Technical Report

Department of Computer Science
and Engineering
University of Minnesota
4-192 EECS Building
200 Union Street SE
Minneapolis, MN 55455-0159 USA

TR 08-032

Repeated Auctions for Robust Task Execution by a Robot Team

Maitreyi Nanjanath and Maria Gini

October 20, 2008

Repeated auctions for robust task execution by a robot team

Maitreyi Nanjanath and Maria Gini
Department of Computer Science and Engineering, University of Minnesota
`nanjan@cs.umn.edu, gini@cs.umn.edu`

October 20, 2008

Abstract

We present empirical results of an auction-based algorithm for dynamic allocation of tasks to robots. The results have been obtained both in simulation and using real robots. A distinctive feature of our algorithm is its robustness to uncertainties and to robot malfunctions that happen during task execution, when unexpected obstacles, loss of communication, and other delays may prevent a robot from completing its allocated tasks. Therefore tasks not yet achieved are resubmitted for bids every time a task has been completed. This provides an opportunity to improve the allocation of the remaining tasks, enabling the robots to recover from failures and reducing the overall time for task completion.

1 Introduction

An autonomous team of robots may be deployed in a situation that is dangerous or inaccessible to humans, such as a building collapsed during an earthquake. The robot team can be used to map the building, identifying unsafe areas, and to locate and rescue survivors. Robots in the team will have different tasks. The tasks could be assigned to each robot before deployment, but this would reduce the team's ability to adapt to the situation. Thus it is preferable to have the robots determine the task assignments dynamically through negotiations within the team.

In this paper we propose a method for distributing tasks dynamically among a group of cooperating robots. We are interested in situations where each task can be done by a single robot, but sharing tasks will reduce the time to complete the tasks and thus has the potential to increase the efficiency of the robot team.

What makes task allocation to robots challenging is the fact that robots have to physically move to reach the locations of their assigned tasks, hence the cost of accomplishing a task depends not only on the location of the task itself but also on the current location of the robot.

In this paper we present empirical results obtained both in simulation and with real robots using the algorithm we originally presented in [16]. The algorithm, which is based on auctions, does not guarantee an optimal allocation, but is specially suited to dynamic environments, where execution time might deviate significantly from estimates, and where the ability to adapt dynamically to changing conditions is crucial.

The algorithm is totally distributed. There is no central controller and no central auctioneer, each robot auctions its own tasks and clears its own auctions. The only assumption we make is that robots can communicate with each other.

The auction mechanism we propose is based on a combination of *parallel single-item auctions* [4] and *sequential single-item auctions* [15, 13]. It attempts to minimize the total time to complete all the tasks and, at the same time, the total path length for all the robots. It tries to minimize the total completion time by minimizing the length of the longest path, and to minimize the total path length for all the robots by assigning each task to the nearest robot. With the simplifying assumption of constant and equal speed of travel for all the robots, the first objective, i.e. minimize the total time, is equivalent to minimizing the maximum path cost over all the robots (called miniMAX objective in [18]). The second objective, i.e. minimize the total path length, is equivalent to minimizing the sum of path costs over all the robots (called miniSUM objective in [18]).

The algorithm we present is simple but robust to failures during execution. If a robot finds an unexpected obstacle, experiences any other delay, loses communication, or is otherwise disabled, the rest of the team continues to operate.

In this paper we describe the algorithm, analyze its complexity, and we report empirical results obtained both in simulation and with real robots in a variety of environments.

2 Related Work

A recent survey [7] covers in detail the state of the art in using auctions to coordinate robots for tasks such as exploration [5, 12], navigation to different locations [18], and box pushing [9]. Auction-based methods for allocation of tasks are becoming popular in robotics [10, 18] as an alternative to other allocation methods, such as centralized scheduling [3], blackboard systems [8], or application-specific methods, which do not easily generalize [1] to other domains.

Combinatorial auctions, where combinations of tasks are bid at once, have been tried as a method to allocate navigation tasks to robots [2] but generating bids and clearing them is slow because of the computational complexity of combinatorial auctions, and they do not scale well.

Sequential single-item auctions [13, 15, 18] can instead be computed in polynomial time and produce solutions that, when the objective is to minimize the sum of the path costs for all the robots, are a constant factor away from the optimum [15]. 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 each auction. However, this requires each robot to keep track of its own costs and of the other robot costs, and so it is not robust to robot malfunctions. Robots are expected to know the exact cost of completing each task at the start. It is unclear how changes to this cost caused by unexpected changes can be handled.

Repeated parallel single-item auctions [4] are fast to compute and more robust. They make use of a pulse that is sent out at fixed time intervals to all the robots to restart the single-item auction between robots. This enables robots to switch tasks if the allocation can be improved and helps in case of unexpected problems, but has the undesirable effect that the length of the entire path covered by the team might be unbounded [18].

We also use single-item auctions, and we repeat the auctions multiple times while the tasks are being executed. However, instead of repeating the auctions at regular intervals, we repeat them whenever a task has been completed. This reduces the need for communication and the time spent in clearing auctions. while still providing the ability to react to changes in the environment or in robot functioning, typically without degrading performance. In addition, we account for tasks won from an auctioneer before bidding on subsequent tasks from the same auctioneer. This reduces the chance of an oscillatory situation where tasks keep getting transferred back and forth between two robots, a problem that affects the total path length in repeated parallel single-item auctions. We discuss later in Section 4 the complexity of our algorithm.

Our approach is similar to the method presented in [6] where a group of robots is given a set of tasks and robots are selectively disabled in different manners in order to measure their performance, i.e. the percentage of tasks completed, under different conditions. Our approach differs in that we assume a time limit for task completion. Additionally we use robots that are simpler and more prone to errors, hence the ability to change task allocation is critical.

Our approach aims at finding a tradeoff between computational complexity, quality of allocations, and ability to adapt.

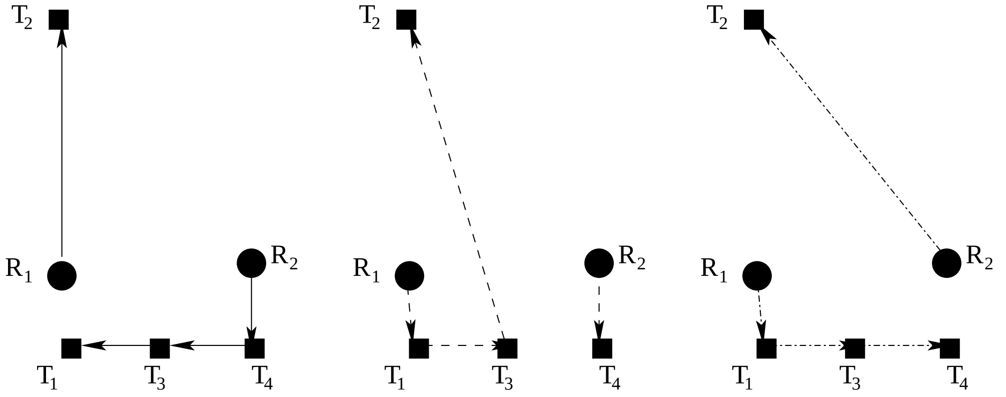

Figure 1: Task allocation using a combinatorial auction, a parallel single-item auction, and a sequential single-item auction.

