot teams are the same problems that currently plague these approaches when they are applied to individual situated robots. As argued by Brooks in [8] and elsewhere, approaches using a *sense-model-plan-act* framework have been unable to deliver real-time performance in a dynamic world because of their failure to adequately address the situatedness and embodiment of physical robots. Thus, a behavior-based approach to cooperation was utilized in ALLIANCE to increase the robustness and adaptivity of the cooperative team.

Refer to [11] for a detailed review of much of the existing work in cooperative robotics.

III. ALLIANCE

A. Assumptions

In the design of any control scheme, it is important to make explicit those assumptions underlying the approach. Thus, before describing the ALLIANCE architecture in detail, we first discuss the assumptions that were made in the design of this architecture. Note that these assumptions are made within the context (described earlier) of solving the problem of multi-robot cooperation for small- to medium-sized teams of heterogeneous robots performing missions composed of independent subtasks that may have ordering dependencies.

Our assumptions are as follows:

1. The robots on the team can detect the effect of their own actions, with some probability greater than 0.
2. Robot r_i can detect the actions of other team members for which r_i has redundant capabilities, with some probability greater than 0; these actions may be detected through any available means, including explicit broadcast communication.
3. Robots on the team do not lie and are not intentionally adversarial.
4. The communications medium is not guaranteed to be available.
5. The robots do not possess perfect sensors and effectors.
6. Any of the robot subsystems can fail, with some probability greater than 0.
7. If a robot fails, it cannot necessarily communicate its failure to its teammates.
8. A centralized store of complete world knowledge is not available.

We make the first assumption — a robot's detection of the effect of its own actions — to ensure that robots have some measure of feedback control and do not perform their actions purely with open-loop control. However, we do not require that robots be able to measure their own effectiveness with certainty, because we realize this rarely happens on real robots.

The second assumption deals with the problem of *action recognition* — the ability of a robot to observe and interpret the behavior of another robot. Previous research in cooperative robotics has investigated several possible ways of providing action recognition to robot teams — from implicit cooperation through sensory feedback to explicit cooperation using the exchange of communicated messages. For example, Huber and Durfee [20] have developed a multiple resolution, hierarchical plan recognition system to coordinate the motion of two interacting mobile robots based upon belief networks. Other researchers have studied the effect of communication in providing action knowledge. For example, MacLennan [24] investigates the evolution of communication in simulated worlds and concludes that the communication of local robot information can result in significant performance improvements; Werner and Dyer [38] examine the evolution of communication which facilitates the breeding and propagation of artificial creatures; and Balch and Arkin [3] examine the importance of communication in robotic societies performing forage, consume, and graze tasks, finding that communication can significantly improve performance for tasks involving little implicit communication through the world, and that communication of current robot state is almost as effective as communication of robot goals.

In the current article, we do not require that a robot be able to determine a teammate's actions through passive observation, which can be quite difficult to achieve. Instead, we enable robots to learn of the actions of

their teammates through an explicit communication mechanism, whereby robots broadcast information on their current activities to the rest of the team.

Third, we assume that the robots are built to work on a team, and are neither in direct competition with each other, nor are attempting to subvert the actions of their teammates. Although conflicts may arise at a low level due to, for example, a shared workspace, we assume that at a high level the robots share compatible goals. (Note, however, that some multi-robot team applications may require the ability to deal with adversarial teams, such as military applications or team competitions (e.g. robot soccer). This is not covered in this article.)

We further assume that any subsystem of the team, such as communications, sensors, and effectors, or individual robots, are subject to failure, thus leading to assumptions four through seven.

Finally, we assume that robots do not have access to some centralized store of world knowledge, and that no centralized agent is available that can monitor the state of the entire robot environment and make controlling decisions based upon this information.

B. Overview of ALLIANCE

As already discussed, a major design goal in the development of ALLIANCE is to address the real-world issues of fault tolerance and adaptivity when using teams of fallible robots with noisy sensors and effectors. Our aim is to create robot teams that are able to cope with failures and uncertainty in action selection and action execution, and with changes in a dynamic environment. Because of these design goals, we developed ALLIANCE to be a fully distributed, behavior-based software architecture which gives all robots the capability to determine their own actions based upon their current situation. No centralized control is utilized, so that we can investigate the power of a fully distributed robotic system to accomplish group goals[1]. The purpose of this approach is to maintain a purely distributed cooperative control scheme which affords an increased degree of robustness; since individual agents are always fully autonomous, they have the ability to perform useful actions even amidst the failure of other robots.

ALLIANCE defines a mechanism that allows teams of robots, each of which possesses a variety of high-level functions that it can perform during a mission, to individually select appropriate actions throughout the mission based on the requirements of the mission, the activities of other robots, the current environmental conditions, and the robot's own internal states. This mechanism is based upon the use of mathematically-modeled motivations within each robot, such as impatience and acquiescence, to achieve adaptive action selection. Under the behavior-based framework, the task-achieving behaviors of each robot receive sensory input and control some aspect of the actuator output. 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 necessary. Within each layer of competence may be a number of simple modules interacting via inhibition and suppression to produce the desired behavior. This approach has been used successfully in a number of robotic applications, several of which are described in [7].

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 are hibernating. Figure 1 shows the general architecture of ALLIANCE and illustrates three such behavior sets. Each behavior set of a robot corresponds to those levels of competence required to perform some high-level task-achieving function. Because of the alternative goals that may be pursued by the robots, the robots must have some means of selecting the appropriate behavior set to activate. This action selection is controlled through the use of *motivational behaviors*, each of which controls the activation of one behavior set. Due to conflicting goals, only one behavior set is active at any point in time (implemented via cross-inhibition of behavior sets). However, other lower-level competences such as collision avoidance may be continually active regardless of the high-level goal the robot is currently pursuing. Examples of this type of continually active competence are shown generically in figure 1 as layer 0, layer 1, and layer 2.

C. Motivational Behaviors

In ALLIANCE, the ability for robots to respond to unexpected events, robot failures, and so forth, is provided through the use of motivations. These motivations are designed to allow robot team members to perform tasks

[1] Note that we are not claiming that centralized or hierarchical control approaches are never needed.

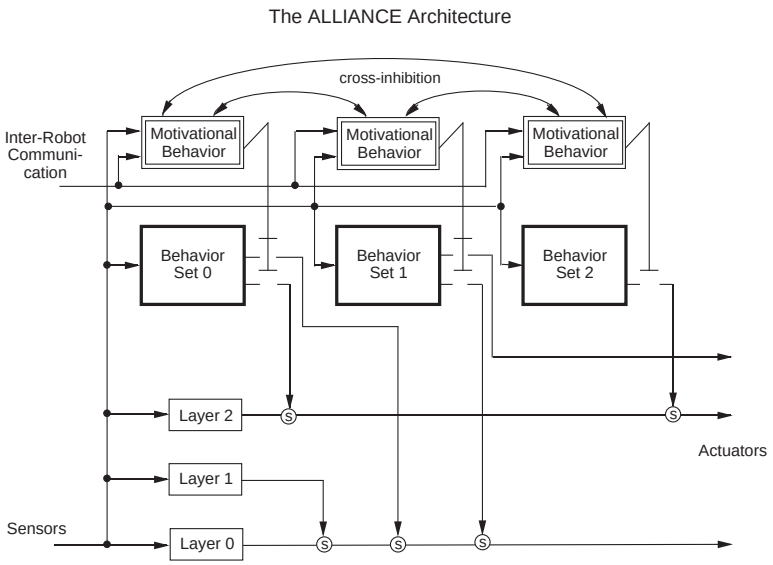

Fig. 1. The ALLIANCE architecture, implemented on each robot in the cooperative team, delineates several behavior sets, each of which correspond to some high-level task-achieving function. The primary mechanism enabling a robot to select a high-level function to activate is the *motivational behavior*. The symbols that connect the output of each motivational behavior with the output of its corresponding behavior set (vertical lines with short horizontal bars) indicate that a motivational behavior either allows all or none of the outputs of its behavior set to pass through to the robot's actuators. The non-bold, single-bordered rectangles correspond to individual layers of competence that are always active.

only as long as they demonstrate their ability to have the desired effect on the world. This differs from the commonly used technique for task allocation that begins with breaking down the mission (or part of the mission) into subtasks, and then computing the "optimal" robot-to-task mapping based upon agent skill levels, with little recourse for robot failures after the allocation has occurred.

The motivations are utilized in each *motivational behavior*, which is the primary mechanism for achieving adaptive action selection in this architecture. 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. This input is combined (in a manner described below) to generate the output of a motivational behavior at any point in time. This output defines 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.

Two types of internal motivations are modeled in ALLIANCE — *robot impatience* and *robot acquiescence*. The impatience motivation enables a robot to handle situations when other robots (outside itself) fail in performing a given task. The acquiescence motivation enables a robot to handle situations in which it, itself, fails to properly perform its task. Intuitively, a motivational behavior works as follows. A robot's motivation to activate any given behavior set is initialized to 0. Then over time, that robot's motivation to perform a given behavior set increases at a fast rate of impatience (defined explicitly below) as long as the task corresponding to that behavior set is not being accomplished by any robot team member. The robot, however, should also be responsive to the actions of its teammates, adapting its task selection to the activities of other robot team members. Thus, if the ith robot r_i is aware that the kth robot r_k is working on a particular task, r_i should be satisfied for some period of time that that task is going to be accomplished even without its own participation in the task, and thus go on to some other applicable action. Robot r_i's motivation to activate its corresponding behavior set continues to increase, but at a slower rate of impatience. This characteristic prevents robots from replicating each other's actions and thus wasting needless energy.

As a simple example, consider a team of two robots unloading boxes from a truck and placing them on one of two conveyor belts, depending upon the labeling on the box. Both of the robots, call them A and B, have the ability to unload boxes from the truck to a temporary storage location, and the ability to move them from the temporary storage location to the appropriate conveyor belt. (We assume that, due to the way the loading dock

is designed, the robots cannot move boxes immediately from the truck to the conveyor belt). At the beginning of the mission, say robot A elects to unload the boxes from the truck. Robot B is then satisfied that the boxes will be unloaded, and proceeds to move the boxes from the temporary location to the conveyor belt when applicable. As the mission progresses, however, let us assume that robot A's box-detection sensor (e.g. a camera) becomes dirty and prevents A from locating boxes. Since no more boxes are arriving at the temporary location, robot B becomes more and more impatient to take over the task of unloading boxes. After a period of time, B takes over the task of unloading boxes, even though robot A is still attempting to accomplish that task — unaware that its sensor is returning faulty readings.

Before continuing with this example, we note here that detecting and interpreting the actions of other robots is not a trivial problem, and often requires perceptual abilities that are not yet possible with current sensing technology. As it stands today, the sensory capabilities of even the 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 sensing through the world. Thus, 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 in the effect of communicated messages, 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 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. However, even if another robot r_k has not taken over the task, robot r_i may give up its task anyway if the task is not completed in an acceptable period of time. This allows r_i the possibility of working on another task that may prove to be more productive rather than becoming stuck performing the unproductive task forever. With this acquiescence characteristic, therefore, a robot is able to adapt its actions to its own failures. Continuing our earlier simple example, this characteristic of acquiescence is demonstrated in robot A, which gives up trying to unload the truck when robot B takes over that task. Depending upon its self-monitoring capabilities, robot A will then either attempt to perform the second task of moving boxes to the conveyor belt (probably unsuccessfully) or give up altogether. Robot B would then be motivated to move the boxes from the storage site to the conveyor belt after it had completed unloading the task. In ALLIANCE, the mission could also be designed in such a way that robot B interleaves a period of unloading the truck with a period of moving boxes to the conveyor belt.

The design of the motivational behaviors in ALLIANCE 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 (e.g. another truck of boxes arrives in our simple example), or existing environmental conditions can disappear and thus relieve the robots of certain tasks (e.g. the truck of boxes drives away). In either case, the motivations fluidly adapt to these situations, causing robots to respond appropriately to the current environmental circumstances.

We note that the parameters controlling motivational rates of robots under the ALLIANCE architecture can adapted over time based upon learning. A learning mechanism we have developed allows the rates of impatience and acquiescence to be context and environment sensitive. An extension to ALLIANCE, called L-ALLIANCE (for Learning ALLIANCE), provides the mechanisms for accomplishing this adaptation. This parameter adaptation capability is described in section III-E. For more details, refer to [31].

C.1 Comparison to Negotiation

We now compare the ALLIANCE task allocation mechanism to a common approach used in distributed artificial intelligence (DAI) systems for multi-agent task allocation — a negotiation-based mechanism. In general, the

negotiation schemes have no centralized agent that has full control over which tasks individual team members should perform. Instead, many agents know which subtasks are required for various portions of the mission to be performed, along with the skills required to achieve those subtasks. These agents then broadcast a request for bids to perform these subtasks, to which other agents may respond if they are available and want to perform these tasks. The broadcasting agent then selects an agent from those that respond and awards the task to the winning agent, who then commences to perform that task, recruiting yet other agents to help if required. One example of a popular negotiation protocol that has been used extensively is the contract-net protocol [15], [33]; other negotiation schemes are described in [18], [21], [32], [39].

