Auctions for robust task execution in multi-robot teams

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

Abstract

We present an auction-based method for dynamic allocation of tasks to robots. The robots have to visit locations in a 2D environment for which they have a map. Unexpected obstacles, loss of communication, and other delays may prevent a robot from completing its allocated tasks. Therefore tasks not yet achieved are rebid every time a task has been completed. This provides an opportunity to improve the allocation of the remaining tasks and to reduce the overall time for task completion. We present experimental results that we have obtained in simulation using Player/Stage.

Introduction

There are many real-world problems in which a set of tasks has to be distributed among a group of robots. We are interested in situations where each task can be done by a single robot, but sharing tasks with other robots will reduce the time to complete the tasks. Search and retrieval tasks as well as pickup and deliveries are examples of the types of tasks we are interested in.

What distinguishes task allocation to robots from other task allocation problems is the fact that robots have to physically move to reach the task locations, hence the cost of accomplishing a task depends on the current robot location and not just on the task itself.

We describe an efficient algorithm based on auctions to perform task allocation. Our 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 it is important to adapt dynamically to changing conditions. The algorithm is totally distributed. There is no central controller and no central auctioneer, each robot auctions its own tasks. This increases robustness and scalability.

The auction mechanism we propose attempts to minimize the total time to complete all the tasks. Given the simplifying assumption of constant and equal speed of travel for all the robots, this is equivalent to minimizing the sum of path costs over all the robots (Tovey *et al.* 2005). 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, or experiences any other delay, or loses communication, or is otherwise disabled, the system continues to operate and tasks get accomplished.

Our algorithm is greedy, and finds close-to-optimal solutions that are fast to compute. It is flexible, allowing robots to rebid every time a task is achieved. This provides an opportunity to produce a better allocation and increases the robustness of the system in case of unexpected problems or delays. Rather than forcing a costly re-computation of the entire optimal solution when a task cannot be achieved, the algorithm uses multiple auctions to reallocate tasks.

In this paper we report scalability results with varying numbers of tasks and robots (already reported in (Nanjanath & Gini 2006a)) and we specifically address the problem of performance in case of communication malfunctions.

Related Work

The problem we address 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, but are otherwise independent.

A recent survey (Dias *et al.* 2005) covers in detail the state of the art in using auctions to coordinate robots for accomplishing tasks such as exploration (Dias & Stentz 2000; Kalra, Ferguson, & Stentz 2005), navigation to different locations (Tovey *et al.* 2005), or box pushing (Gerkey & Matarić 2002). Auction-based methods for allocation of tasks are becoming popular in robotics (Dias & Stentz 2000; Gerkey & Matarić 2003; Tovey *et al.* 2005) as an alternative to other allocation methods, such as centralized scheduling (Chien *et al.* 2000), blackboard system (Engelmore & Morgan 1988), or application-specific methods, which do not easily generalize (Agassounon & Martinoli 2002) to other domains.

Combinatorial auctions have been tried as a method to allocate navigation tasks to robots (Berhault *et al.* 2003) 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.

Sequential single-item auctions tend to miss opportunities for optimal allocation, even though they can be computed in polynomial time. Our approach tries to find a tradeoff be-

Figure 1: The hospital environment. The top part of the figure shows the Stage simulation, with the locations of the tasks and of the robots. (The active robot has its range sensor traces shown). The lower part shows the paths generated by the RRT algorithm, with the location of the active robot on the paths indicated by a square. This is one of the single robot experimental runs, where only one robot is active.

tween computational complexity and optimality of allocations. We do not use combinatorial auctions, but we reauction tasks multiple times while they are being executed, so allowing for a better allocation.

