Learning Opportunity Costs in Multi-Robot Market Based Planners

Jeff Schneider, David Apfelbaum, Drew Bagnell, and Reid Simmons
The Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
{*schneide,da0g,dbagnell,reids*}*@cs.cmu.edu*

***Abstract*— Direct human control of multi-robot systems is limited by the cognitive ability of humans to coordinate numerous interacting components. In remote environments, such as those encountered during planetary or ocean exploration, a further limit is imposed by communication bandwidth and delay.**

Market based planning can give humans a higher-level interface to multi-robot systems in these scenarios. Operators provide high level tasks and attach a reward to the achievement of each task. The robots then trade these tasks through a market based mechanism. The challenge for the system designer is to create bidding algorithms for the robots that yield high overall system performance.

Opportunity cost provides a nice basis for such bidding algorithms since it encapsulates all the costs and benefits we are interested in. Unfortunately, computing it can be difficult. We propose a method of learning opportunity costs in market based planners. We provide analytic results in simplified scenarios and empirical results on our FIRE simulator, which focuses on exploration of Mars by multiple, heterogeneous rovers.

***Index Terms*— Market-based planning, learning, opportunity cost, multi-robot systems.**

I. Introduction

Multi-robot systems are particularly challenging for human operators to control or even to observe and assess progress. The sheer number of control and sensor variables and the potential interactions between them can quickly overwhelm an operator's cognitive abilities. Furthermore, one popular use of such systems is in remote environments where it is too unsafe or expensive for humans to be present. Exploration of other planets, the oceans, and the antarctic are examples of these environments. These applications have the additional difficulty that communication bandwidth and delay further limit what humans can do. A common approach to these problems is to make the interaction between humans and robots occur at a fairly high level using a language that requires only a small amount of communication. The robots fulfill the high-level directives in an autonomous mode of operation that requires them to make all lower level control decisions.

Market-based planning provides one way to define the language of communication between operators and robots as well as a mechanism for handling the inter-robot planning required for autonomous execution of the operators' requests. Operators send high level tasks to the multi-robot system and may assign each task a reward value to indicate the relative priority of performing it. The robots then participate in an auction or other market-based mechanism to distribute the tasks and coordinate their efforts. The challenge in these systems is devising a bidding strategy for the robots that yields good performance.

Fig. 1. Screen shot from the FIRE simulator

In this paper, we begin by describing our multi-robot simulator and the multi-agent, three-tier planning and control architecture we use to control both the simulation and, in other work, real robots. We then describe the specifics of the market mechanism and propose an opportunity cost based method of bidding and a corresponding learning algorithm to determine those costs. We show analytic results showing the convergence of the method in a simplified version of the problem. Then we demonstrate the algorithm's peformance empirically in our simulator.

II. The FIRE Simulator

We first describe the FIRE (Federation of Intelligent Robotic Explorers) simulator. It provides both the testbed for our empirical results and a motivating scenario to use during the description of our methods. Figure 1 shows

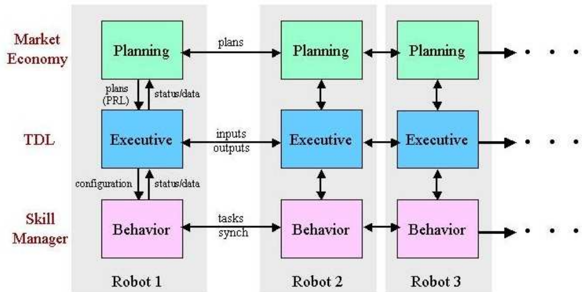

Fig. 2. Diagram of the distributed 3 layer control architecture. The "plans" being exchanged between planning levels are actually interpreted as bids, solicitations for bids, and awards of individual tasks.

a screen shot from the simulator. The scenario involves multiple heterogeneous rovers exploring the surface of Mars. There are different types of rovers with each having different capabilities (e.g. carrying different sensors) and performance (e.g. driving at different speeds). The environment provides different types of rocks and terrain.

For this paper all tasks will take the form of analyzing a rock at a known location. Each task description will specify the location of the rock, the type of sensor to be used, and the reward to be paid for successful completion of the task. We evaluate the performance of our multi-robot planning algorithms by the total reward they collect For further background on the FIRE simulator, see [4].

III. Distributed Three Layer Control Architecture

Each robot is controlled with a three tiered architecture consisting of a planning, executive, and behavioral layer with the behavioral layer responsible for direct interaction with a robot's sensors and actuators, or a simulator. The control architecture is built directly on the system described in [5], [1]. For the FIRE project, a distributed version of the architecture was developed as shown in figure 2. For more details about the communication vertically and horizontally between layers see [4].

