

MULTIROBOT CONTROL USING TASK ABSTRACTION IN A MARKET FRAMEWORK

Robert Zlot
Anthony Stentz

Robotics Institute
Carnegie Mellon University
Pittsburgh, PA 15213

ABSTRACT

It is well-recognized that the utilization of multiple robots is of great importance for robust task execution in many application domains. Often, tasks can be performed more efficiently and reliably using several robots, and, in many cases, objectives can be achieved exclusively by using multiple robots. In this paper, we look at a novel approach to multirobot coordination that simultaneously distributes task allocation, planning, and execution among members of a robot team. By combining traditional hierarchical task decomposition techniques with recent developments in market-based multirobot control, we obtain an efficient and robust system capable of solving complex distributed problems. Results are presented for a simulated area reconnaissance scenario.

INTRODUCTION

Many applications of multirobot systems can be viewed in terms of problems dealing with scarce resources in highly uncertain, dynamic environments. The optimization problems involved are usually highly unscalable making centralized solutions impractical. Distributed solutions are faster and more reliable; although these benefits are often obtained in exchange for guarantees on solution quality. We introduce a new approach that addresses many of the challenges inherent to multirobot coordination, which can deliver efficient solutions quickly in a completely distributed setting.

Our research addresses both the task allocation problem as well as multirobot planning. The multirobot task allocation problem can be stated as follows: *given a set of tasks and a group of robots, what is the optimal way to distribute the tasks among the robots?* In

terms of mission planning, it is desirable to divide the planning among the robot team to exploit the presence of multiple processors and varying local information, but it is difficult to do this while still ensuring that the resulting global plan is efficient.

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

RELATED WORK

We handle the coordination of planning and execution through the use of the *TraderBots*¹ market architecture. Well-known market mechanisms (such as auctions and contracts) are applied to a team of robots and other agents who participate in a virtual economy. Within the economy, participants can negotiate contracts to execute tasks, or trade in resources (such as use of specialized sensors or actuators, storage capacity in transport tasks, computation time) required to complete such tasks. Market architectures are typically distributed; however Dias and Stentz [4] propose a system which includes opportunistic centralized optimization within subgroups of the team. Market-based implementations for multirobot systems have been successfully demonstrated in simulation [3, 2, 9] and several physical robot applications [13, 14, 6].

Our work is also related to research in the area of AI

¹The name *TraderBots* was recently adopted for our market-based coordination work that was started by Stentz and Dias in 1999 [12].

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

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

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

TASK TREES

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

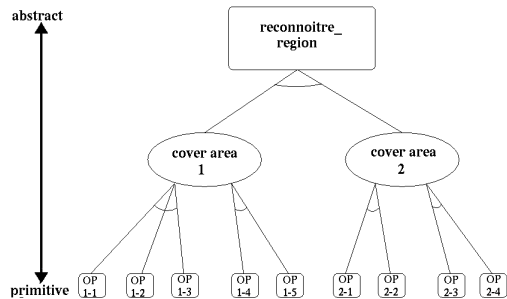


Figure 1: An example task tree for an area reconnaissance scenario with two areas. The arcs on the edges represent *AND* operators, and the absence of an arc represents an *OR* operator.

TASK TREE AUCTIONS

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

Any agent in the system can act as an auctioneer at any time. Some robots or agents may start with knowledge about the mission objectives, new tasks may be discovered while processing information acquired during execution or may be introduced into the system by a human operator. When a robot has new information about tasks or mission objectives, it can decompose those tasks (if applicable) and call an auction for the corresponding task tree. This decomposition need not be complete; that is, the leaves of the task tree need not be executable actions at this point. Currently, each auction is for a single task tree.