Recent work (Tovey *et al.* 2005; Lagoudakis *et al.* 2005) has focused on producing bidding rules for robot navigation tasks that lower significantly the computational costs but give up the guarantee of finding an optimal solution. 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 differs in many ways. First, the auctioneer determines the winner of the auction, so if a robot fails to submit a bid (perhaps because of communication failure), the auction can continue. Second, our approach is designed for highly dynamic situations where unexpected delays during execution or communication failures can prevent a robot from accomplishing its tasks, or can make task accomplishment more time consuming than originally thought. By continuously rebidding and reallocating tasks among themselves during task execution, the robots adjust to changing situations. When the environment is highly dynamic, computing the optimal path to achieve all the tasks allocated to a robot, as in (Lagoudakis *et al.* 2005), might not pay off, because tasks are reallocated often.

Our approach is similar to the method presented in (Dias

et al. 2004) where a group of robots is given tasks to accomplish, and robots are selectively disabled in different manners, in order to examine their performance under different conditions. Performance is measured in terms of task completion. Their approach differs in that they do not assume a time limit for task completion. Additionally they use more complex robots, whose navigation and obstacle detection abilities are much better.

We have reported elsewhere (Nanjanath & Gini 2006b) results of our algorithm when tasks have priorities, which means the first few tasks receive more attention and later tasks may be abandoned in favor of accomplishing earlier ones. 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.

Proposed Algorithm

In this work we assume that each robot is given a map that shows its own location and the positions of walls and rooms in the environment. No information is given about where the other robots are located. The map allows a robot to estimate its cost of traveling to the task locations, and to compute the path to reach them from its original location.

Suppose a user has a set R of m robots $R = \{r_1, r_2, ...r_m\}$, and a set T of n tasks $T = \{t_1, t_2, ...t_n\}$, where each task is a location a robot has to visit. The user partitions the tasks into m disjoint subsets, such that

$T_1 \cup T_2 \cup ... \cup T_m = T$ and $T_i \cap T_j = \phi \ \forall i, j 1 \leq i, j \leq m.$ and allocates each subset to a robot. Note that a subset can be empty.

The initial task distribution done by the user might not be optimal. Some robots might have no task at all assigned to them, while others might have too many tasks, the tasks assigned to a robot might be spread all over the environment, and might be within easy reach of another robot, some tasks may be in an unreachable location.

A robot must complete all its tasks unless it can pass its commitments to other robots. Since the robots are cooperative, they will pass their commitments only if this reduces the estimated task completion time. 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. This process is accomplished via single-item reverse auctions, in which the lowest bid wins, that are run independently by each robot for their tasks.

Each bid 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.

To generate paths efficiently, robots use Rapidly-expanding Random Trees (RRTs) (Kuffner & LaValle 2000). Generation of RRTs is very fast, and scales well with large environments. An example of a RRT is shown in Figure 1.

Auctions are parallel, i.e. many auctioneers may put up their auctions at once, but since each 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. Robots compute their bids for all the parallel auctions assuming they start at their current location. This can results in bids that over- (or under-)estimate the true cost.

Repeat for each robot $r_i \in R$:

1. Activate r_i with a set of tasks T_i and a list of 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 the path found.
5. Order task list T_i by ascending order of c_j.
6. r_i does in parallel:
   (a) Auction the assigned tasks:
      i. Create a Request For Quotes (RFQ) with tasks T_i.
      ii. Broadcast the RFQ to R_{-i} and wait for bids.
      iii. Find the lowest bid b_{jk} among all the bids for task t_j.
      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 for the tasks left (r_k has now new tasks).
      vi. Repeat from 6(a)iii until all tasks are assigned. Robots that do not bid on tasks are ignored in the auction.
   (b) Bid on RFQs received from other robots:
      i. Find a RRT path for each task t_r in the RFQ.
      ii. Create a cost estimate c_r for each t_r that the robot found a path to.
      iii. Send the list of costs to the auctioneer that sent the RFQ.
   (c) Begin execution of first assigned task:
      i. Start executing the first task t_j by finding a path in the RRT and following it as closely as possible.
      ii. If new tasks are added as result of winning auctions, insert them in T_i keeping it sorted in ascending order of cost, and repeat from 6(c)i.
      iii. If r_i is stuck, auction r_i's tasks.
      iv. If t_j is completed successfully, restart from 4.

until timeout.