In Figure 1 we show an example of how a combinatorial auction, a parallel single-item, and a sequential single-item auction differ from each other. The figure shows how the methods would behave in an environment with 4 tasks, T_1, T_2 T_3 and T_4, and 2 robots, $R_1$and R_2. Since some auction methods are sensitive to task order, we assume in what follows that tasks are bid in the order T_1, T_2 T_3, and T_4.

The combinatorial auction (shown in Figure 1 with solid arrows) examines the bids for every possible combination of the tasks, and finds the optimal solution, which is to allocate task T_2 to R_1, and to send R_2 to do tasks T_4, T_3 and T_1, in that order. The parallel single-item auction (shown with dashed arrows) makes R_1 to do T_1, T_3 and T_2 (assuming some path optimization is done), while R_2 does only T_4. This is because R_1 starts closer to the three tasks, even though R_2 could accomplish T_3 and T_1 more easily, after completing T_4. The ratio of path costs of the single-item auction compared to the combinatorial auction is 1.155 : 1. The third diagram shows how the sequential single-item auction (shown with dot-dash arrows) would work - it achieves a better solution than the parallel single-item auction, but it is sub-optimal compared to the combinatorial auction solution. The ratio of path costs in this case is 1.079 : 1.

Our algorithm combines both the sequential single-item auction and parallel single-item auction, by running several sequential auctions in parallel. When tasks are initially put up for auction from a single source, it starts like the sequential single-item auction, but the remaining tasks are auctioned again after each task is completed. Therefore the algorithm will behave differently if an obstacle between two of the tasks is found after motion has begun; in this situation the tasks are re-allocated accounting for the modified path for that robot, thus allowing for recovery from blockades. If tasks are initially distributed randomly between the robots, then the tasks of each robot are sold in parallel, which may lead to a worse allocation, but this gets corrected once the first task (which is necessarily the nearest) is completed. Our algorithm is also geared to address failure on the part of one or more of the robots; the other robots will take over those tasks and finish them themselves.

3 Auction Algorithm

In this work we assume that each robot is given a map that shows its own location and the position of walls and rooms in the environment. No information is given about where the other robots are located and about other moving objects present in the environment, or about any temporary change, such as closed doors. The map is used by each robot to estimate, using Rapidly-exploring Random Trees [14], its cost of traveling to the task locations and to compute the path to reach them from its original location. Generation of RRTs is very fast, and scales well with large environments, so they are particularly appropriate for dynamic situations where computing the optimal path to achieve all the tasks allocated to a robot, as in [15], might not pay off, because tasks are likely to be reallocated. Examples of RRTs for our experimental setups are shown later in Figure 3 and Figure 4.

Each robot is also given a list of all the robots in the team. We assume the robots can communicate with each other for the purpose of notifying potential bidders about auctioned tasks, for submitting their own bids, and for receiving notification when they won a bid. However, robots do not know all the tasks, they are aware only of the ones they have been assigned and discover the other tasks when they are auctioned.

Let R be the set of n robots $R = \{r_1, r_2, ...r_n\}$, and T the set of m tasks $T = \{t_1, t_2, ...t_m\}$, where each task is a location a robot has to visit. We partition the tasks into n disjoint subsets T_j, such that $\cup_{j=1}^{n} T_j = T$ and $T_i \cap T_j = \phi \;\; \forall i \neq j \;\; 1 \leq i, j \leq n$, and allocate each subset to a robot. Note that a subset can be empty.

The initial task distribution might not be optimal. For instance, some robots might have no task at all while others might have too many tasks, the tasks assigned to a robot might be spread all over the environment, might be closer to another robot, or tasks may be unreachable by the robot.

A robot must complete all its tasks unless it can pass its commitments to other robots. Since the robots are cooperative and are trying to minimize task completion time, they will pass their commitments only if this reduces the estimated task completion time. A robot can choose not to bid on a particular task, based on its distance and accessibility to that task. The ability to pass tasks to other robots is specially useful when robots become disabled since it allows the group as a whole to increase the chances of completing all the tasks. Any task that cannot be completed by any of the robots, for instance because it is not accessible, is abandoned. We assume that there is value in accomplishing the remaining tasks even if not all of them can be completed.

This process is accomplished via multiple single-item reverse auctions, in which the lowest bid wins. Auctions are run independently by each robot for its own tasks. The algorithm that each robot follows is outlined in Figure 2.

Each bid submitted by a robot is an estimate of the time it would take for that robot to reach that task location (assuming for simplicity a constant speed) from its current location.

```
Repeat for each robot $r_i \in R$:
  1. Activate $r_i$ with a set of tasks $T_i$ and a list of the other robots $R_{-i} = R$ - $\{r_i\}$.
  2. Create an RRT using $r_i$'s start position as root.
  3. Find paths in the RRT to each task location in $T_i$.
  4. Assign cost estimate $c_j$ to each task $t_j \in T_i$ based on length of the path found starting from the current position.
  5. Order task list $T_i$ by ascending order of $c_j$.
  6. Establish communications with the other robots and build a list of all the tasks (system task list) for reference.
  7. $r_i$ does in parallel:
    (a) Auction its tasks:
      i. Create a Request For Quotes (RFQ) with tasks in $T_i$.
      ii. Broadcast the RFQ to $R_{-i}$ and wait for bids for a fixed time limit.
      iii. Determine the lowest bid $b_{jk}$ among all the bids received for task $t_j$. Let $r_k$ be the robot that submitted the winning bid.
      iv. If $b_{jk} < c_j$ then send $t_j$ to robot $r_k$, else keep $t_j$. If $r_k$ does not acknowledge receipt, return $t_j$ to $r_i$. Mark $t_j$ as assigned.
      v. Ask $r_k$ to update its bids, if any, for the remaining tasks in $T_i$ ($r_k$ has now a new task). If $r_k$ does not acknowledge receipt of the message, return $t_j$ to $r_i$.
      vi. Repeat from Step 7(a)iii until all tasks are assigned.
    (b) Bid on RFQs received from other robots:
      i. Find a RRT path for each task $t_r$ in the RFQ.
      ii. Compute cost estimate $c_r$ for each $t_r$ to which the robot found a path, starting from its current position.
      iii. If a bid is won, recompute the bids for the remaining tasks in that RFQ, accounting for the tasks assigned from that RFQ and submit bids to the auctioning robot.
    (c) Begin execution of the assigned tasks:
      i. Find a path in the RRT to the first task and start following it as closely as possible.
      ii. If new tasks are added as a result of winning new auctions, insert them in $T_i$ keeping it sorted in expected execution order, from the nearest task to farther away ones, and repeat from Step 7(c)i.
      iii. If the robot is stuck, or could not complete its tasks within a set time limit, starts a new auction to reassign its tasks.
      iv. If $t_j$ is completed successfully, notify all robots of task completion, update the system task list, and restart from Step 4.
until time limit is reached or all tasks are completed.
```

Figure 2: Task allocation algorithm.