However, although DAI work has demonstrated success with this approach in a number of domains (e.g. distributed vehicle monitoring [23] and distributed air traffic control [10]), the proposed solutions have not been adequately demonstrated in *situated* agent (i.e. robotic) teams, which have to live in, and react to, dynamic and uncertain environments amidst noisy sensors and effectors, frequent agent failures, and a limited bandwidth, noisy communication mechanism. The DAI approaches typically assume the presence of "black boxes" to provide high-level, nearly-perfect sensing and action capabilities. However, these DAI approaches typically ignore or only give brief treatment to the issues of robot performance of those tasks after they have been allocated. Such approaches usually assume the robots will eventually accomplish the task they have been assigned, or that some external monitor will provide information to the robots on dynamic changes in the environment or robot performance. However, to realistically design a cooperative approach to robotics, we must include mechanisms within the software control of each robot that allow the team members to recover from dynamic changes in their environment, or failures in the robot team or communication mechanism.

Thus, we developed the ALLIANCE architecture to explicitly address the issue of fault tolerance amidst possible robot and communication failures. The ALLIANCE architecture provides each robot with sufficient autonomy to make decisions on its actions at all times during a mission, taking into consideration the actions of other agents and their effects upon the world. While some efficiency may be lost as a consequence of not negotiating the task subdivision in advance, robustness is gained if robot failures or other dynamic events occur at any time during the mission. We speculate that a hybrid combination of negotiation and the ALLIANCE motivational mechanisms would enable a system to experience the benefits of both approaches. This is a topic for future research.

D. Discussion of Formal Model of ALLIANCE

Let us now look in detail at how our philosophy regarding action selection is incorporated into the motivational behavior mechanism.

First, let us formally define our problem as follows. Let the set $R = \{r_1, r_2, ..., r_n\}$ represent the set of n heterogeneous robots composing the cooperative team, and the set $T = \{task_1, task_2, ..., task_m\}$ represent m independent subtasks which compose the *mission*. We use the term *high-level task-achieving function* to correspond intuitively to the functions possessed by individual robots that allow the robots to achieve tasks required in the mission. These functions map very closely to the upper layers of the subsumption-based control architecture [6]. In the ALLIANCE architecture, each behavior set supplies its robot with a high-level task-achieving function. Thus, in the ALLIANCE architecture, the terms *high-level task-achieving function* and *behavior set* are synonymous. We refer to the high-level task-achieving functions, or behavior sets, possessed by robot r_i in ALLIANCE as the set $A_i = \{a_{i1}, a_{i2}, ...\}$. Since different robots may have different ways of performing the same task, we need a way of referring to the task a robot is working on when it activates a behavior set. Thus, we define the set of n functions $\{h_1(a_{1k}), h_2(a_{2k}), ..., h_n(a_{nk})\}$, where $h_i(a_{ik})$ returns the task in T that robot r_i is working on when it activates behavior set a_{ik}.

In the discussion below, we first discuss the threshold of activation of a behavior set, and then describe the five primary inputs to the motivational behavior. We conclude this section by showing how these inputs are combined to determine the current level of motivation of a given behavior set in a given robot. Throughout this section, a number of parameters are defined that regulate the motivational levels. A discussion on how the parameter values are obtained and adapted over time is provided in section III-E.

D.1 Threshold of activation

The threshold of activation of a behavior set is given by one parameter, θ. This parameter determines the level of motivation beyond which a given behavior set will become active. Although different thresholds of activation

could be used for different behavior sets and for different robots, in ALLIANCE one threshold is sufficient since the rates of impatience and acquiescence can vary across behavior sets and across robots.

D.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. This feedback is assumed to be noisy, and can originate from either physical robot sensors or *virtual* robot sensors. We define a simple function to capture the notion of sensory feedback in the motivational level computation as follows:

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

D.3 Inter-robot communication

Two parameters are utilized in ALLIANCE to control the broadcast communication among robots: ρ_i and τ_i. The first parameter, ρ_i, gives the rate at which robot r_i broadcasts its current activity. The second parameter, τ_i, provides an additional level of fault tolerance by giving the period of time robot r_i allows to pass without receiving a communication message from a specific teammate before deciding that that teammate has ceased to function. While monitoring the communication messages, each motivational behavior a_{ij} of robot r_i must also note when a team member is pursuing task $h_i(a_{ij})$. To refer to this type of monitoring in the formal model, the function *comm_received* is defined as follows:

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

D.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". In the formal model, this activity suppression is modeled by the following simple function:

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

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.

D.5 Robot impatience

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

The next two parameters, $\delta\_slow_{ij}(k,t)$ and $\delta\_fast_{ij}(t)$, give the rates of impatience of robot r_i concerning behavior set a_{ij} either while robot r_k is performing the task corresponding to behavior set a_{ij} (i.e. $h_i(a_{ij})$) or in the absence of other robots performing the task $h_i(a_{ij})$, respectively. We assume that the fast impatience parameter corresponds to a higher rate of impatience than the slow impatience parameter for a given behavior set in a given robot. The reasoning for this assumption should be clear — a robot r_i should allow another robot r_k the opportunity to accomplish its task before becoming impatient with r_k; however, there is no reason for r_i to remain idle if a task remains undone and no other robot is attempting that task.

The question that now arises is the following: what slow rate of impatience does a motivational behavior controlling behavior set a_{ij} use when more than one other robot is performing task $h_i(a_{ij})$? The method used in

ALLIANCE is to increase the motivation at a rate that allows the slowest robot r_k still under its allowable time $\phi_{ij}(k,t)$ to continue its attempt.

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

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

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

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

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

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.

D.6 Robot acquiescence

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

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

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

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

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

D.7 Motivation calculation

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

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

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

A topic of future research is to investigate other possible combinations of the above input values to compute the motivational behavior. The current linear form was selected for its simplicity, and resulted in good performance. Additional studies are needed to determine the utility of any specific combination.

E. Parameter Settings

As with any parameter-based system, the parameter settings in ALLIANCE strongly influence the global performance of the system. A robot's selection of actions under ALLIANCE is dependent upon the parameter settings of the motivational behaviors. In addition, a number of efficiency issues are affected by the parameter settings, such as the amount of robot idle time, the time of response to the need for task reallocation, the selection between alternative methods of accomplishing a task, and so forth. In practice, finding the proper parameter settings in ALLIANCE has not proved to be difficult. This architecture has been implemented on a number of different robotic applications [31] — one of which is reported later in this article — and parameter tuning did not prove to be a problem. However, ideally, the robot team members should be able to initialize and adapt these values with experience to find the proper parameter settings, rather than relying on human tuning. In the design of a parameter update strategy, we wanted to ensure that we did not sacrifice the desirable characteristics of fault tolerance and adaptivity that are present in ALLIANCE while enabling increases in robot team efficiency. We have thus developed an extension to ALLIANCE, called L-ALLIANCE, which provides mechanisms that allow the robots to dynamically update their parameter settings based upon knowledge learned from previous experiences. In this section, we outline the approach utilized in L-ALLIANCE for parameter tuning. Refer to [31] for a more complete discussion of L-ALLIANCE.

The general idea for dynamically updating the parameters is to have each robot "observe", evaluate, and catalogue the performance (measured in task execution time) of any robot team member whenever it performs a task of interest to that robot. The performance measurements are then used to update the ALLIANCE parameters in the manner specified below. These updates allow the robots to adapt their action selections over time in response to changes in the robot team composition, the robot team capabilities, or in the environment.

In this approach to parameter updates, we make two assumptions: (1) 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 in the future, and (2) 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.

Figure 2 illustrates the L-ALLIANCE extension to ALLIANCE, which incorporates the use of performance monitors for each motivational behavior within each robot. Formally, robot r_i, programmed with the b behavior sets $A = \{a_{i1}, a_{i2}, ..., a_{ib}\}$, also has b monitors $MON_i = \{mon_{i1}, mon_{i2}, ..., mon_{ib}\}$, 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. Monitor mon_{ij} then uses the mechanism described below to update the control parameters of behavior set a_{ij} based upon this learned knowledge.

Once the robot performance data has been obtained, it must be input to a control mechanism that allows the robot team to improve its efficiency over time while not sacrificing the fault tolerant characteristics of the behavior-based ALLIANCE architecture. Through an extensive series of empirical studies reported in [31], we derived an action selection algorithm that specifies how the parameters should be updated to achieve increased efficiency in the robot team's action selection.

The algorithm derived in [31] specifies that each robot r_i should obey the following algorithm in selecting its tasks:

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.

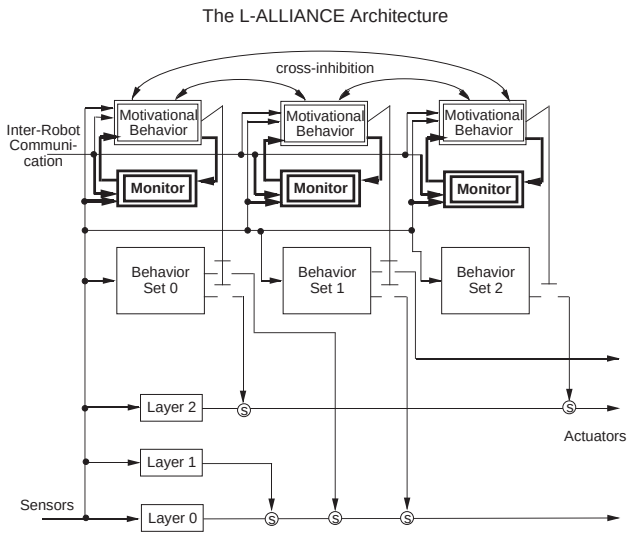

Fig. 2. The L-ALLIANCE architecture, which adds in the capability for dynamic parameter adaptation.

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 fall into the second category.

Note that although the above algorithm is stated in terms of a centralized controlling mechanism, the algorithm is in fact distributed across the behavior sets of ALLIANCE through the motivational behavior parameter settings.

The key parameters in ALLIANCE that affect the action selection are:

- $\delta\_fast_{ij}(t)$: the rate of impatience of r_i at time t concerning the behavior set a_{ij} when no other robot is performing task $h_i(a_{ij})$
- $\delta\_slow_{ij}(k,t)$: the rate of impatience of r_i at time t concerning the behavior set a_{ij} when robot r_k is performing task $h_i(a_{ij})$
- $\psi_{ij}(t)$: the time r_i will maintain a_{ij}'s activity before acquiescing to another robot

We now describe how these parameters are set and updated during the mission. First, we define the performance metric —task completion time — as:

$$\begin{aligned} task\_time_i(k,j,t) &= (\text{average time over last } \mu \text{ trials of } r_k\text{'s performance of task } h_i(a_{ij})) \\ &\quad + (\text{one standard deviation of these } \mu \text{ attempts}), \text{ as measured by } r_i \end{aligned}$$

The robot impatience parameters of $\delta\_slow_{ij}(k,t)$ and $\delta\_fast_{ij}(t)$ are then updated as follows. For $\delta\_slow_{ij}(k,t)$:

$$\begin{aligned} \phi_{ij}(k,t) &= \text{Time during which robot } r_i \text{ is willing to allow robot } r_k\text{'s communication} \\ &\quad \text{message to affect the motivation of behavior set } a_{ij}. \\ &= task\_time_i(i,j,t) \end{aligned}$$

$$\begin{aligned} \delta\_slow_{ij}(k,t) &= \text{Rate of impatience of robot } r_i \text{ concerning behavior set } a_{ij} \text{ after} \\ &\quad \text{discovering robot } r_k \text{ performing the task corresponding to this behavior set} \\ &= \frac{\theta}{\phi_{ij}(k,t)} \end{aligned}$$

For $\delta\_fast_{ij}(t)$:

min_delay = minimum allowed delay $\qquad$ max_delay = maximum allowed delay

$high = \max_{k,j} task\_time_i(k,j,t)$ $\qquad$ $low = \min_{k,j} task\_time_i(k,j,t)$

$$scale\_factor = \frac{max\_delay - min\_delay}{high - low}$$

$\delta\_fast_{ij}(t)$ = Rate of impatience of robot r_i concerning behavior set a_{ij} in the absence of other robots performing a similar behavior set

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

The robot acquiescence parameters are updated as follows:

$\psi_{ij}(t)$ = Time robot r_i wants to maintain behavior set a_{ij}'s activity before yielding to another robot.
= $task\_time_i(i,j,t)$

These parameter settings cause the robot team to effectively select their actions according to the algorithm given earlier in this subsection. Refer to [31] for the derivation of this algorithm.

