The 4th International Conference on Field and Service Robotics, July 14–16, 2003

Market-based Multirobot Coordination Using Task Abstraction

Robert Zlot **Anthony Stentz**

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213
robz@ri.cmu.edu tony@cmu.edu

Abstract

In this paper, we introduce a novel approach to multirobot coordination that works by simultaneously distributing task allocation, mission planning, and execution among members of a robot team. By combining traditional hierarchical task decomposition techniques with recent developments in market-based multirobot control, we obtain an efficient and robust distributed system capable of solving complex problems. Essentially, we have extended the TraderBots *market-based architecture to include a mechanism that distributes tasks among robots at multiple levels of abstractions, represented as task trees. Results are presented for a simulated area reconnaissance scenario.*

1 Introduction

Many applications of multirobot systems can be viewed in terms of problems dealing with scarce resources in highly uncertain, dynamic environments. The optimization problems involved are usually highly unscalable, making centralized solutions impractical. Distributed solutions can be faster and more reliable; although these benefits are often obtained in exchange for guarantees on solution quality. By extending market-based multirobot coordination architectures to deal with hierarchical representations of tasks, we produce a distributed mechanism that can quickly generate efficient solutions to complex optimization problems.

Our research addresses both the task allocation problem as well as multirobot planning. The multirobot task allocation problem can be stated as follows: *given a set of tasks and a group of robots, what is the optimal way to distribute the tasks among the robots?* In terms of mission planning, it is desirable to divide the planning among the robot team to exploit the presence of multiple processors and asymmetric local information, but it is difficult to do this while still ensuring that the resulting global plan is efficient.

This paper presents a task-oriented market mechanism in which the participating agents exchange tasks at multiple levels of abstraction, represented as task trees. This effectively distributes task decomposition among the robots, and creates a market where the best plans will dominate.

In the next section we discuss related work in both market-based multirobot coordination and in hierarchical task decomposition. We then detail our approach, which introduces a unique market framework based on task trees. We first define the task tree representations that are used, followed by a description of our task tree market mechanism. Results from simulation for an area reconnaissance problem follow.

2 Related Work

We handle the coordination of multiple robots through the use of the *TraderBots*[1] market architecture. Well-known market mechanisms (such as auctions and contracts) are applied to a team of robots and software agents participating in a virtual economy. The market is designed such that the act of each robot trying to maximize its individual profit results in a globally efficient outcome. Market mechanisms have long been applied to software agents, beginning with Smith's Contract Net Protocol in 1980 [12]. Within the *TraderBots* economy, participants can negotiate contracts to execute tasks, or trade in resources required to complete such tasks (such as the use of specialized sensors or tools, storage capacity, and processor time). Market architectures are typically based on distributed control; however Dias and Stent propose a system which includes opportunistic centralized optimization within subgroups of the team [4]. Market-based implementations for multirobot systems have been successfully demonstrated in simulation [3, 2, 9] and several physical robot applica-

[1] The name *TraderBots* was recently adopted for our market-based coordination work that was started by Stentz and Dias in 1999 [13].

tions [6, 14, 16]. Approaches taken in some market-based systems make use of a central auctioneer to allocate tasks among the robots [9, 6]. In contrast, we allow all of the participating robots and software agents to dynamically take on the roles of buyer and seller. To date, market-based multirobot coordination mechanisms have only considered trading tasks at a single level of abstraction.

Our work is also related to research in the area of AI hierarchical task networks (HTN). Task networks are collections of tasks which are related through ordering constraints. HTNs extend task networks by allowing the tasks in the network to be abstract tasks which can be decomposed to ultimately form an executable task network. Erol *et al*. [5] present an overview of HTN planning, and provide a sound and complete HTN planning algorithm.

Although there has been a great deal of focus on task decomposition and HTN or hierarchical planning in the AI literature, few deal explicitly with the case of multiagent planning. Clement and Durfee [1] present an approach for coordinating hierarchical plans of multiple agents. Several agents send their top-level plans to a central planner which searches for a way to merge them. If it cannot do so, then some plans may need to be expanded, which requires agents to send *summary information* at increasingly deeper levels of the tree until a feasible global plan can be produced.

