Department of Computer Science
George Mason University Technical Reports

4400 University Drive MS#4A5
Fairfax, VA 22030-4444 USA
http://cs.gmu.edu/ 703-993-1530

Bounties

Drew Wicke
dwicke@gmu.edu

David Freelan
dfreelan@gmu.edu

Technical Report $-705-991i$

Abstract

This paper presents a mechanism for task allocation for groups of robots. The mechanism is based on bounties, a utility based motivation strategy common in human societies. We define task-allocation problems for which the bounty mechanism is suitable and demonstrate a greedy and reinforcement learning approaches can be used to solve the role allocation problem. Experiments in simulation show that our learning approach consistently does similar to the greedy approach. Experiments on real robots show that the mechanism is also able to perform after learning is done offline.

1 Introduction

Task allocation is an integral part of coordinating the behaviors of multiple agents by simplifying the interactions between the agents. It is also useful in bringing about cooperation between agents in order to accomplish cooperative tasks. For the purposes of this paper we take similar definitions of task, activity and role from the often referenced definitions by Krieger and Billeter [?]. Tasks define what needs to be accomplished, activities are tasks that are being completed, and role is the set of tasks that define the normal activity of an individual.

There have been multiple methods proposed to solve the task allocation problem. Greedy algorithms, optimization approaches and market/auctions.

Auctions are a popular mechanism for distributing tasks in a multi robot system. Auctions rely on the bid of the robots to determine how to distribute tasks. However, robots' sensors are noisy and therefore the bid may not reflect a true valuation and the task may not get done. Removing the need for trust by using Bounties results in a competitive environment that leads to the successful completion of tasks. Bounties are an alternative mechanism for getting multiple robots to do tasks. Tasks are incentivized with a bounty. Bounties are a time dependent utility for accomplishing a task.

This paper will describe some previous work in the area of task allocation. Then we will describe our Bounty task allocation mechanism and provide experimental evidence of there ability to allocate tasks. During this we will argue for our new learning algorithm DDLearner which is made specifically for our Bounties mechanism. We will also describe our results when implemented on real robots. Finally, we will propose future work to be completed.

2 Related Work

There has been much focus on solving the task allocation problem because it is an important problem when more than one robot can do tasks.

One of the first to attempt multi-robot coordination through task allocation is in the work by Parker with both ALLIANCE and L-ALLIANCE [?, ?]. These approaches created an entire robot architecture around the idea of distributing tasks. They focused on the concept of mathematically defined motivations that allow the robots to select to perform particular tasks. L-ALLIANCE extended this architecture to include learning behaviors.

Gerkey and Mataric created, MURDOCH, an auction mechanism for solving the task allocation problem and coordinating multi robot systems [?]. They focused their mechanism on minimizing resource usage, task completion time and communication overhead. This work is based off of the contract net protocol [?]. MURDOCH has since been extended and studied greatly as a means of allocating tasks.

Dahl took a different approach and used vacancy chains as a method for allocating tasks [?]. In order to use vacancy chains to solve the task allocation problem the tasks need to have a value and can be repeated indefinitely.

An excellent survey on the task allocation problem by Gerkey [?] defines three major axes for the problem, including the type of task, type of robot and the information available to determine the assignment. In the experiment section we define our problem along these axes.

Another aspect of the paper is role allocation. Role allocation is the problem of determining the set of tasks that the individuals will normally take on. It is determined by the agent's learning mechanism in our experiments. Campbell provides a detailed overview of the problem, previous work and well thought out open problems in the area [?].

3 Bounties

Robot sensor information is noisy especially localization data. This limitation causes the system to not be ideal when dealing with multiple robots. One popular method of distributing tasks to robots is through an auction. This method involves the robots biding on tasks based on their sensor infoformation. The centralized auctioneer will then determine the winner based on submitted bids. Usually the robot that has bid the most will get the task. Therefore, the assignment of the task may result in the robot failing to accomplish the task due to its inability to accurately predict its success. Then either the auctioneer has to re-auction the task or take the loss. This can be an unacceptable outcome in scenarios where the task must be completed.

Bounties are a lazy task allocation method that avoids the problem of trusting bidders with the market/auction approach. This method is similar in nature to real life bounty mechanisms. We have a bondsman that manages and creates the tasks with associated bounty. The bounty is defined to be a monotonically increasing with time value associated with completing the task that the bounty is associated with. Bounty hunters are the entities, in our case robots, that undertake the tasks based on the bounty.