Here we focus only on the planning layer and the inter-robot communication between planning layers. The planning layer of each robot communicates with the planning layers of the others in order to hold and participate in task auctions. By doing so, the system (implicitly) constructs a global plan and coordinates all of the robots' activities. The main components of the planning layer are the trader and the scheduler.

The optimization algorithm used by the scheduler is described in [2]. The input to the scheduler is a list of tasks with their rewards. Using the task information, the scheduler is able to compute a net profit for any specific schedule and it performs an efficient heuristic search to find the best one. In the FIRE architecture, it is also responsible for communicating with the executive layer to perform the scheduled tasks at the appropriate time.

The trader performs two separate roles. One is to hold auctions where it takes tasks it currently owns and puts them up for sale. If they can be sold profitably, the tasks are transferred to the winning bidder. The specifics of what makes a profitable transaction are given in the next section. The second role of the trader is to bid in auctions being held by others. Again, the bids are made according to estimated profitability.

IV. Bidding Based on Opportunity Cost

It is not obvious what market or bidding mechanism should be used when implementing market-based planners. A precursor to this work focused on the costs of performing tasks [3] and added features such as continuous arrival of new tasks into the system and dynamic reallocation of tasks during execution. The principles of that system are similar to Contract Nets [6], which inspired many market-based planning efforts. In a cost-based framework, each bidder estimates how much it will cost to perform a task and bids that cost. The task is allocated to the lowest bidder. The contract reflects an *obligation* to perform a task and the bidder is paid to take on that obligation. It is assumed that the bidder will end up paying much of this out as the "cost" of performing the task though the cost may be hypothetical. In the work by Dias, the cost will be real if the winning bidder then pays someone else to perform the task. If there is no continuous trading, it may not even be necessary to have a concept of "money" in the system.

It is preferable not to use a cost-only framework for planning. Doing so leaves no way to specify the relative importance of various tasks. Instead we use a reward-based framework. Every task has a reward which will be paid to whoever completes it successfully. The flow of money and meaning of a contract are different under this framework. A contract consists of an agent paying for the *opportunity* to perform a task and collect the corresponding reward. During bidding, each agent computes how much that opportunity would be worth and offers to pay that much for it. The highest bidder gets the task. Later it might resell that opportunity or it might complete the task and collect the reward, in either case hopefully gaining a net profit.

Before describing the bidding, we address the issue of temporal discounting. As specified so far, our planning problem is unconstrained. Eventually the robots will accomplish all tasks and thus all schedules are equally good. A common constraint in practice is a limit on the availability of resources such as time or power. We choose to impose a limit through the use of temporal discounting as is common in reinforcement learning and financial analysis. A reward, R, that will be received t time steps into the future is given a value of $\gamma^t R$ where γ is a discount factor between zero and one. γ reflects uncertainty about future events and is equivalent to assuming that the system will stop executing with probability $1-\gamma$ on each time step. That interpretation is particularly useful in our Martian rover domain since there is considerable uncertainty about the amount of time until a rover ceases functioning due to loss of power, failure of components, or difficulties navigating the terrain.

We now address the question of how an agent determines what the opportunity to perform a task is worth. The bidding algorithm is as follows:

1) Compute the expected net profit from the current schedule, P_{old}.
2) Use the scheduler to compute the best schedule including the new task being bid on and report the expected net profit from it, P_{new}.
3) Bid P_{new} - P_{old} if that quantity is positive. If the quantity is negative do not bid since no agent would be willing to pay another to accept an opportunity. It can always retain the opportunity and not take advantage of it for free.

A robot auctioning off a task follows exactly the same procedure except that it checks to see if the highest bid it receives more than compensates for the expected loss due to removal of that task from its schedule. If the highest bid would not compensate this loss, it keeps the task. In order to complete this bidding algorithm, we need to specify how a schedule is evaluated. The schedule evaluation algorithm considers each task in the schedule and does the following:

1) Determine the amount of time it will take to complete the task. This may include consideration of adjacent tasks and travel time between them as well as the amount of time required to perform the requested sensing operation.
2) Compute the opportunity cost as the product of the time required and cost per unit time.
3) Compute the time discounted value of the reward for the task minus the opportunity cost for it. Sum this value over all tasks in the schedule.

