L-ALLIANCE: TASK-ORIENTED MULTI-ROBOT LEARNING IN BEHAVIOR-BASED SYSTEMS

Lynne E. Parker

Center for Engineering Systems Advanced Research, Oak Ridge National Laboratory, P. O. Box 2008, Mailstop 6364, Oak Ridge, TN 37831-6364 USA

Abstract— A large application domain for multi-robot teams involves task-oriented missions, in which potentially heterogeneous robots must solve several distinct tasks. Previous research addressing this problem in multi-robot systems has largely focused on issues of efficiency, while ignoring the real-world situated robot needs of fault tolerance and adaptivity. This paper addresses this problem by developing an architecture called L-ALLIANCE that incorporates task-oriented action selection mechanisms into a behavior-based system, thus increasing the efficiency of robot team performance while maintaining the desirable characteristics of fault tolerance and adaptivity. We present our investigations of several competing control strategies and derive an approach that works well in a wide variety of multi-robot task-oriented mission scenarios. We provide a formal model of this technique to illustrate how it can be incorporated into any behavior-based system.

Key words: Multi-robot learning, behavior-based, L-ALLIANCE, multi-robot cooperation, action selection, fault tolerance.

1. INTRODUCTION

Consider the following problem: a team of heterogeneous mobile robots is required to perform a task-oriented mission. Each robot on the team is programmed with the capabilities necessary to perform a subset of the tasks required by the current mission. In order to reduce the effects of bottlenecks and single points of failure, the robots are designed to overlap in the tasks they are able to accomplish, although they may demonstrate different levels of performance in accomplishing the same task due to robot heterogeneity. The capabilities of the robots in such a mission may change over time, due either to robot subsystem failure, or perhaps due to robot action learning. The problem we are to solve, then, is that of enabling these heterogeneous robot team members to select the proper actions to perform during a mission so as to maintain a high level of fault tolerance and adaptivity while not sacrificing the efficiency of the mission performance.

A real-world instance of this type of problem is a material handling mission involving multiple types of containers and multiple types of robots that have heterogeneous abilities to carry or to push the containers from point to point. The robots need to be able to select which container to move based upon their

own capabilities and the capabilities of their functioning teammates.

Traditional solutions to this type of task-oriented multi-agent system would likely employ one of two approaches. Either, (1) the robots would be preprogrammed to always move containers in a fixed order, or (2) the system would employ a planning and hierarchical problem decomposition process, followed by allocation of tasks to agents (either through a centralized assignment, or through a negotiation/bidding process), and then agent execution; if an agent were to fail at its task, then a replanning process would occur that seeks to remedy the problem by reassigning the task(s).

However, these solutions are unsatisfactory for most real robot systems, since the first solution is quite inflexible (e.g. what if a new type of robot were added to the system with a different capability profile, or if the robots were frequently moved from one material handling job to another, working with different types of robots) and the second solution is prone to single-point failure (e.g. due to communication failure, robot failure, central planning failure, etc.), leading to the severe disruption or total breakdown of the system.

In previous research [1], Brooks has shown that behavior-based systems provide a high level of fault tolerance and adaptivity that is generally difficult to demonstrate in systems using a more traditional AI approach. Thus, researchers in multi-robot systems have commonly used behavior-based systems to achieve cooperative control (see the following section for a review). However, the majority of these techniques are not well-suited for task-oriented problems, in which various distinct tasks must be accomplished by the robot team.

Our objective is to extend the usefulness of cooperative behavior-based systems by developing a methodology that takes advantage of the fault-tolerant, adaptive characteristics of behavior-based systems while also enabling efficient task-oriented solutions by multi-agent teams. In this paper, we introduce a method called L-ALLIANCE that achieves this objective. Our approach is built upon an earlier system we have developed, called ALLIANCE, for fault tolerant multi-robot control. In the next section, we briefly describe related work in this area. Section 3 then describes the L-ALLIANCE approach, first discussing the NP-hardness of the problem we are addressing, then introducing several potential approaches to incorporating task-oriented efficiency into the ALLIANCE behavior-based multi-robot system. Experimental results are presented in Section 4 that compare the effectiveness of these strategies in simulation. In Section 5, we discuss these results and briefly describe the implementation of our derived approach on a real robot team to verify its applicability. We conclude the paper in Section 6.

2. RELATED WORK

A significant amount of previous research has been accomplished in the areas of multi-robot systems and optimal task allocation/scheduling; research interest in multi-agent learning systems has also grown in recent years. The multi-robot

systems work can be loosely divided into two categories: *swarm* cooperation and *intentional* cooperation. The largest body of this research is in swarm cooperation, which deals with the study of large numbers of (usually) homogeneous robots (e.g., [2,3,4,5,6,7,8,9,10,11,12]). The difficult problem addressed in these systems is predicting the global behavior of the collective from the design of the control laws in the individual agent. Such approaches usually rely on mathematical convergence results (such as the random walk theorem [13]) that indicate the desired outcome over a sufficiently long period of time.