IV. Results

The ALLIANCE architecture has been successfully implemented in a variety of proof-of-concept applications on both physical and simulated mobile robots. The applications implemented on physical robots include two versions of a hazardous waste cleanup mission and a cooperative box pushing demonstration [28]. The applications using simulated mobile robots include a janitorial service mission [27] and a bounding overwatch mission (reminiscent of military surveillance) [31]. All of these missions using the ALLIANCE architecture have been well-tested. Over 60 logged (and many videotaped) physical robot runs of the hazardous waste cleanup mission and over 30 physical robot runs (many of which were videotaped) of the box pushing demonstration were completed to elucidate the important issues in heterogeneous robot cooperation. The missions implemented on simulated robots encompass dozens of runs each, most of which were logged in the study of the action selection mechanism.

The experimental mission we describe here to illustrate the fault tolerant action selection features of ALLIANCE is a laboratory version of hazardous waste cleanup. (Refer [29], [31] for a somewhat different version of the hazardous waste cleanup mission, which involved the use of only one spill, rather than the two spills described below.) We first describe the robots used in these experimental studies, followed by a description of the mission the robots were given. We then describe the behavior set design of the robots for this mission, followed by the results of the implementation. The results described below are available on videotape [30].

A. The Robots

Our empirical studies were conducted on teams of three R-2 robots purchased commercially from IS Robotics. Each of these robots is a small, fully autonomous wheeled vehicle measuring approximately 25 centimeters wide, 31 centimeters deep, and 35 centimeters tall. The R-2 has two drive wheels arranged as a differential pair, two caster wheels in the rear for stability, and a two-degree-of-freedom parallel jaw gripper for grasping objects. The robot sensory suite includes eight infrared proximity sensors for use in collision avoidance, piezoelectric bump sensors distributed around the base of the robot for use in collision detection, and additional bump sensors inside the gripper for use in measuring gripping force.

We note here that although these robots are of the same type and thus have the potential of maximum redundancy in capabilities, mechanical drift and failure can cause them to have quite different actual abilities. For example, one of our robots had full use of its side infrared (IR) sensors which allowed it to perform wall-following, whereas the side IR sensors of two of the other robots had become dysfunctional. The L-ALLIANCE learning and parameter update system outlined in section III-E gives these robots the ability to take advantage of these differences and thus determine from trial to trial which team member is best suited for which task.

A radio communication system allows robot team members to communicate with each other. This radio system is integrated with a positioning system, which consists of a transceiver unit attached to each robot plus two sonar base stations for use in triangulating the robot positions. The positioning system is accurate to about 15 centimeters and is useful for providing robots with information on their own position with respect to their environment and with respect to other robot team members.

B. The Hazardous Waste Cleanup Mission

Illustrated in figure 3, the laboratory version of the hazardous waste cleanup mission requires two artificially "hazardous" waste spills in an enclosed room to be cleaned up by a team of three robots. This mission requires robot team members to perform the following distinct tasks: the robot team must locate the two waste spills, move the two spills to a goal location, while also periodically reporting the team progress to humans monitoring

the system. These tasks are referred to in the remainder of this article as *find-locations*, *move-spill(left)*, *move-spill(right)*, and *report-progress*, where *left* and *right* refer to the locations of the two spills relative to the room entrance.

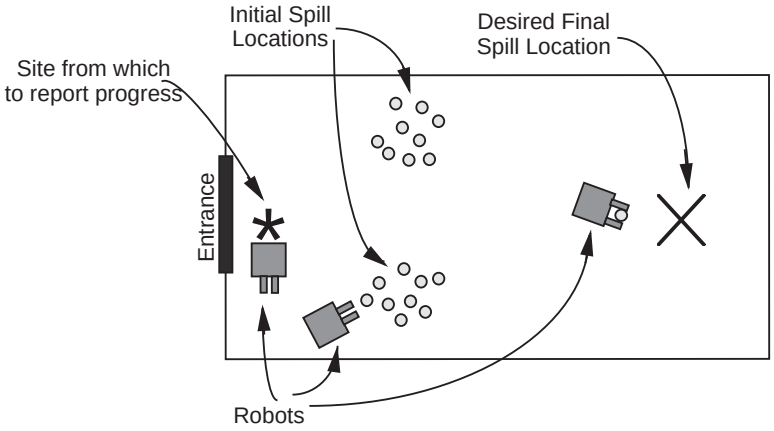

Fig. 3. The experimental mission: hazardous waste cleanup.

A difficulty in this mission is that the human monitor does not know the exact location of the spills in robot coordinates, and can only give the robot team qualitative information on the initial location of the two spills and the final desired location to which the robots must move the spills. Thus, the robots are told qualitatively that one spill is located in the right half of the front third of the room, while the other spill is located in the left half of the front third of the room. Furthermore, the robots are also told that the desired final location of the spill is in the back, center of the room, relative to the position of the entrance. This information is used as described below to locate the initial and final spill locations. To prevent interference among robots, ideally only one robot at a time attempts to find the spill, broadcasting the computed locations to the other team members once the task is complete.

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

Two behavior sets are provided which both accomplish the task of finding the initial and final spill locations — *find-locations-methodical* and *find-locations-wander* — both of which depend upon the workspace being rectangular and on the sides of the room being parallel to the axes of the global coordinate system. Because of these assumptions, these behavior sets do not serve as generally applicable location-finders. However, we made no attempt to generalize these algorithms, since the point of this experiment is to demonstrate the adaptive action selection characteristics of ALLIANCE. Shown in more detail in figure 5, the methodical version of finding the spill location is much more reliable than the wander version, and involves the robot first noting its starting (or home) x, y position and then following the walls of the room using its side IRs until it has returned to its home location while tracking the minimum and maximum x and y positions it reaches. It then uses these x, y values to calculate the coordinates of the right and left halves of the front third of the room (for the two initial spill locations) and the back center of the room (for the final spill location). These locations are then made available to the *move-spill(loc)* behavior set, which requires this information to perform its task.

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

The *move-spill(loc)* behavior set, shown in more detail in figure 7, can be activated whenever there are spill objects needing to be picked up at *loc*, the locations of the initial and final spill positions are known, and the

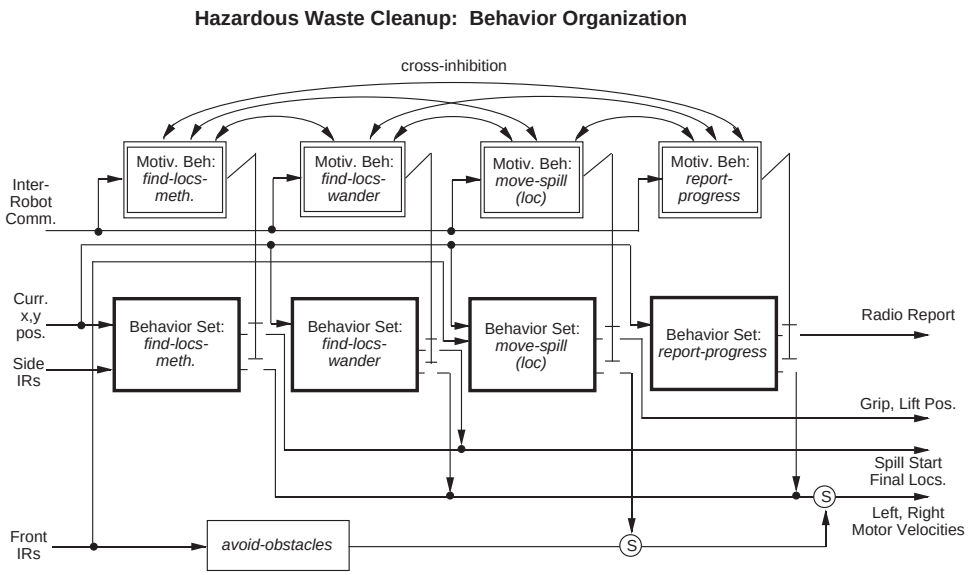

Fig. 4. The ALLIANCE-based control of each robot in the hazardous waste cleanup mission. Not all sensory inputs to the behavior sets are shown here. In this figure, the high-level task achieving functions *find-locations-methodical* and *find-locations-wander* are abbreviated as *find-locs-meth* and *find-locs-wander*, respectively.

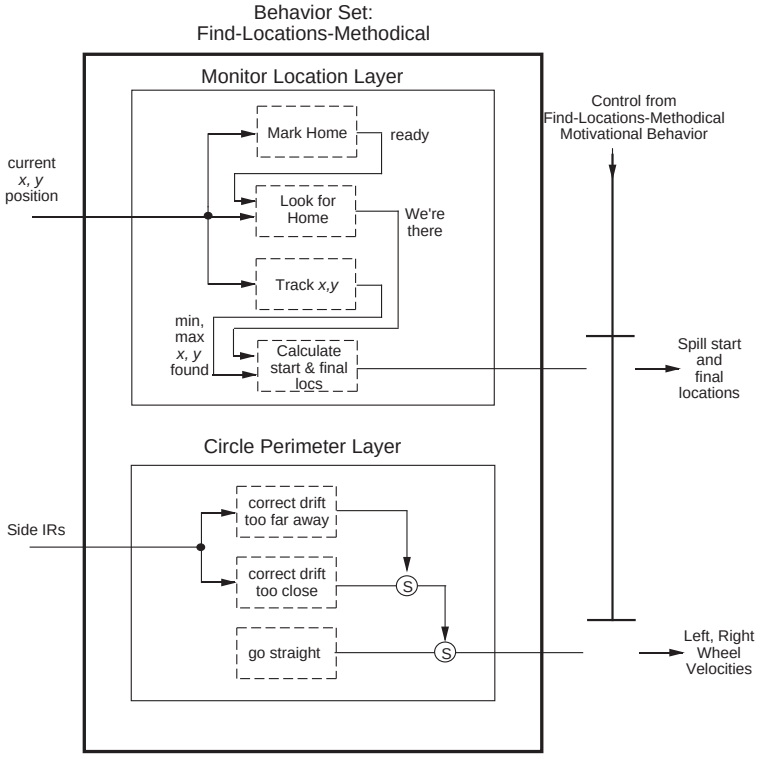

Fig. 5. The robot control organization within the *find-locations-methodical* behavior set.

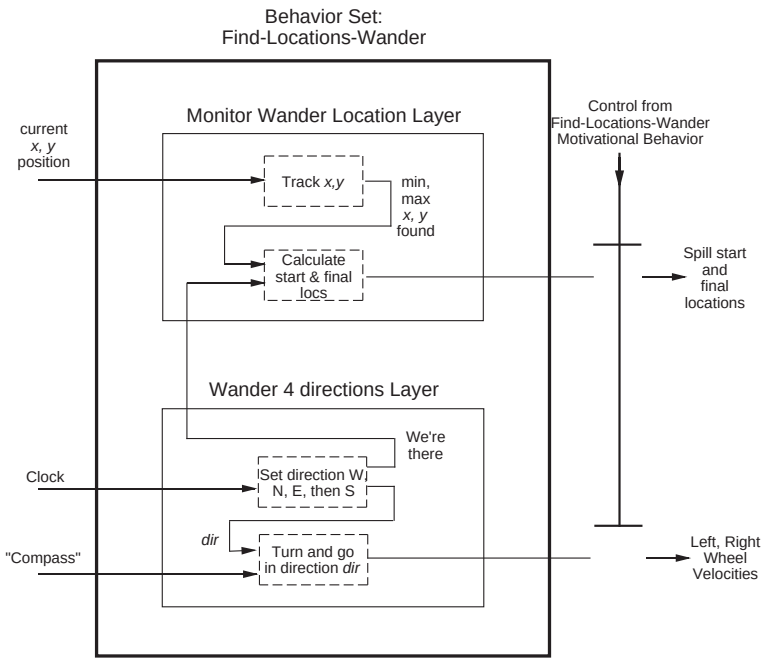

Fig. 6. The robot control organization within the *find-locations-wander* behavior set.

robot is not aware of any other robot currently working on the spill at *loc*. It involves having the robot (1) move to the vicinity of the initial spill location, (2) wander in a straight line through the area of the spill while using its front IR sensors to scan for spill objects, (3) "zero in" on a spill object once it is located to center it in the gripper, (4) grasp and lift the spill object, (5) move to the vicinity of the final spill location, and then (6) lower and release the spill object. To minimize interference among robots in a relatively small space, ideally only one robot at a time should work on a given spill.

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

C. Experiments

We report here the experiments we conducted to test the ability of ALLIANCE to achieve fault-tolerant cooperative control of our team of mobile robots performing the hazardous waste cleanup mission. In all of the following experiments, teams of three R-2 robots were utilized in an environmental setup very similar to that depicted in figure 3; we will refer to these robots individually as GREEN, BLUE, and GOLD. All the robots began their missions at the room entrance, as shown in figure 9.