Hunsberger and Grosz [7] describe a system where multiple agents are able to bid on subtasks which are related through task network structures allowing nodes to be grouped together as *roles*. The roles are then assigned by clearing multiple centralized combinatorial auctions. Our work differs in that we dynamically construct task hierarchies rather than using predefined recipes, and we allow the distribution of planning computation among agents. We also use the tree structure to dictate which tasks can be bid on in combination, rather than specifying roles or considering all combinations of tasks/roles to bid on.

3 Task Tree Auction Mechanism

3.1 Task Trees

We represent composite tasks using task trees. A task tree is defined as $\mathcal{T} = (T, r, E)$, where T is a set of tasks or nodes, $r \in T$ is a unique root task (node), and E is a directed edge set which specifies parent-child relationships between the tasks in T (see Figure 1). Each successive level of the tree represents a further refinement of an abstract task; the root is the most abstract task, and the lowest level contains primitive actions that can be executed by a robot or another agent in the system. Subtasks are related to their parents through logical operators, such as *AND* and *OR*. To execute an *AND* task, all of its subtasks must be executed; to execute an *OR* task, any one of its subtasks may be executed. Constructing a task tree involves performing a hierarchical decomposition on an abstract task.

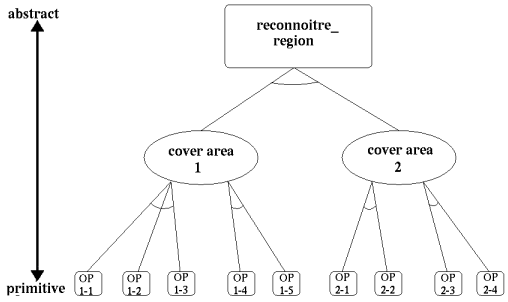


Figure 1: An example task tree for an area reconnaissance scenario with two areas. The arcs on the edges represent *AND* operators, and the absence of an arc represents an *OR* operator. Note that for the *cover area* tasks, there are two alternative decompositions specified in the tree.

3.2 Task Tree Auctions

The market in our system is primarily based on task tree auctions; contracts are sold for executing trees of tasks. When an auction is announced, each robot determines its valuation (which is based on cost estimates) for each task in the task tree, whether it is an interior node (abstract task) or a leaf (primitive/executable task). After computing its costs, a robot can then bid on the tasks that it expects are going to be profitable, attempting to win a contract to perform those tasks in exchange for payment. A specially designed auction clearing algorithm ensures that the auction is cleared optimally (minimizing cost) in polynomial time. The winner of the auction is responsible to the seller and must ensure that the tasks are completed (either by executing the task itself, or by subcontracting parts of the task to other teammates in future negotiations) before receiving payment. In general, the payment can depend on the quality of the job completed.

Any agent in the system can act as an auctioneer at any time. Some robots or agents may start with knowledge about the mission objectives, new tasks may be discovered while processing information acquired during execution, or a human operator may introduce them into the system. When a robot has new information about tasks or mission objectives, it can decompose those tasks (not necessarily completely) and call an auction for the corresponding task tree. Currently, each auction is for a single task tree.

3.2.1 Bid Valuation

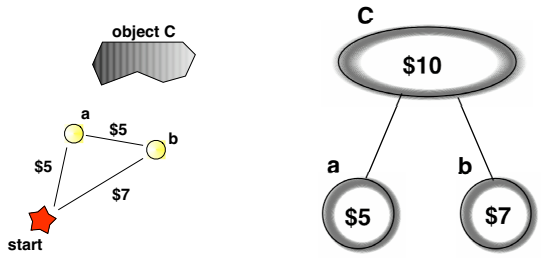
In the case of a primitive (leaf) task, a robot's bid for the task is based on the expected marginal cost of executing the task. If the robot wins a primitive task, it can insert it into its schedule and perform it at the appropriate time.

For an abstract (interior) task, the valuation is the cost of minimally satisfying the task represented by the corresponding tree node. For example, if the connective on the tree node is *AND*, the value of the task is the expected marginal cost of performing *all* of the subtasks (children) of the node. If it is an *OR* node, then it is the *minimum* expected cost of executing *one* of the children. However, the bidding robot may have a better plan than the auctioneer. This can be due to a difference in the current state or resources of the bidder, or due to asymmetric local information. The bidder can perform its own decomposition of the abstract task, and if the resulting plan is of lower cost, then it bases its bid on the abstract node on this new plan. If the plan is indeed a better plan, the bidder wins the task in the auction and it can replace the auctioneer's original plan with its own (it also may subcontract some or all of the new plan in subsequent auctions). This enables the distribution of planning among the robots.

