Book Title
Book Editors
IOS Press, 2003

Auctions for task allocation to robots

Maitreyi Nanjanath [a] and Maria Gini [a,1]
[a] *Department of Computer Science and Engineering, University of Minnesota*

Abstract. We present an auction-based method for dynamic allocation of tasks to 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, in any order. Unexpected obstacles 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.

Keywords. multi-robot systems, auctions, task allocation, Player/Stage

1. Introduction

There are many real-world problems in which a set of tasks has to be distributed to a group of robots. We are interested in situations where, while a single robot could do all the tasks, sharing the work with other robots will reduce the time to complete the tasks and increase the success rate. Search and retrieval tasks, which have been studied extensively in robotics (see, for instance [10,14]), are examples of the types of tasks we are interested in.

In our study, tasks are locations in the map that have to be visited by the robots, but we could easily add other activities the robot has to perform at each task location. 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 highly on the current robot location.

We describe an efficient method based on auctions to perform task allocation. 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 it is important to adapt dynamically to changing conditions. The method is totally distributed. There is no central controller and no central auctioneer, which increases robustness.

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. 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

[1]Correspondence to: Maria Gini, Dept of Computer Science and Engineering, 200 Union St SE. Minneapolis, MN 55455. Tel.: +1 612 625 5582; Fax: +1 612 625 0572; E-mail: gini@cs.umn.edu.

robot finds an unexpected obstacle, or experiences any other delay, or is 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 when solutions are unobtainable, rather than forcing a costly re-computation of the entire optimal solution.

2. 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 $T = \{t_1, t_2, ...t_n\}$. In this study tasks are simply locations in the map that have to be visited, but the algorithm can take into account additional costs of doing the task once its location has been reached. The user partitions the tasks into m disjoint subsets, such that

$T_1 \cup T_2 \cup ... \cup T_m = T$ and $T_i \cap T_j = \phi \ \forall i, j1 \leq i, j \leq m.$

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 distributed 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) [12]. Generation of RRTs is very fast, and scales well with large environments. An example of a RRT is shown later in Figure 2.

Auctions are simultaneous, 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 the auctions done sequentially.

The algorithm each robot follows is outlined in Figure 1. 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. When estimating costs for tasks in different auctions, a robot uses only its current position, without accounting for possible movements in between task locations. 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

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 assign t_j to robot r_k else keep t_j. 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.

   (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 1. Task allocation algorithm.

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.

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 3, the set of tasks and of robots is known at start and does not change during the execution.

Figure 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 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.

3. Experimental setup and analysis

We conducted experiments in the Player/Stage simulation environment [9]. 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 2. 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 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 in the execution. Tasks are modeled as beacons placed at different positions in the environment.

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 final set of experiments with a single auction (with no rebidding) to use as a baseline.

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.

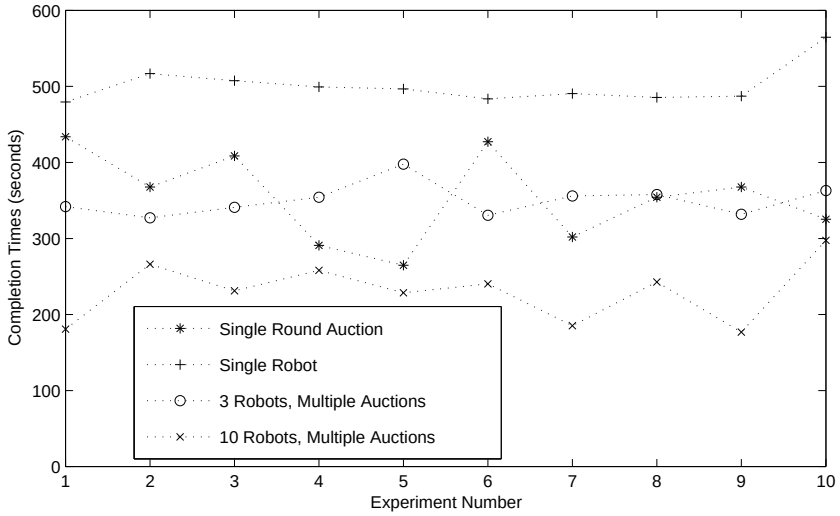

Figure 3. Time spent trying to complete tasks in different robot-auction combinations.