Figure 10 shows the action selection results of a typical experimental run when no robot failures occur; figure 11 shows the corresponding motivation levels during this run. As reflected in these figures, at the beginning of the mission, GREEN has the highest motivation to perform behavior set *find-locations-methodical*, causing it to initiate this action. This causes BLUE and GOLD to be satisfied for a while that the initial and final spill locations are going to be found; since no other task can currently be performed, they sit idle, waiting for the locations to be found.

However, they do not idle forever waiting on the locations to be found. As they wait, they become more and

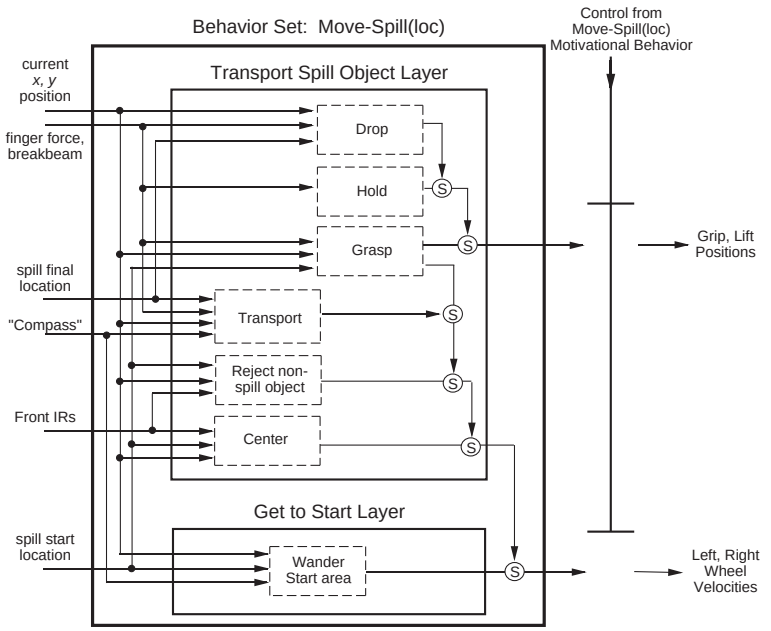

Fig. 7. The robot control organization within the *move-spill(loc)* behavior set.

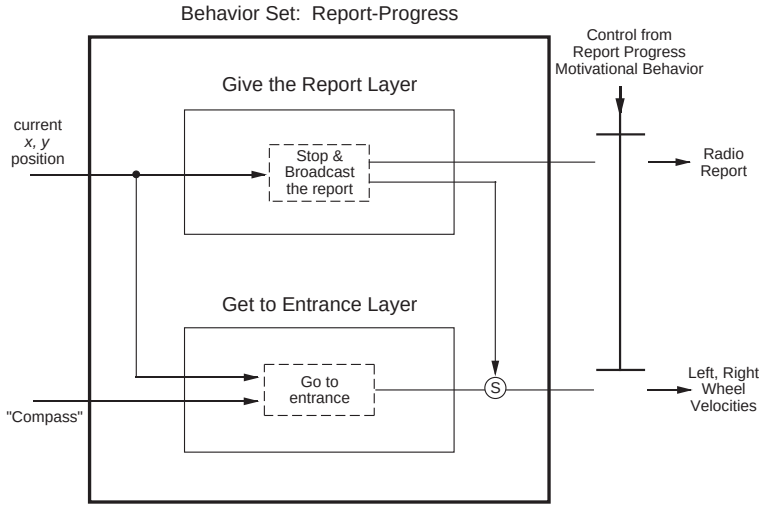

Fig. 8. The robot control organization within the *report-progress* behavior set.

more impatient over time, which can cause one of BLUE or GOLD to decide to find the spill and goal locations. Indeed, this does happen in a situation as shown in the photograph in figure 12, in which we intentionally interfere with GREEN's ability to find the spill and goal locations. As shown in the action trace of figure 13, (with the corresponding motivational trace in figure 14) this leads to one of the remaining robots — namely, BLUE — to activate its *find-locations-wander* behavior set. (Note that BLUE does not activate its *find-locations-methodical* behavior set because its side infrared sensors failed during previous runs, preventing BLUE from successfully accomplishing that behavior set. This behavior set is left in BLUE's repertoire to allow it to respond to some potential future event that may restore the working-order of the infrared sensors. Its motivations were altered based upon the L-ALLIANCE mechanism outlined in section III-E.) In this case, GREEN acquiesces its attempt to find the spill and goal locations to BLUE, since GREEN realized it was encountering difficulties of some sort. In either case, the robot finding the spill and goal locations reports these locations to the rest of the team.

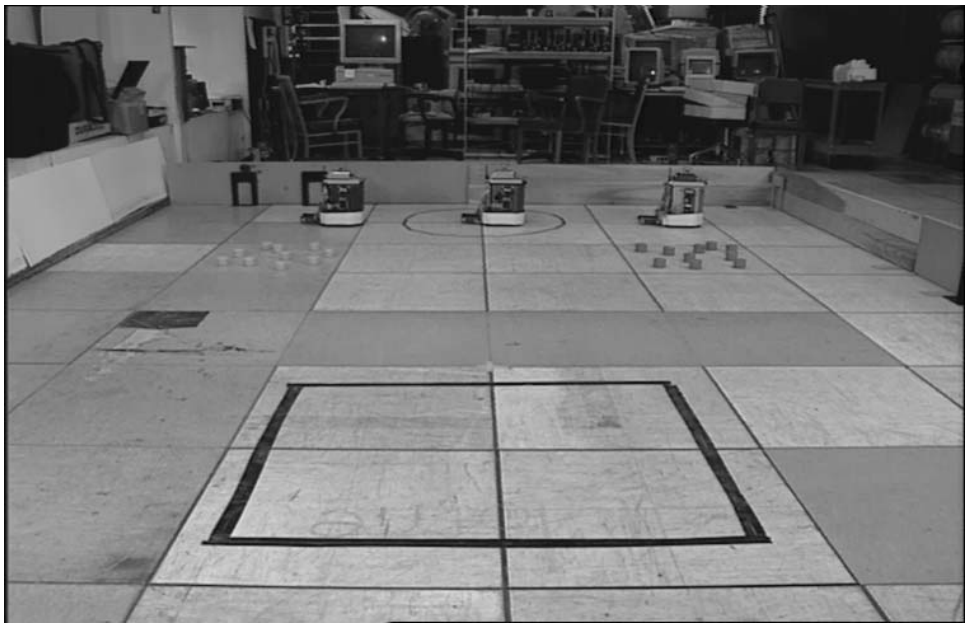

Fig. 9. The robot team at the beginning of the hazardous waste cleanup mission.

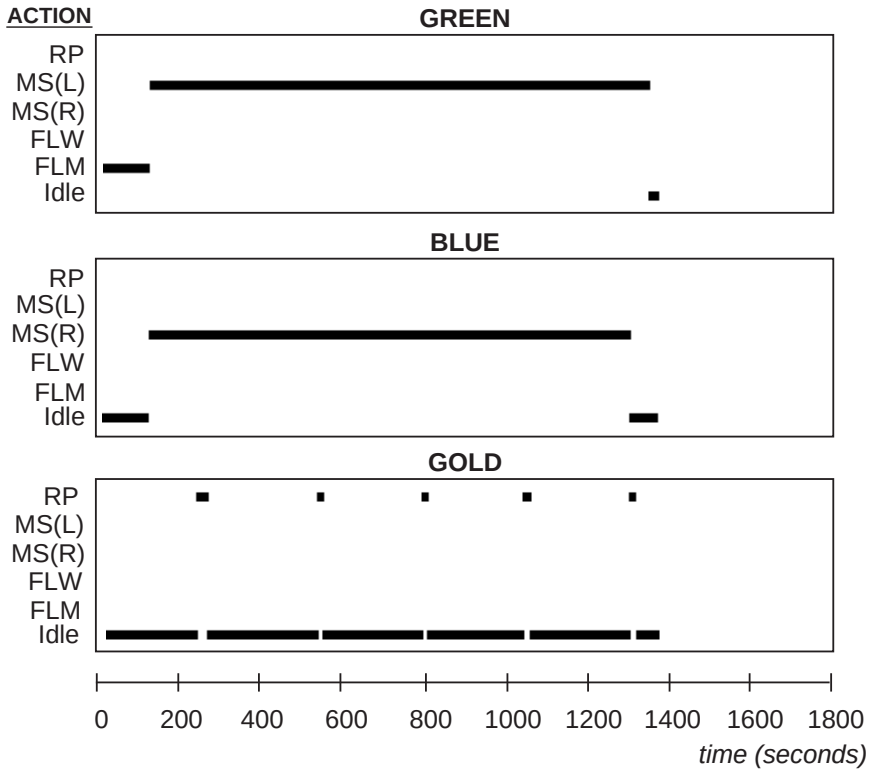

Fig. 10. Typical robot actions selected during experiment with no robot failures. This is one instance of many runs of this mission. In this and the following figures showing traces of action selections, the meanings of the abbreviations are as follows: RP stands for *report-progress*; MS(L) and MS(R) stand for *move-spill(left)* and *move-spill(right)*, respectively; FLW stands for *find-locations-wander*; and FLM stands for *find-locations-methodical*.

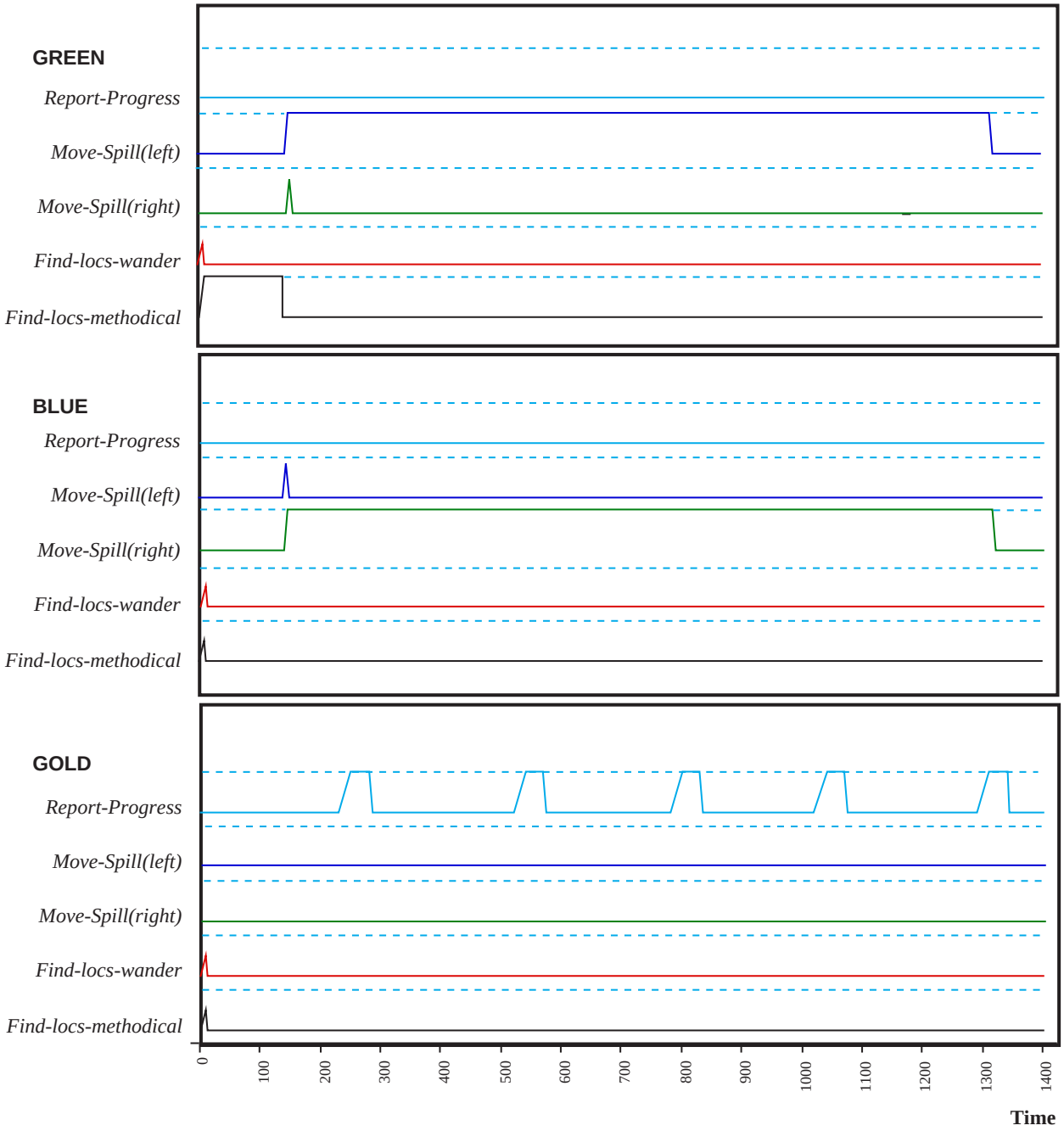

Fig. 11. Motivational levels for behavior sets during experiment with no robot failures. The dashed lines represent the thresholds of activation of each behavior set.