Auctions are parallel, i.e. many auctioneers put up their auctions at the same time, but since a bidder generates bids in each auction independently of the other auctions, the effect is the same as having each auction done as a single-item auction that the bidder either wins or loses. Since a robot can bid for tasks in multiple parallel auctions, the order in which tasks are to be executed might be different from the order in which bids for tasks are submitted and won. The robot cannot compute its bids according to the order of execution, since the order is unknown at the time of bidding. Therefore, the robot treats each auction in a round in isolation. It computes its bids for each parallel auction assuming it starts at its current location, taking into account tasks that were won in that auction, but ignoring tasks won in other auctions. This can results in bids that over- (or under-)estimate the true cost. However, because tasks can be reallocated, this does not impact significantly the quality of the solution. In each auction the bid for the closest task is a correct estimate of its cost. Once that task is achieved the robot starts another auction for its remaining tasks.

Each bidder re-orders its tasks each time a new task is added to its list, and then moves towards the nearest task (i.e. the task with the lowest cost) over its entire set of tasks. When a robot completes its current task, it starts an auction again for its remaining tasks in an attempt to improve the overall task allocation. This is specially useful if a robot gets delayed, because this redistribution of tasks enables it to change its commitments and to adapt more rapidly.

The robots are given a time limit to complete the tasks, so that they do not keep trying indefinitely. 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.

The algorithm allows for dynamic addition of new tasks during the execution, but for simplicity in the experiments described in Section 5 the set of tasks and of robots is known at start and does not change during the execution.

4 Auction Analysis

In analyzing the auction algorithm described in the previous Section we make the following assumptions: (1) all robots are working, (2) communications is perfect, (3) all tasks are accessible, and (4) all tasks are initially assigned to a single robot.

Formally, the problem is defined as follows: Given n robots and m tasks, the setup of the tasks can be represented as a graph G where tasks are a set of nodes T and paths between tasks are a set of undirected edges E. Each robot associates a cost with an edge according to the robot's own ability to navigate along that edge. The cost measure we use is path length. Since we assume constant and equal speed for all the robots, in the experiments we measure travel time to reach each task. A different function could be used for cost, such as, for instance, the power consumption.

As the auction algorithm proceeds, it assigns a subset of tasks T_j to each robot r_j, such that $T_j = \{t_j | t_j \text{ is assigned to } r_j \text{ and } t_j \in T\}$ and $\cup_{j=1}^{n} T_j = T$.

Each robot r_j needs to find a path to the task subset T_j assigned to it. This is equivalent to solving the traveling salesman problem for that robot. An approximation can be made using a greedy path algorithm that takes the shortest path to the nearest unvisited node. This has provable bounds, as follows. Build a Minimum Spanning Tree (MST) over T_j rooted at the node nearest to r_j. Let the sum of costs of edges in the MST be denoted by K_j. Then, the greedy path algorithm has a cost bound of $2 \times K_j + C_j$ where C_j is the cost for the robot to reach the root of the MST.

The overall team cost is then bounded by

$$C_{total} = \sum_{i=1}^{m} Ct_i + \sum_{j=1}^{n} (2 \times K_j + C_j)$$

where Ct_i is the cost to do task t_i. In this paper for simplicity we assume that all task costs Ct_i are 0.

The objective is to find an allocation S of tasks over T, that minimizes C_{total}, subject to the constraint $time_{total} \leq timelimit$. If multiple solutions are found with the same minimum C_{total}, the objective is to minimize $time_{total}$ over those solutions.

Method	Time	Sum Path Costs	Initial Comm.	Overall Comm.
Sequential Single Item Auctions [15]	$O(n \times m)$	$2 \times n \times d$	$n \times m$	N/A
Repeated parallel single item auction [6]	$O(n \times m)$	unbounded	$n \times m^2$	$n \times m \times t/i$
Our Algorithm	$O(n \times m^2)$	$(3 \times n - 2) \times d$	$n \times m$	$n \times m$

Table 1: Performance comparison between auction methods. n the number of robots, m the number of tasks, and d the total path cost for all the robots in the optimal solution.

In Table 1 we compare the computational complexity of our algorithm with the complexity of sequential single-item auctions and repeated parallel single-item auctions. We use i for the communication pulse interval, i.e. a signal broadcast to all the robots which triggers a new round of auctions [6]), and d for the total path cost for all the robots in the optimal solution, i.e., the one that minimizes the sum of path costs for all the robots. Since the initial task allocation in our algorithm matches that of a sequential single-item auction [15], we can use their complexity analysis results to our algorithm. Subsequent auctions can result in added path costs; these are accounted for in our complexity analysis.

4.1 Path Length

In our algorithm, items are sold in unit bundles and each robot accounts for tasks it already won from the current auctioneer before bidding further on new tasks from the same auctioner. To bid on a new task the robot computes the difference in the cost of the path that includes the new task from the previously computed path cost, and bids that difference. This is similar to the method of bidding described in [15] for the MiniMAX objective, but differs in the handling of multiple auctions.

The upper bound on the sum of path costs if the robots follow their initial allocation is $2 \times n \times d$ ([15]). Following the initial allocation, in subsequent auctions, tasks may either stay with the same robot or be reassigned. With the exception of a special case, reassignment is equivalent to having the initial auction with tasks in the reassigned order, and hence will only result in improvement. The special case occurs when two task allocations are nearly equivalent, and the robots keep switching between the two allocations in each auction. In this situation, since the number of auctions is limited by the number of remaining tasks, the maximum increase is $(n - 2) \times d$ (the time taken by the remaining robots to reach those tasks). Thus, the bound on the cost becomes $(3 \times n - 2) \times d$.

Our auction method avoids the trap of parallel single-item auctions, where robots may all travel a long distance to reach a cluster of close-together tasks, instead of having just one robot completing the tasks in that cluster [18]. This is achieved by making robots to account in further bidding from an auctioneer for tasks already won from that auctioneer. This ensures that if an auctioneer is auctioning tasks that are close to each other, the robot winning a task from that auctioneer will continue to win subsequent tasks from the same auctioneer that are near that task. Each time tasks are auctioned, a robot will win the tasks nearest to it. If nearby tasks were incorrectly given to a different robot previously, they will get reassigned to the closet robot, even when they are auctioned by different auctioneers, because the closest robot will bid its distance to that task, and that will be the smallest bid, hence the robot will win the bid. Over multiple auction rounds, this implies that tasks will tend to get assigned in groups to specific robots, based on their positions.

4.2 Communications

The robots in our algorithm have more communication needs than the robots in [15], since in our case communication continues after the initial allocation, whenever there is an auction. There are n messages per auction, one per robot, and m auctions (The initial auction + 1 auction per task completed, with the exception of the last task). Therefore, a total of $n \times m$ messages are sent.

Failure of communication before the start of execution is a problem because tasks may never get shared between robots, and some tasks may remain undone. However, if communication failure takes place later, then the working robots can handle the additional tasks, and the problem can be treated as a modified one where the number of robots has gone down to $n - k$, if k robots are out of commission. Given k possible breakdowns, we need extra rounds of auctions for the tasks of the failed robots, thus resulting in $n \times m + n \times k = n \times (m + k)$ communication messages.

The number of messages for our algorithm is considerably smaller than the number needed for repeated parallel auctions, where tasks are placed for bidding continuously, so that communications takes place all the time.