The "intentional" cooperation research employs a more directed type of cooperation that usually requires several distinct tasks be performed. These missions usually require a smaller number of possibly heterogeneous mobile robots than the swarm-type of applications. Key issues in these systems include robust task allocation, efficient team performance, and multi-robot coordination. Nearly all of the previous work on heterogeneous physical robots (e.g. [14,15,16,17]). uses a traditional artificial intelligence approach, which breaks the robot controller into modules for sensing, world modeling, planning, and acting rather than the functional decomposition of behavior-based approaches. However, although the need for fault tolerance is often noted in this intentional cooperation research, the approaches are typically vulnerable to single-point failures, and are largely limited in their ability to deliver real-time performance in a dynamic world because they do not adequately address the situatedness and embodiment of physical robots.

A huge amount of research in optimal task allocation and scheduling has been accomplished previously (e.g. [18]). 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.

Previous work in multi-robot learning is currently limited, although the topic is gaining increased interest. See [19] for several recent efforts in this area. Our work is different from most of this earlier work in that it applies multi-robot learning in behavior-based systems to task-oriented missions.

3. APPROACH

To achieve efficiency in task-oriented multi-robot systems, we built upon our behavior-based architecture, called ALLIANCE [20,21] that facilitates the execution of task-oriented missions by teams of heterogeneous mobile robots. We implemented a learning mechanism in this architecture through the use of parameter tuning. Since the problem we are addressing is NP-hard, we investigated several heuristic approaches and tested and compared them extensively in simulation. We then validated our results in a simple robot application to verify its applicability in real robot systems.

We begin this section by showing that the efficient action selection problem addressed in this paper is NP-hard, even if all information is known in advance

(which is not generally true). We then present the L-ALLIANCE mechanism and describe the heuristic approaches we studied for achieving efficiency in task-oriented behavior-based systems.

3.1. NP-Hardness

To understand the difficulty of our action selection problem, we now show that even a simplified version of this problem, in which the performance capabilities of all robots are known in advance, is NP-hard.

Let $R = \{r_1, r_2, ..., r_n\}$ represent the set of n robots on a cooperative team, and the set $T = \{task_1, task_2, ..., task_m\}$ represent the m independent tasks required in the current mission. Each robot in R has a number of high-level task-achieving functions that it can perform, represented by the set $A_i = \{a_{i1}, a_{i2}, ...\}$. Since different robots may have different ways of performing the same task, we define the set of n functions H, where $H : A_i \rightarrow T$, $H = \{h_1(a_{1k}), h_2(a_{2k}), ..., h_n(a_{nk})\}$, and $h_i(a_{ik})$ returns the task $task_j$ that robot r_i is working on when it performs the high-level function a_{ik}.

We denote the metric evaluation function as $q(a_{ij})$, which returns the "quality" of the action a_{ij} as measured by a given metric. Here, we consider the metric of average task completion time, although many other metrics could be used. Finally, we define the tasks a robot r_i elects to perform during a mission as the set $U_i = \{a_{ij} | \text{robot } r_i \text{ performs task } h_i(a_{ij}) \text{ during the current mission}\}$.

In the most general form of this problem, distinct robots may have different collections of capabilities; thus, we do not assume that $\forall i.\forall j.(A_i = A_j)$. Further, if different robots can perform the same task, they may perform that task with different qualities; thus, we do not assume that if $h_i(a_{ix}) = h_j(a_{jy})$, then $q(a_{ix}) = q(a_{jy})$. For the simplified case, we will assume that these robot performance measurements are known in advance. The formal Heterogeneous Robot Action Selection Problem (HRASP) can then be stated for each robot r_i as follows: Given T, A_i, and $h_i(a_{ik})$, determine the set of actions U_i such that $\forall i.U_i \subseteq A_i$ and $\forall j.\exists i.\exists k.((task_j = h_i(a_{ik}))$ and $(a_{ik} \in U_i))$ and the following is minimized, according to the time performance metric: $max_i(\sum_{a_{ik} \in U_i} q(a_{ik}))$.

It can be easily shown that the efficiency problem, HRASP, is NP-hard by restriction to the well-known NP-complete problem PARTITION [22]. (See [21] for the proof.) Thus, since this heterogeneous action selection problem is NP-hard, we cannot expect the robot team to be able to derive an optimal action selection policy in a reasonable length of time. We look instead to heuristic approximations to the problem that work well in practice.

