Dynamic Task Allocation for Robots via Auctions

Maitreyi Nanjanath and Maria Gini
Department of Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455
Email: {nanjan,gini}@cs.umn.edu

Abstract— **We present an auction-based method for the allocation of tasks to a group of robots. The robots operate in a 2D environment for which they have a map. Tasks are locations in the map that have to be visited by the robots. Unexpected obstacles and other delays may prevent a robot from being able to complete its allocated tasks. Therefore tasks not yet achieved are rebid every time a robot accomplishes a task. This provides an opportunity to improve the allocation of the remaining tasks and to reduce the overall task completion time. We present experimental results that we have obtained in simulation using Player/Stage with this task allocation mechanism.**

I. Introduction

There are many real-world problems in which a group of robots is required to accomplish a set of tasks. We are interested in situations where, while a single robot could do all the tasks, sharing the work with other robots will reduce the completion time. Therefore, we need a method to distribute tasks efficiently to the robots.

We describe a fast method based on auctions to perform this task redistribution. The method does not guarantee an optimal allocation, but it is specially suited to dynamic environments, where execution time might deviate significantly from estimates, and where the ability to adapt to changing conditions is the key to success.

Auctions have been suggested for allocation of computational resources since the 60s [1]. The Contract Net [2] is perhaps the most well known and most widely used protocol for distributed problem solving, where the agents that participate in the bidding are cooperative and do not intentionally lie about their commitments and costs.

Auctions are traditionally used with self-interested agents, but there is a growing body of work on using auctions to allocate tasks among cooperative agents. Auction-based methods for allocation of tasks are becoming popular in robotics [3]–[5] as an alternative to other allocation methods, such as centralized scheduling [6] or application-specific methods, which do not easily generalize [7] to other domains.

Task allocation can be done using combinatorial or single-item auctions. In combinatorial auctions, all the tasks are presented for bidding and agents can bid on groups (bundles) of tasks that they wish to have. Combinatorial auctions therefore involve having to compute bids for every bundle of tasks in the set of tasks, and to compute winners for the bundles. Their computational and communication costs are too high to make them practical for robots.

In sequential single-item auctions tasks are auctioned one at a time, and they are assigned to the winner promptly. This simplifies bidding and clearing the auction, but can miss the optimal allocation. Because of their simplicity, single-item auctions are typically used for robots [8].

In this paper we study the problem of allocating tasks to robots, where tasks are simply locations in a map that have to be visited by the robots. The problem of optimal allocation then becomes a problem of optimal routing. It has been shown [9] that producing an optimal allocation that minimizes either (a) the sum of path costs over all robots, or (b) the average path cost over all robots, or (c) the maximum path cost over all targets is NP-hard.

The auction mechanism we propose tries to shorten the total time taken to complete all the tasks. This is equivalent to minimizing the sum of path costs over all robots, given the assumption of constant and equal speed of travel for all the robots. The algorithm we use finds close-to-optimal solutions that are fast to compute. It is flexible, allowing robots to rebid when solutions are unobtainable, rather than forcing a costly re-computation of the optimal solution.

We are not as much interested in obtaining a theoretically optimal solution, as in providing a method that is both simple and robust to failure during execution. If a robot finds an unexpected obstacle on its way, or experiences any other delay, or is disabled, or loses communication we want to ensure the system continues to operate and tasks get accomplished.

II. The proposed method

Our method uses auctions to distribute tasks among a group of robots. The tasks are modeled as locations on a map that the robots have to move to within a fixed period of time. The robots start at different points in the environment, and have to move to their assigned task locations. We assume the robots can communicate with each other, for the purpose of notifying potential bidders about the auctioned tasks and for providing bids to the auctioneer.

At the start, one of the robots in the group is given the entire set of tasks to accomplish (perhaps by its human supervisor). This robot then redistributes the tasks among the other robots available. This redistribution is accomplished by means of a first-price reverse auction, where one task at a time is auctioned. The robot with all the tasks acts as the auctioneer. The auction is a reverse auction, since the auctioneer is the buyer.

Tasks are ordered by priority, which means the first few tasks receive more attention; later tasks may be abandoned in favor of accomplishing earlier ones. The tasks are auctioned one at a time, starting with the highest priority task. As each task is auctioned, the auctioneer selects the best bid and assigns the task to the corresponding bidder. Tasks with no bids are discarded, since the lack of bids implies they are not reachable. There is no guarantee that each task can be accomplished; some tasks may be in locations that are inaccessible to the robots.