5 Experimental Setup

We evaluated our algorithm through experiments done both in simulation and with real robots. Due to space and equipment constraints, we were limited to two robots for the real robot experiments, but were able to perform different and more complex experiments in simulation.

Experiments were performed in the Player/Stage [11] simulator. Player/Stage has the advantage that implementation details do not change significantly when shifting from simulation to real robots, thus making comparison easier. The experiments performed in our robotics lab used two Pioneer I robots, each mounted with a laptop and equipped with wireless cards for communication with each other. Communication was done through Java Sockets, as they provide features nearest to what the simulated system had.

Additional simulation experiments done earlier have been reported in [16, 17]. Their purpose was to evaluate the effectiveness of our auction algorithm in comparison to using a single initial auction, to measure the impact of loss of communication and of changes in the environment, and to measure the robustness of the algorithm. The earlier experiments used robots with 5 sonar sensors and differential drives, scattered in the hospital world environment provided by Player/Stage.

In this paper we report results on experiments conducted in two scenarios, a lab scenario, where we performed experiments both with real robots and in simulation, and a building scenario, where we performed experiments only in simulation.

5.1 Adaptations for Real Robots

There were some non trivial differences we had to deal with between the simulation and the real robot experiments.

1. Player 2.0 has some significant difference in the way motion is dealt with in real robots in comparison to the simulation. The same command produced in simulation a differing range of motion than when given to a real robot. Thus, motion commands had to be reconfigured to suit the robots.
2. Data for ranges of goals, sonar ranges, and collision ranges had to be modified to suit the real robots, since the form factor of the real robots was considerably different from that of the simulation.
3. In the simulations, all obstacles were detectable through sonars. In the real robot experiments however, robots occasionally could not detect obstacles, such as table legs, because the sonar sensors were too far apart and missed the obstacle. This resulted in several collisions and near collisions in the real robot experiments, and produced far more variability in task completion times than what we had seen in the simulations. Details on the task completion times can be seen in Table 2 and Table 3.
4. Odometry in the real robots was significantly worse than that accounted for in the simulations. In most cases, unless there was a tight fit, the robots managed to complete all the tasks without collision. Tasks were considered to be complete when the robots arrived within 30 cm of the task (i.e. an approximate robot-length away from the task). Collisions were tolerated in simulation; in the real runs, robots that had collided with obstacles were given one chance to recover and then shut down, to avoid damage.

6 Experimental Results

The main purpose of the experiments was to evaluate the performance of different aspects of the auction algorithm, such as auction time and communication overhead during execution, and to validate the simulation results by comparing simulation with real robots results.

6.1 Lab scenario

For the real robot experiments, the robots were given a map of the lab which did not include chairs but included table positions, and also a description of the team, including the wireless ids of the other robots. The robots started at different locations, and were given their own approximate position in the map. The tasks were scattered randomly in the lab and were initially divided equally between the robots.

When a robot had completed all its assigned tasks, it would wait a fixed amount of time (usually the amount of time the other robot had provided as its lowest bid) waiting for another robot to start a new auction. If any task in the system task list maintained by the robot was still incomplete and no auction had been started, the robot would start a new auction for the incomplete tasks.

The two experimental setups in the lab are illustrated in Figure 3 and Figure 4. The figures also show the RRTs formed by each robot in one of the runs.

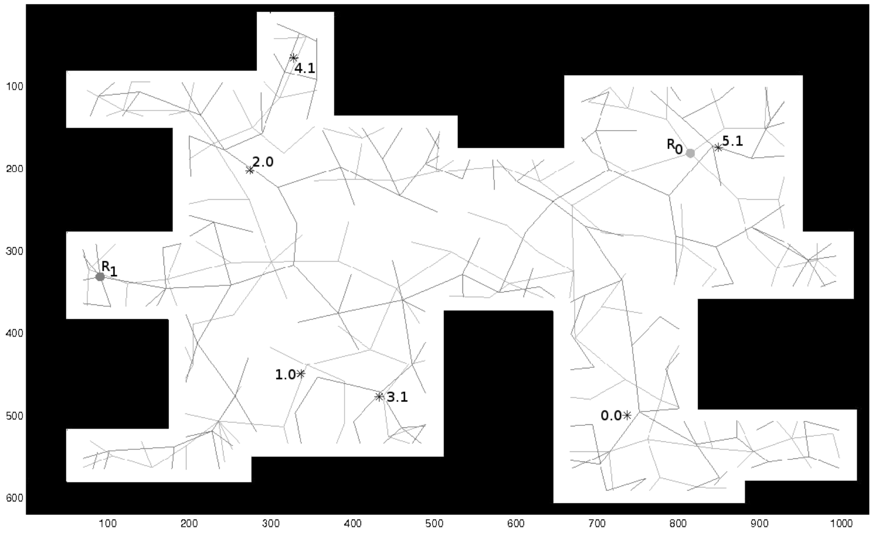

Figure 3: Experiment I map: robots are circles and tasks are asterisks. The RRTs for run 4 are shown.

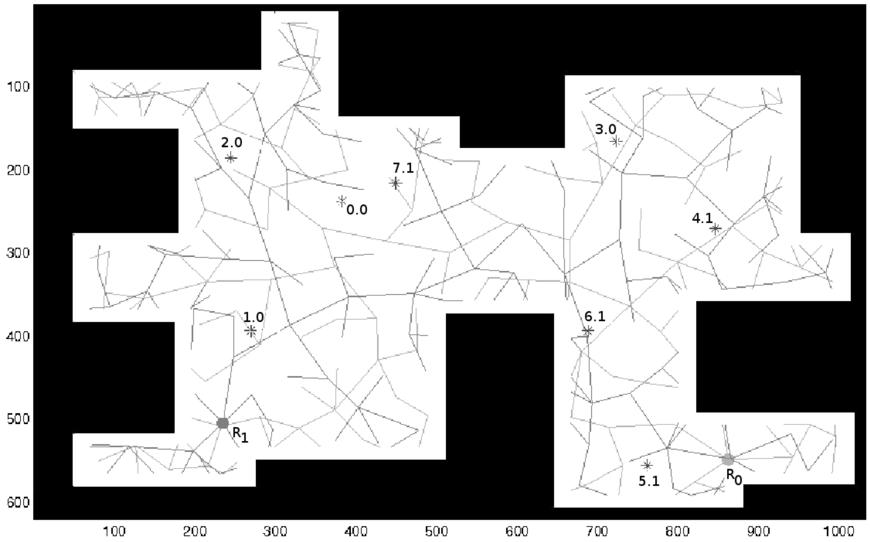

Figure 4: Experiment II map: robots are circles and tasks are asterisks. The RRTs for run 3 are shown.

In Experiment I there were six tasks scattered randomly in such a way that an optimal task allocation would result in an uneven distribution of the tasks between the robots. In Experiment II there were eight tasks distributed initially such that the majority of the tasks given to Robot 0 was closer to Robot 1 and vice versa. This was done to examine if the robots exchanged tasks successfully and completed them correctly.

We performed 5 runs of each experiment type individually both in simulation and with the real robots.