In order for robots to accurately value tasks, a cost function must be suitably defined which maps a set of resources required to a positive real-valued cost. Additionally, in some situations the bidder may have to consider synergies between tasks not accounted for by the structure of the tree. If these relationships are not taken into account, the bidder may end up paying more

to execute two or more tasks which would have been less costly to execute individually. For example, suppose the robot is considering two tasks which are for sale and which are not siblings in the task tree: the first task is *go to point A*, and the second is *go to point B*. The robot calculates its cost to go to point *A* to be \$5 and its cost to go to point *B* to be \$7, while the cost to go from *A* to *B* is \$10. Now suppose the robot offers to do task *A* for \$6 and task *B* for \$8. If the robot were to win both tasks, it would cost a minimum of \$15 to go to point *A* and then to point *B* (assuming costs are symmetric). Yet the robot would incur a loss as it is only being paid \$14 to execute the pair of tasks.

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

For an abstract (interior) task, the valuation is the cost of minimally satisfying the task represented by the corresponding tree node. For example, if the connective on the tree node is *AND*, the value of the task is the expected marginal cost of performing *all* of the sub-tasks (children) of the node. If it is an *OR* node, then it is the *minimum* expected cost of executing *one* of the children. However, the bidding robot may have a better plan than the auctioneer. This can be due to a difference in the current state or resources of the bidder, or due to differing local information. The bidder can perform its own decomposition of the abstract task, and if the resulting plan is of lower cost, then it makes its bid based on this new plan. If the plan is indeed a better plan, the bidder wins the task in the auction and it can replace the auctioneer’s original plan with its own. It is this part of the mechanism which allows for the distribution of planning among the robots.

Auctions can be called in a series of rounds. From an optimization standpoint, a round can be thought of as an improvement step in a distributed local search algorithm. If the auctions in a round are not simultaneous, then after each round the global solution is guaranteed to be no worse than it had been before the round started. Therefore, when starting with an initial feasible solution (or once the initial auction round has terminated) the system behaves like an *anytime* algorithm – a feasible solution is available quickly, and the quality of this solution can only be improved over time. In the case of a multirobot application, the robots can start to execute a feasible plan and at the same time improve

the plan through further auctions while executing. The same mechanism can be applied to auction online tasks which are added to the system during execution (*e.g.* new tasks may be added by a human operator or autonomously generated based on information gathered by the robots).

EXAMPLE

As an example, consider a planetary exploration scenario. There are several robots on a team tasked with observing a cliff face below them as part of the global science mission. The robots are able to perform cliff-descent by having one robot anchor a tether and lower another robot down the rockface. The robots are also equipped with a repair capability in which one robot can tow another robot to a base for repair. The partial tree in figure 2 shows one possible decomposition of the task in which a robot can drive along the cliff to location *A* where it can easily descend the slope, and then proceed back along the bottom of the cliff to location *G* from where the rockface can be observed. Suppose this plan is part of the task tree up for auction, and a robot R_1 is considering a bid for these tasks. R_1 calculates that it would cost \$50 to drive to *A*, \$10 to descend the slope, and \$40 to drive to *G* and take readings, for a total cost of \$100. However, an alternative plan for robot R_1 is to repair robot R_2 and have R_2 anchor R_1 ’s descent of the cliff face, for a total cost of \$70 (figure 2). Thus robot R_1 can bid based on a \$70 cost, thus replacing the auctioneer’s subplans of the `observe_cliff` plan with its new decomposition if it wins the auction.

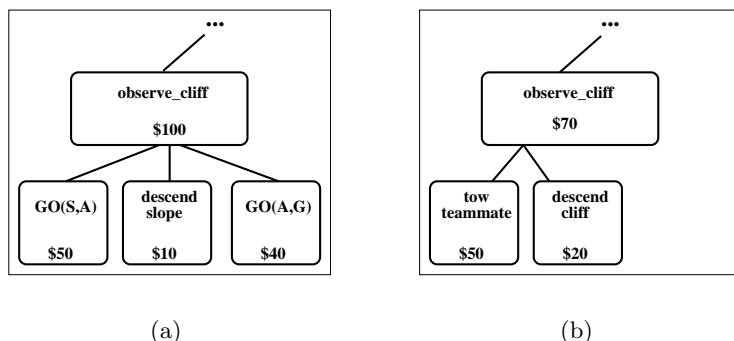


Figure 2: Two decompositions for an abstract task. Tasks costs for robot R_1 are indicated on each task node. (a) The initial decomposition offered by the auctioneer (with R_1 ’s bid labeled). (b) The decomposition after R_1 ’s winning bid.