Figure 2: Task allocation algorithm.

The algorithm that each robot follows is outlined in Figure 2. 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. We show later experimental results in case of partial communications loss. A robot can choose not to bid on a particular task, based on its distance from and accessibility to that task.

Once the auctioned tasks are assigned, the robots begin to move to their task locations, attempting the nearest task first (i.e. the task with the lowest cost).

When a robot completes its first task, it starts an auction again for its remaining tasks, in an effort to improve the task allocation. In case robots get delayed by unexpected obstacles, this redistribution of tasks allows them to change their commitments and to adapt more rapidly to the new situation.

If a robot is unable to complete a task it has committed to, it can auction that task. Any task that cannot be completed by any of the robots is abandoned. We assume that there is value in accomplishing the remaining tasks even when not all of them can be completed.

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 dynamical additions of new tasks during the execution, but for simplicity, in the experiments described in Section , the set of tasks and of robots is known at start and does not change during the execution.

Experimental Setup and Analysis

We conducted experiments in the Player/Stage simulation environment (Gerkey, Vaughan, & Howard 2003). We simulated robot deployment in complex 2-D worlds, using as our test environment the section of the hospital world from Player/Stage shown in Figure 1. The hospital world consists of several rooms with small doorways and limited accessibility, covering a total area of $33 \times 14\mathrm{m}^2$.

Each robot is simulated as a small differential drive vehicle placed at an arbitrary location in the world. It is equipped with 5 sonar sensors mounted at 45° angles across its front, which are used for obstacle avoidance. While these sensors allow the robot to avoid colliding into straight walls, robots tend to get stuck on corners where they cannot detect the corner before colliding into it. This tend to produce unexpected delays during the execution that vary greatly between runs. Tasks are modeled as beacons placed at different positions in the environment.

The experiments were run for 10 minutes each, to avoid long runs when robots were unable to make much progress. This also allowed us to test how often the robots could not accomplish all the tasks in the allocated amount of time.

We ran each experiment 10 times, with the same initial conditions, but with different initial task allocations. The auction algorithm is sensitive to the order in which tasks are given to the robots. To reduce this effect we supplied the tasks to the robots in a random order each time an experiment was run. This, combined with the inherently random nature of the RRT generation algorithm, resulted in significant variations across runs both in the allocation of tasks and time taken to complete the tasks.

Experiments with different numbers of robots

We used different experimental setups, each with 16 tasks placed in different rooms. We tested the setups with 1, 3, and 10 robots, and ran a set of experiments with a single auction (with no rebidding) to use as a baseline.

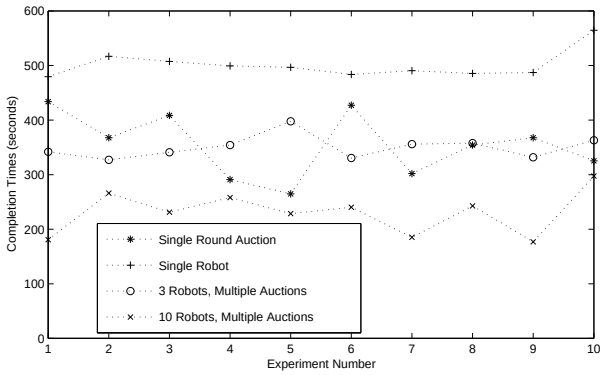

Figure 3: Time spent trying to complete tasks in different robot-auction combinations.