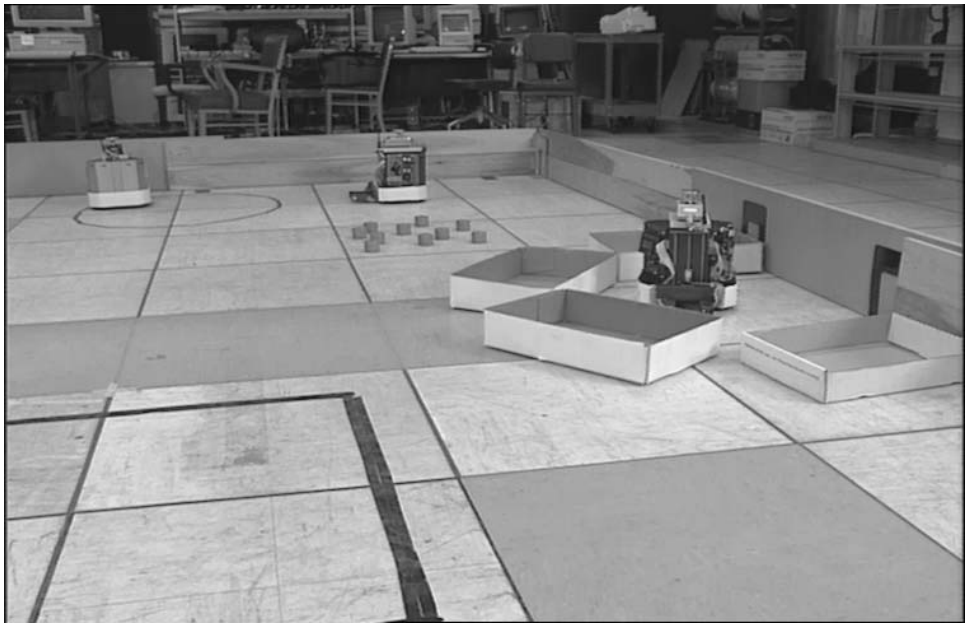

Fig. 12. Here, we interfere with GREEN, which is attempting to locate the spill and goal locations, thus preventing it from completing this task.

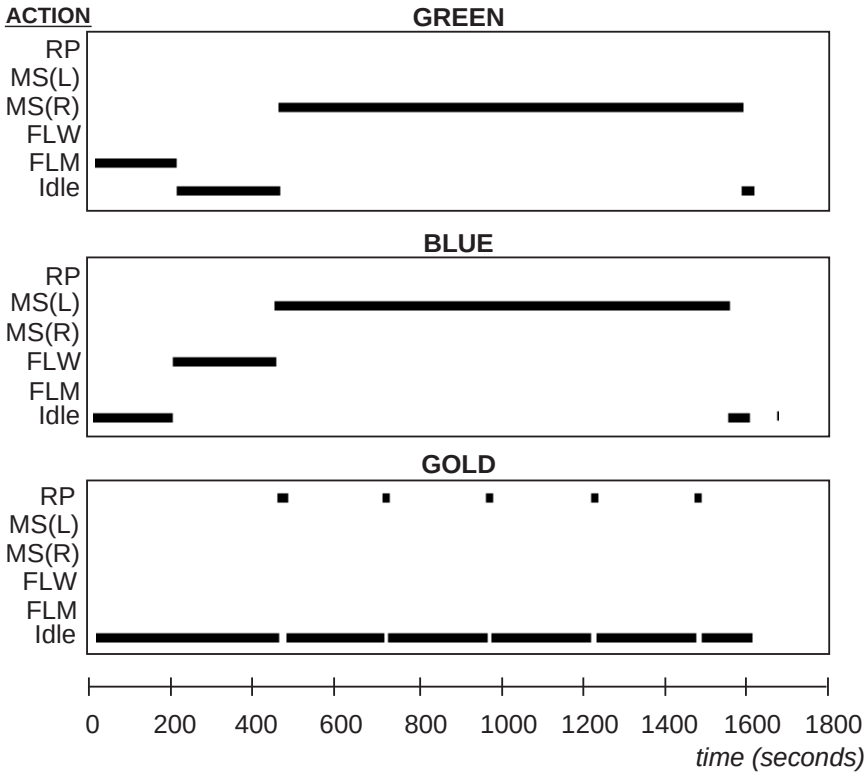

Fig. 13. Typical robot actions selected during experiment when the initial robot action of finding the spill fails. This is one instance of many runs of this mission.

At this point, the environmental feedback and knowledge of the spill and goal locations indicate to the robot team that the *move-spill(loc)* behavior set is applicable. As we see in figure 10, GREEN selects to move the *left* spill while BLUE selects to move the *right* spill. Since only one robot at a time should work on a given spill (as described in section IV-B), GOLD sits idle, satisfied that the left and right spills are going to be moved. Figure 15 shows a photograph of the robots at this stage in the mission.

In the meantime, the robots' impatience motivations to report the team's progress are increasing. Since GOLD is not performing any other tasks, it is the first to activate its *report-progress* behavior set. This reporting satisfies the remainder of the team, so they continue to move the two spills. This periodic reporting of the progress throughout the mission by GOLD is reflected in the diagrams in figures 10 and 13. In these particular examples, GOLD has effectively specialized as the progress reporting robot, whereas GREEN and BLUE have specialized as the move-spill robots. The mission continues in this way until both spills are moved from their starting location to the goal destination. Figure 16 shows two of the robots delivering spill objects to the goal destination.

To illustrate the effect of unexpected events on the action selection of the team, we next experimented with dynamically altering the composition of the team during the mission. Figure 17 shows the effect on the mission when we removed BLUE from the team. Figures 18, 19, and 20 show the corresponding motivation, impatience, and acquiescence levels for the three robots GREEN, BLUE, and GOLD. The removal of BLUE caused GOLD to become impatient that the *right* spill was not being moved, which in turn caused GOLD to activate its behavior set to move the *right* spill. However, this then effectively removes from the team the robot that is performing all of the progress reports, leading the remaining two robots — GREEN and GOLD — to have to interrupt their spill-moving activities to occasionally report the progress. A similar effect can be observed in figure 21 (with the corresponding motivations shown in figure 22), when we remove GOLD from the team.

D. Discussion

These experiments illustrate a number of primary characteristics we consider important in developing cooperative robotic teams. First of all, the cooperative team under ALLIANCE control is robust, in that robots are allowed to continue their actions only as long as they demonstrate their ability to have the desired effect on the world. This was illustrated in the experiments by BLUE and GOLD becoming gradually more impatient with GREEN's search for the spill. If GREEN did not locate the spill in a reasonable length of time then one of the remaining robots would take over that task, with GREEN acquiescing the task.

Secondly, the cooperative team is able to respond autonomously to many types of unexpected events either in the environment or in the robot team without the need for external intervention. As we illustrated, at any time during the mission, we could disable or remove robot team members, causing the remaining team members to perform those tasks that the disabled robot would have performed. Clearly, we could also have easily increased or decreased the size of the spill during the mission and the robots would not be adversely affected.

Third, the cooperative team need have no *a priori* knowledge of the abilities of the other team members to effectively accomplish the task. As previously noted, the learning/parameter update system, L-ALLIANCE, allows the team to improve its efficiency on subsequent trials whenever familiar robots are present.

Other characteristics of ALLIANCE have also been studied which show that this architecture allows robot teams to accomplish their missions even when the communication system providing it with the awareness of team member actions breaks down. Although the team's performance in terms of time and energy may deteriorate, at least the team is still able to accomplish its mission. Refer to [31] for a deeper discussion of these and related issues.

The primary weakness of ALLIANCE is its restriction to independent subtasks. As it is designed, one has to explicitly state ordering dependencies in the preconditions if the order of subtask completion is important. No mechanism for protecting subgoals is provided in ALLIANCE. For example, consider a janitorial service team of two robots that can both empty the garbage and clean the floor. However, let us say that the robots are clumsy in emptying the garbage, and usually drop garbage on the floor while they empty the trash. Then, it is logical that the trash should be emptied before beginning to clean the floor. Under ALLIANCE, this mission can be accomplished in one of three ways:

1. An explicit ordering dependency can be stated in advance between the two subtasks, causing the robots to automatically choose to empty the garbage first
2. The robots may fortuitously select to empty the garbage first, because of the parameter settings, or

3. The robots will be inefficient in their approach, cleaning the floor before or during the garbage emptying task, which leads to the need to clean the floor again.

Although all of these events lead to the mission being completed, efficiency is lost in the third situation, and additional explicit knowledge has to be provided in the first situation. For the applications that have been implemented in ALLIANCE (e.g. janitorial service, bounding overwatch, mock hazardous waste cleanup, cooperative box pushing), this limitation has not been a serious problem. Nevertheless, enhancing the ALLIANCE action selection mechanism to enable more efficient execution of these special case situations without the need for providing additional controlling information is a topic of future research.

V. Conclusions

We have presented a fully distributed, behavior based architecture called ALLIANCE, which facilitates fault tolerant mobile robot cooperation. A number of key characteristics of ALLIANCE provide these fault tolerant cooperative features. ALLIANCE enhances team robustness through the use of the motivational behavior mechanism which constantly monitors the sensory feedback of the tasks that can be performed by an individual robot, adapting the actions selected by that robot to the current environmental feedback and the actions of its teammates. Whether the environment changes to require the robots to perform additional tasks or to eliminate the need for certain tasks, ALLIANCE allows the robots to handle the changes fluidly and flexibly. This same mechanism allows robot team members to respond to their own failures or to failures of teammates, leading to adaptive action selection to ensure mission completion. ALLIANCE further enhances team robustness by making it easy for robot team members to deal with the presence of overlapping capabilities on the team. The ease with which redundant robots can be incorporated on the team provides the human team designer the ability to utilize physical redundancy to enhance the team's fault tolerance. The feasibility of this architecture for achieving fault tolerance has been illustrated through an example implemented on a physical robot team performing a laboratory version of hazardous waste cleanup.

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 three anonymous reviewers who provided valuable suggestions for improving earlier versions of this paper.

Appendix

A. Proofs of Termination

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

ALLIANCE is designed for applications involving a significant amount of uncertainty in the capabilities of robot team members which themselves operate in dynamic, unpredictable environments. Within this context, a key point of interest is whether the architecture allows the team to complete its mission at all, even in the presence of robot difficulties and failure. This section examines this issue by evaluating the performance of ALLIANCE in certain dynamic environments.

Let us consider realistic applications involving teams of robots that are not always able to successfully accomplish their individual tasks; we use the term *limitedly-reliable* robot to refer to such robots. The uncertainty in the expected effect of robots' actions clearly makes the cooperative control problem quite challenging. Ideally, ALLIANCE's impatience and acquiescence factors will allow a robot team to successfully reallocate actions as robot failures or dynamic changes in the environment occur. With what confidence can we know that this will happen in general? As we shall see below, in many situations ALLIANCE is guaranteed to allow a limitedly-reliable robot team to successfully accomplish its mission.

It is interesting to note that with certain restrictions on parameter settings, the ALLIANCE architecture is guaranteed to allow the robot team to complete its mission for a broad range of applications. We describe these circumstances here, along with the proof of mission termination.

We first define the notions of *goal-relevant capabilities* and *task coverage.*

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

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

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

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

We use the term *task coverage* to give a measure of the number of capabilities on the team that may allow some team member to achieve a given task. However, we cannot always predict robot failures; thus, at any point during a mission, a robot may reach a state from which it cannot achieve a task for which it has been designed. This implies that the expected task coverage for a given task in a mission may not always equal the *true* task coverage once the mission is underway.

Definition 2: Task coverage is given by:

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

The task coverage measure is useful for composing a team of robots to perform a mission from an available pool of heterogeneous robots. At a minimum, we need the team to be composed so that the task coverage of all tasks in the mission equals 1. This minimum requirement ensures that, for each task required in the mission, a robot is present that has some likelihood of accomplishing that task. Without this minimum requirement, the mission simply cannot be completed by the available robots. Ideally, however, the robot team is composed so that the task coverage for all tasks is greater than 1. This gives the team a greater degree of redundancy and overlap in capabilities, thus increasing the reliability and robustness of the team amidst individual robot failures.