The performance of the real robots in each experiment is shown in Figure 5 and Figure 7. We can notice in the timeline that the allocation of tasks is not the same in the different runs, in particular in the experiments with the real robots. Since a task can be allocated to different robots at different times, to keep track of which robot does what task, we indicate tasks with a pair of numbers, the task number and the robot that task is assigned to. For instance, in run number 4 of Experiment I, shown in Figure 3, task 0 was auctioned first but due to the way the RRT curved, the estimated cost for task 5 by Robot 0 was very high (it added the cost of going to and returning from task 0 to its cost estimate). Robot 1 initially won task 2 because it had a lower cost estimate, but Robot 0 won it back after it completed task 0.

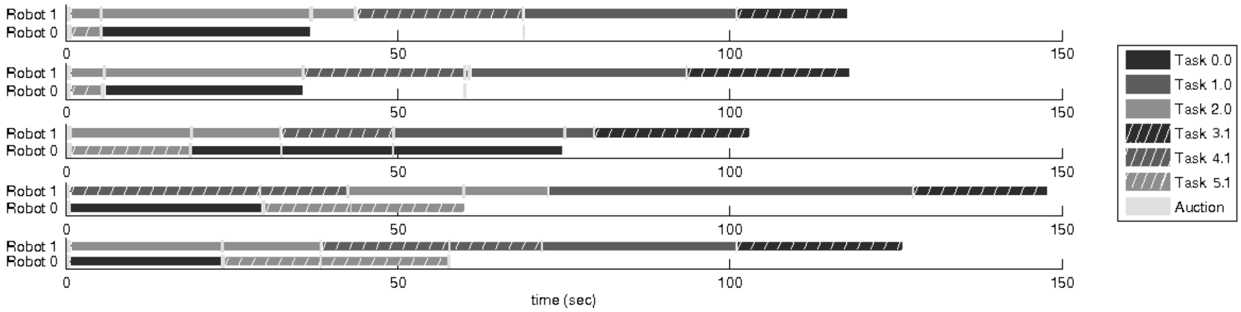

Figure 5: Experiment I real-robot timeline. Runs 1 through 5 (top to bottom). Task IDs show task number followed by robot number.

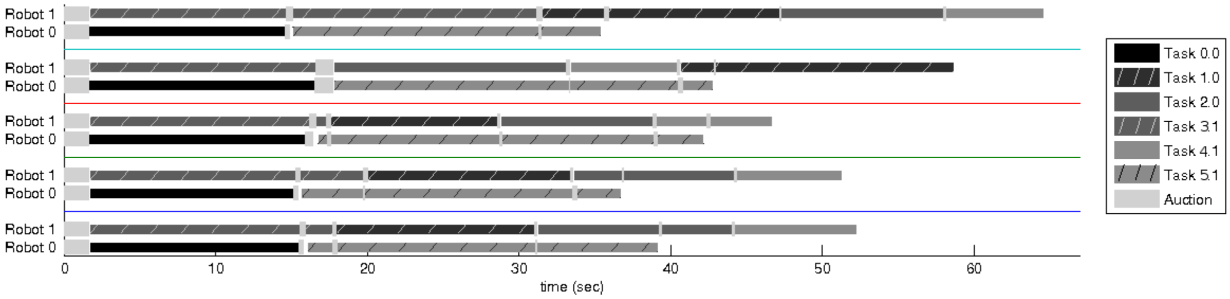

Figure 6: Experiment I simulation timeline. Runs 1 through 5 (top to bottom)

The task completion times for the lab scenario experiments are summarized in Table 2 and Table 3. In each case, the robots completed the assigned tasks within 2 minutes, staying well within the 10 minute time limit provided. Task completion times in the simulation were significantly shorter than the corresponding times in the real robot experiments, as shown in Table 2 and Table 3.

In run number 3 in Experiment II (Figure 4), Robot 0 initially got stuck trying to get to task 6, and then completed the remaining tasks, but was much slower than usual in completing the first two tasks, probably because of low battery.

The simulation experiments, whose timelines are shown in Figure 6 and Figure 8, in comparison did not show robots getting stuck as often. A significant difference was a long initial auction time in simulation as compared to the real robots. This was likely caused by the fact that the computers

Table 2: Task completion times (in seconds) for Experiment I.

Task	Robot	Real Robots		Simulation	
ID		Avg. time	σ	Avg. time	σ
0.0	0	33.478	12.78	13.796	0.75
1.0	1	35.443	10.82	14.180	2.67
2.0	1	35.018	5.12	11.828	2.21
3.1	1	21.707	3.56	18.755	6.06
4.1	1	28.041	9.48	7.135	0.62
5.1	0	17.872	12.28	22.910	2.27
Total		121.618	16.53	52.955	7.01

used in the simulation shared a network and hence took longer to initially establish connections than the robots which had a dedicated network. This resulted in initial auction times being on the order of 1.6 seconds in the first auction, dropping to 0.3 seconds subsequently. While the real robots also had a longer initial auction, such a large drop was not seen in the auction times.

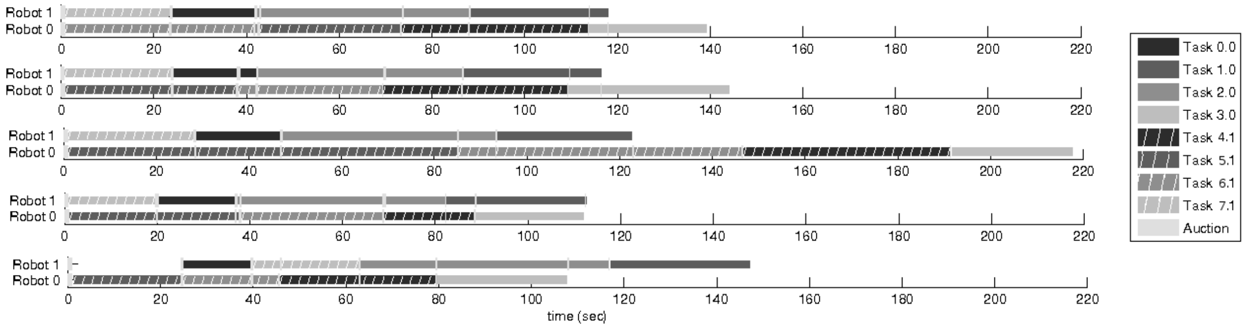

Figure 7: Experiment II real-robot timeline. Runs 1 through 5 (top to bottom)

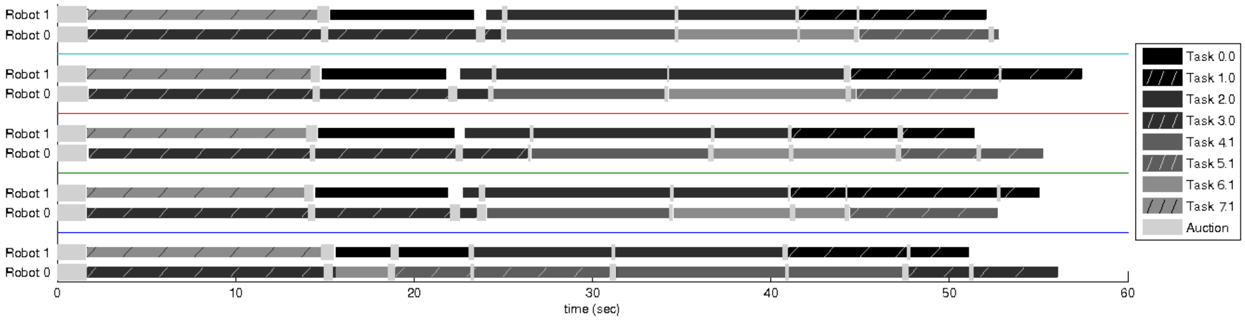

Figure 8: Experiment II simulation timeline. Runs 1 through 5 (top to bottom)