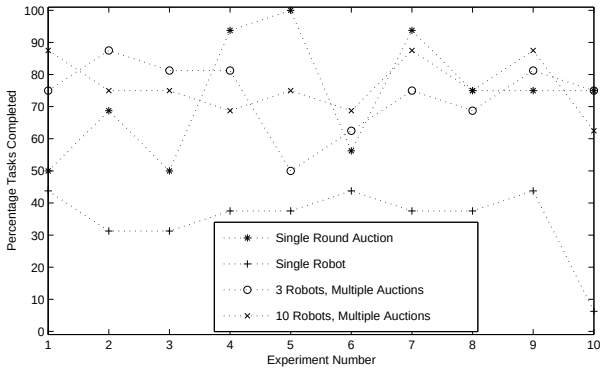

Figure 4: Relative task completion rates for different robot-auction combinations

Performance results are shown in Figure 3. The results show the time taken to complete all the tasks that were accomplished in each run. We can observe that a single robot takes longer, but, as expected, the speedup when using multiple robots is sublinear. A single round auction tends to perform worse than multiple auctions and has more variability in the time needed to complete the tasks. This is consistent with the observation that reallocation of tasks via additional bidding tends to produce on average a better allocation. The results are best when the number of robots and tasks is balanced. When the task are few some of the robots stay idle, when the tasks are too many with respect to the number of robots the completion time increases, since each robot has more work to do.

Figure 4 shows the percentage of tasks completed for each run. Since the number of tasks was relatively large with respect to the time available and the distance the robots had to travel, very few runs had all the tasks completed. We can observe that with a single robot only a small percentage of the 16 tasks get accomplished in the time allocated. With a more balanced number of tasks and robots a much larger percentage of tasks gets done. We can see the differences between runs when using a single round auction versus using multiple rounds. The performance of multiple rounds of

auctions is not consistently better than when using a single round. Recall that in each experiment the initial allocation of tasks to robots was different, and some allocations were clearly better than others.

Experiments with Communication Loss

We ran a second set of experiments, where we modeled loss of communication among the robots and its impact on task performance. For this set of experiments we kept the number of robots fixed at 10, with 16 tasks placed in different rooms in the hospital environment as in the previous set of experiments.

Communications among the robots is modeled in terms of two variables: range and efficiency. The range measures how far the signal of each robot reaches. For the purposes of these experiments, range has been set to the size of the environment - all robots are within range. Communication efficiency encodes the rate at which messages sent from one robot to another actually reach the other robot. The current experiments have been done keeping the efficiency uniformly across all the robots as 75%. With this setup, each time a message is sent (the message may be the response to an RFQ, or a task assignment on winning a bid), the message has a 75% chance of actually reaching the desired recipient. Results are reported in Table 1.

We assume that as long as a signal is available, actual communication is instantaneous - issues such as signal being interrupted in the middle of a transaction are not considered.

The possible error conditions are dealt with as follows:

1. If a robot does not receive notification of an RFQ, then the robot simply does not respond, and hence its bids are not considered in the computations for the task allocation.
2. If after assigning a task to a robot, the robot fails to accept the assigned task, the task is taken back by the auctioneer, and the original agent puts it back on its own task list.
3. In the event a set of tasks is missed because of communication failures, the robots bring the tasks back to auction after a 10 second wait. This continues until either all the tasks are completed or the time is up.

In order to maximize the chance of completing tasks, the robots initially send a list of assigned tasks to each other, and use the list to update their own task lists. Thus, a complete copy of all the tasks assigned to date is maintained with each robot. This list of tasks is updated whenever a task is completed by any robot (and the information reaches that robot). This can lead to redundancy with multiple robots trying to accomplish the same tasks. However, since the task lists are synchronized periodically, this redundancy should be caught before time is wasted.

Table 1: Communications Experiments. Results obtained with 10 robots and 16 tasks with 75% and 100% communications efficiency. Task locations are the same across all the experiments, but the initial task allocation to robots varies randomly. Times are in seconds. Results averaged over 10 runs.