Let us now define the notion of *sufficient task coverage* as follows:

Condition 1: **(Sufficient task coverage):**

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

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

Now, we define the notion of an *active* robot team, since we consider our robots to be useful only if they can be motivated to perform some action:

Definition 3: An *active* robot team is a group of robots, R, such that:

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

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

Finally, 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 ϵ such that z is reduced by a finite amount which is at least some constant δ greater than 0, or (2) r_i experiences a failure with respect to task w. Additionally, if z ever increases, the increase is due to an influence external to the robot team.

Condition 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 ϵ. One exception, however, is if a robot failure has occurred that prevents robot r_i from accomplishing task w, even if r_i has been designed to achieve task w.

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. As we shall see, this assumption is necessary to prove the effectiveness of ALLIANCE in certain situations. Of course, this does not preclude dynamic environmental changes from increasing the workload of the robot team, which ALLIANCE allows the robots to handle without problem.

What we now show is that whenever conditions 1 and 2 hold for a limitedly-reliable, active robot team, then either ALLIANCE allows the robot team to accomplish its mission, or some robot failure occurs. Furthermore, if a robot failure occurs, then we can know that any task that remains incomplete at the end of the mission is either a task that the failed robot was designed to accomplish, or a task that is dependent upon the capabilities of that robot.

We can now show the following:

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

Proof:

First, we show that the calculation of the motivational behavior guarantees that each robot eventually activates a behavior set whose sensory feedback indicates that the corresponding task is incomplete. From equation 1 in section III-D.7, we see that at time t, robot r_i's motivation $m_{ij}(t)$ to perform behavior set a_{ij} either (1) goes to 0, or (2) changes from $m_{ij}(t-1)$ by the amount $impatience_{ij}(t)$. The motivation goes to 0 in one of four cases: (1) if the sensory feedback indicates that the behavior set is no longer applicable, (2) if another behavior set becomes active, (3) if some other robot has taken over task $h_i(a_{ij})$, or (4) if the robot has acquiesced its task. If the sensory feedback indicates that the behavior set is no longer applicable, we know either that the task $h_i(a_{ij})$ must be successfully accomplished, or the robot's sensory system has failed. If another behavior set a_{ik} becomes active in r_i, then at some point task $h_i(a_{ij})$ will either become complete, thus allowing r_i to activate behavior set a_{ij}, or the robot has failed. If some other robot has taken over task $h_i(a_{ij})$, then either that other robot will eventually accomplish task $h_i(a_{ij})$, thus eliminating the need to activate task a_{ij}, or robot r_i will become impatient with that other robot. Since r_i is *active*, then we know that $impatience_{ij}(t)$ is greater than or equal to $\min_k(\delta\_slow_{ij}(k,t))$, which is greater than 0. Therefore, we can conclude that an idle, yet *active* robot always has a strictly increasing motivation to perform some incomplete task. At some point, the finite threshold of activation, θ, will thus be surpassed for some behavior set, causing r_i to activate the behavior set corresponding to task $h_i(a_{ij})$.

We now build upon these observations to prove that either the mission becomes accomplished, or a robot failure occurs.

PART I (Either ALLIANCE succeeds or a robot fails):

Assume no robot fails. Then after a robot r_i has performed a task w for any period of time greater than ϵ, one of five events can occur:

1. Robot r_j takes over task w, leading robot r_i to acquiesce.

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

Since Condition 2 holds, we know that the first four cases reduce the amount of work left to complete task w by at least a positive, constant amount δ. Since the amount of work left to accomplish any task is finite, the task must eventually be completed in finite time. In the fifth case, since task w is completed, the sensory feedback of the robots no longer indicates the need to perform task w, and thus the robots will go on to some other task required by the mission.

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

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

First, we show that if a task is not in A_F, then it will be successfully completed. Let w be some task required by the mission that is not included in A_F. Since Condition 1 holds and this robot team is active, there must be some robot on the team that can successfully accomplish w. Thus, as long as w remains incomplete, one of these successful robots will eventually activate its behavior set corresponding to the task w; since condition 2 holds, that task will eventually be completed in finite time. Thus, all tasks not dependent upon the capabilities of a failed robot are successfully completed in ALLIANCE.

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

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

Note that it is not required here that robot team members be *aware* of the actions of their teammates in order to guarantee that ALLIANCE allows the team to complete its mission under the above conditions. However, awareness does have an effect on the *quality* of the team's performance, both in terms of the time and the energy required to complete the mission. These effects on team performance are discussed in [31].

References

[1] 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.

[2] 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.

[3] Tucker Balch and Ronald C. Arkin. Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1(1):1–25, 1994.

[4] Gerardo Beni and Jing Wang. On cyclic cellular robotic systems. In *Japan – USA Symposium on Flexible Automation*, pages 1077–1083, Kyoto, Japan, 1990.

[5] Alan Bond and Less Gasser. *Readings in Distributed Artificial Intelligence*. Morgan Kaufmann, 1988.

[6] 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.

[7] Rodney A. Brooks. Elephants don't play chess. *Robotics and Autonomous Systems*, 6:3–15, 1990.

[8] Rodney A. Brooks. New approaches to robotics. *Science*, 253:1227–1232, September 1991.

[9] 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.

[10] Stephanie Cammarata, David McArthur, and Randall Steeb. Strategies of cooperation in distributed problem solving. In *Proceedings of 8th International Joint Conference on Artificial Intelligence*, pages 767–770, 1983.

[11] Y. Uny Cao, Alex Fukunaga, Andrew Kahng, and Frank Meng. Cooperative mobile robotics: Antecedents and directions. In *Proceedings of 1995 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS '95)*, pages 226–234, 1995.

[12] K. L. Chung. *Elementary Probability Theory with Stochastic Processes*. Springer-Verlag, New York, 1974.

[13] Paul Cohen, Michael Greenberg, David Hart, and Adele Howe. Real-time problem solving in the Phoenix environment. COINS Technical Report 90-28, U. of Mass., Amherst, 1990.
[14] R. W. Conway, W. L. Maxwell, and L. W. Miller. *Theory of Scheduling*. Addison-Wesley, Reading, Massachusetts, 1967.
[15] Randall Davis and Reid Smith. Negotiation as a metaphor for distributed problem solving. *Artificial Intelligence*, 20(1):63–109, 1983.
[16] 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.
[17] 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.
[18] Edmund Durfee and Thomas Montgomery. A hierarchical protocol for coordinating multiagent behaviors. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, pages 86–93, 1990.
[19] 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.
[20] Marcus Huber and Edmund Durfee. Observational uncertainty in plan recognition among interacting robots. In *Proceedings of the 1993 IJCAI Workshop on Dynamically Interacting Robots*, pages 68–75, 1993.
[21] Thomas Kreifelts and Frank von Martial. A negotiation framework for autonomous agents. In *Proceedings of the Second European Workshop on Modeling Autonomous Agents and Multi-Agent Worlds*, pages 169–182, 1990.
[22] 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.
[23] Victor R. Lesser and Daniel D. Corkill. The distributed vehicle monitoring testbed: A tool for investigating distributed problem solving networks. *AI Magazine*, pages 15–33, Fall 1983.
[24] Bruce MacLennan. Synthetic ethology: An approach to the study of communication. In *Proceedings of the 2nd interdisciplinary workshop on synthesis and simulation of living systems*, pages 631–658, 1991.
[25] Maja Mataric. Designing emergent behaviors: From local interactions to collective intelligence. In J. Meyer, H. Roitblat, and S. Wilson, editors, *Proc. of the Second Int'l Conf. on Simulation of Adaptive Behavior*, pages 432–441. MIT Press, 1992.
[26] 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.
[27] Lynne E. Parker. Adaptive action selection for cooperative agent teams. In Jean-Arcady Meyer, Herbert Roitblat, and Stewart Wilson, editors, *Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, pages 442–450. MIT Press, 1992.
[28] Lynne E. Parker. ALLIANCE: An architecture for fault tolerant, cooperative control of heterogeneous mobile robots. In *Proc. of the 1994 IEEE/RSJ/GI Int'l Conf. on Intelligent Robots and Systems (IROS '94)*, pages 776–783, Munich, Germany, Sept. 1994.
[29] 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.
[30] Lynne E. Parker. Fault tolerant multi-robot cooperation. MIT Artificial Intelligence Lab Videotape AIV-9, December 1994.
[31] 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).
[32] Jeffery Rosenschein and M. R. Genesereth. Deals among rational agents. In *Proceedings of the International Joint Conference on Artificial Intelligence*, pages 91–99, 1985.
[33] Reid G. Smith and Randall Davis. Frameworks for cooperation in distributed problem solving. *IEEE Transactions on Systems, Man, and Cybernetics*, SMC-11(1):61–70, January 1981.
[34] Luc Steels. Cooperation between distributed agents through self-organization. In Yves Demazeau and Jean-Pierre Muller, editors, *Decentralized A.I.* Elsevier Science, 1990.
[35] 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.
[36] 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.
[37] 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.
[38] Gregory M. Werner and Michael G. Dyer. Evolution of communication in artificial organisms. In *Proceedings of the 2nd interdisciplinary workshop on synthesis and simulation of living systems*, pages 659–687, 1991.
[39] Gilad Zlotkin and Jeffery Rosenschein. Negotiation and conflict resolution in non-cooperative domains. In *Proceedings of the Eighth National Conference on AI*, pages 100–105, 1990.

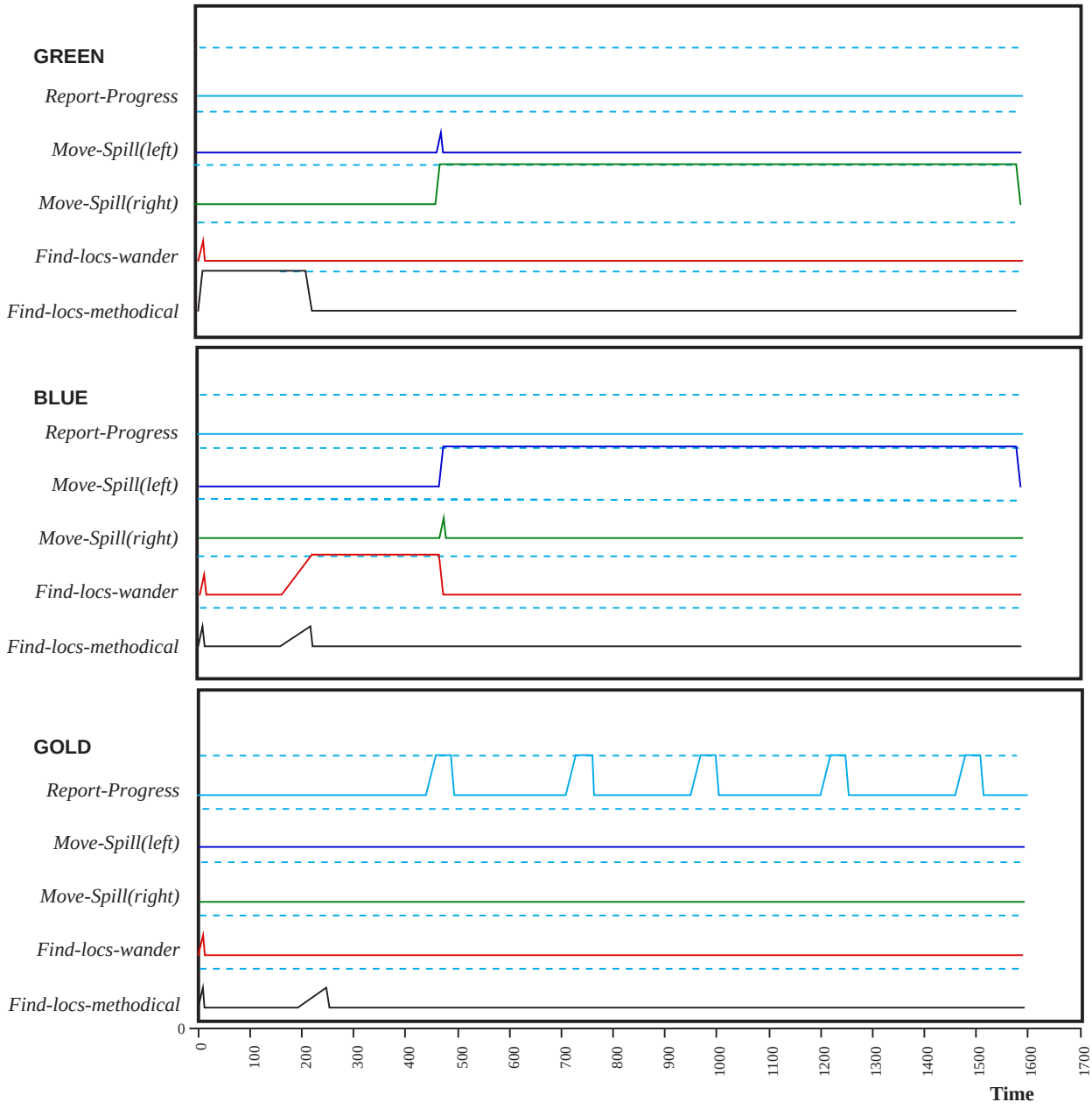

Fig. 14. Motivational levels for behavior sets during experiment with spill finding error. The dashed lines represent the thresholds of activation of each behavior set.