The bounty mechanism works by following the following steps:

1. Bondsman announces to the the robots the list of available tasks and their current bounty

2. Bounty hunters decide which task to take by weighting their decision based on the current bounty

3. Bounty hunters approaching a task and who have not started to complete the task may jump-ship and approach another task at that current bounty value

4. Bounty hunters who have finished approaching the task notify the bondsman that the task should not be available to other bounty hunters (also once the robot has started the task it must finish the task before changing)

5. If a Bounty hunter completes a task the Bondsman awards the bounty they agreed upon

The Bounties mechanism does not make any assumptions about the type, number or location of tasks. The mechanism also does not limit the number of bounty hunters needed to complete the task and through the use of a hierarchical task network as the mechanism for describing a task we can have complex subtasks. The Bounty mechanism does not require the bondsman to know anything other than the post conditions for successful completion of the task. By our definition of bounty hunter this mechanism does not place any bounds on the number or abilities of the robots. This means that the robots can be homogeneous or heterogeneous and that robots can be added and removed at any time. However, for the purposes of this paper we do make a number of limiting assumptions in order to focus on the ability of bounties to act as a task allocation mechanism. We assume that the bondsman knows the location of the tasks and they do not move. We also assume that the robots are homogenous.

4 Robot Behaviors

We created three different methods for picking tasks to solve the role allocation problem. The Jump-ship approach is a greedy approach with full information that follows a static protocol. Q-learning which follows the standard Q-learning algorithm. And our new approach DDLearner that was created specifically for bounties. However, both Q-learning and the DDLearner approach define dynamic role.

4.1 Jump-ship Robot

The Jump-ship robot is a greedy approach to picking the task to do. The robot can also change tasks after starting to go after the task. However, once the task is in progress it can not switch. Also, instead of adding a fixed amount of utility to stay in our current role as in [?] we penalize all the tasks that the robot is not currently doing in order for it to not oscillate.

$$ {}_aR_t^b = v^b P_a^b $$

$$ P_a^b = \Pi_{i \in A} \begin{cases} 1, E(T_a^b) < E(T_i^b) \\ 1/2, E(T_a^b) = E(T_i^b) \\ 0, E(T_a^b) > E(T_i^b) \end{cases} $$

- ${}_aR_t^b$: reward for agent a to grab task b at time t

- $v^b P_a^b$: v^b is the value of the bounty, P_a^b is the approximated probability that the agent will attain the bounty b

- $E(T_i^j)$: defined as the expected time is takes j to go after bounty i

- A the set of agents currently working toward doing task b not including agent a

4.2 Q-Learner

We also tried using a standard q-learning algorithm. The states were the different tasks and the actions were to change to doing one of the tasks. Therefore the goal was to learn what a good sequence of tasks would be.

4.3 DDLearner

The DDLearner is a learning algorithm created with the goal to learn how to scale the bounties appropriately in order to identify the set of tasks to take. Essentially to learn a role.

$$S = \{s_1, s_2, s_3 .... s_n\}$$
$$s \in S$$
$$s^* = max_s\{R_s(B, DD(s))\}$$
$$R_s = B * (DD(s) + \epsilon)$$

$$DD(s) = \begin{cases} (1-\alpha)DD(s) + \alpha r, & \text{if } r! = 0 \\ \alpha * DD(s), & \text{if } r = 0 \end{cases}$$

- S: set of all tasks which are available
- s^*: best task to take, it is a function of the reward of a given task s
- R_s: the reward of taking task s. Multiplies the bounty by the sum ϵ and the learned function $DD(s)$
- $DD(s)$: learned function. If you complete the task, you chose the first equation, which is essentially a Q-update, without looking at future reward calculation. Else, you actually learn at a different rate, we chose α as an example, but the reality is that it can be perform well by replacing it with some different constant

The reasoning for a different learning vs unlearning rate, was to avoid robots from learning, or unlearning, at an even rate to other robots. Allowing them to learn at the same rate appeared to cause unwanted oscillation with multiple robots in the system.

Another important feature of this equations is the the reward function, specifically the ϵ. ϵ is a necessary addition to make exploration of the field always possible. It is possible most of the agents could find a task not worthy of doing very often, however you never want that probability to go as low as zero, so an epsilon becomes necessary.