The robots are given the environment layout. They have to generate paths through the environment and estimate the distance to each task position according to the path generated. They then use their estimate of the distance as their path cost (and hence the value of their bid).

Each robot computes the distance to all the tasks that are up for auction, considering each task in isolation. This simplifies the computations. When a robot wins a bid, it has to recompute its costs for all the remaining tasks, to account for the fact that its position will be different due to the assigned task.

Since the environment is complex and dynamic, obstacles may appear that were not originally in the map. Therefore, there is a chance that a robot may fail to achieve its tasks or take longer than expected. To deal with the situations where robots fail to achieve their assigned tasks, we allow for re-auctioning tasks when certain conditions are met. The auction is done as follows.

Whenever a robot completes a task, all the tasks that are still being executed by this or other robots, or are still pending, are re-auctioned. This includes tasks that were not allocated in previous rounds (thus new tasks can be assigned from external sources after the robots have begun execution of their current tasks). Until the re-auctioned tasks are assigned, the other robots continue to do their tasks. Upon reassignment, if a robot has received a higher priority task than its current task, it postpones execution of the current task to complete the higher priority one first.

When all the achievable tasks (determined by whether at least one robot was able to find a path to that task) are completed, the robots idle until the remainder of the time given to them is over. New tasks can be assigned to any robot anytime during the entire process, and the robot would simply include those tasks in its allocation process.

The task reallocation allows for tasks that were difficult for one robot to achieve (and hence took longer) to be reassigned to others. In addition when a robot fails (gets stalled or completely lost), it can put its tasks up for auction again, for other robots to try and accomplish. This reallocation is especially useful when the environment is highly dynamic or the robots are prone to failures.

We show in Figure 1 an example of how our method of auction with re-allocation of tasks differs from a single-task auction and from a combinatorial auction. We show how the methods would behave in an environment with 3 tasks, T_1, T_2 and T_3, and 2 robots $R_1$and R_2. The combinatorial auction (shown with dash-dot arrows) would examine the bids for every possible combination of the tasks, and find the optimal solution. This solution is to send R_1 to do tasks T_1 and T_2, and to send R_2 to do task T_3. The single-item auction (dashed arrows) forces R_2 to do both T_2 and T_3. This is because R_2 starts closer to T_2 initially, even though R_1 could accomplish T_2 more easily, after completing T_1. Our auction with re-allocation method (solid arrows) first does the same as the single-item auction, but when R_1 reaches T_1, it redirects R_2 to T_3, due to rebidding on the remaining tasks.

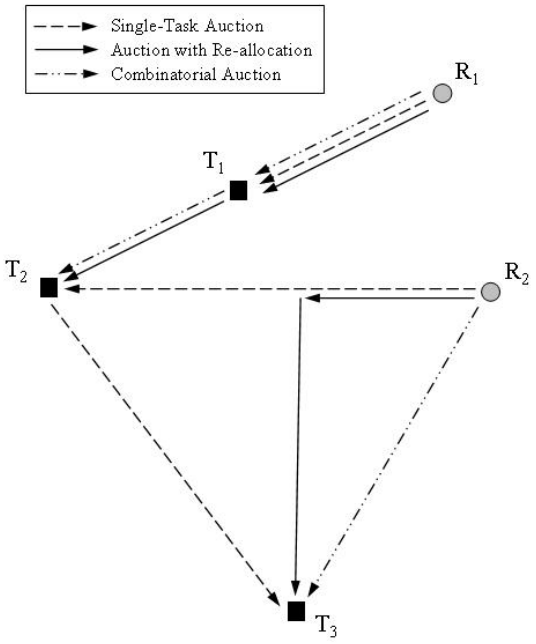

Fig. 1. Task allocation using a single-task auction, an auction with re-allocation of tasks, and a combinatorial auction. Allowing re-allocation of tasks gives considerable improvement in performance over the single-task auction, but does not always yield an optimal allocation.

During task execution, the robots follow paths generated by the method outlined below. To generate paths efficiently in a complex environment, we use Rapidly-expanding Random Trees (RRTs) [10]. RRTs are used for path planning when good area coverage is required. They are appropriate for environments where it is desired to reach every region, without having costly extra computations for inaccessible areas. Generation of RRTs is very fast, and scales well with large environments. An example of a RRT is shown in Figure 2.