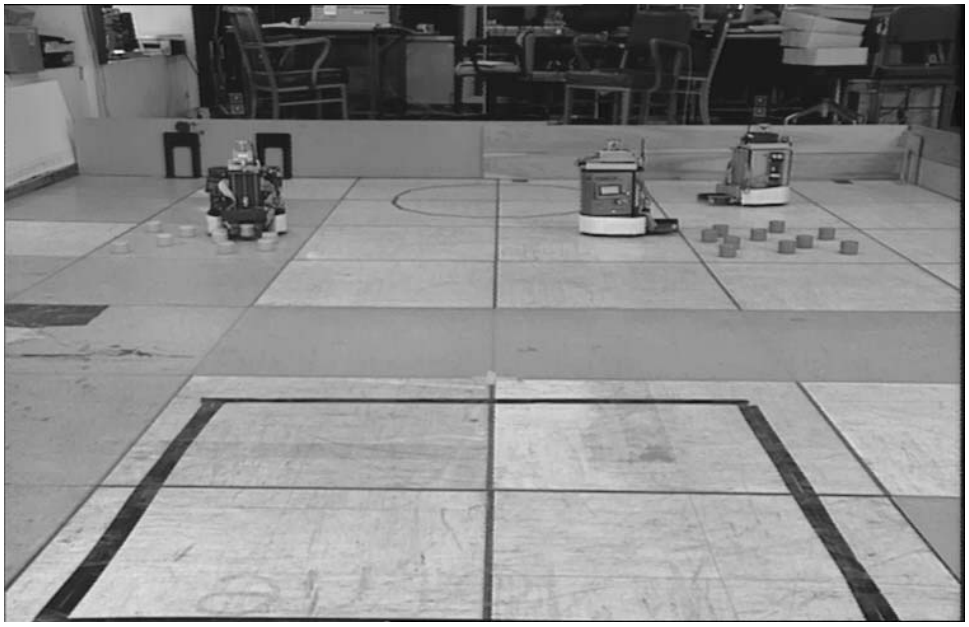

Fig. 15. Now knowing the location of the two spills, two R-2 robots are in the process of moving their respective spills to the goal location.

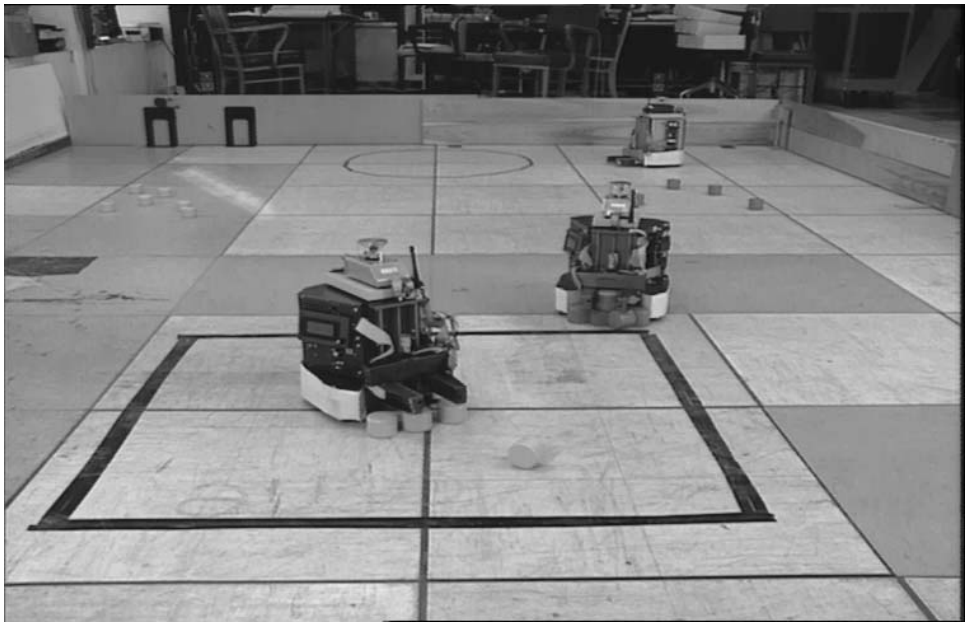

Fig. 16. Robots delivering spill objects to the goal destination.

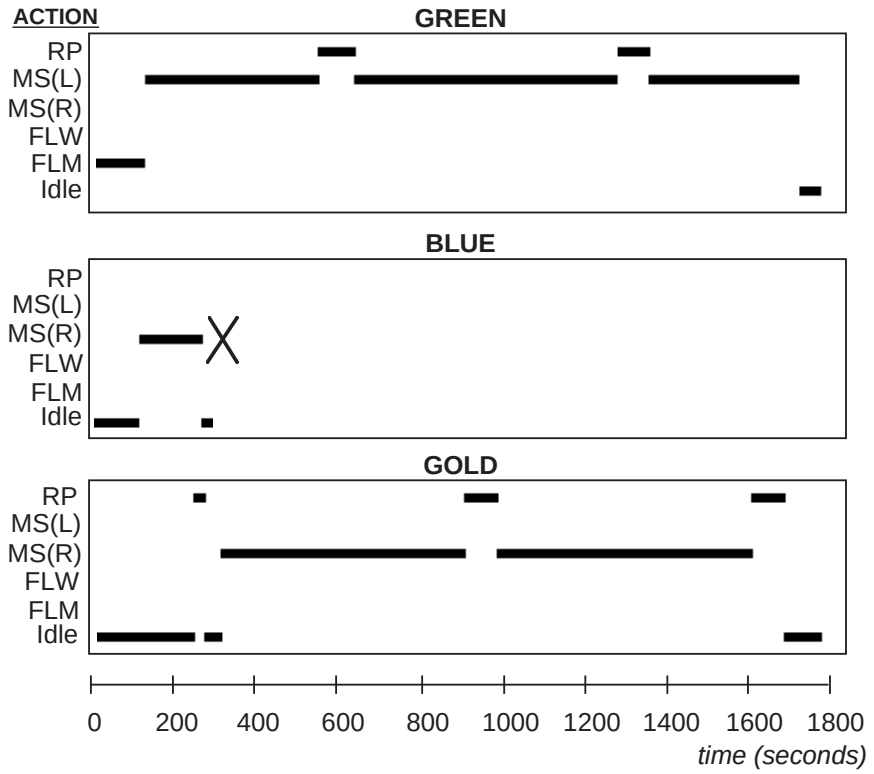

Fig. 17. Typical robot actions selected during experiment when one of the robot team members which is moving a spill is removed. This is one instance of many runs of this mission. In this figure, the "X" indicates the point in time when the corresponding robot was removed from the robot team.

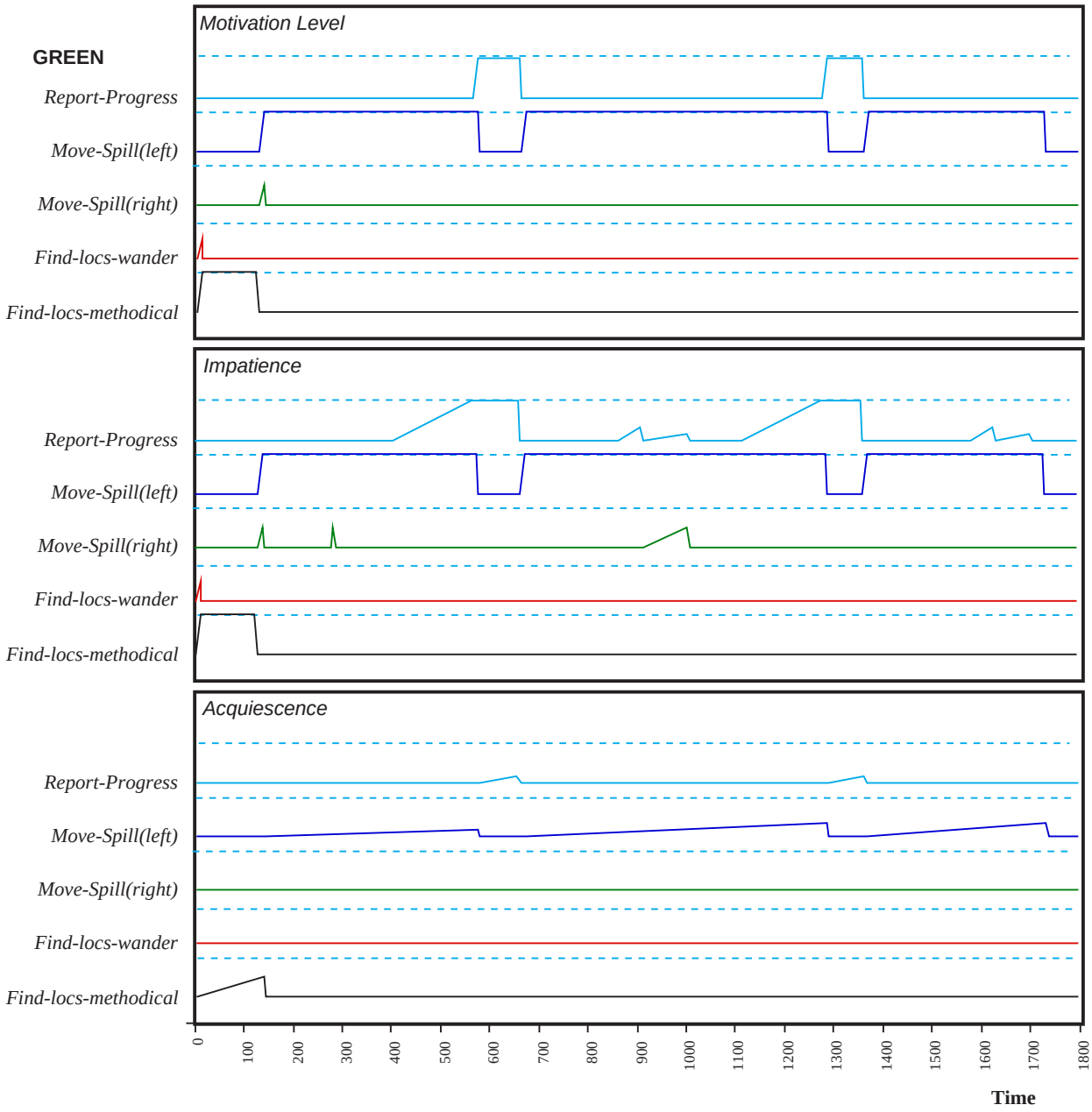

Fig. 18. Motivational, impatience, and acquiescence levels for behavior sets of GREEN during experiment when a team member that is moving a spill is removed. The dashed lines represent the thresholds of activation.