The auctions took a very small percentage of the total time (as shown by the light grey bands in the figures, and summarized in Table 4), and caused small delays between one task and the next. This accounted for less than 1% of the time spent in performing the tasks. Communications time

Table 3: Task completion times (in seconds) for Experiment II

Task ID	Robot	Real Robots		Simulation	
		Avg. time	σ	Avg. time	σ.
0.0	1	16.771	1.51	7.495	0.39
1.0	1	29.678	0.47	11.528	1.79
2.0	0	46.727	3.90	18.478	1.75
3.0	0	27.470	4.31	29.268	14.08
4.1	1	35.404	9.76	11.004	2.79
5.1	0	42.060	23.96	8.773	1.81
6.1	0	36.862	15.44	8.593	3.18
7.1	1	22.719	3.39	12.610	0.40
Total		151.185	39.04	53.642	1.72

was also a very small fraction of the time taken to complete tasks (on average, communications took up less than 1% of the work-time).

We can summarize the comparative performance between simulation and real robots as follows:

- Algorithm performance: In simulation, the task allocation found was identical to that found in the real robot experiments, thus the simulation results were acceptable. However, the impact of the time taken to perform the auctions was significantly less with the real robots compared to simulation.
- Time: the simulated robots moved faster than the real robots, despite the fact that we tried to find an equivalent velocity setting; thus, the auctions took a more significant portion of simulation time than they did in the real robot experiments. This speed difference also required modifications to the range parameter settings to get equivalent settings for the real robots as compared to simulation.
- Robot performance: The simulation was much more optimistic about the ability of the robots to detect obstacles and recover from errors; in the real robots, there was a tendency to get stuck that was not seen as frequently in simulation.

Table 4: Auction Times for Experiments I and II

Expt Num	Real Robots		Simulation	
	Mean (s)	σ	Mean (s)	σ
I	0.4052	0.1861	0.5527	0.5797
II	0.4322	0.2412	0.4865	0.4938

In conclusion, the simulation experiments were good indicators of real world performance, though some of the problems faced by actual robots were not perfectly mirrored in simulation.

6.2 Building scenario

We have also evaluated the auction algorithm in the environment described in [18], with 18 tasks and three robots (Figure 9). This environment is more complex than the lab environment, because there are numerous rooms and doors connecting them, so the navigation is harder. The major reason for choosing this environment is to enable comparison of results produced by different algorithms in the same environment. We used two different experimental setups. In Experiment III we used the same layout as the one used in [18]. In Experiment IV, we added moving obstacles in four locations that hinder robot movement. We performed 10 runs for each of these experiments.

The paths followed by the robots in one of the runs for Experiment III are shown in Figure 10. The path followed by the robot on the left shows squiggly lines where the RRT was following the wall too closely. The obstacle avoidance routines would force the robot away from the wall, but the path to be followed would bring it back to being close to the wall. This kind of movement was happening often due to the tendency of RRT nodes to be generated close to walls when in an environment with a lot of rooms. This did not cause a significant negative impact on the motion of the robot, when overall performance is considered.

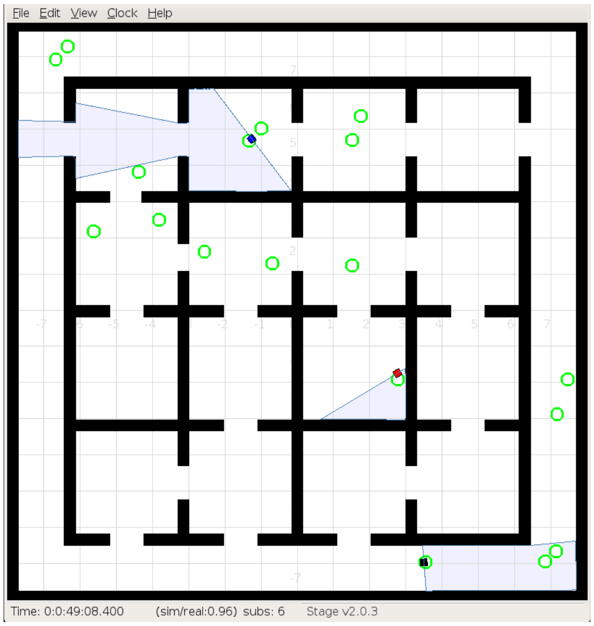

Figure 9: Stage image of the building scenario used in Experiment III. The environment is the same as the one used in [18].

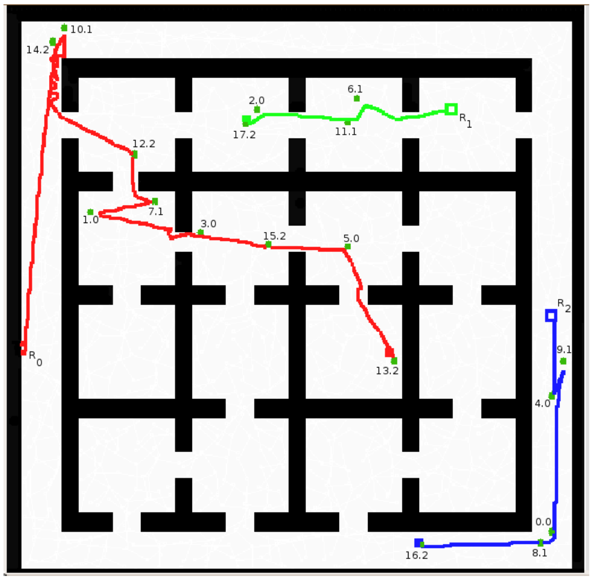

Figure 10: Simulation experiment III: An example showing the paths followed by robots 0, 1 and 2.

The experiments show that the robots were able to successfully complete the tasks scattered in the environment, generating paths comparable to those shown in [18]. However, a direct comparison with [18] is not possible due to differences in scale, number of tasks, and positions.

The timeline in Figure 11 shows the length (by task) of the path followed by each robot in each run in Experiment III. Short gaps indicate intervals where the robot was attempting a task that was completed by a different robot later (it counts as part of the distance traveled by the robot, but is not productive in terms of task completion). The large gaps are intervals where the robot had completed all its tasks, and took on another robot's tasks if the other robot was getting delayed too long. This is done as a means to ensure that as many tasks as possible are completed within the time limit (the overarching objective), thus allowing for some inefficiency in favor of completeness.

