# Complex Task Allocation For Multiple Robots*

Robert Zlot and Anthony Stentz
*The Robotics Institute*
*Carnegie Mellon University*
*Pittsburgh, PA, 15213 USA*
{*robz* | *axs* }@*ri.cmu.edu*

***Abstract*— Recent research trends and technology developments are bringing us closer to the realization of autonomous multirobot systems performing increasingly complex missions. However, existing multirobot task allocation mechanisms treat tasks as simple, indivisible entities and ignore any inherent structure and semantics that such complex tasks might have. These properties can be exploited to produce more efficient team plans by giving individual robots the ability to come up with new ways to perform a task, or by allowing multiple robots to cooperate by sharing the subcomponents of a task, or both. In this paper, we introduce the *complex task allocation problem* and describe a distributed solution for efficiently allocating a set of complex tasks to a robot team. The advantages of explicitly modeling complex tasks during the allocation process is demonstrated by a comparison of our approach with existing task allocation algorithms in an area reconnaissance scenario. An implementation on a team of outdoor robots further validates our approach.**

***Index Terms*— multirobot coordination, task allocation**

## I. Introduction

Multirobot systems are becoming increasingly more capable and the types of achievable applications for teams of robots are becoming progressively more complex. Many approaches to multirobot coordination rely on a mechanism for task allocation to determine an efficient assignment of tasks to robots; however, existing techniques do not fully consider the complexity of the tasks to be allocated. For the most part, tasks are assumed to be atomic units that can be performed by one or more robots on the team. In practice, this usually means that tasks are either acquired from a central planner that decomposes the mission goals, or that tasks are specified as input by a system user. In any case, existing task allocation algorithms consider the tasks only in terms of the level of description provided by the user or the planner.

In this paper, we address the problem of allocating *complex* tasks to a team of autonomous robots. Complex tasks are tasks that may have many potential solution strategies; finding a plan to achieve a complex task often involves solving an $\mathcal{NP}$-hard problem. In contrast, *simple* tasks can be executed by a robot in a straightforward, prescriptive manner. In particular, we focus here on complex tasks that can be decomposed into multiple inter-related subtasks (which may also be complex). This class of tasks is natural for describing many application domains, including reconnaissance, automated construction, search and rescue, hazardous waste cleanup, and planetary exploration. As an example, the problem of area reconnaissance involves sending a team of scout robots to observe a number of named areas of interest. Each area can be viewed from a set of observation points. Thus the scenario can be modeled as a set of "*area coverage*" tasks, each of which can be decomposed into several "*visit observation point*" subtasks.

The *simple* task allocation problem can be defined as follows: *given a set of tasks, $T$, a set of robots, $R$, and a cost function for each robot $r \in R$ specifying the cost of performing each subset of tasks, $c_r : 2^T \rightarrow \mathbb{R} \cup \{\infty\}$, find the assignment of tasks to robots $\langle T_1, \ldots, T_{|R|} \rangle$ that minimizes a global cost function*[1] *(e.g. $\sum_{r \in R} c_r(T_r)$ or $\max_{r \in R} c_r(T_r)$)*. However, when dealing with complex tasks, this definition does not suffice as the set of tasks that is supplied to the team is not fixed since it may be achieved in many possible ways. While it is possible to first decompose each complex task and then assign the resulting simple tasks, this approach can result in highly suboptimal solutions. The reason for this, essentially, is that there are two problems that must be addressed by the team: *what do we do?*; and *who does what?* That is, it is not possible to know how to efficiently allocate complex tasks among the robots if it is not known how these tasks will be decomposed. Similarly, it is not possible to optimally decompose complex tasks before deciding which robots are to execute them (or their subtasks). In this paper, we address these problems by proposing a distributed algorithm in which robots simultaneously and continuously allocate and decompose complex tasks. Our solution uses a market-based approach in which both efficient task allocations and efficient task decompositions are favored.

In the next section, we review existing approaches to task allocation that can be applied to complex task domains. We then outline our approach in section III. Results follow from an area reconnaissance domain implemented both in simulation and on team of outdoor robots.