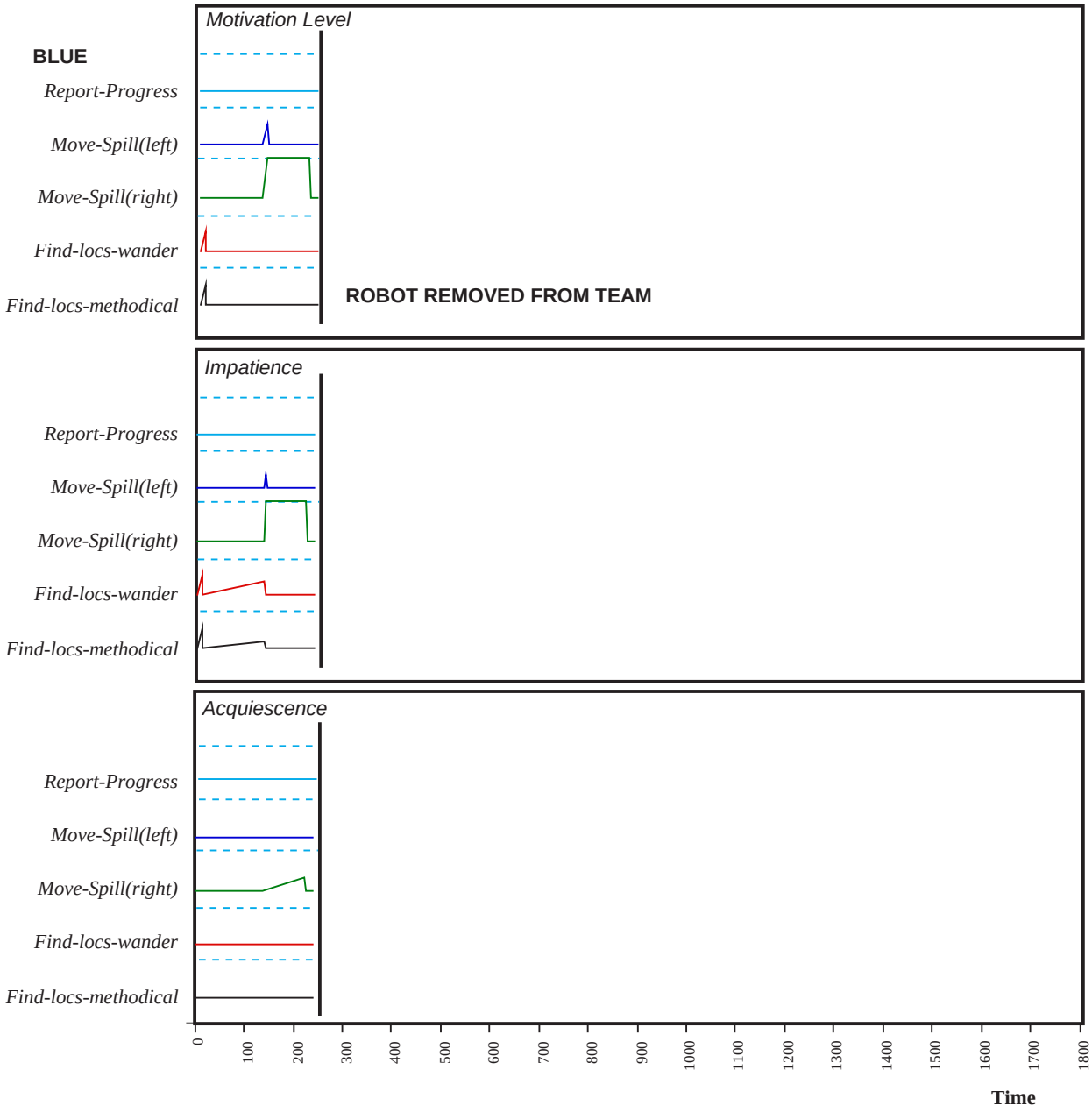

Fig. 19. Motivational, impatience, and acquiescence levels for behavior sets of BLUE during experiment when a team member that is moving a spill is removed. The dashed lines represent the thresholds of activation.

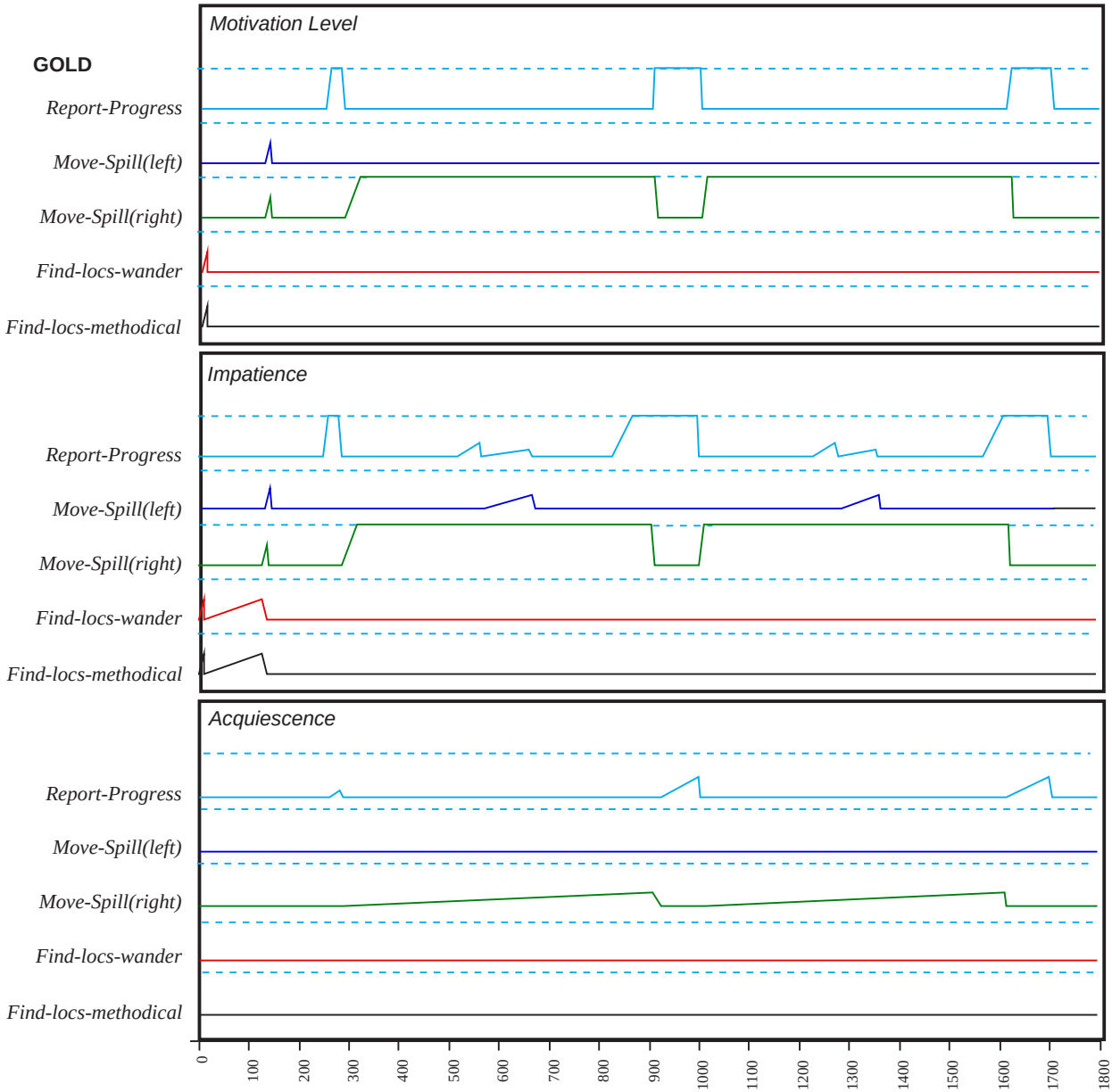

Fig. 20. Motivational, impatience, and acquiescence levels for behavior sets of GOLD during experiment when a team member that is moving a spill is removed. The dashed lines represent the thresholds of activation.

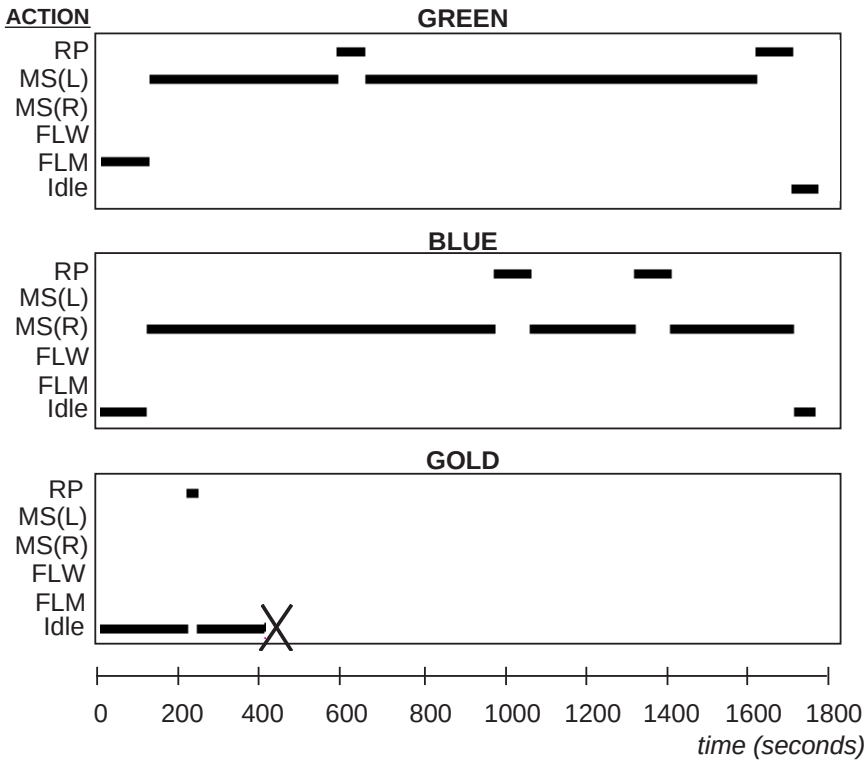

Fig. 21. Typical robot actions selected during experiment when one of the robot team members which is reporting the progress is removed. In this figure, the "X" indicates the point in time when the corresponding robot was removed from the robot team. This is one instance of many runs of this mission.

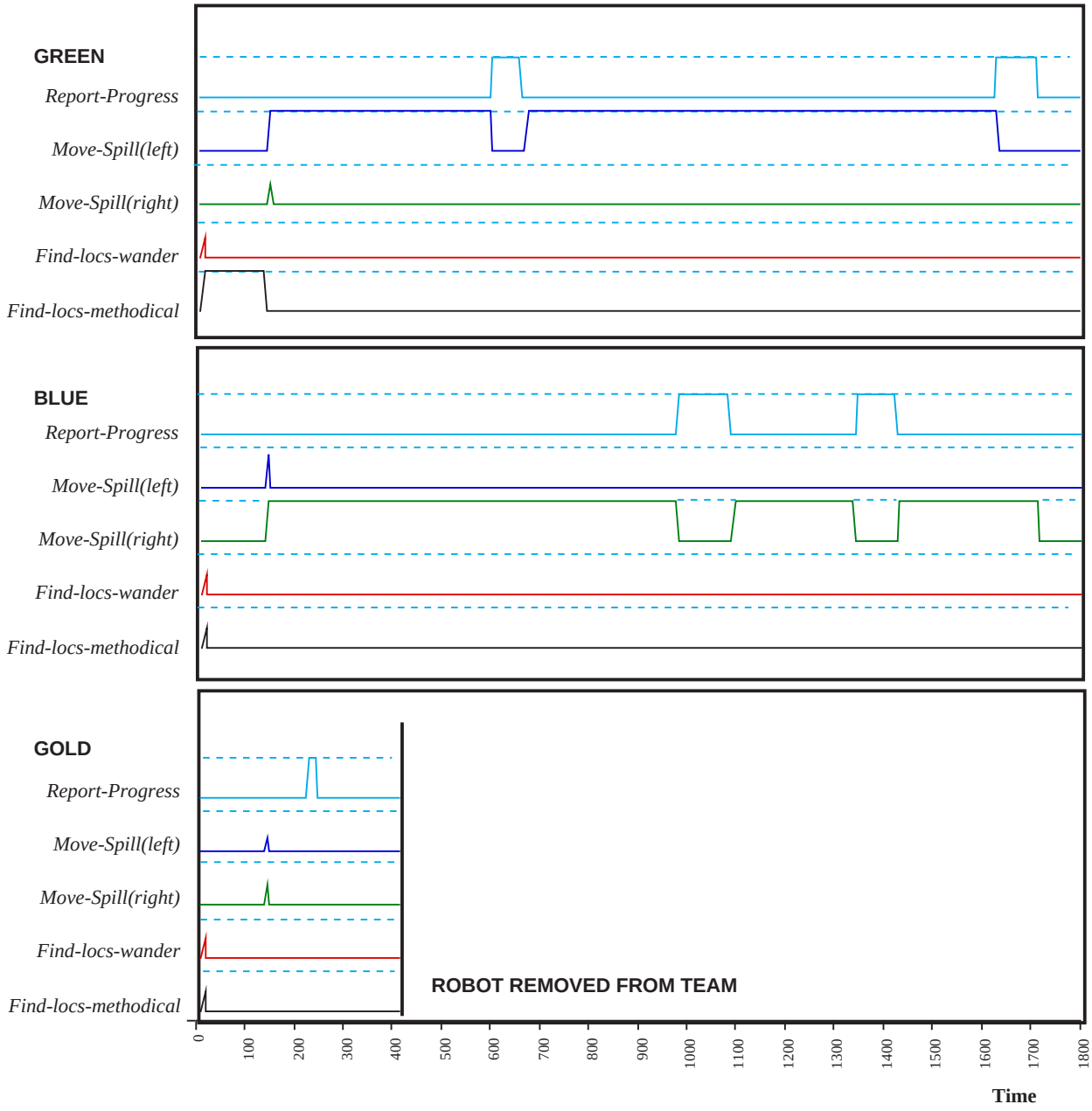

Fig. 22. Motivational levels for behavior sets during experiment when a team member that is reporting the progress is removed. The dashed lines represent the thresholds of activation of each behavior set.