POLYNOMIAL TIME AUCTION CLEARING ALGORITHM

The process of allocating the tasks in the task tree to the highest bidder, is an instance of a combinatorial auction winner determination problem (CAWDP). In this framework, the leaf nodes of the tree can be viewed as *items* and the interior nodes as *bundles* of items containing its leaf descendants. The objective for the auctioneer is to clear the auction by selling the combination of items that would maximize revenue, in which case tasks have been sold to the robots that are expected to be best suited to execute them. CAWDP is known to be \mathcal{NP} -hard in general [11]. However, since the bids are structured as a tree, this is a special case where winner determination can be computed in polynomial time. Additional constraints are dictated by the structure of the tree: for example, if a task node is in the optimal CAWDP solution, then none of its descendants can be; in the case of an *OR*-connective node, at most one child of the node can be in the optimal solution. We use a slightly modified version of an algorithm due to Rothkopf et al. [10], which has time complexity $O(n|T|)$ (where T is the set of all nodes in the tree and n is the number of leaves in the original tree).

AUCTION CLEARING ALGORITHM

Let T be the set of bundles, and \mathcal{C}^* be the set of bundles in the optimal solution. Let $op(B)$ be the connective (logical operator) of node (bundle) B , and R be the root of the tree. The variables $v(\cdot)$ represents the value of a bundle.

1. Set $\mathcal{C}^*(B) = \{B\}$ for every leaf B . Initialize B_p (a *parent* node) to anything other than the root of the tree.
2. Calculate the depth of each node $B \in T$ from the root R .
3. while $B_p \neq R$
 - (a) Find a leaf with the greatest depth from the root R (ties are broken arbitrarily). Call this leaf node B_{max} .
 - (b) Let B_p be the parent of B_{max} , and let \mathcal{S} be the set of children of B_p .
 - (c) if $op(B_p) = AND$

$$v(\mathcal{S}) := \sum_{B \in \mathcal{S}} v(B).$$
else if $op(B_p) = OR$

$$v(\mathcal{S}) := \max_{B \in \mathcal{S}} v(B).$$
else (*other connectives possible*)

- ...
 - (d) if $v(B_p) > v(\mathcal{S})$

$$\mathcal{C}^*(B_p) := B_p.$$
else
$$v(B_p) := v(\mathcal{S})$$

$$\mathcal{C}^*(B_p) := \bigcup_{B \in \mathcal{S}} \mathcal{C}^*(B).$$
 - (e) $T := T - \mathcal{S}$.
- end

EXPERIMENTS

Our approach was tested using a graphical simulation of an area reconnaissance scenario. The objective for the robot team is to view all of several predefined areas of interest (AIs) within a large geographical area while minimizing (distance-based) travel costs. The robots and AIs are randomly placed at the start of the simulation. Each robot is equipped with a range-limited 360° line-of-sight sensor. One robot (robot R) is initially aware of the global mission, *i.e.* it starts with an abstract task describing the overall mission. This robot then performs a task decomposition, decomposing the global task into subtasks representing the task of covering each AI, and in turn each of the AI subtasks is broken down into several groups of observation points (OPs) from which a robot can view part of an AI (Figure 3). For computational purposes, we limit the number of possible OPs considered for each area to twelve per AI (which surround the perimeter of the AI). The objective is achieved by the robot team visiting all of the required OPs, minimizing overall distance traveled. (This optimization problem is an instance of the multi-depot traveling salesman path problem which is known to be \mathcal{NP} -hard, and thus our solution is an approximation to the optimal solution). Robot R then holds a tasktree auction, distributing subtasks among the team. Other robots may bid node in any level in the tree, allowing them to perform their own decompositions for interior nodes. The auctions then proceed in rounds in which each robot optionally holds a tasktree auction for one of its subtrees in a round-robin fashion. In each round, the global cost is monotonically non-increasing. Eventually a local minimum in cost space is reached; the robots may execute their tasks at or before this time. Negotiation may carry on during execution.