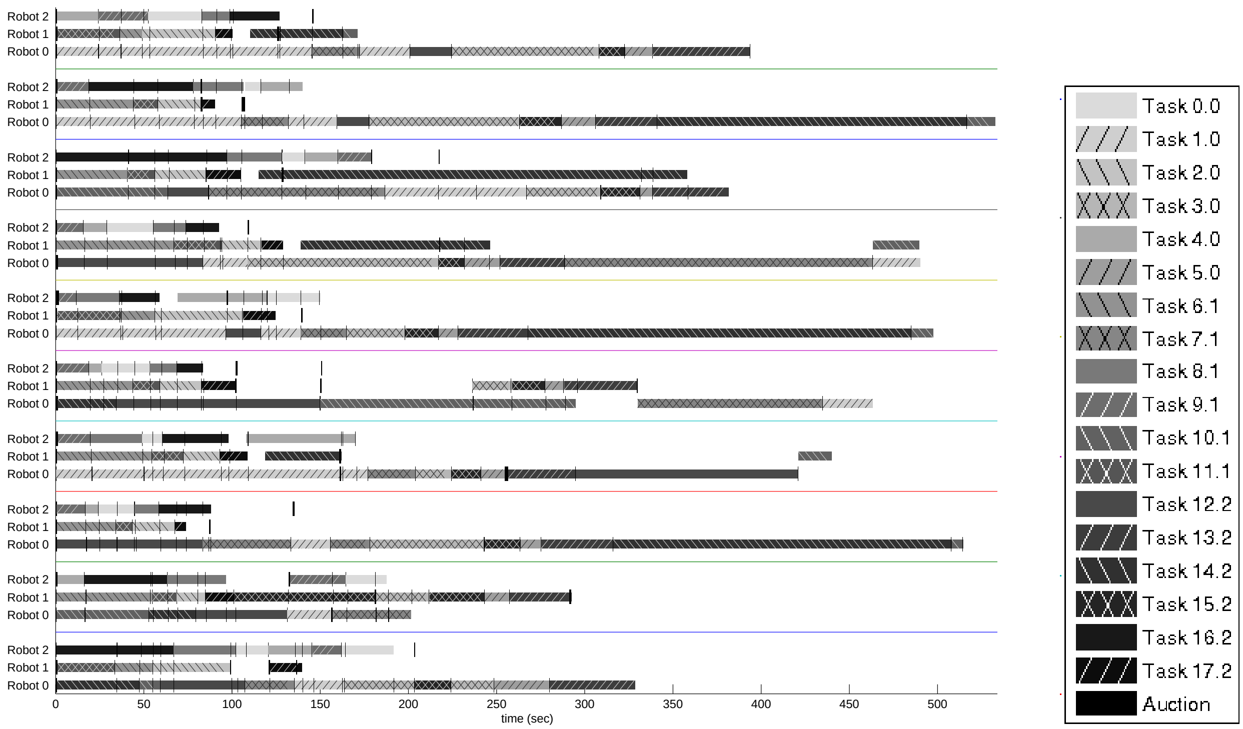

Figure 11: Experiment III timeline in the environment shown in Figure 9

In Figure 12 we show the environment used for Experiment IV. There are four obstacles, shown as small rectangles, that move across the corridor or in front of a door. The paths followed by the robots in one of the runs for Experiment IV are shown in Figure 13.

The experiments with moving obstacles showed only small differences from the ones without obstacles, as can be seen in Table 5. In the runs with obstacles the robots successfully coped with moving obstacles, showing on average only a 5% increase in path length. Similarly completion time averaged 6 min and 47 sec without obstacles, and showed an increase of approximatively 10% (to 7 min and 30 sec) with obstacles. The variance in average distance traveled was greater in the runs with obstacles, as expected. The robots dealt with obstacles by auctioning tasks again, and trying to access blocked areas repeatedly until the tasks in those areas were completed.

One difference we noted with previous experiments was that the robots had a tendency to follow a different order of task completion in each run. This is likely due to the environment and the RRT paths. The re-ordering did not appear to affect performance in terms of average path traveled by the three robots, however it did affect the maximum path traveled, as shown in the difference between runs 2 and run 9 in Figure 11.

Variations in the order in which tasks were accomplished was caused primarily by the RRTs which tend to bias distances according to the manner in which the RRT tree was formed. In an environment like this one, with many ways to access the same room, different experimental runs would often find different non-overlapping routes around the tasks. Overall, the distance traveled

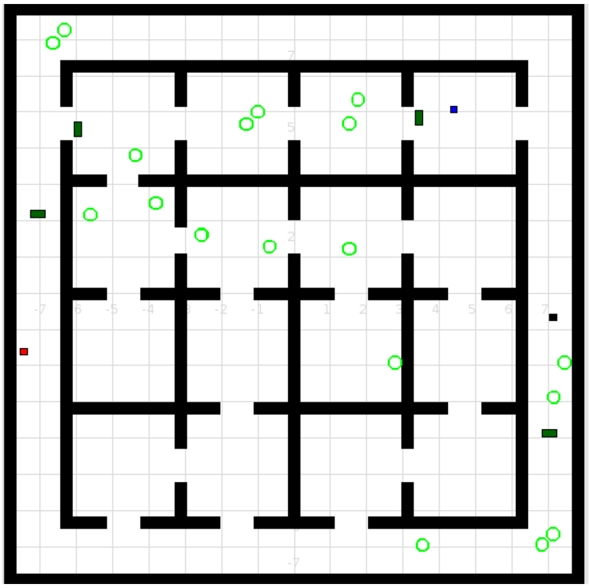

Figure 12: Stage image of the building scenario with obstacles used in Experiment IV. The obstacles move along their longer axis.

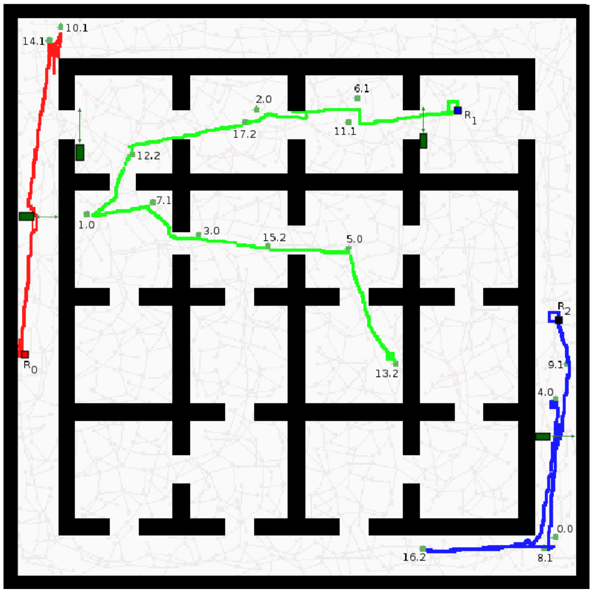

Figure 13: Simulation experiment IV: An example showing the paths followed by the robots when obstacles are present.