3.2. ALLIANCE Overview

The foundation of our approach to task-oriented multi-robot learning in behavior-based systems is the ALLIANCE architecture, which has been reported earlier in [20,21]. ALLIANCE is a behavior-based, fully distributed architecture

for multi-robot cooperative control in robot missions involving loosely coupled, largely independent tasks. Robots under this architecture possess a variety of high-level functions (modeled as *behavior sets*) that they can perform during a mission, and must at all times select an appropriate action based on the requirements of the mission, the activities of other robots, the current environmental conditions, and their own internal states. Since cooperative robotic teams often work in dynamic and unpredictable environments, this software architecture allows the team members to respond robustly and reliably to unexpected environmental changes and modifications in the robot team that may occur due to mechanical failure, the learning of new skills, or the addition or removal of robots from the team by human intervention. This is achieved through the interaction of mathematically modeled motivations of behavior, such as impatience and acquiescence, within each individual robot. These motivations allow robots to take over tasks from other team members (i.e., become *impatient*) if those team members do not demonstrate their ability — through their effect on the world — to accomplish those tasks. Similarly, they allow a robot to give up its own current task (i.e., *acquiesce*) if its sensory feedback indicates that adequate progress is not being made to accomplish that task. The rate at which robots become impatient or acquiesce is dependent upon control parameter settings. It is these control parameters that are automatically updated by the L-ALLIANCE mechanism to achieve increased efficiency in robot team performance.

3.3. L-ALLIANCE Mechanism

In this subsection, we describe our approach to the development of a task-oriented multi-robot control architecture that increases robot team performance without sacrificing the desirable characteristics of fault tolerance and adaptivity. We first state two assumptions under which we developed our methodology, followed by a description of the performance monitors incorporated into L-ALLIANCE. We then describe the various control strategies we studied to increase the efficiency of multi-robot team performance in terms of the L-ALLIANCE formal model. The formal model derived as a result of our studies is given in the Appendix.

In the L-ALLIANCE approach, 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.

3.3.1. Performance Monitors. Figure 1 L-ALLIANCE-arch illustrates the L-ALLIANCE architecture. This architecture extends the ALLIANCE architecture by incorporating the use of performance monitors for each motivational behavior within each robot. Each monitor is responsible for observing, eval-

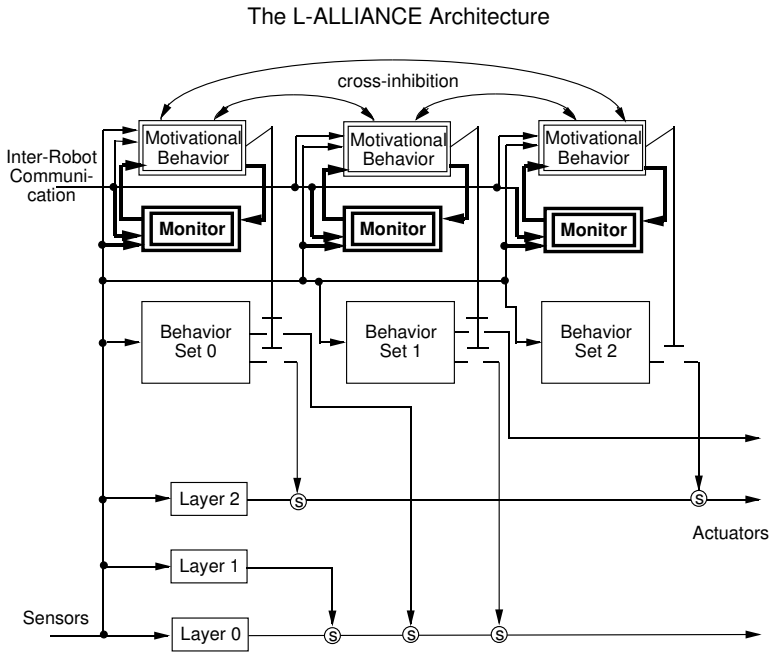

Figure 1: The L-ALLIANCE architecture.

uating, and cataloging the performance of any robot team member (including itself) whenever it performs the task corresponding to that monitor's respective behavior set. Formally, robot r_i, programmed with the b behavior sets $A = \{a_{i1}, a_{i2}, ..., a_{ib}\}$, 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. It is important to note that a robot r_i does not keep track of the task completion times for capabilities of other robots that r_i does not share. This allows the L-ALLIANCE architecture to scale favorably as the mission size increases.