3.2.2 Auction Clearing

A task tree auction can be viewed as a special case of a combinatorial auction. A combinatorial auction is an auction in which there are multiple items up for sale, and the bidders can bid on any combination, or *bundle*, of items. In a task tree auction, *primitive tasks* correspond to *items* and *abstract tasks* roughly correspond to *bundles*[2]. While in a general combinatorial auction any arbitrary set of items may form a bundle, in a task tree auction the structure of the tree dictates which bundles may be traded. For example, in figure 2 the available bundles are primitive tasks *a*, *b* (singleton bundles) and abstract task *C*. Combinatorial auctions allow bidders to express complimentarity and substitutability between items or tasks. For example, in figure 2 the cost to drive to goal *a* followed by goal *b* ($10) is less than the sum of the costs of driving to each one separately ($12). A bidder can express this complimentarity between the tasks by bidding on the 'bundle' *C* which has a cost that is less than the sum of the individual costs of its subtasks.

The process of allocating the tasks in the task tree to the highest bidder, is an instance of a combinatorial auction winner determination problem (CAWDP). The objective for the auctioneer is to clear the auction by selling the combination of items that would maximize revenue (in which case tasks have been sold to the robots that are expected to be best suited to execute them). CAWDP is known to be $\mathcal{NP}$-hard in general [11]. However, since the bids are structured as a tree, winner determination can be computed



(a) (b)

Figure 2: An example partial task set with its task tree representation. The goal is to create a stereo image of object *C*, which can be achieved by capturing images from multiple viewpoints. **(a)** The starting location of a robot is shown along with the locations of two viewpoints. The estimated mobility costs between points are shown. **(b)** A task tree corresponding to the tasks in *(a)*. The two primitive tasks *a* and *b* are shown together with the abstract task *C*.

in polynomial time. We use a modified version of an algorithm due to Rothkopf *et al.* [10], which has time complexity $O(n \cdot |T|)$ (where T is the set of all nodes in the tree and n is the number of leaves in the original tree)[3].

One important design consideration is the bidding language used for task tree auctions. There is inherently a tradeoff between the expressiveness and the simplicity of the bidding language for both the bidders and the auctioneers. The auctioneer must ensure that every bidder cannot win more than one bundle in a single auction because the individual bids do not take into account dependencies between the bundles. The simplest possible bidding language for a task tree auction is to allow bids on only one node of the tree. A slightly more complex specification (and the language that we currently implement) is to choose all nodes along a root-to-leaf path, allowing the path to branch out to become multiple paths at *OR* nodes. Both of these languages ensure that bidders are awarded only one bundle with the current auction clearing algorithm. In contrast, an *exclusive-or* bid language would allow the robots to bid on all tree nodes, but with the connotation that the robot can be awarded only one of the bundles. This can lead to a more efficient outcome, but at a cost to the auctioneer of running a more complex auction-clearing algorithm. We are currently developing algorithms to deal with this more expressive bidding language efficiently.

Auctions can be called in a series of rounds. From an optimization standpoint, a round can be thought of as an improvement step in a distributed local search algorithm. If the auctions in a round are not simultaneous, then after each round the global solution is guaranteed to be no worse than it had been before the round started. Therefore, when starting with an initial feasible solution (or once the

[2]There is a slight distinction between an abstract *AND* task and a traditional bundle in that abstract tasks may be decomposed differently by a bidder – but, for the purposes of auction-clearing, this distinction is unimportant. Abstract *OR* tasks are also handled differently since they represent alternatives among their subtasks, rather than a combination of subtasks.

[3]Our algorithm is described in detail in [15].

initial auction round has terminated) the system behaves like an *anytime* algorithm – a feasible solution is available quickly, and the quality of this solution can only be improved over time. In the case of a multirobot application, the robots can start to execute a feasible plan and at the same time improve the plan through further auctions while executing. The same mechanism can be applied to auction online tasks added to the system during execution (*e.g.* new tasks may be added by a human operator or autonomously generated based on information gathered by the robots).