## II. Related Work

The majority of multirobot systems that utilize an explicit task allocation mechanism assume either that a static set of tasks is given to the system as input [2], [5], [7], [8], or that tasks arrive dynamically, either from external [5], [7]

*This work was sponsored by the U.S. Army Research Laboratory, under contract **Robotics Collaborative Technology Alliance** (contract number DAAD19-01-2-0012).

[1]Often utility functions are used rather than cost functions, and the problem becomes one of utility maximization.

or internal [11], [15] sources. In any case, such approaches search for an efficient assignment of the current task set to robots, assuming that all tasks are indivisible. When this type of mechanism is applied to complex tasks, a robot assigned a task can decompose it and then execute the resulting simple tasks (*e.g.* [2]). In reality, however, it may be beneficial to allocate subcomponents of these tasks to more than one, and generally the preferred task decomposition will depend on the subtask assignments. Therefore, treating tasks as atomic entities during allocation is not always prudent.

A common alternative among systems that explicitly handle complex tasks is a two-stage approach: first decompose all tasks and then distribute the resulting set of subtasks [1], [4], [12]. The main drawback of this approach is that task decomposition is performed without knowledge of the eventual task allocation; therefore the cost of the final plan cannot be fully considered. Since there is no backtracking, costly mistakes in the central decompositions cannot be rectified. In some instances, the central plan is left intentionally vague, which allows for a limited amount of flexibility in modifying it later. For example, in GOFER [4], the central planner produces a general plan structure for which individual robots can later instantiate some variables; while in the "playbook" system of Simmons *et al.* [12], the planner relies on a central executive to both allocate tasks and to fill in some plan details.

The M+ cooperative task achievement scheme allows some reallocation of subcomponents of complex tasks [3]. Tasks are first allocated using the M+ task allocation mechanism [2], which uses a bidding protocol to distribute predefined abstract tasks among the team. Each robot locally decomposes its tasks into *actions* (which can be viewed as primitive subtasks). The task achievement scheme allows the elimination or transfer of actions to remove some inefficiencies or redundancies in the global plan. While this additional step can potentially improve the solution quality to some degree, the initial task allocation does not consider how the actions are to be shared among the robots, nor does the decomposition step consider the eventual allocation of its resulting actions. Additionally, the task achievement scheme does not explicitly model the costs of the actions that get reassigned or canceled.

## III. Approach

Our approach to the complex task allocation problem builds upon the success of existing market-based multirobot coordination techniques (*e.g.* [2], [5], [7], [8], [11], [15]). By generalizing the definition of a task and developing appropriate mechanisms to handle these new task descriptions, we create a marketplace capable of distributing complex tasks among a robot team in an efficient manner.

### A. Market-based Task Allocation for Multiple Robots

Market-based approaches to multirobot coordination treat a team of robots as participants in a virtual economy. Essentially, robots are contracted to complete required tasks in exchange for payment. Each robot has well-defined cost and revenue functions that can be used to compute the expected gains and losses for performing tasks. Since the participants are self-interested, they work to maximize their individual profits. Costs and revenues are designed so that individual optimizations lead to globally efficient solutions.

Our approach can be considered an extension of *TraderBots* [5]. In *TraderBots*, agents called *traders* – one running on each robot (*RoboTraders*), and possibly others representing human operators (*OpTraders*), computers, sensors, or other resources – participate in a market, trading tasks via auctions. When an auction is announced, participants compute bids based on their expected profit for the tasks on offer, and the robots that can perform the tasks for the best price are awarded the resulting contracts. Since only profitable trades occur, each auction acts to improve the global solution. Each *RoboTrader* maintains a schedule of tasks to which it has committed, and can evaluate new tasks by computing the marginal costs of adding them to its schedule. Traders can take on the roles of auctioneer and bidder dynamically, thus facilitating peer-to-peer trades amongst the team. This implies that tasks can be reallocated, allowing for solution improvements over initial assignments and for adapting the task assignments as new information is ascertained. Having no single auctioneer also avoids the presence of a central agent becoming a critical point of failure for the system.