Step 3 of the bidding algorithm may cause some concern. In a cost-based framework, an agent is bound by its contract to complete a task. In our framework, the agent has the option to ignore the task. This will be the right thing to do if the entire system is already loaded with more valuable tasks, so we will instead consider the case where there is spare capacity somewhere in the system. If the owner of a task has available time it will always do the task since the purchase price is a sunk cost and it can get additional reward by completing it. If there is no available time then the task owner expects to receive zero reward from it. It will accept any positive bid it receives when auctioning the task and the new owner will complete it.

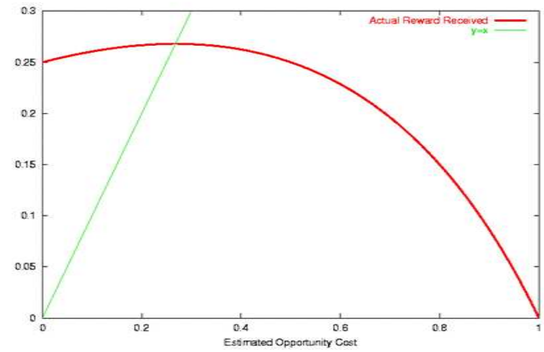

Fig. 3. Plot showing estimated opportunity cost vs reward achieved using that estimate. The straight line is y=x and is provided to make it easy to see the effects of applying the learning rule.

A. Learning Opportunity Cost

The missing component in the bidding algorithm is the determination of the opportunity cost per unit time. Since we are using an infinite time horizon with dynamic arrival of tasks, we operate under the assumption that there is a fixed opportunity cost per unit time and we only need to estimate that one number. We do this by observing the *actual* reward collected while the system runs. This allows us to use a very simple learning rule: $C^{t+1} = (1-\alpha)C^t + \alpha O^t$, where C^t is the estimated opportunity cost at time t, O^t is the observed average reward during time period t, and α is a learning rate.

Note that we assume opportunity cost applies only to time. This is not necessary and any other resource such as power or machine wear could be included using the same method. Another obvious extension is to generate additional features such as the amount of some resource remaining (useful for finite time horizon and other resource bounds), the utilization in the current schedule, the number and value of tasks currently in the system, the number of robots still functioning, etc. Then we would use an appropriate non-linear function approximator in place of the learning rule above. For this paper, we focus only on the simple case and attempt to understand whether and how it works.

V. Analytic Results

We have proposed a market-based planning system using bidding based on learned opportunity costs, but so far have no indication whether it will result in stable and/or optimal performance. To gain insight, we step back from the proposed algorithm and focus on a simplified example that can be understood analytically. Consider a single agent participating in the following market:

- A single new task is up for auction at each time step.

- Each task requires two time steps to complete.
- The agent uses no scheduler. If it accepts the task, it must perform the work immediately and will be unavailable to bid on the task offered on the next time.
- All other agents in the market are using a fixed strategy (they are not learning). The effect of their strategies and the distribution of task rewards available is that the agent sees tasks with profits available that are drawn from a uniform distribution on [0,1]. As presented earlier, the bidding algorithm never bids to include a profit for itself but this effect may be observed if a second price auction is used [7].

In this market, the agent merely has to decide on its threshold for accepting and rejecting tasks at each time step. Suppose the agent currently estimates that it will receive an average profit of x per time step and that the task available on the current time step offers a profit of y. Its choice is between accepting the task and getting a profit of y during the next two time steps or rejecting the task and getting zero profit this time step plus an expected x profit on the next time step. Therefore, it accepts the task if $y > x$.

Now assume that the agent learns slowly such that it keeps using the above policy until it discovers its true expected reward exactly. Then it changes its estimate, x, to be that value and repeats (the learning rate, α, is 1, but the amount of time covered by one observation is infinite). Using the rule above, the agent accepts a portion $1 - x$ of the tasks offered to it and rejects the other x of them. The total time spent doing this per task offered is (2 for the accepts and 1 for the rejects): $2 - x$. The agent receives an average reward of $(1 + x)/2$ for each task it accepts and thus accumulates a total of $(1 - x)(1 + x)/2$ profit per task offered. Dividing profit per task offered by time per task offered we get an average profit per time step as a function of estimated opportunity cost: $f(x) = (1-x)(1+x)/(4-2x)$. The agent then updates its estimate of profit per time step to be this value. Solving $f(x) = x$, we find a fixed point for these updates at $x = 2 - \sqrt{3}$. By checking the derivatives of f we see that $x = 2 - \sqrt{3}$ is also a local maximum and the fixed point is stable.