4 Experiments

Our approach was tested using a graphical simulation of an area reconnaissance scenario. The objective for the robot team is to view all of several predefined named areas of interest (NAIs) within a large geographical area while minimizing (distance-based) travel costs. Each robot is equipped with a range-limited 360° line-of-sight sensor. The terrain is represented as a grid where each cell has associated mobility cost and utility values, representing the cost to traverse the cell and the benefit of viewing the cell respectively.

One robot (robot *R*) is initially aware of the global mission, *i.e.* it starts with an abstract task describing the overall mission. Robot *R* then performs a task decomposition, refining the global task into subtasks representing the task of covering each NAI, and in turn each of the NAI subtasks is broken down into several groups of observation points (OPs) from which a robot can view part of an NAI (Figure 3). The global reconnaissance task is decomposed using a connected components algorithm, where connected areas that were marked with high utility are taken to be the NAIs. The NAI tasks are then decomposed by greedily selecting OPs that have the highest utility-cost tradeoff (a weighted sum of utility and cost is used). For computational purposes, we limit the number of possible OPs considered for each area to twelve per NAI (which surround the perimeter of the NAI). The NAIs are decomposed twice, producing two alternative plans for robot *R*'s initial decompositions.

Once the decomposition is complete, robot *R* holds a task tree auction, thereby distributing subtasks among the team. Other robots may bid on nodes in any level in the tree, allowing them to perform their own decompositions for interior nodes. The auctions then proceed in rounds in which each robot optionally holds a task tree auction for one of its subtrees in a round-robin fashion.

The mission objective is achieved by the robot team visiting all of the required OPs, minimizing the overall distance traveled. This optimization problem is an instance of the multi-depot traveling salesman path problem (MD-

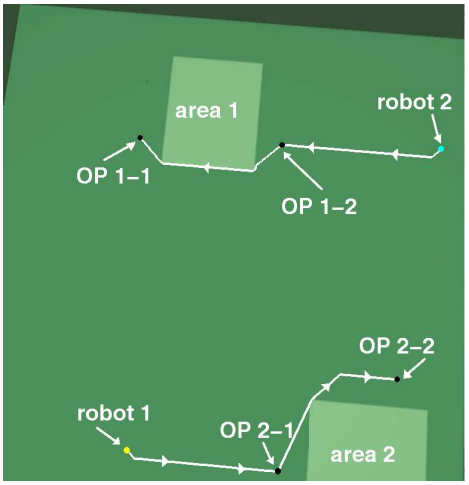


Figure 3: Example scenario. The objective is to generate observation points (OPs) to view the two areas with minimum travel cost.

TSPP), which can be stated as follows: *given a list of n cities (OPs), the inter-city costs, and m salesmen (robots) located at different start cities, find the set of m paths (each starting at one of the salesmen locations) that visits all n cities and minimizes the total travel cost.* The traveling salesman path problem (TSPP) is a special case of the MD-TSPP (with one salesman) and is a well-known $\mathcal{NP}$-hard problem. Thus MD-TSPP is also $\mathcal{NP}$-hard (and we cannot be guaranteed to find the optimal solution in polynomial time). Since it is not known which group of cities minimizes the objective function and still ensure area coverage, there is another layer of complexity added to the problem; that is, we may have to solve the MD-TSPP for exponentially many subsets of OPs.

5 Results

The above scenario was run varying the number of robots and NAIs, with 100 runs for each case. The robots and NAIs are randomly placed with a 255*x*255-cell grid at the start of the simulation. NAIs were randomly-sized rectangles with edge lengths drawn uniformly at random in the range of 22 to 32 grid cells. We compared the task tree algorithm with three other task allocation algorithms.

The first algorithm simply auctions off the leaf-level nodes of the task tree (decomposed by the first robot) to the highest bidder one at a time, in random order, until the entire tree is satisfied. If an interior node is satisfied by an intermediate allocation, then none of that node's children are sold in subsequent auctions. This algorithm is representative of the outcome that would be reached if replanning (when bidding on interior nodes) were not permitted (we will refer to this algorithm as "Fixed-Tree Leaf auction"

algorithm or FTL).