### B. Task Trees

The types of tasks used by *TraderBots* and other multirobot task allocation mechanisms are treated as atomic units, in that the allocation algorithms only consider one static description for each task – thus the only degree of freedom is determining to which robot to assign it. To handle complex tasks, this description must be broadened.

We use *task trees* to represent complex tasks. A task tree is defined as a rooted set of task nodes connected by directed edges that specify parent-child relationships between the tasks. Each successive level of the tree represents a further refinement of a complex task; the root is the most abstract description, and the lowest level contains primitive tasks that can be executed by a robot or another agent in the system.

The way in which subtasks are related to their parents can vary depending on the application and the degree of coordination desired. In this paper, we look specifically at a subclass of complex tasks: loosely-coupled tasks related through AND and OR logical operators; to satisfy an AND-labeled task, all of its subtasks must be executed, while an OR task is satisfied when at least one of its subtasks is executed. The AND/OR tree shown in figure 1 represents a task decomposition for an area reconnaissance mission.

### C. Complex Task Markets

In order to effectively incorporate complex tasks, multirobot task markets can be extended to include *task tree auctions*. Instead of trading contracts for simple tasks, trees of tasks are offered in auctions. Participants can bid on

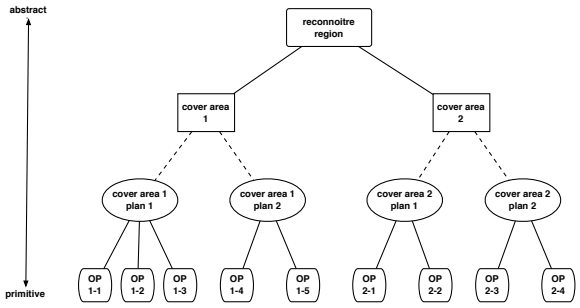



Fig. 1. An example $AND/OR$ task tree for an area reconnaissance scenario. The solid edges represent $AND$ operators, and the dashed lines represent $OR$ operators. Note that for the *cover area* tasks, there are two alternative decompositions specified in the tree.

any combination of nodes in the tree, and the auctioneer can choose to award several nodes from the same tree to multiple winning bidders. The winners of the auction are responsible to the seller and must ensure that the tasks are completed before receiving payment (either by executing the task themselves, or by subcontracting parts of the task to other teammates in future negotiations). Because bids can be on tasks at multiple levels of abstraction, task tree markets have the flexibility to allocate tasks at whichever granularity of abstraction is most appropriate.

One benefit of a task tree market is that task tree structures allow robots to express their valuations for both tasks *and* plans. Since robots have different states, information, resources and capabilities, they may prefer different decompositions for the same task. The calculation of these preferences is a recursive process. In the base case, primitive tasks are handled in much the same way as in simple task markets: bids are based on the expected marginal cost of the task, and awards can be immediately inserted into the schedule of the winner. Abstract tasks are evaluated in two ways. First, the cost of performing the task as offered (*i.e.* the auctioneer's decomposition) is considered. This is essentially the cost of performing a set of the descendant primitive tasks; the chosen set depends on the connectives (*e.g.* the logical operators) present in the subtree. In the second step, the bidder can try to find a better plan for the task: it computes its own decomposition of the task, and, if the resulting plan is of lower cost, it uses this cost for its bid. If the bidder is awarded this task, it can insert the primitive subtasks of whichever plan was more beneficial into its schedule. It may later choose to subcontract some or all of the new plan in future auctions.

### D. An example

Perhaps the best way to illustrate the mechanics of the task tree market is through a simple example. Figure 2 displays a series of auctions for an abstract *cover area* task. At the bottom of each subfigure is a geometric representation of the task. The large square shows the area to be covered, and the labeled points are observation points from which the area can be viewed to achieve the required coverage. The edges between the tasks are labeled with the costs of navigating between the points. At the top of each figure is a task tree representing the current decomposition of the task, labeled with the task prices. Also shown is the global cost of the current solution.