3.3.2. Two L-ALLIANCE Control Phases. If robot team members are on a *training* mission, whose sole purpose is to allow robots to become familiar with themselves and with their teammates, then they can explore their capabilities without concern for possible mission failure. On the other hand, if the robots are on a *live* mission, then the team has to ensure that the mission is completed as efficiently as possible. Even so, as they perform a live mission, the robots should take advantage of the opportunity to learn about the robot capabilities

that are demonstrated. Thus, one of two high-level control phases are utilized for robot team members under L-ALLIANCE, depending upon the type of the team's mission. During training missions, the robots enter the *active learning* phase, whereas during live missions, they enter the *adaptive learning* phase. In the *active learning* phase, each robot selects its next action randomly from those actions that are: (1) currently incomplete, as determined from the sensory feedback, and (2) currently not being executed by any other robot, as determined from broadcast communication messages. While they perform their tasks, the robots are maximally patient and minimally acquiescent, meaning that a robot neither tries to preempt another robot's ongoing task, nor does it acquiesce its own current action to another robot. During the active learning phase, each monitor mon_{ij} in each robot r_i monitors and catalogues the performance of all robots r_k that are performing task $h_i(a_{ij})$, as well as updates the appropriate control parameters.

When a robot team is applied to a "live" mission, it enters the *adaptive learning* phase, in which the robots acquiesce (give up tasks) and become impatient (take over tasks) according to their learned knowledge and the control strategies described in the remainder of this paper, rather than being maximally patient and minimally acquiescent as they are in the active learning phase. However, the monitors within each robot continue to monitor and catalog robot performances during this phase, updating their control parameters based upon these performances.

3.1.3. Parameter Update Strategies. Once the robot performance data has been obtained, it must be input to some 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. This control problem translates into two related subproblems: (1) *dynamic task reallocation* — how an individual robot determines whether to interrupt the task currently being performed by another robot (i.e. become impatient), or whether it should acquiesce its own current task and (2) *task ordering* — how an individual robot selects from among a number of incomplete tasks that no other team member is currently performing. These issues are modeled in the L-ALLIANCE formalism (shown in detail in the Appendix) as three control parameters:

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

To study these issues, we identified three potential control strategies for the two dynamic task reallocation parameters ($\delta\_slow_{ij}(k,t)$ and $\psi_{ij}(t)$), and three

Strategy	Impatience ($\delta\_slow_{ij}(k,t)$)	Acquiescence ($\psi_{ij}(t)$)
I	own time	own time
II	own time	min. time of team
III	time of robot performing $h_i(a_{ij})$	own time

Table 1: The impatience and acquiescence parameters are functions of the values shown in the table, for each of three dynamic task reallocation strategies.

potential control strategies for the task ordering parameter, $\delta\_fast_{ij}(t)$. These potential control strategies for parameter learning are discussed below.

Dynamic Task Reallocation Strategies

The three dynamic task reallocation strategies that we tested for task-oriented multi-agent learning are summarized in Table 1 three-strategies. The first dynamic task reallocation strategy, which we call Strategy I: "Distrust Performance Knowledge about Teammates", takes a minimalist approach to the problem by requiring a robot to use only the knowledge it learns about its own performance. Under strategy I, r_i becomes impatient with any other robot r_k that does not complete $h_i(a_{ij})$ in the same length of time required for r_i to complete $h_i(a_{ij})$.

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

The third dynamic task reallocation strategy, called Strategy III: "Give Robots a Fighting Chance", results in a robot team that judges performances of robot team members based on each team member's own individual expected performance, rather than its comparison to other team members' performances. Under strategy III, r_i becomes impatient with r_k's performance only after r_k begins performing worse than its (r_k's) normal abilities. Additionally, each robot is unwilling to acquiesce its own action until it believes it has had a fair chance to accomplish the task, according to its own expected performance requirements.

Task ordering strategies

We also investigated three task ordering strategies (i.e. how an individual robot selects from among a number of incomplete tasks that no other team member is currently performing) in which each robot's next action selection is either a greedy choice based upon the expected execution time of the tasks it is

able to perform, or is a random choice of actions. The following paragraphs describe these three task ordering approaches, called Longest Task First, Modified Shortest Task First, and Modified Random Task Selection.

In the multi-processor scheduling community, a centralized greedy approach called Descending First Fit has been shown to result in mission completion times within 22% of optimal [22] for identical processors. In this approach, the tasks are assigned to processors in order of non-increasing task length. Thus, we first attempted a distributed version of Descending First Fit to determine its effectiveness for the multi-robot application domain. The distributed version, which we call Longest Task First, requires each robot to select as its next task that which is expected to take the robot the longest length of time to complete.