RESULTS

The above scenario was run varying the number of robots and AIs, with 100 runs for each case. Each AI

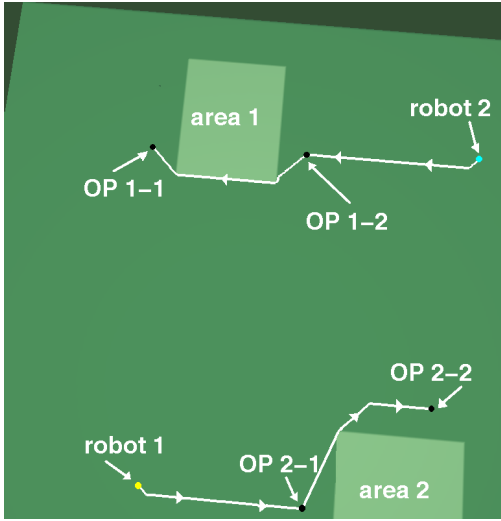


Figure 3: Example scenario. The objective is to generate observation points (OPs) to view the two areas with minimum travel cost.

was a randomly-sized rectangular region from which 12 candidate OPs surrounding the region were considered as solution points.

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

The second algorithm is a greedy algorithm (which we will refer to as *GR*) that looks at all possible OP candidates (12 per AI) and auctions non-redundant OPs off one at a time until coverage is complete. All of the potential OPs are bid on in each round, but only the OP with the lowest-cost bid is sold. There is no tree decomposition involved – all potential OPs are considered. This algorithm behaves similarly to multirobot coordination architectures which greedily allocate the best suited task to the best suited robot (*e.g.* [13, 8, 6]).

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

We compared the overall solution cost of the task tree auction algorithm (c_{TT}) to the solutions produced by

each of the other algorithms (*c_{FTL}*, *c_{GR}* and *c_{OPT}*). We also compared the execution times (t_{FTL} , t_{GR} , t_{OPT} and t_{TT}) of the different algorithms. Results are presented in tables 1 and 2.

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

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

In terms of solution cost, the task tree algorithm is 10-15% better than the *FTL* algorithm. One reason for this is that *TT* allows distributed replanning, so the *TT* solution is expected to be no worse than the *FTL* – in the worst case no replanning is done by the task tree algorithm and the original tree decomposition is preserved². In addition, the task tree algorithm also has an advantage because it allows reallocation through task subcontracting, permitting agents to discard tasks that they may have been too hasty in purchasing earlier on. It should also be noted that if the solutions are, as we suspect, close to optimal, then a 10-15% margin of improvement is significant. We can see this in the 2-robot 1-area case in which we were able to compute the optimal solution. Here the task tree algorithm was only 19% worse than optimal, as compared to *FTL* which was almost 40% worse than *OPT*. The execution time listed for the task tree algorithm (t_{TT}) is the time taken to reach the local minimum cost; although *FTL* appears to run much faster, *TT* is an anytime algorithm and often reaches a lower cost than *FTL* long before it reaches its local minimum.

On average, the task tree algorithm and the *GR* algorithm produce about the same solution quality; however, the task tree algorithm is faster and does not rely on a central auctioneer. The execution time for the task tree algorithm shown in table 2 reflects the time taken to reach the locally optimal solution. Though *TT* found its local optimum three to four times faster than *GR*, the task tree algorithm produced a feasible solution in an even shorter time, which appears to improve in its advantage over *GR* as the problem size increases. The

²This is not always true: since the task tree algorithm is a local search based on myopic cost estimates, the solution reached will depend on the order the tasks are allocated to each robot.