In the example of figure 2, the area coverage task is initially allocated to robot R1. Robot R2 is unable to perform any of the tasks at a lower cost than R1 given the initial decomposition; but, after computing its own plan R2 is able to perform the coverage task at a lower price. R2 therefore wins the task tree, and subsequently subcontracts one of the subtasks to a third robot on the team for a further reduction in cost.

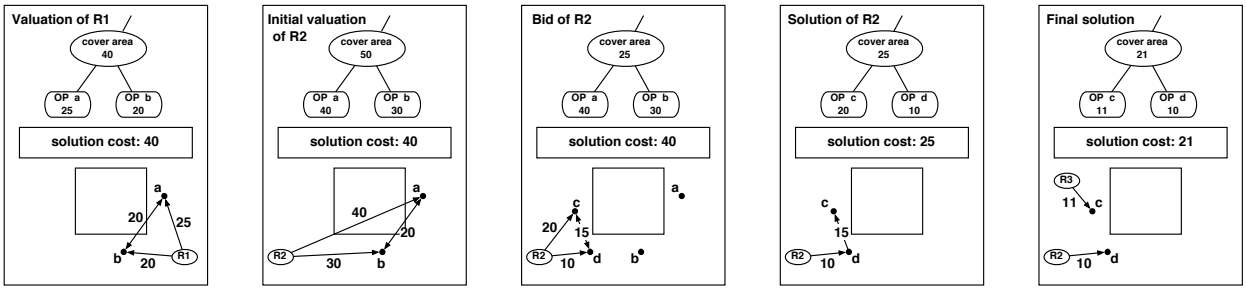
### E. Bidding and Auction Clearing

Auction clearing is the process of determining to which bidders to award which tasks. In a task tree auction, the goal of the auctioneer is to find the allocation which maximizes the auctioneer's profit (minimizing team cost). The winning allocation must satisfy several constraints. Firstly, the tree structure defines which tasks can be awarded in combination. For example, a task node should not be awarded to a bidder in a task tree auction if one of its ancestor nodes has already been assigned. Second, due to the dependency of task costs on existing commitments, only one tree node can be awarded to each bidder.

The bidding language is a specification of which types of bids participants are permitted to submit to an auctioneer. There is inherently a tradeoff between the expressiveness and the simplicity of the bidding language. In the context of task trees, the bidding language can range from allowing the bidders to bid on only one node in the tree, to bidding on any arbitrary set of nodes in the tree. The *one-node* bidding language is very simple, in that it is easy to specify and the auction can be cleared optimally in polynomial time: the auctioneer does not have to consider which bid to award to each bidder – each either wins the one task bid on or not. However this language is not very expressive, as the bidders are filtering out most of their preference information. As a result, the possible solution space is limited and the resulting allocations are inefficient. The *any-nodes* bidding language is far more expressive, but it introduces a more complex auction clearing problem. Clearing becomes hard because although each bid is for multiple nodes in the tree, only one node can be awarded per bidder to ensure the bid prices are valid (once a task is awarded to a particular bidder, its marginal costs for all other tasks are different from what was originally estimated at the time of the bid).

In a previous publication [14], we describe a clearing algorithm for a bidding language in which traders can bid on all nodes along a root-to-leaf path in a task tree (a single path starts at the root, but can branch at OR nodes). This algorithm is described in algorithm 1.

While the bidding language accepted by algorithm 1 is more expressive than the *one-node* bidding language, a great deal of preference information is still being lost by ignoring most (all but $O(\log n)$) of the tree nodes when submitting a bid. Therefore, in order to achieve more




(a) (b) (c) (d) (e)

Fig. 2. Simple example of a task tree auction for an area coverage task. *(a)* R1 holds a task tree auction. The initial plan of R1 is displayed along with R1's reserve price of 40 for the task tree. *(b)* R2's valuation of R1's tree, without replanning is more costly for all tasks in the tree. *(c)* R2 comes up with a different decomposition for the cover area task with a cost of 25, and updates its bid accordingly. *(d)* The auction is cleared. R2 is awarded the abstract area coverage task. R2's new plan is shown along with the associated task tree decomposition. The global solution cost has been reduced from 40 to 25. *(e)* R2 holds another auction round, which results in task $c$ being subcontracted to R3 for a cost of 11 units. The global solution cost has been further reduced to 21.