As a logical next step, we studied the dual of the Longest Task First approach in which each robot to selects its next action as that which it expects to perform the quickest. The centralized version of this greedy approach for identical multi-processors has been shown to result in minimizing the *mean flow* of the mission, which means that the average completion time of the tasks in the mission is minimized [18]. Here, we modify the pure Shortest Task First technique somewhat (hence, the name *Modified* Shorted Task First) to compensate for the fact that heterogeneous robots have different sets of tasks which they are able to pursue. We thus require a robot to first select from among those actions which it expects to perform better than any other robot on the team.

Finally, as a baseline against which to compare the other task ordering approaches, a random selection of tasks was also studied, which we call Modified Random Task Selection. In this case, the robots divide the tasks into the same two categories used in the Modified Shortest Task First approach. However, in this case, the tasks are randomly selected, initially from the first category, and then from the second category.

4. RESULTS

To determine the relative merits of these strategies, we executed a large number of test runs in simulation, comparing the results of all of the combinations of the dynamic task reallocation and task ordering strategies in terms of the time required to complete the mission. We collected performance data by varying the number of robots on the team (n) from 2 to 20, the number of tasks the team must perform (m) from 1 to 40, the task coverage[1] from 1 to 10, the degree of heterogeneity[2] from 0 percent to 3200 percent, and the value of the *Progress When Working* (PWW) condition[3] as either true or false. For this study, the

[1] The *task coverage* is a measure of the total number of capabilities on the robot team that may allow a team member to achieve a given task — see [21].

[2] In this context, *degree of heterogeneity* refers to a relative comparison of $q(a_{ij})$ and $q(a_{kl})$, in which $h_i(a_{ij}) = h_k(a_{kl})$; in other words, r_i and r_k can both perform the same task, but with different levels of performance.

[3] We define the *Progress When Working* condition as follows (see also [21]). Let z be the

missions were composed of completely independent subtasks involving no ordering constraints, the capabilities were distributed uniformly across the robots based upon the given task coverage, and the same task coverage was assumed for all tasks in the mission. For each 5-tuple (n, m, *task_coverage*, *heterogeneity*, PWW) of a given run of the simulation, which we call a *scenario*, 200 randomly generated test runs were executed. The average over these 200 runs was then considered the characteristic performance of that scenario.

Our studies revealed that all of the above variables have an impact on the relative performances of the parameter learning strategies. Limited space in this paper prohibits a thorough analysis of these results; instead, we focus here on the most important, high-level results. (Refer to [21] for a more detailed analysis.) The relative performance results comparing the three dynamic task reallocation strategies for the fixed task ordering strategy of Modified Shortest Task First are shown succinctly in Fig. 2 time-venn. In this figure, the numbers I, II, and III in large parentheses indicate the relative performance of the three dynamic task reallocation strategies in each of the regions, where the top row indicates the best performer(s). When more than one set of values are given (e.g., in regions 2 and 3), the relative performances depend upon the *Progress When Working* (PWW) condition. The four points noted with small black squares are exemplar missions of their corresponding regions. The three values in the small parentheses by each of these four points describe the corresponding cooperative scenario by giving the number of robots, the number of tasks, and the task coverage used in the exemplar. As can be seen, the relative performances of the various parameter update strategies vary depending upon the particular scenario being investigated. In this figure, "highly" heterogeneous means roughly a degree of heterogeneity above 600%, whereas "mildly" heterogeneous means a degree of heterogeneity between approximately 300% and 600%.

We then compared the impact on these task reallocation strategies when changing the task order strategy from Modified Shortest Task First to the Longest Task First and Modified Random Task Selection approaches. Our results led to a quick dismissal of the distributed Longest Task First approach, since it turned out to be disastrous for heterogeneous cooperative teams in which robot failures can occur — in general, this approach caused each task in the mission to be attempted by the robot team member with the worst ability to accomplish that task. Clearly, this does not result in collectively efficient task execution.

Our experiments showed that the relative performances of the three dynamic task reallocation strategies change when using a Modified Random Task Selection task ordering approach instead of the Modified Shortest Task First ap-

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.

proach, regardless of the scenario's region in Fig. 2. A typical result of comparing the relative performances of the three dynamic task reallocation strategies under the Modified Random Task Selection task ordering approach instead of with the Modified Shortest Task First approach is shown in Fig. 3 for one of the exemplar scenarios. As this figure shows, although the Random Task Selection approach degrades the performance of teams controlled by strategies I and III as heterogeneity increases, it actually improves the performance of teams controlled with strategy II (Let the Best Robot Win).

We discuss the significance of these results in the next section.

5. DISCUSSION

Our goal in this work is to derive a single approach to multi-robot action selection that performs well regardless of the particular instantiation of the robot team and mission. We do not want to require a human system designer to perform extensive analysis of specific multi-robot applications to determine the appropriate multi-robot learning strategy. Thus, we now analyze the results of the previous section to attempt to find a general approach that performs well in any of the situations studied.