The second algorithm is a greedy algorithm (which we will refer to as *GR*) that looks at all possible OP candidates (12 per AI) and auctions non-redundant OPs off sequentially until coverage is complete. All of the potential OPs are bid on in each round, but only the OP with the lowest-cost bid is sold. There is no tree decomposition involved – all potential OPs are considered. This algorithm behaves similarly to multirobot coordination architectures that greedily allocate the best suited task to the best suited robot from a central agent (*e.g.* [14, 8, 6]).

The third algorithm (*OPT*) computes the globally optimal solution. Since the problem complexity is highly exponential, we could only compute the optimal solution for very small problem instances.

We compared the overall solution cost of the task tree auction algorithm (denoted by c_{TT}) to the solutions produced by each of the other algorithms (c_{FTL}, c_{GR} and c_{OPT}). We also compared approximations of the execution times (t_{FTL}, t_{GR}, t_{OPT} and t_{TT}) of the different algorithms. The execution times were estimated by timestamping the start and end times of each algorithm. Results are presented in tables 1 and 2.

Robots	Areas	c_{TT}/c_{FTL}	c_{TT}/c_{GR}	c_{TT}/c_{OPT}
2	1	.85	1.03	1.19
4	2	.86	.92	
5	3	.88	1.00	
7	5	.88	1.01	
4	8	.91	1.10	

Table 1: Experimental results: comparison of solution costs (results shown are averages of the ratio of solution costs taken over 100 runs).

In terms of solution cost, the task tree algorithm is 10-15% better than the FTL algorithm. One reason for this is that TT allows distributed replanning, so the TT solution is intuitively expected to be no worse than the FTL – in the worst case no replanning is done by the task tree algorithm and the original tree decomposition is preserved[4]. In addition, the task tree algorithm also has an advantage because it allows reallocation through task subcontracting, permitting agents to discard tasks that they may have been too hasty in purchasing earlier on. It should also be noted that if the solutions are, as we suspect, close to optimal, then a 10-15% margin of improvement is significant. We can see this in the 2-robot 1-area case in which we were able to compute the optimal solution. Here the task tree algorithm was only 19% worse than optimal, as compared to FTL which was almost 40% worse than *OPT*. The execution time listed for the task tree algorithm (t_{TT}) is the time taken to reach the local minimum cost; although FTL appears to run much faster, TT is an anytime algorithm and often reaches a lower cost than FTL long before it reaches its local minimum.

[4] TT almost always performed better that FTL, but there were a few exceptional cases where FTL did slightly better: since the task tree algorithm is a local search based on myopic cost estimates, the solution reached can depend on the order the tasks are allocated to each robot.

Robots	Areas	t_{FTL}/t_{TT}	t_{GR}/t_{TT}	t_{OPT}/t_{TT}
2	1	.21	3.9	117
4	2	.14	2.7	
5	3	.10	2.8	
7	5	.08	3.4	
4	8	.06	3.1	

Table 2: Experimental results: comparison of approximate execution times (results shown are averages of the ratio of execution times taken over 100 runs).

On average, the task tree algorithm and the *GR* algorithm produce about the same solution quality; however, the task tree algorithm is faster and does not rely on a central auctioneer. The execution time for the task tree algorithm shown in table 2 reflects the time taken to reach the locally optimal solution. Though TT found its local optimum three to four times faster than *GR*, the task tree algorithm produced a feasible solution in an even shorter time and improved that solution until it reached equilibrium at the time reflected in the table. The task tree algorithm guides the search quickly through a reduced search space without compromising the solution quality, while also allowing for a distributed search.

Table 3 shows the effects of allowing auction winners to replan abstract tasks that they have won. Task tree auctions were run twice on each of 100 instances. In one run task decomposition was performed only once at the beginning by the initial auctioneer – tasks were not permitted to be decomposed once the original plan was developed. In the second case, the full task tree algorithm was used, allowing further decompositions after abstract tasks exchanged hands. Table 3 compares the solution costs from these two scenarios. The improvement ranged between 3 and 10% of total cost. Intuitively, we would expect the solution to be more efficient when allowing full decomposition; however, the magnitude of the improvement will depend on the specifics of the application, such as the size of the grid used and how many alternative plans the auctioneers choose to offer in the initial decomposition.