5 Experimental Setup

We experimented with bounties in simulation and demonstrated the system on two Darwin-OP humanoid robots. To simulate the bounty system we created a MASON simulation [?]. We created the simulation so that we can easily change between real robots and virtual robots. In order for the simulation field to align to the actual field we made the simulation 60 tiles long by 40 wide where each tile is a decimeter. Our experiments are ST-SR-IA, Single Task, Single Robot, Instantaneous Assignment as defined by [?]. Meaning that the robot can complete at most a single task at a time, only a single robot is required to accomplish each task and that there is not future planning on who will do what task ahead of time. This is also therefore in the computational complexity class of P [?, ?].

5.1 Experiment Tasks and Measures

We conducted four experiments to determine the convergence of the robots to a particular role. The experiment consisted of the agent traveling to the task location and then taking the item at the task location to the goal location. The tasks, robots, and goal location are randomly placed on the field unless specified for the particular experiment. The agents were given the coordinates of both the task location and the goal location and knew how to get to both locations. The agents did not perform any collision detection and are able to occupy the same tile. When moving to either the task or goal location the agents first travel along the x-axis and then along the y-axis.

To measure the proficiency of the different methods, Jump-ship, Q-Learning and DDLearner, we looked at the average ticks. Average ticks are defined to be the average bounty on the field at that time. We chose this method as our measure because we wanted to see how fair to all the tasks the algorithms were and to illustrate the effectiveness of bounties at manipulating which tasks get done. The graphs shown are a rolling average over 1000 time steps.

5.2 2 Robots 10 Tasks

This experiment consisted of randomly placing 10 tasks, 1 goal location, and 2 robots on the field and then running the experiment for 20,000 time steps.

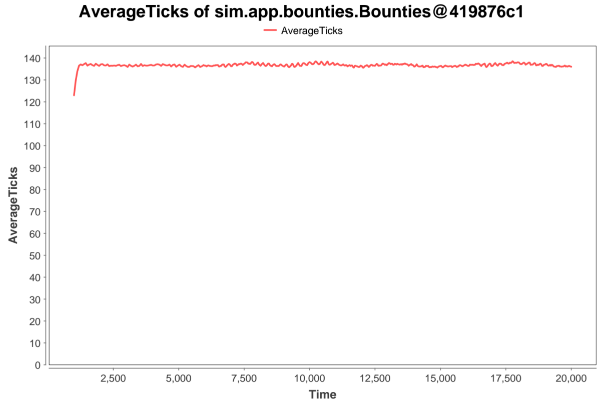

Figure 1: Jump-ship approach

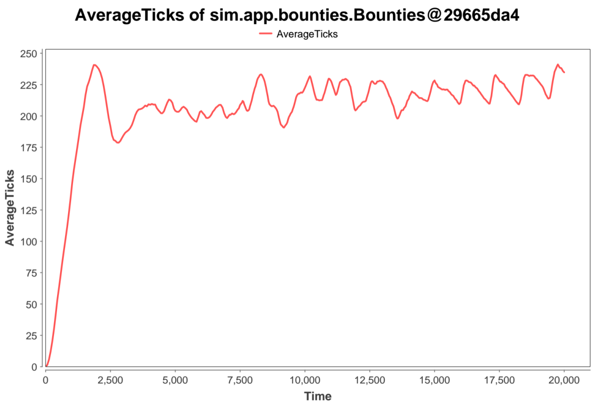

Figure 2: Q-Learning approach

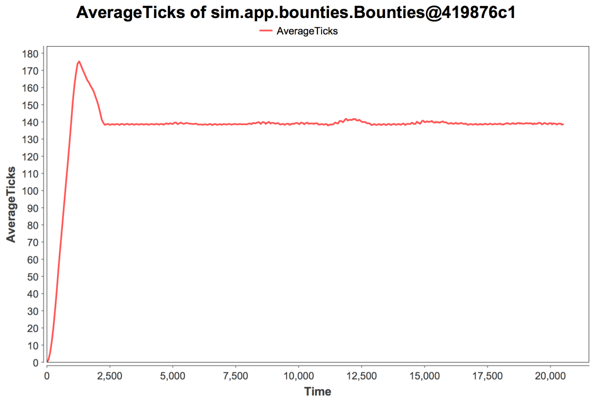

Figure 3: DDLearning approach

5.3 8 Robots 30 Tasks

This experiment consisted of randomly placing 30 tasks, 1 goal location, and 8 robots on the field and then running the experiment for 20,000 time steps.