Combining the results of Figs. 2 and 3 leads first to the conclusion that strategy II (Let the Best Robot Win) is the favored strategy for experimental scenarios in regions 1 and 4, as well as in regions 2 and 3 when the Progress When Working condition is true, regardless of whether the Modified Random or the Modified Shortest Task First ordering approach is used. Strategy III (Give Robots a Fighting Chance) is preferred over strategy II only when using the Modified Shortest Task First ordering approach and when the PWW condition is false in the following regions: region 3, and in region 2 for "mildly" heterogeneous teams. Strategy I (Distrust Performance Knowledge about Teammates) is never a superior strategy.

Returning to Fig. 3, let us now determine why strategy II performs so much better under the Modified Random approach than the Modified Shortest Task First approach. The reason for this performance improvement relates to the theoretical advantages mentioned earlier of using the Longest Task First selection strategy. The Longest Task First approach theoretically results in shorter mission completion times for homogeneous robot teams because the longer tasks are pursued first while available robots perform the shorter tasks in parallel. Of course, we dismissed this approach for heterogeneous robot teams which can perform the same tasks with different qualities, since it caused each task to be pursued by the robot with the longest task completion time. However, if we allow robots to divide their tasks into two categories — (1) those which the robot expects to be able to perform quicker than any other robot, and (2) all remaining tasks that robot can perform — and then use a Longest Task First mechanism to select among the first category and a Shortest Task First mechanism for selecting among the second category, the problem of heterogeneity is

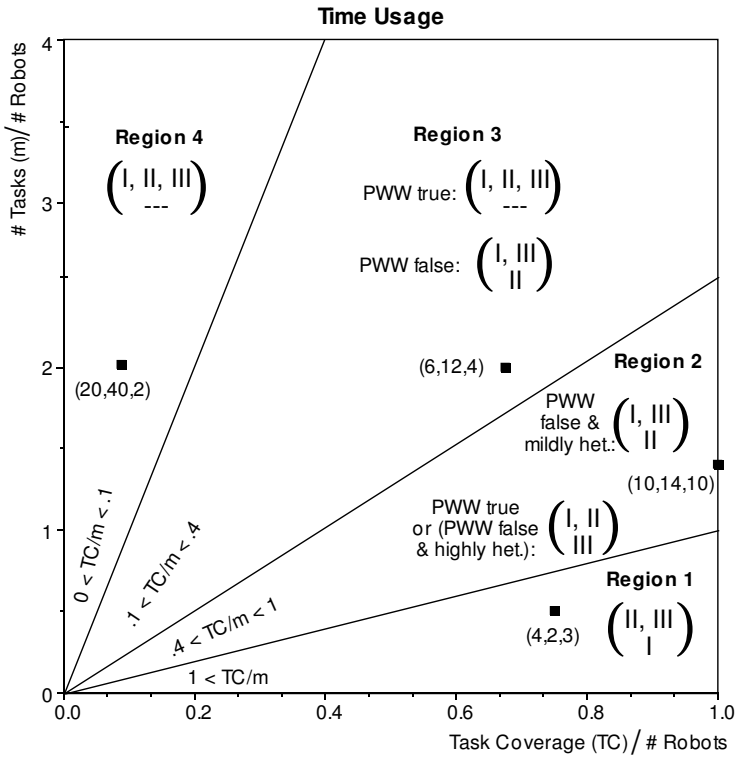

Figure 2: Summary of time usage for the dynamic task reallocation strategies using a fixed task ordering strategy (Modified Shortest Task First). (Refer to the text for an explanation of this figure.)

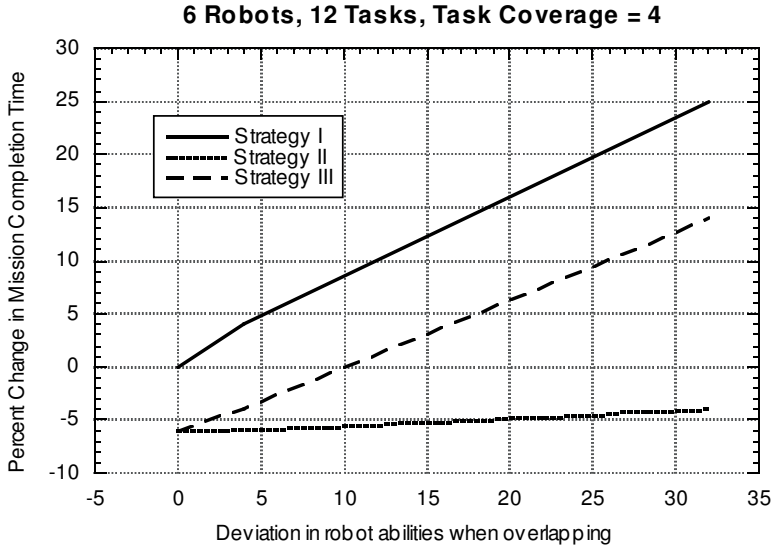

Figure 3: Typical change in mission completion time when using the Modified Random Task Selection ordering approach instead of the Modified Shortest Task First approach.