The RRT algorithm works as follows. We start with a root node, which in our case is the initial position of the robot. The RRT is expanded outwards in all directions from the root node. To do this, a point P is chosen at random in the environment, and an attempt is made to link P to points already in the RRT (which initially is just the root node). If there is a line l joining P to an RRT point which does not intersect any obstacles, then a point Q is generated on l at a fixed distance from the RRT point, and Q is added to the RRT. This is repeated till a certain number of nodes is in the RRT, or a certain predetermined

Fig. 2. The hospital environment. The top part of the figure shows the Stage simulation, with the locations of the tasks and of the robots. (The robots have their range sensor traces shown). The lower part of the figure shows the paths generated by the RRT algorithm. This is from the Mixed Task experiment, with 3 robots and 6 tasks.

amount of area coverage is achieved.

If the RRT node candidate points are chosen at random from a uniform distribution over the environment, we get a structure that grows outwards from the RRT root node very rapidly, and does not have a large concentration of points in any single area. With this method, we can efficiently cover a complex environment, which can have many doorways, small rooms and obstacles.

The RRT generated can then be used to find paths, by simply tracing backward from a node that has line-of-sight access to the task position, till the robot position (which is used as the root node for the RRT).

A drawback of RRTs is that the path generated is not necessarily the shortest possible path. Since the RRT nodes in the path were generated randomly, the path often wanders all over the place before reaching its destination. Therefore, after each path is generated, the robots optimize it. The path is optimized by finding shortcuts between RRT nodes that lie on the path, and updating the path with the shortcuts obtained. Thus, any direct routes existing between non-adjacent points in the path is used to replace the original route between the points in the path.

We present next experimental results and analyze the performance of the auction with re-allocation method.

III. Experimental results

We tested our algorithm using Player/Stage [11] as the simulation environment, using a section of the hospital world.

The hospital world section, shown in Figure 2, is a large environment with many rooms, and is sufficiently complex to provide a good test for the algorithm. Each grid square in the figure is $1\mathrm{m}^2$, so the entire world measures approximately $33 \times 14\mathrm{m}^2$.

We modeled each robot as a small mobile device equipped with range sensors (sonar) and differential drives. Each robot has 5 sensors, mounted at 45° angles across the front of the robot. Tasks were modeled as beacons placed at different positions in the environment.

We used different experimental setups, each with a different number of tasks and robots, placed in different initial locations in the world. We did two sets of experiments, one (Experiment Set I) where the only moving obstacles were robots, the other (Experiment Set II) where additional moving obstacles were present.

Each experiment was run for 10 minutes and repeated 10 times with the same initial starting conditions. The results were averaged across experiments. Because of the random nature of the RRTs generated, and because of unexpected situations, the outcome of runs for each setup varied significantly both in the allocation of tasks to robots and in the time taken to accomplish the tasks. This allowed us to test what happens when the robots cannot accomplish all the tasks in the allocated amount of time, and provided a way of avoiding very long runs in hard cases when the robots did not make much progress.

In some runs the RRT nodes ended up being very close to walls or corners. This made the navigation of the robot harder. The robots could do only coarse-resolution control of their speed and turn rate, so they often got stuck on corners, and spent considerable time trying to free themselves. We could have avoided most of these problems by forcing nodes to be farther away from the walls, but we decided to use this to see the effects of real world uncertainty on the robots.

A. Experiment Set I

The experiment setups for the first set of experiments are summarized in Table I. The first three setups have tasks in different places with the three robots starting at the same initial positions. The robots ignored tasks that were unreachable, but performed well in achieving the tasks that were reachable (see Table I). Despite the fact that the set with one unachievable task had harder to reach tasks, the robots were able to devote more attention to the remaining tasks, and completed their task assignments much faster than in the other two scenarios.

Figure 3 shows a breakup of the task completion times for the Easy Task set. Each box shows the distribution of the time taken to accomplish that task, over 10 runs of the experiment.

When we increased the number of tasks, so that there were many more tasks than robots, the robots became busier. The bidding process slowed as the number of tasks increased, but not to an unmanageable extent. The robots also tended to run out of time before they could finish all the tasks given. The algorithm scaled up to 20 robots with 30 tasks without noticeable slowdown in the achievement of tasks.

On the contrary, when we increased the number of robots to be much larger than the number of tasks. the robots achieved their tasks much faster. Each robot was generally assigned only one task, and some were never allocated any tasks.