Figure 3 can give additional intuition. When the agent over-estimates its expected profit it will accept very few tasks and get a small reward because of it. When it under-estimates, it will accept too many low profit tasks and thus perform sub-optimally. In either case, it will end up with an under-estimate of its optimal performance. Using that under-estimate results in performance better than the estimate and repeating the process moves it toward the optimum.

VI. Empirical Results

We now present an empirical evaluation of our proposed algorithms using the FIRE simulator described earlier. Except where noted, the algorithm follows that presented in section IV rather than the simplified method used for the analysis in section V. In all of these experiments, each trader repeatedly tries to auction off the tasks it owns in a round robin fashion.

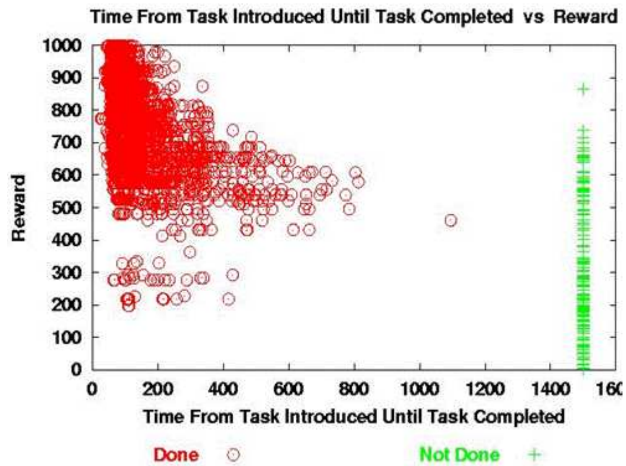

Fig. 4. Plot showing task reward vs completion time. The crosses on the far right represent tasks that were not complete when the simulation was terminated.

A. Experiments with No Learning

In the first experiments, we demonstrate the operation of the overall system without learned opportunity costs. We fix the estimated opportunity cost at a value of 10 dollars/second since this is known to be a conservative estimate. Based on figure 3 and our experience, under-estimates are much less devastating to performance than over-estimates. The rewards for tasks are drawn from a uniform distribution ranging between 0 and 1000 dollars. 50 new tasks are generated every 40 seconds (the parameters of the simulator are set such that robots act faster than real-time in order to speed up our experiments). There are six identical robots in the system and all tasks are identical except for their reward and the location of the rock, which is drawn from a uniform distribution over a square.

During a long run of the simulation the time between the introduction of a task and its completion were recorded. A plot of this time against the reward for the task shows the system exhibiting the desired behavior (see figure 4). Tasks of very high reward are always completed almost immediately. Moderate reward tasks are generally completed less urgently and low reward tasks are left undone. These tasks might be done eventually, but new tasks are arriving at a rate faster than they can be completed, so a backlog of low reward tasks builds up as the simulation runs. Also observe that there is some spread in the points. Since the robot's also consider how far they will have to drive to a rock, they sometimes do a low-reward job because it is nearby or put off a valuable one because it is far away.

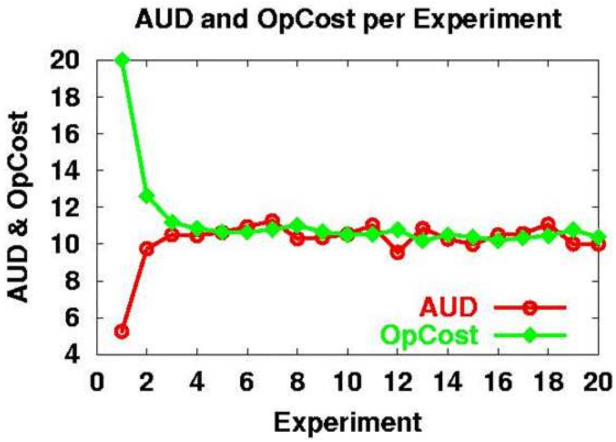

Fig. 5. Plot showing estimated opportunity cost and corresponding average undiscounted reward received (AUD) as a function of time. Each “experiment” refers to one 15 minute block of time after which the opportunity cost estimate was updated.

On average each robot collected 10.8 dollars/second in the simulation. A test using the cost-only version of bidding reported in [4] yielded an average of 8.6 dollars/second. The difference is not surprising, but does further demonstrate the smart use of rewards in the market.