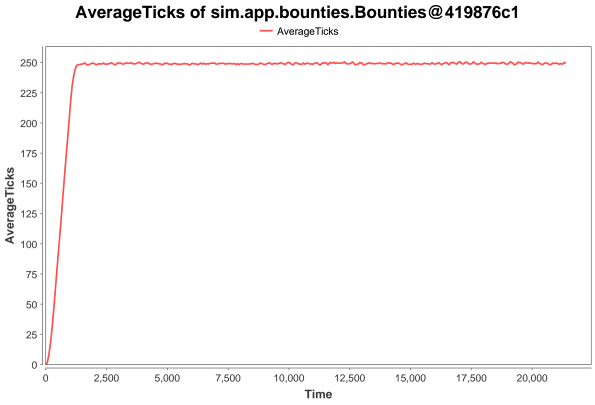

Figure 4: Jump-ship approach

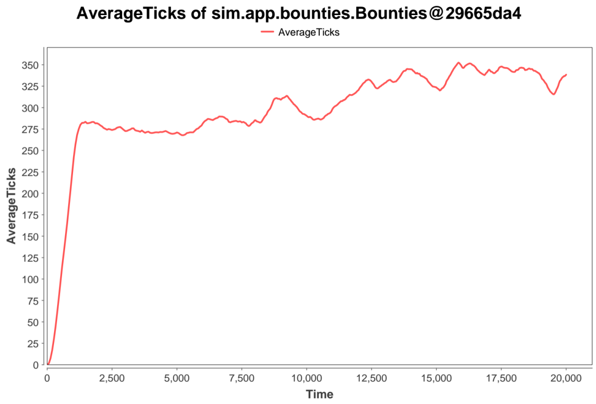

Figure 5: Q-Learning approach

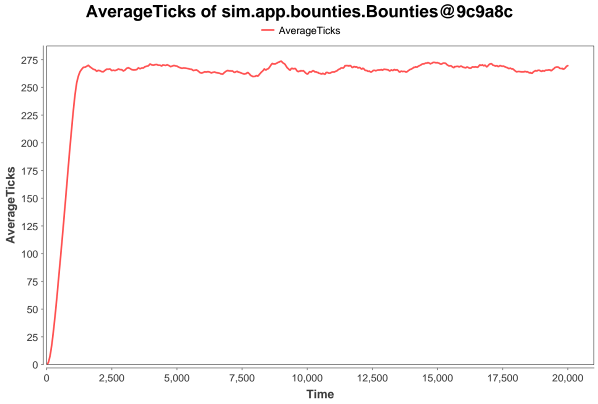

Figure 6: DDLearning approach

5.4 8 Robots 300 Tasks

This experiment consisted of randomly placing 300 tasks, 1 goal location, and 8 robots on the field and then running the experiment for 20,000 time steps. Due to the excessive computation time greedy approach was not used.