A side-effect of having many robots was that the robots tended to get in each other's way more often. The sonar sensors could detect collisions only when the robots were approached from certain angles. This meant they often did not detect each other until collision had already occurred (since the robots were small, they were often not detectable on the sonar).

Table I shows a comparison of average completion times over ten runs for all the tasks with the different environment configurations. The experiment set with many robots and fewer tasks had a very low completion time, while the one with many tasks and few robots showed a much higher completion time than the more mixed distributions. The table also shows the percentage of tasks completed for the different environments.

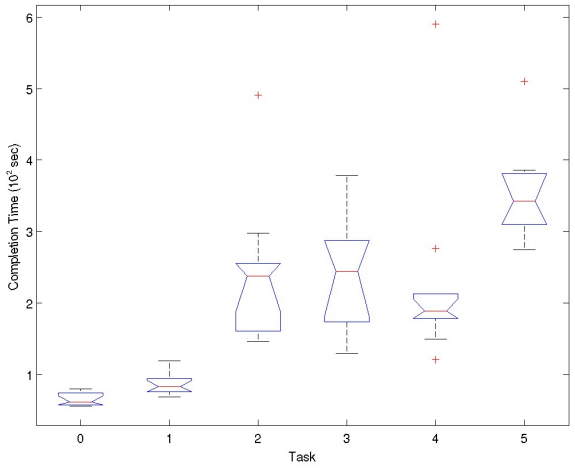

Fig. 3. Time for task completion for 6 tasks in the Easy Task experiment. For each task the figure shows the time it took to complete it. Times are plotted using a box representation where the center line is the median value, the top and bottom lines of the box represent the upper and lower quartile values respectively, and the lines at the top and bottom of each plot represent the rest of the distribution. The notches in the box represent an estimate of the distribution's mean. Outliers are marked with $+$.

B. Experiment Set II

To verify the performance of our algorithm with dynamic objects, we did a set of experiments with additional moving obstacles. The obstacles are modeled as thin rectangular sliding doors placed in the long corridor at the top of the hospital section. The obstacles move along the corridor to restrict access to the rooms there. There are two obstacles to each side of the corridor, for a total of four obstacles.

The intent was to measure deterioration in performance due to the introduction of moving obstacles, so obstacles were added until a significant deterioration in performance was seen. We performed experiments in the Easy, Mixed, and More

TABLE I

EXPERIMENT SET I

Experiment Type	Number of robots	Number of tasks	Task Completion Times Mean	Task Completion Times Std Dev	Task Completion Rates Mean	Task Completion Rates Std Dev
Easy Task Distribution	3	6	228.73	52.55	91.67	14.16
Mixed Task Distribution	3	6	229.70	55.56	90.00	16.10
More Tasks than Robots	3	16	412.13	28.28	58.75	7.90
Unachievable Task Dist.	3	6	310.15	42.90	73.33	11.65
More Robots than Tasks	10	6	89.03	28.32	98.33	5.27
Large Task/Robot Set	10	16	239.46	39.04	76.25	10.94
Huge Task/Robot Set	20	30	280.24	22.80	65.00	5.27

TABLE II

EXPERIMENT SET II

Experiment Type	Number of robots	Number of tasks	Obstacles	Optimal Allocation Task Compl. Time Mean	Optimal Allocation Task Compl. Time Std Dev	Optimal Allocation Task Compl. Rate Mean	Optimal Allocation Task Compl. Rate Std Dev	Auction with Reallocation Task Compl. Time Mean	Auction with Reallocation Task Compl. Time Std Dev	Auction with Reallocation Task Compl. Rate Mean	Auction with Reallocation Task Compl. Rate Std Dev
Easy Task Distribution	3	6	None	170.66	50.09	93.33	11.65	177.45	59.82	91.67	14.16
			Slow	276.92	96.02	73.33	22.50	240.46	54.74	85.00	12.30
			Fast	193.27	61.20	90.00	14.05	229.83	84.61	85.00	24.15
Mixed Task Distribution	3	6	None	174.99	68.30	91.67	14.16	185.59	63.68	90.00	16.10
			Slow	276.92	67.33	81.67	12.30	225.55	46.96	83.33	11.11
			Fast	193.27	80.10	80.00	15.32	258.46	65.34	85.00	18.34
More Tasks than Robots	3	16	None	342.88	47.57	55.00	10.12	334.11	46.06	58.75	7.91
			Slow	409.92	68.59	41.25	14.19	364.44	56.05	52.50	12.57
			Fast	437.97	39.70	35.62	8.86	373.00	85.05	51.25	16.61

Tasks than Robots environments. The results are summarized in Table II.