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 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.

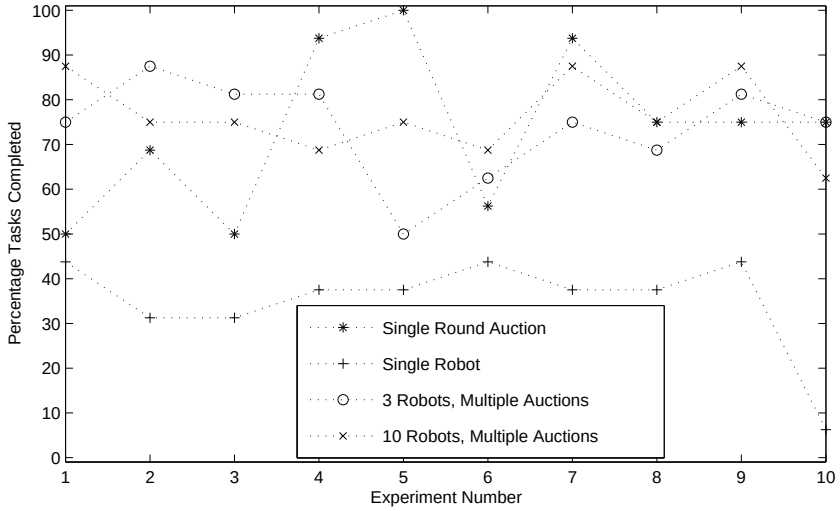

Figure 4. Relative task completion rates for different robot-auction combinations

4. Related Work

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, but are otherwise independent.

A recent survey [5] covers in detail the state of the art in using auctions to coordinate robots for accomplishing tasks such as exploration [4,11], navigation to different locations [15], or box pushing [7]. Auction-based methods for allocation of tasks are becoming popular in robotics [4,8,15] as an alternative to other allocation methods, such as centralized scheduling [3], blackboard system [6], or application-specific methods, which do not easily generalize [1] to other domains.

Combinatorial auctions have been tried as a method to allocate navigation tasks to robots [2] 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 tend to miss opportunities for optimal allocation, even though they can be computed in polynomial time. Our approach tries to find a tradeoff between 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 [15,13] has focused on producing bidding rules for robot navigation tasks that lower computational costs while providing 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 dynamic situations where unexpected delays during execution can prevent a robot from accomplishing its tasks, or can make task accomplishment more expensive than originally thought. By continuously rebidding and reallocating tasks among themselves during task execution, the robots react and adjust to changing situations.

5. 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 reassess 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 [6], 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.

In addition, we have left addressing communication malfunctions for future research. It is our experience that robots can become disabled but rarely lose the ability to communicate. Disabling communication will introduce new challenges. Since each robot is initially given its own tasks, it will have to maintain separately the complete list of tasks given to the system as a whole. This can be done by having an initial communication phase that involves broadcasting the list of tasks to all the robots (assuming no communication failure at that time). Each robot will also need to track task completion by the other robots, and periodically broadcast its own state.

Acknowledgments
Work supported in part by NSF under grants EIA-0324864 and IIS-0414466.

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. Int'l Conf. on Robotics and Automation*, 2003.

[3] 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*, pages 100–101. ACM Press, 2000.

[4] 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.

[5] M. B. Dias, R. M. Zlot, N. Kalra, and A. T. 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.

[6] R. S. Engelmore and A. Morgan, editors. *Blackboard Systems*. Addison-Wesley, 1988.

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

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

[9] 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*, pages 317–323, June 2003.

[10] D. Goldberg and M. J. Matarić. Design and evaluation of robust behavior-based controllers. In T. Balch and L. E. Parker, editors, *Robot Teams: From Diversity to Polymorphism*. A K Peters Ltd, Natick, MA, Apr. 2002.

[11] 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.

[12] 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.

[13] 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.

[14] K. Sugawara and T. Watanabe. Swarming robots – foraging behavior of simple multi-robot systems. In *Proc. IEEE/RSJ Int'l Conf. on Intelligent Robots and Systems*, Lausanne, Switzerland, 2002.

[15] 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.