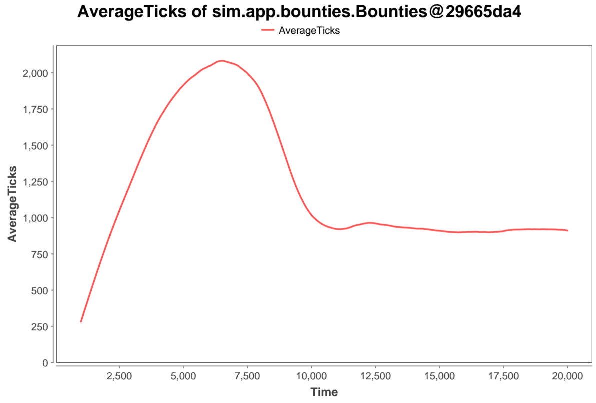

Figure 7: Q-Learning approach

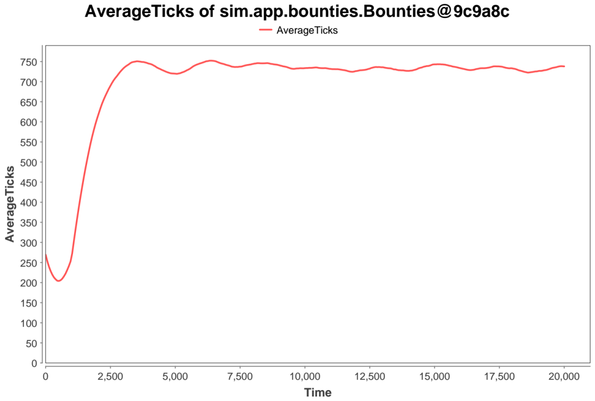

Figure 8: DDLearning approach

5.5 2 Robots 10 Tasks Configured

This experiment consisted of configuring the locations of 10 tasks and randomly placing 1 goal location, and 2 robots on the field and then running the experiment for 20,000 time steps. The tasks were configured in order to accentuate the bounty mechanism by placing half of the tasks far away from the goal and the other half around the goal. This is similar to the *forager-protector* scenario described in [**?**], except with ours the distance implies the type of task. Where there are two types of tasks one to protect the goal (close) and the other to go forage (far from goal).

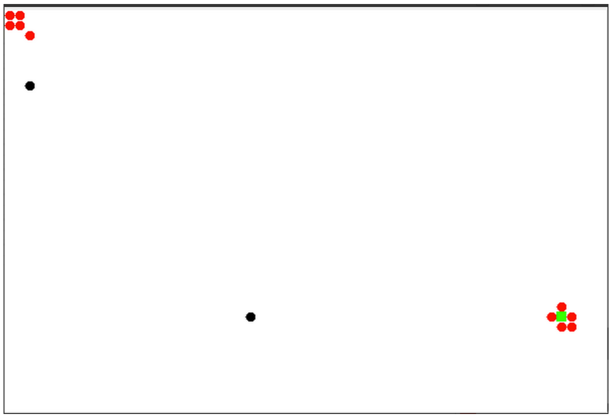

Figure 9: Configuration of field. Red = task, Green = goal, Black = Robot

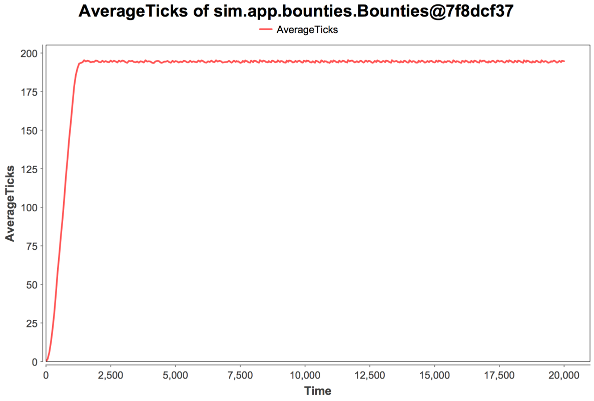

Figure 10: Jump-ship approach

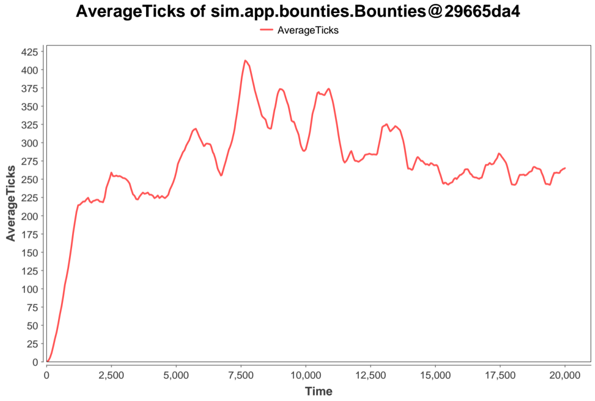

Figure 11: Q-Learning approach

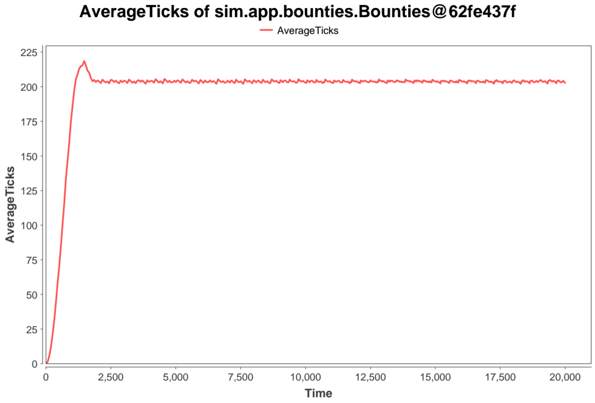

Figure 12: DDLearning approach

6 Analysis and Discussion

As the graphs show in each experiment the Jump-ship approach (the greedy approach) outperforms both DDLearner and the Q-Learning algorithms. This is mainly because this algorithm has full knowledge of what everyone is doing and their measure of how good they are going to be at it every time step. This approach is not a viable option to use when the number of tasks or number of robots increases as the communication overhead is too great. As is seen in the case with 8 robots and 300 tasks the Jump-ship approach was not able to perform in a reasonable amount of time compared to the DDLearner or the Q-Learner.