Three cases were examined: (1) there are no extra obstacles other than the robots, (2) the extra obstacles move slowly relative to the robots (at about 1/5th the robot speed), and (3) the obstacles move at about the same speed as the robots.

We compared the performance of our method against a method that generates the optimal allocation at the beginning of the run and never changes it. For the Easy and Mixed environments, the optimal allocation of tasks to robots was computed geometrically. For the More Tasks than Robots environment the optimal allocation was approximated by running multiple single round auctions and selecting the task allocation that occured most often.

In the optimal allocation, robots are programmed to wait till the obstacle, if any, moves out of the way. In our method when a task cannot be achieved it is put up for bids. This implies that with fast moving obstacles, the optimal allocation should perform better than the rebidding method, simply because waiting for a short time is faster that assigning the task to a different robot.

The results show that the optimal algorithm performs slightly faster in the default situation. When obstacles block the way however, our algorithm performs better. What is interesting is the change in performance when there are many tasks, as seen in Table II, supporting our assertion that adapting to the changing environment helps improve performance. The number of tasks completed also shows a clear increase when the robots follow our algorithm.

Our algorithm perfoms clearly better when stopping and waiting for obstacles to move is no longer a rewarding strategy. The robots tend to run out of time when doing so - and this applies to the slow obstacles and the many tasks scenarios. When the obstacles move fast, the optimal allocation gains by being able to wait for the obstacles to move out of the way - our algorithm wastes time rebidding and exchanging tasks around. This could be prevented by adjusting a time-out after which only rebidding occurs, or by adding some form of obstacle tracking to find the speed of the obstacle and decide how long to wait for completing the targeted task.

IV. RELATED WORK

A recent survey [12] covers in detail the state of the art in using auctions to coordinate robots while accomplishing different tasks such as exploration [3], navigation to different locations [5], or box pushing [8].

The problem we studied is a subset of the larger problem of coordination in a team. Our robots have to coordinate so that all the locations of a given set are reached by a robot. A single robot could in theory reach the entire set of locations to be visited, assuming all the locations are accessible to it, but having a team should increase efficiency and make the system more robust to robot failures.

Combinatorial auctions have been tried as a method to allocate navigation tasks to robots [13] but are too slow to be practical and do not scale well. They allow tasks to be accomplished with maximum efficiency, but the time taken in determining whom to assign which tasks often ends up being more than the time for the tasks themselves. Single item

auctions miss opportunities for optimal allocation, even though they can be computed in polynomial time.

Box-pushing is a challenging task which requires the robots to coordinate their behavior while moving boxes into demarcated regions. The method described in [8] uses a single-task auction process. Task completion is dependent on robots with the required capabilities being available. Using a single-task auction resulted in resources getting assigned sub-optimally, so that the robots could not complete some tasks that would have been achievable with an optimal allocation.

Auctions for exploration tasks [3], [14] have multiple robots placed in an unknown environment that is to be mapped. The robots generate tasks themselves to decide what to explore next, and auction the tasks to other robots in a single-item auction. Exploring a region leads to a reward for the robot, though the cost of resources used by the robot is deducted. The robots try to maximize their own gain. Thus, failure of a robot in the system is not critical since other robots would move to explore the areas missed. The system however needs a centralized agent to handle the rewards to be provided to the robots.

Recent work [5], [9] has focused on producing bidding rules for robot navigation tasks that lower computational costs while providing a close to optimal solution. The method they propose for generating bids has low cost, but requires all the robots to communicate with each other, so that each robot can compute its bid. The method uses multi-round auctions, where each robot bids in each round on the task for which its bid is the lowest. The overall lowest bid on any task is accepted, and the next round of the auction starts for the remaining tasks. Once all the tasks have been allocated, each robot plans its path to visit all the sites for the tasks it won. The bidding rules are such that there is no need for a central controller, as long as each robot receives all the bids from all the robots, each robot can determine the winner of the auction.