circumvented. What we find in Fig. 3 is that the Random Task Selection approach for strategy II actually moves the robot control toward a Longest Task First approach, since any random selection of an action must result in a longer task than that chosen with the Shortest Task First approach.

This then leads to the following general task oriented learning mechanism for this application domain, under which each robot r_i does the following:

1. Divide the tasks into two categories:
   (a) Those tasks which r_i expects to be able to perform better than any other team member, and which no other robot is currently performing.
   (b) All other tasks r_i can perform.
2. Repeat the following until sensory feedback indicates that no more tasks are left:
   (a) Select tasks from the first category according to the longest task first approach, unless no more tasks remain in the first category.
   (b) Select tasks from the second category according to the shortest task first approach.

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

The implementation of this control strategy in terms of the control parameters of the L-ALLIANCE architecture is formally presented in the Appendix.

Since this action selection problem is NP-hard, it is very difficult to empirically compare the performance of the L-ALLIANCE approach to the theoretically optimal result. The optimal solution becomes impractical to calculate directly, even for small values of n and m. However, the optimal result can be directly calculated for many small problems in which the value of n^m is reasonable. We thus experimentally compared the results of the derived L-ALLIANCE control strategy with the optimal solution for those problems in which we could derive the optimal result. What we discovered, for a total of 496 scenarios in regions 1, 2, and 3, is that L-ALLIANCE performs within 20% of the optimal solution for these smaller scenarios, which is certainly acceptable. Of course, the key issue is how seriously the performance of L-ALLIANCE will degrade as the size of the problem increases. This is a topic of future research.

To validate the results of our empirical studies in simulation, we successfully implemented this L-ALLIANCE control strategy in a cooperative box pushing demonstration on real mobile robots. This demonstration was useful, because if offered a simple and straight-forward illustration of the key characteristics of the L-ALLIANCE architecture: fault tolerant, adaptive, and efficient control. This demonstration was implemented using a heterogeneous robot team of two R-2 robots and a Genghis-II robot. It successfully illustrated how the L-ALLIANCE architecture endows robot team members with fault tolerant and efficient action selection in the midst of robot failures and team heterogeneity. Space limitations prohibit a discussion of these results; refer to [21] for more details.

6. CONCLUSIONS

Most previous work in multi-robot systems has addressed either the need for fault tolerance in a cooperative system, or the need for efficiency in multi-robot systems. Very little previous research has addressed a combination of these two issues for task-oriented multi-robot control. We have addressed this problem by developing a mechanism called L-ALLIANCE that allows teams of heterogeneous robots to improve their task efficiency without sacrificing the desirable characteristics of fault tolerance and adaptivity. This mechanism is based upon the ALLIANCE architecture, and was developed through extensive testing of competing strategies in simulation. We validated our results on a team of physical robots performing a simple manipulation task. Future research in this area includes comparing our results to the results of a evolutionary search approach, as well as deriving analytical proofs of the theoretical effectiveness of our approximate solution.

Acknowledgments

This research was funded in part by the Advanced Research Projects Agency of the Department of Defense under Office of Naval Research contract N00014-91-J-4038 at the Massachusetts Institute of Technology's Artificial Intelligence Laboratory, and in part by the Office of Engineering Research, Basic Energy Sciences, of the U.S. Department of Energy, under contract No. DE-AC05-96OR22464 with Lockheed Martin Energy Research Corporation.

REFERENCES

1. Rodney A. Brooks A robust layered control system for a mobile robot. *IEEE Journal of Robotics and Automation* **RA-2**, No 1 14–23. (1986).

2. Gerardo Beni and Jing Wang On cyclic cellular robotic systems. In: *Japan – USA Symposium on Flexible Automation.* 1077–1083. (1990).

3. Rodney A. Brooks, Pattie Maes, Maja Mataric, and Grinnell Moore Lunar base construction robots. In: *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems.* 389–392. (1990).

4. 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.* 1093–1098. (1990).

5. 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.* 451–459. (1992).

6. 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).* 145–150. (1988).