B. Experiments with Learning on Homogeneous Robots

In the second set of experiments, the robots were allowed to learn as a group. All robots pooled their observations to obtain a joint estimate of their opportunity cost. During the runs, the average undiscounted reward for the previous 15 minutes was computed every 15 minutes and then the learning rule was applied with $\alpha = 0.5$. The entire simulation was run for 5 hours (yielding 20 such updates). Several runs were completed with the initial estimate of opportunity cost set at a variety of levels. In all cases, the cost estimates and performance converged to a range between 10 and 11 dollars/second.

Figure 5 shows one of these runs where the opportunity cost was initialized to a very high value. From the plot we see that the initial estimate was 20 and using that estimate yielded an average reward received of a little under 6 in the first 15 minute period. Applying the learning rule then gave an estimated opportunity cost of almost 13 for the second 15 minutes of the experiment. By the 5th update, the learning has converged and the values continue to oscillate due to the randomness of the simulation and the relatively high learning rate.

C. Experiments with Learning on Heterogeneous Robots

Consider a scenario with different types tasks. One type requests the use of an A sensor and the other requests the

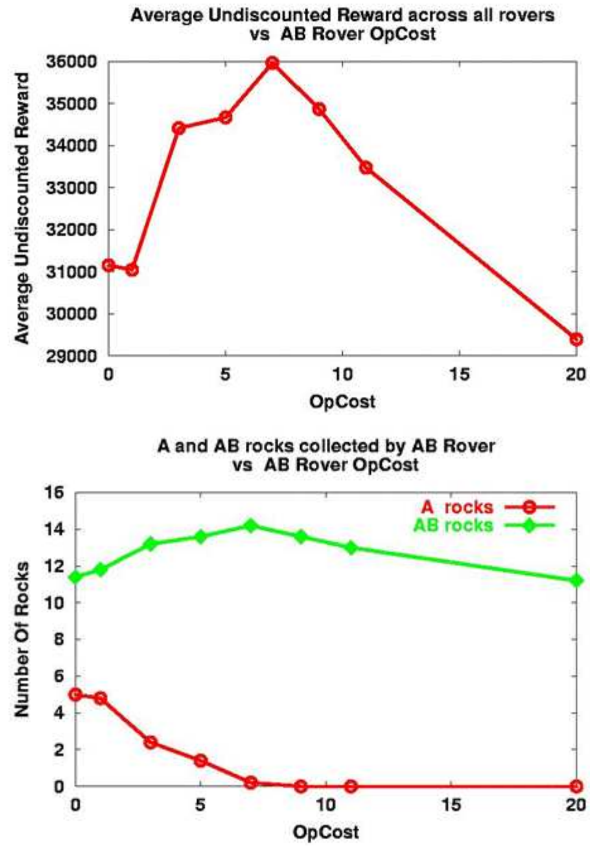

Fig. 6. Plots showing the effect of varying AB robots' estimate of their opportunity cost. The top plot shows overall system performance measured by the total undiscounted reward collected. The bottom plot shows numbers of A and B rocks chosen by AB robots.

use of a B sensor. Suppose we have two different types of robots. One carries only an A sensor and thus can only perform tasks requiring an A sensor. The other carries both sensors and can perform any task.

Intuitively, we would like AB robots to choose mostly B tasks even though they are equally well suited for A tasks. This would leave most of the A tasks for the A robots and ensure that B tasks don't get neglected, thus achieving the highest overall reward for the system. Is there a way to achieve this automatically in our system? Consider the actual opportunity costs for these robots. The AB robots have a higher opportunity cost (higher reward typically achieved on average) because they can do any task and thus have the best ability to pick high-value tasks near their current position. A robots have lower opportunity cost because they only have A tasks available to them. Now suppose an A robot and an AB robot are bidding on the same A task and suppose it is the same distance from both of them. The AB robot should lose this auction because its opportunity cost is higher and thus its estimated profit and the amount it is willing to bid for the task is lower. Of course, AB robots will not always lose since the current

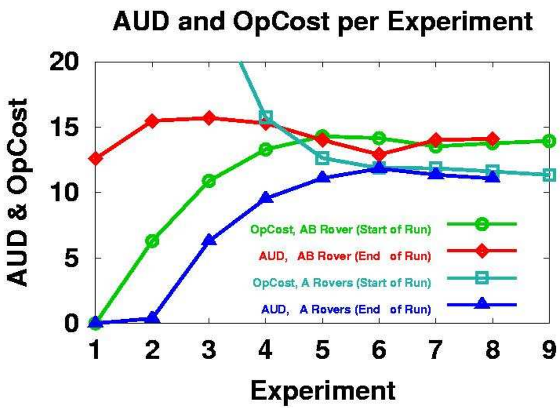

Fig. 7. Plots showing the results of opportunity cost learning with heterogeneous robots. AUD is average undiscounted reward received. OpCost is estimated opportunity cost for that time period. Each time period ("experiment") is 15 minutes long

state of their schedule and the location of the rock also contribute to the expected costs and rewards for the tasks. The high-cost AB robots will still win B tasks because no other robots can perform them.