6 Conclusions

We have introduced a new method for distributing execution and planning over a team of robots and software

Robots	Areas	$c_{replan}/c_{no-replan}$
2	1	.97
4	2	.96
5	3	.90
7	5	.91
4	8	.97

Table 3: Experimental results: comparison of solution quality of TT algorithm with replanning vs. the solution quality of TT algorithm without replanning (results shown are averages of the ratio of the solution cost with replanning compared to the solution cost without replanning).

agents, which combines ideas from hierarchical planning with a market-based multirobot coordination architecture. Empirical results in computer simulation show that task tree auctions can produce cost-efficient solutions to difficult optimization problems in a time-efficient manner.

In future work we will look at extending the task description language to handle richer task constraints and interactions, such as partial order precedence relations and conflicts arising between tasks competing for the same resources. Another interesting issue to explore will be decisions on when decomposition should take place. In some instances it can be costly to decompose abstract tasks, so it may be beneficial to leave the decomposition to the buyers of the task, or until immediately before the task must be executed. This would require the ability to adequately reason about the costs and benefits of tasks at an abstract level. We are also working on alternative auction clearing algorithms that will allow the bidders to use a more expressive bidding language, leaving the burden of selecting which of a robot's bids to accept to the auctioneer. Additionally, we would like to address other issues such as further generalization of the tree structures, permitting commitments to be broken and subcontracts to be sold, and how to efficiently deal with replanning when new information is discovered that affects upcoming plans.

We are currently performing further experiments and are in the process of porting the system to a team of 10 ActivMedia Pioneer P2DX robots. The hardware and software infrastructure is in place and first results are expected in the very near future.

7 Acknowledgments

This work was sponsored by the U.S. Army Research Laboratory, under contract **Robotics Collaborative Technology Alliance** (contract number DAAD19-01-2-0012). The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U. S. Government. The U. S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation thereon. The authors also thank Bernardine Dias for helpful contributions and advice.

References

[1] B. J. Clement and E. H. Durfee. Top-down search for coordinating the hierarchical plans or multiple agents. In *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 252–259, 1999.

[2] M. B. Dias, D. Goldberg, and A. Stentz. Market-based multirobot coordination for complex space applications. In *The 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, 2003.

[3] M. B. Dias and A. Stentz. A free market architecture for distributed control of a multirobot system. In *6th International Conference on Intelligent Autonomous Systems (IAS-6)*, pages 115–122, 2000.

[4] M. B. Dias and A. Stentz. Opportunistic optimization for market-based multirobot control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002.

[5] K. Erol, J. Hendler, and D. S. Nau. Semantics for hierarchical task-network planning. Technical Report CS-TR-3239, University of Maryland College Park, 1994.

[6] B. P. Gerkey and M. J. Matarić. Sold!: Auction methods for multirobot control. *IEEE Transactions on Robotics and Automation Special Issue on Multi-Robot Systems*, 18(5):758–768, October 2002.

[7] L. Hunsberger and B. J. Grosz. A combinatorial auction for collaborative planning. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS)*, 2000.

[8] L. Parker. Adaptive heterogeneous multi-robot teams. *Neurocomputing, special issue of NEURAP '98: Neural Networks and Their Applications*, 28:75–92, 1999.

[9] G. Rabideau, T. Estlin, S. Chien, and A. Barrett. A comparison of coordinated planning methods for cooperating rovers. In *Proceedings of the AIAA 1999 Space Technology Conferece*, 1999.

[10] M. H. Rothkopf, A. Pekec, and R. M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.

[11] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1-2):1–54, 2002.

[12] R. Smith. The contract net protocol: High-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C-29(12), 1980.

[13] A. Stentz and M. B. Dias. A free market architecture for coordinating multiple robots. Technical Report CMU-RI-TR-99-42, Robotics Institute, Carnegie Mellon University, December 1999.

[14] S. Thayer, B. Digney, M. B. Dias, A. Stentz, B. Nabbe, and M. Hebert. Distributed robotic mapping of extreme environments. In *Proceedings of SPIE: Mobile Robots XV and Telemanipulator and Telepresence Technologies VII*, 2000.

[15] R. Zlot and A. Stentz. Multirobot control using task abstraction in a market framework. In *Proceedings of the Collaborative Technology Alliances Conference*, 2003.

[16] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *Proceedings of the International Conference on Robotics and Automation*, 2002.