7. C. Ronald Kube and Hong Zhang Collective robotic intelligence. In: *Proceedings of the Second International Workshop on Simulation of Adaptive Behavior.* 460–468. (1992).

8. Maja Mataric Designing emergent behaviors: From local interactions to collective intelligence. (J. Meyer, H. Roitblat, and S. Wilson, Eds.). In: *Proc. of the Second Int'l Conf. on Simulation of Adaptive Behavior.* 432–441. MIT Press. (1992).

9. David Miller Multiple behavior-controlled micro-robots for planetary surface missions. In: *Proceedings IEEE Systems, Man, and Cybernetics Conference.* (1990).

10. Luc Steels Cooperation between distributed agents through self-organization. (Yves Demazeau and Jean-Pierre Muller, Eds.). In: *Decentralized A.I.*. El-

sevier Science. (1990).

11. 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.* 766–771. (1993).

12. 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.* 346–355. (1990).

13. K. L. Chung Elementary Probability Theory with Stochastic Processes. Springer-Verlag. (1974).

14. 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**, No 2 122–127. (1992).

15. 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.* 67–72. (1990).

16. Paul Cohen, Michael Greenberg, David Hart, and Adele Howe Real-time problem solving in the Phoenix environment. In: *COINS Technical Report 90-28, U. of Mass., Amherst.* (1990).

17. Fabrice R. Noreils Toward a robot architecture integrating cooperation between mobile robots: Application to indoor environment. The International Journal of Robotics Research **12**, No 1 79–98. (1993).

18. R. W. Conway, W. L. Maxwell, and L. W. Miller Theory of Scheduling. Addison-Wesley. (1967).

19. Sandip Sen Report on the IJCAI-95 workshop on adaptation and learning in multiagent systems. (1995).

20. Lynne E. Parker ALLIANCE: An architecture for fault tolerant, cooperative control of heterogeneous mobile robots. In: *Proc. of the 1994 International Conference on Intelligent Robots and Systems (IROS '94).* 776–783. (1994).

21. Lynne E. Parker Heterogeneous Multi-Robot Cooperation. In: *PhD thesis, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, Cambridge, MA.* (1994). MIT-AI-TR 1465

22. Michael R. Garey and David S. Johnson Computers and Intractability: A Guide to the Theory of NP-Completeness. W. H. Freeman and Company. (1979).

APPENDIX: FORMAL MODEL OF L-ALLIANCE

The formal definition of the L-ALLIANCE cooperative robot architecture, including mechanisms for learning and efficiency considerations, is provided below.

Given θ, which is the threshold of activity of a behavior set, and *strategy*, which is the current impatience/acquiescence update strategy, seven sources of input affect the motivation to perform a particular behavior set. These inputs are defined as:

Sensory feedback:

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

Inter-robot communication:

ρ_i = Rate of messages per unit time that robot r_i sends concerning its current activity

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

τ_i = Maximum time robot r_i allows to pass without hearing from a particular robot before assuming that that robot has ceased to function

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

Suppression from active behavior sets:

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

Learned robot influence:

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

μ = Number of trials over which task performance averages and standard deviations are maintained

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

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

$boredom\_threshold_i$ = level of boredom at which robot r_i ignores the presence of other robots able to perform some task not currently being executed

$boredom\_rate_i$ = Rate of boredom of robot r_i

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

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

Robot impatience:

$\phi_{ij}(k,t)$ = Time during which robot r_i is willing to allow robot r_k's communication message to affect the motivation of behavior set a_{ij}.

$$= \begin{cases} task\_time_i(k,j,t) & \text{if } (strategy = \text{III}) \\ task\_time_i(i,j,t) & \text{if } (strategy = \text{II}) \end{cases}$$

$\delta\_slow_{ij}(k,t)$ = Rate of impatience of robot r_i concerning behavior set a_{ij} after discovering robot r_k performing the task corresponding to this behavior set

$$= \frac{\theta}{\phi_{ij}(k,t)}$$

min_delay = minimum allowed delay

max_delay = maximum allowed delay

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

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

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

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

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

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

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

Robot acquiescence:

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

$$= \begin{cases} task\_time_i(i,j,t) & \text{if } (strategy = \text{III}) \\ \min\limits_{k \in robots\_present(i,t)} task\_time_i(k,j,t) & \text{if } (strategy = \text{II}) \end{cases}$$

$\lambda_{ij}(t)$ = Time robot r_i wants to maintain behavior set a_{ij}'s activity before giving up to try another behavior set

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

Motivation calculation:

The motivation of robot r_i to perform behavior set a_{ij} at time t is calculated as follows:

DURING ACTIVE LEARNING PHASE:

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

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

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

DURING ADAPTIVE PHASE:

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