**Algorithm 1:** Auction clearing algorithm for restricted bids

**Instance –** $T$: a set of nodes in a tree with root $R$;
$p(N)$, $N \in T$: the lowest price bid on each tree node;
Initialize the solution by marking all leaf nodes;
**while** $N_p \neq R$ **do**
  Find $N_{max}$, the maximum depth node in $T$. Let $N_p$ be the parent of $N_{max}$, and $S$ be the set of children of $N_p$;
  **if** $Op(N_p)$ = *AND* **then**
    $p(S) := \sum_{C \in S} p(C)$;
  **else if** $Op(N_p)$ = *OR* **then**
    $p(S) := min_{C \in S} p(C)$;
  **if** $p(N_p) < p(S)$ **then**
    Mark $N_p$, unmark all descendants of $N_p$;
  **else**
    $p(N_p) := p(S)$
  $T := T - S$

efficient solutions, we allow the bidders to bid on the entire tree; for which we require two new task tree auction clearing algorithms.

*1) RoboTrader auctions:* This clearing algorithm is used in auctions held by individual robots. The distinguishing property of these auctions is that any tree nodes that aren't sold remain contracted to the auctioneer. That is, the auctioneer has a reserve price for the tree (the cost to perform the tasks itself). A rough outline of this clearing algorithm is given as algorithm 2.

*2) OpTrader auctions:* This algorithm is used in auctions held by the *OpTrader*, an agent acting on behalf of a system operator. When a user introduces a new complex task for the robots to accomplish, the *OpTrader* decomposes the task and holds an auction to allocate the resulting task tree to the robots. In this case, the agent holding the auction cannot perform the tasks itself, therefore there is no reserve price. A sketch of the clearing algorithm is given as algorithm 3. Note that this algorithm does not necessarily award the entire tree – if some of the tree remains unsold,

**Algorithm 2:** *RoboTrader* auction clearing

**while** *the tree is not satisfied* **do**
  Determine and mark the cost minimizing set of remaining bids that satisfy the remaining parts of the tree, using algorithm 1;
  Sort marked nodes by profit (*i.e.* reserve minus lowest bid). Accept as many of the highest-profit bids as possible, ensuring that no bidder wins multiple nodes;
  Remove the bids of the winners from future consideration. If any of these nodes have marked descendants, unmark them and reintroduce the bids of their previous winners for future consideration;

further auctions are called until the tree is satisfied by the combination of awards.

**Algorithm 3:** *OpTrader* auction clearing

Determine and mark the cost minimizing set of bids that satisfy the remaining part of the tree, using algorithm 1;
For each bidder, award the node for which it has the highest margin of victory (*i.e.* first- minus second-price bids), if any. Do not include a node if an ancestor has also been awarded;

## IV. Experiments

We have tested our approach on an area reconnaissance application. In this scenario, a team of robots is tasked with a reconnaissance mission which involves scouting a number of specified named areas of interest (NAI). To cover each NAI the robots select and navigate to a set of observation points (OP) and view the area with range-limited $360^\circ$ line-of-sight sensors. Because the NAIs may contain enemies, the robots cannot enter them without incurring a large cost. The mission is achieved when the robots visit a sufficient number of OPs to cover a predefined fraction (75%) of each area.

We use a 2.5-dimensional occupancy grid representation of the environment, in which every cell can be marked

as free, occupied, or unknown space. Additionally, each cell is marked with a height, and a benefit gained for viewing it (*i.e.* if it is within an NAI). We use the D* path planning algorithm [13] to compute navigation costs, and we calculate area visibility from observation points using a point-to-point inter-visibility algorithm.

### A. Task Decomposition

To construct a task tree for a problem instance, a mission-level reconnaissance task is decomposed into a set of NAIs, and each NAI in turn is decomposed into a set of OPs. An example of a task tree for an area reconnaissance problem with two NAIs is given in figure 1.

*1) Mission-level decomposition:* A connected components algorithm is used to determine connected areas within the region that have high viewing benefits. Each separate area (NAI) found becomes a child to the mission-level task, and a bounding-box representation is used to describe the area coverage task.