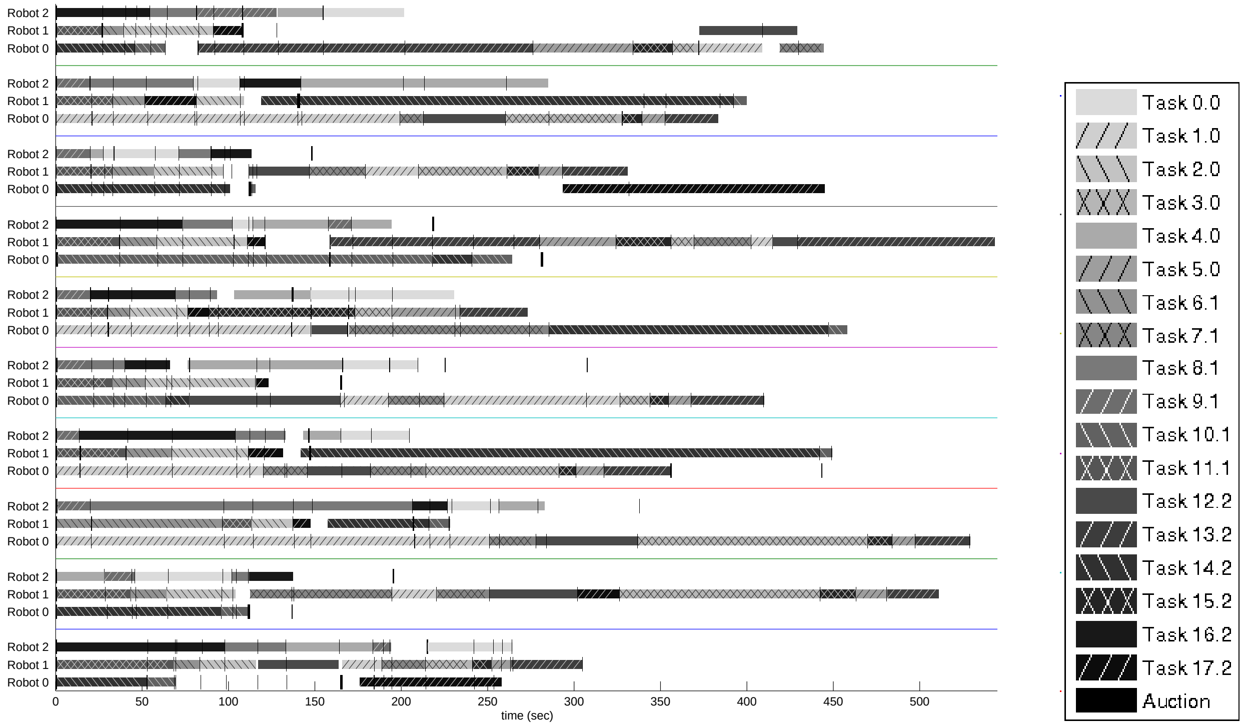

Figure 14: Experiment IV timeline in the environment shown in Figure 12 with obstacles.

Table 5: Average path and longest path traveled in Experiments III (no obstacles) and IV (with obstacles)

Expt Type	Average path		Average longest path	
	Mean (m)	σ	Mean (m)	σ
No Obst.	21.14	1.91	31.53	7.12
Obst.	22.31	2.22	33.62	5.77

did not show too great a variation between runs, despite this effect.

7 Conclusions and Future Work

We have presented an algorithm based on auctions for allocation of tasks to robots, which is robust to robot failure and environmental uncertainty. We have analyzed the algorithm's complexity compared with other algorithms in current use.

The experiments with real robots showed similar performance to the simulation experiments, even if the real robots were slower than the simulated ones and more prone to delays. In particular, the experiments showed that the task allocations found did not suffer significantly from the change in speed in the robots. As a side effect, the time for the auctions compared to the time to execute the tasks improved when experiments were done with real robots.

The robots proved adaptable, tasks were exchanged during execution, and the final task assignment was close to optimal. The comparison of performance between simulation and real robots showed that simulation results may be relied on.

Acknowledgements:

Work supported in part by the National Science Foundation under grants EIA-0324864 and IIS-0414466, and by the Industry/University Cooperative Research Center for Safety, Security, and Rescue at the University of Minnesota.

References

[1] W. Agassounon and A. Martinoli. Efficiency and robustness of threshold-based distributed allocation algorithms in multi-agent systems. In *Proc. of the 1st Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, pages 1090–1097, July 2002.

[2] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt. Robot exploration with combinatorial auctions. In *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, 2003.

[3] Steve Chien, Anthony Barrett, Tara Estlin, and Gregg Rabideau. A comparison of coordinated planning methods for cooperating rovers. In *Proc. of the Int'l Conf. on Autonomous Agents*, pages 100–101. ACM Press, 2000.

[4] M. B. Dias. *TraderBots: A Market-Based Approach for Resource, Role, and Task Allocation in Multirobot Coordination.* PhD thesis, Carnegie-Mel;on University, 2004.

[5] 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*, pages 115–122, Venice, Italy, July 2000.

[6] M Bernardine Dias, Marc B Zinck, Robert Michael Zlot, and Anthony (Tony) Stentz. Robust multirobot coordination in dynamic environments. In *Proc. Int'l Conf. on Robotics and Automation*, April 2004.

[7] M Bernardine Dias, Robert Michael Zlot, Nidhi Kalra, and Anthony (Tony) Stentz. Market-based multirobot coordination: A survey and analysis. Technical Report CMU-RI-TR-05-13, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, April 2005.

[8] Robert S. Engelmore and Anthony Morgan, editors. *Blackboard Systems.* Addison-Wesley, 1988.

[9] Brian P. Gerkey and Maja J Matarić. Sold!: Auction methods for multi-robot coordination. *IEEE Trans. on Robotics and Automation*, 18(5), October 2002.

[10] Brian P. Gerkey and Maja J Matarić. Multi-robot task allocation: Analyzing the complexity and optimality of key architectures. In *Proc. Int'l Conf. on Robotics and Automation*, September 2003.

[11] Brian P. Gerkey, Richard T. Vaughan, and Andrew Howard. The Player/Stage project: Tools for multi-robot and distributed sensor systems. In *Proc Int'l Conf on Advanced Robotics*, pages 317–323, June 2003.

[12] N. Kalra, D. Ferguson, and A. Stentz. Hoplites: A market-based framework for planned tight coordination in multirobot teams. In *Proc. Int'l Conf. on Robotics and Automation*, 2005.

[13] S. Koenig, C. Tovey, M. Lagoudakis, V. Markakis, D. Kempe, P. Keskinocak, A. Kleywegt, A. Meyerson, and S. Jain. The power of sequential single-item auctions for agent coordination. In *Proc. of the National Conf. on Artificial Intelligence*, pages 1625–1629, 2006.

[14] 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*, pages 995–1001, 2000.

[15] Michail G. Lagoudakis, Evangelos Markakis, David Kempe, Pinar Keskinocak, Anton Kleywegt, Sven Koenig, Craig Tovey, Adam Meyerson, and Sonal Jain. Auction-based multi-robot routing. In *Robotics: Science and Systems*, Cambridge, USA, June 2005.

[16] Maitreyi Nanjanath and Maria Gini. Auctions for task allocation to robots. In *Proc. of the Int'l Conf. on Intelligent Autonomous Systems*, pages 550–557, Tokyo, Japan, March 2006.

[17] Maitreyi Nanjanath and Maria Gini. Dynamic task allocation in robots via auctions. In *Proc. Int'l Conf. on Robotics and Automation*, 2006.

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