%	Task Completion Rate		Task Completion Time	
Comm	Mean	Std Dev	Mean	Std Dev
75%	56.875	12.65	397.6748	31.92
100%	51.875	5.92	395.2361	24.18

The results of the experiments with communication loss show that 75% communication does slightly better in task completion than 100% communication but has a larger variance. This may be because the robots with no communication had less interference while performing tasks, though they spent more time in completing them, or because of a better initial task allocation. Something similar was observed in the experiments reported in (Dias *et al.* 2004). We will be conducting further analysis with different communication rates, to track this further.

Conclusions and Future Work

We have presented an algorithm for allocation of tasks to robots. The algorithm is designed for environments that are dynamic and where failures are likely.

We assume the robots are cooperative, and try to minimize the total time to complete all the tasks assigned to the group. Each robot acts as an auctioneer for its own tasks and tries to reallocate its tasks to other robots whenever this reduces the cost. Robots also re-assess the current situation and attempt to improve the current task allocation by putting their remaining tasks up for bid whenever they complete a task. The process continues until all the tasks have been completed or the allocated time has expired.

We removed any need for central coordination; tasks are assigned in a distributed fashion, so that the system can recover from single or even multiple points of failure. This prevents us from using any centralized system, such as a blackboard system (Engelmore & Morgan 1988), since this will create a single point of failure.

Future work will include considering additional costs to do tasks over the cost of reaching the task location, and introducing heterogeneous robots having different speeds and capabilities. We will also compare the performance of our algorithm against the performance of other auction-based task allocation algorithms, such as the one reported in (Lagoudakis *et al.* 2005). There are tradeoff between quality of the solution found and rate of change in the environment that we will explore.

Acknowledgments

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

References

Agassounon, W., and Martinoli, A. 2002. 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*, 1090–1097.

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

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

Dias, M. B., and Stentz, A. 2000. A free market architecture for distributed control of a multirobot system. In *Proc. of the Int'l Conf. on Intelligent Autonomous Systems*, 115–122.

Dias, M. B.; Zinck, M. B.; Zlot, R. M.; and Stentz, A. T. 2004. Robust multirobot coordination in dynamic environments. In *Proc. Int'l Conf. on Robotics and Automation*.

Dias, M. B.; Zlot, R. M.; Kalra, N.; and Stentz, A. T. 2005. Market-based multirobot coordination: A survey and analysis. Technical Report CMU-RI-TR-05-13, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA.

Engelmore, R. S., and Morgan, A., eds. 1988. *Blackboard Systems*. Addison-Wesley.

Gerkey, B. P., and Matarić, M. J. 2002. Sold!: Auction methods for multi-robot coordination. *IEEE Trans. on Robotics and Automation* 18(5).

Gerkey, B. P., and Matarić, M. J. 2003. Multi-robot task allocation: Analyzing the complexity and optimality of key architectures. In *Proc. Int'l Conf. on Robotics and Automation*.

Gerkey, B. P.; Vaughan, R. T.; and Howard, A. 2003. The Player/Stage project: Tools for multi-robot and distributed sensor systems. In *Proc Int'l Conf on Advanced Robotics*, 317–323.

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

Kuffner, J. J., and LaValle, S. M. 2000. RRT-connect: An efficient approach to single-query path planning. In *Proc. Int'l Conf. on Robotics and Automation*, 995–1001.

Lagoudakis, M. G.; Markakis, E.; Kempe, D.; Keskinocak, P.; Kleywegt, A.; Koenig, S.; Tovey, C.; Meyerson, A.; and Jain, S. 2005. Auction-based multi-robot routing. In *Robotics: Science and Systems*.

Nanjanath, M., and Gini, M. 2006a. Auctions for task allocation to robots. In *Proc. of the Int'l Conf. on Intelligent Autonomous Systems*, 550–557.

Nanjanath, M., and Gini, M. 2006b. Dynamic task allocation in robots via auctions. In *Proc. Int'l Conf. on Robotics and Automation*.

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