Our approach is designed for environments that are dynamic and where failures are likely. Tasks are re-bid whenever a robot accomplishes a task. This allows for re-assessing the current situation and might result in reallocation of tasks to robots. The advantage this provides is in allowing robots to re-allocate tasks when tasks bring them to more advantageous (or difficult) positions.

We have extended the method presented here to deal with unordered tasks [15]. Task ordering forces a constraint on which task gets allocated when. Unordered tasks give robots more freedom in deciding the order in which to accomplish them.

V. Conclusions and Future Work

We have presented an algorithm for multiple robots to bid on tasks. The algorithm is specially suited to dynamic environments, where unexpected obstacles might prevent a robot from achieving its tasks. The algorithm requires a robot to rebid its tasks as soon as it has achieved one of its task, so providing ways of adapting to changing conditions.

Future work includes improving the robot navigation to use more feedback from the sensors, and to use the sonars for some amount of environment remapping instead of relying on it solely for obstacle avoidance. At present, the sonar sensors allow the robot to avoid colliding into straight walls, but the robots tend to get stuck on corners where they cannot detect the corner before collision.

Adding methods for robots to decide whether or not to bid on tasks could improve performance. At present, robots only take into account their estimated distance from the task(s) while bidding. This should be extended to accounting for other factors such as whether they are stuck, how long they have been trying to complete the current task, and whether it is worthwhile after a substantial time investment to give up the task.

Acknowledgment

Work supported in part by the National Science Foundation under grants EIA-0324864 and IIS-0414466.

References

[1] L. E. Sutherland, "A futures market in computer time," *Comm. of the ACM*, vol. 11, no. 6, pp. 449–451, 1968.

[2] R. G. Smith, "The Contract Net protocol: High level communication and control in a distributed problem solver," *IEEE Trans. Computers*, vol. 29, no. 12, pp. 1104–1113, Dec. 1980.

[3] M. B. Dias and A. Stentz, "A free market architecture for distributed control of a multirobot system," in *Proc. of the Int'l Conf. on Intelligent Autonomous Systems*, Venice, Italy, July 2000, pp. 115–122.

[4] B. Gerkey and M. J. Mataric, "Multi-robot task allocation: Analyzing the complexity and optimality of key architectures," in *Proc. Int'l Conf. on Robotics and Automation*, Sept. 2003.

[5] C. Tovey, M. Lagoudakis, S. Jain, and S. Koenig, "The generation of bidding rules for auction-based robot coordination," in *Multi-Robot Systems Workshop*, Mar. 2005.

[6] S. Chien, A. Barrett, T. Estlin, and G. Rabideau, "A comparison of coordinated planning methods for cooperating rovers," in *Proc. of the Int'l Conf. on Autonomous Agents*. ACM Press, 2000, pp. 100–101.

[7] W. Agassounon and A. Martinoli, "Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems," in *Proc. of the First Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, Bologna, Italy, July 2002, pp. 1090–1097.

[8] B. P. Gerkey and M. J. Matarić, "Sold!: Auction methods for multi-robot coordination," *IEEE Trans. on Robotics and Automation*, vol. 18, no. 5, Oct. 2002.

[9] M. G. Lagoudakis, E. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, S. Koenig, C. Tovey, A. Meyerson, and S. Jain, "Auction-based multi-robot routing," in *Robotics: Science and Systems*, Cambridge, USA, June 2005. [Online]. Available: http://www.roboticsproceedings.org/rss01/p45.html

[10] J. J. Kuffner and S. M. LaValle, "Rrt-connect: An efficient approach to single-query path planning," in *Proc. Int'l Conf. on Robotics and Automation*, 2000, pp. 995–1001.

[11] B. P. Gerkey, R. T. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," in *Proc Int'l Conf on Advanced Robotics*, June 2003, pp. 317–323.

[12] M. B. Dias, R. M. Zlot, N. Kalra, and A. T. Stentz, "Market-based multirobot coordination: A survey and analysis," Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-05-13, April 2005.

[13] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt, "Robot exploration with combinatorial auctions," in *Proc. Int'l Conf. on Robotics and Automation*, 2003.

[14] R. M. Zlot, A. T. Stentz, M. B. Dias, and S. Thayer, "Multi-robot exploration controlled by a market economy," in *Proc. Int'l Conf. on Robotics and Automation*, May 2002.

[15] M. Nanjanath and M. Gini, "Auctions for task allocation to robots," in *Proc. of the Int'l Conf. on Intelligent Autonomous Systems*, 2006.