The Q-Learning approach based off of our limited experiments did not converge (except for the 300 task case where it did not perform as well as the DDLearner) and did not appear to be improving as time increased past 20,000 time steps. This may be due to both the size of the q-table and the non-stationary environment. In contrast, the DDLearner was able to converge and did relatively similar to the Jump-ship approach. One benefit of the DDLearner approach is that it does not require the full information that the Jump-ship algorithm requires. The only thing is that the space required grows linearly with the number of tasks.

7 Robot Demonstration

We also demonstrated the system on two Darwin-OP humanoid robots. We placed a Robocup regulation size ball midway between center and the goal on both sides of the field. The robots were tasked to go to the ball and kick the ball into the goal and then to go to the center of the field. The robots were not told which task they should be responsible for, the goal of the experiment was to show that the robots could learn their role in the simulation and then transfer that behavior to the robots.

We ran the simulation for 5000 time steps until the DDLearner robots converged to a set of roles where one robot took one task and the the other took the other task. Then we switched the controller mechanism to move the Darwin-OPs and update the simulation with their location data. The robots were able to successfully complete their separate tasks using the learned behaviors from the simulation. A video of the demonstration can be found `http://youtu.be/LXiHaUqBWLI`.

8 Future Work

The future work will be focused on tasks that require multiple robots and therefore cooperation within a competitive environment. We intend to carry out experiments in this ST-MR-IA environment to explore how the DDLearner and bounties work. This needs to be done because this type of problem is NP-Hard and will show how well bounties actually do. We would study the emergent coalitions from just our DDLearner and compare it to a Coalition learning algorithm. We would like to analyze both convergence and fairness to the tasks as measured by average time the tasks are on the field.

References

[1] James Brusey, Mark Makies, and Lin Padgham, 'RMIT united', *RoboCup 2000: Robot ...*, (2001).

[2] Adam Campbell and Annie S. Wu, 'Multi-agent role allocation: issues, approaches, and multiple perspectives', *Autonomous Agents and Multi-Agent Systems*, **22**(2), 317–355, (April 2010).

[3] TS Dahl, MJ Mataric, and G Sukhatme, 'Multi-robot task-allocation through vacancy chains', *Robotics and Automation, ...*, 12–17, (2003).

[4] R Davis and RG Smith, 'Negotiation as a metaphor for distributed problem solving', *Artificial intelligence*, (1983).

[5] B. P. Gerkey, 'A Formal Analysis and Taxonomy of Task Allocation in Multi-Robot Systems', *The International Journal of Robotics Research*, **23**(9), 939–954, (September 2004).

[6] B P Gerkey and M J Mataric, 'Sold!: auction methods for multirobot coordination', *IEEE Transactions on Robotics and Automation*, **18**(5), 758–768, (2002).

[7] BP Gerkey and MJ Matarić, 'On role allocation in RoboCup', *RoboCup 2003: Robot Soccer World Cup VII*, **3020**(2003), 43–53, (2004).

[8] Michael J.B. Krieger and Jean-Bernard Billeter, 'The call of duty: Self-organised task allocation in a population of up to twelve mobile robots', *Robotics and Autonomous Systems*, **30**(1-2), 65–84, (January 2000).

[9] S Luke, GC Balan, and L Panait, 'MASON: A Java Multi-agent Simulation Library', *... in Social Simulation*, (2003).

[10] LE Parker, 'L-ALLIANCE: A mechanism for adaptive action selection in heterogeneous multi-robot teams', *ORNL/TM-13000*, (615), (1995).

[11] LE Parker, 'ALLIANCE: An Architecture for Fault Tolerant Multi-Robot Cooperation', *Robotics and Automation, IEEE Transactions on*, (1998).