Before trying learning, we performed some experiments to test the effects of the opportunity cost estimate on system behavior. There are 3 A robots, 3 AB robots, and an equal number of A and B tasks generated. The opportunity cost of the A robots was fixed at 5 (well below the correct value). During separate runs, the opportunity cost for the AB robots was varied over a range of 0 to 20. Figure 6 shows the results of these experiments. When the AB opportunity cost is low, those robots perform a significant number of A tasks in addition to their B tasks and cause under-utilization of the A robots. The overall performance of the system is poor. As the AB opportunity cost is increased, AB robots do more B tasks and less A tasks. Corresponding to this change, overall system performance increases up until the cost reaches 7. Beyond that, the AB robots quit doing some B tasks as well because they expect more valuable tasks to arrive. As a result overall system performance decreases.

Finally, we allowed each robot type to learn its own opportunity costs in this scenario. Again the simulations were initialized with several different values for the opportunity cost and convergence was observed for all of them. Interesting results from one such trial are shown in figure 7. In this run, the initial cost for A robots is set very high at 100, while the initial cost for AB robots is set very low at 0. This is the extreme opposite of their correct values. In the figure we see that during the first 15 minutes, the A robots collected no reward because their cost estimate made no task appear profitable. The AB robots collected a sub-optimal, but reasonable reward. Then for the second 15 minutes, the AB estimate was increased to about 6 and the A estimate was reduced to about 50 (still way off the chart). The result of this was that A robots still accomplished almost no tasks, while AB robots became more selective and achieved higher profit. By the third period, the A estimate has come down to a level where A robots start doing things. As they take more jobs, it begins to cut into the profitability of the AB robots, but the overall system performance is better. By the 7th period, the costs have converged exactly as we had hoped. The A robots have concluded that their opportunity costs are lower than those of the AB robots and they are doing the majority of the A tasks.

VII. Future Work

The current system demonstrates the successful use of opportunity cost learning in a market-based planner. Their remain many extensions that would be helpful and/or necessary in real systems. The most obvious is is learning based on a broad array of features and the ability to handle non-linear opportunity costs.

Further work is also needed to bridge the gap between the analytic and empirical results we presented. The analytic results are similar to what appears as yield management in operations research and we expect to be able to extend those results similarly. The interaction between the scheduler's ability to move low-priority tasks to the end and the trader's ability to get rid of them is particularly challenging to analyze. Generalizing the analysis for simultaneous learning by all agents is the topic of much ongoing effort in the machine learning community.

Acknowledgment

We would like to thank the rest of the FIRE group including Vince Cicirello, Bernardine Dias, Dani Goldberg, Maayan Roth, Steve Smith, Trey Smith, Tony Stentz for the many useful discussions leading up to this work.

References

[1] R. Peter Bonasso, R. James Firby, Erann Gat, David Kortenkamp, David Miller, and Mark Slack. Experiences with an architecture for intelligent, reactive agents. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2), 1997.

[2] V. Cicirello and S. Smith. Distributed coordination of resources via wasp-like agents. In *The First NASA Goddard/JPL Workshop on Radical Agent Concepts*, 2002.

[3] B. Dias and A. Stentz. A market approach to multirobot coordination. Technical Report CMU-RI-TR-01-26, The Robotics Institute, Carnegie Mellon University, 2001.

[4] D. Goldberg, V. Cicirello, B. Dias, R. Simmons, S. Smith, and A. Stentz. Market-based multi-robot planning in a distributed layered architecture. In *Proceedings of the 2003 International Workshop on Multi-Robot Systems*, volume 2, pages 27–38. Kluwer Academic Publishers, 2003.

[5] Reid Simmons. Structured control for autonomous robots. *IEEE Transactions on Robotics and Automation*, 10(1):34–43, February 1994.

[6] R. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, pages 1104–1113, December 1980.

[7] W. Vickrey. Counterspeculation, auctions, and competitive sealed tenders. *Journal of Finance*, 16:8–37, 1961.