Robots	Areas	t_{FTL}/t_{TT}	t_{GR}/t_{TT}	t_{GR}/t_{TTF}	t_{OPT}/t_{TT}
2	1	.21	3.9	4.0	117
4	2	.14	2.7	3.6	
5	3	.10	2.8	5.0	
7	5	.08	3.4	6.3	
4	8	.06	3.1	12.7	

Table 2: Experimental results: comparison of execution times (results shown are averages of the ratio of execution times taken over 100 runs). t_{TTF} is the time taken for the task tree algorithm to find an initial feasible solution.

task tree algorithm guides the search quickly through a reduced search space without compromising the solution quality, while also allowing for a distributed search.

CONCLUSIONS

We have introduced a new method for distributing execution and planning over a team of robots and agents which combines ideas from hierarchical planning with a market-based multirobot coordination architecture. Empirical results in computer simulation show that task tree auctions can produce cost-efficient solutions to difficult optimization problems in a time-efficient manner.

In future work we will look at extending the task description language to handle richer task constraints and interactions, such as partial order precedence relations and conflicts arising between tasks competing for the same resources. Additionally, we would like to address some issues that arise during execution, such as how to efficiently deal with replanning when new information is discovered that affects upcoming plans.

We are currently performing further experiments and are in the process of porting the system to a team of 10 ActivMedia Pioneer P2DX robots.

ACKNOWLEDGMENTS

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

for helpful contributions and advice.

References

- [1] B. J. Clement and E. H. Durfee. Top-down search for coordinating the hierarchical plans of multiple agents. In *Proceedings of the Third International Conference on Autonomous Agents (Agents'99)*, pages 252–259, 1999.
- [2] M. B. Dias, D. Goldberg, and A. Stentz. Market-based multirobot coordination for complex space applications. In *The 7th International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS)*, 2003.
- [3] M. B. Dias and A. Stentz. A free market architecture for distributed control of a multirobot system. In *6th International Conference on Intelligent Autonomous Systems (IAS-6)*, pages 115–122, 2000.
- [4] M. B. Dias and A. Stentz. Opportunistic optimization for market-based multirobot control. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2002.
- [5] K. Erol, J. Hendler, and D. S. Nau. Semantics for hierarchical task-network planning. Technical Report CS-TR-3239, University of Maryland College Park, 1994.
- [6] B. P. Gerkey and M. J. Mataric. Sold!: Auction methods for multi-robot control. *IEEE Transactions on Robotics and Automation Special Issue on Multi-Robot Systems*, 18(5):758–768, October 2002.
- [7] L. Hunsberger and B. J. Grosz. A combinatorial auction for collaborative planning. In *Proceedings of the Fourth International Conference on Multi-Agent Systems (ICMAS)*, 2000.
- [8] L. Parker. Adaptive heterogeneous multi-robot teams. *Neurocomputing, special issue of NEURAP '98: Neural Networks and Their Applications*, 28:75–92, 1999.
- [9] G. Rabideau, T. Estlin, S. Chien, and A. Barrett. A comparison of coordinated planning methods for cooperating rovers. In *Proceedings of the AIAA 1999 Space Technology Conference*, 1999.
- [10] M. H. Rothkopf, A. Pekec, and R. M. Harstad. Computationally manageable combinatorial auctions. *Management Science*, 44(8):1131–1147, 1998.
- [11] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135(1-2):1–54, 2002.
- [12] A. Stentz and M. B. Dias. A free market architecture for coordinating multiple robots. Technical Report CMU-RI-TR-99-42, Robotics Institute, Carnegie Mellon University, December 1999.
- [13] S. Thayer, B. Digney, M. B. Dias, A. Stentz, B. Nabbe, and M. Hebert. Distributed robotic mapping of extreme environments. In *Proceedings of SPIE: Mobile Robots XV and Telem manipulator and Telepresence Technologies VII*, 2000.
- [14] R. Zlot, A. Stentz, M. B. Dias, and S. Thayer. Multi-robot exploration controlled by a market economy. In *Proceedings of the International Conference on Robotics and Automation*, 2002.