*2) NAI decomposition:* To find the set of OPs from which to view a given NAI, we look at potential OPs[2] and compute the expected revenue (line-of-sight coverage of the NAI from the OP) and the expected cost of traveling to the OP. These quantities are combined as a weighted difference (revenue minus weighted cost), and we repeatedly choose the highest scoring OP until we have sufficient coverage of the area. By modifying the weighting factor, we can come up with alternative decompositions for the same area. With a high weight, we put more emphasis on cost, and the resulting OP set has low cost for a single robot. Using a low weight de-emphasizes navigation cost and results in more spread out, but potentially less, coverage points. These decompositions often favor team plans – the robot performing the decomposition often contracts out some of the goal points to other robots. Our implementation computes one 'individual' decomposition and one 'team' decomposition, which are both placed in the tree as alternative plans under an OR node (as in figure 1). In general, if the team is heterogeneous, robots can come up with several alternative plans that draw on the various capabilities of the different types of robots on the team. NAI tasks that have previously sold or executed OP children can still be decomposed and traded by considering those OPs as being fixed in the decomposition.

### B. Clustering

In addition to the mission and area decomposition steps, a clustering algorithm is also run on the observation points and NAIs. The purpose of this algorithm is to group together any sibling nodes that are physically close together. This can improve the allocation efficiency by allowing some groups of synergistic tasks to be sold together, thus circumventing some local minima. The clustering algorithm works by running a $k$-means on the group of NAIs, and each sibling group of OPs. The value or $k$ is varied between 2 and the number of nodes being considered, and the clustering resulting from the value of $k$ that leads to the largest relative improvement over the previous value is the one chosen. With the addition of task clustering, a task tree for an area reconnaissance mission can have a depth of up to six levels.

[2]Because the computation required to consider a single OP is expensive, we limit the candidate OPs to twelve per NAI, which surround the bounding box with four OP candidates per side.

### C. Costing

The goal of the robot team is to complete the reconnaissance mission while minimizing the total distance traveled. Individually, the robots must solve instances of the traveling salesman path problem (TSPP)[3] when deciding in which order to visit observation points. Robots frequently encounter TSPP-related optimization and cost estimation problems when computing reserve prices for auctions, when bidding, and when reordering schedules after trades. For small problems (at most twelve cities) we compute the optimal solution; while for larger instances we run a $\frac{3}{2}$-approximation algorithm [9], which we then improve by a 2-opt local search. Reserve prices and bids for tasks are computed by differencing the costs of a path with and without the tasks under consideration.

### D. Simulation Experiments

A series of experiments was conducted to evaluate the effectiveness of task tree trading for the area reconnaissance scenario, using a multirobot simulator with a graphical display (figure 3). In each test, a number of robots and NAIs are randomly placed within a 200x200-cell grid containing multiple obstacles. The terrain map was constructed from real-world measurements by an autonomous helicopter equipped with a downward looking scanning laser rangefinder[4]. The NAIs are non-overlapping, randomly-sized rectangles with edge lengths drawn uniformly at random in the range of 15 to 30 grid cells.

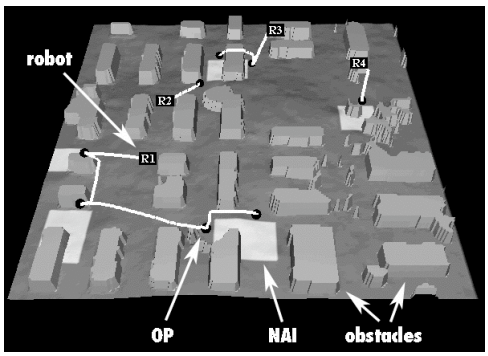



Fig. 3. Screenshot from the multirobot simulator. Here there are four robots tasked with a reconnaissance mission requiring the coverage of five areas. Also pictured are the paths that the robots plan to traverse to visit the OPs.

Our experiments examine the benefit of explicitly handling complex task allocation by comparing task tree

[3]The traveling salesman path problem is similar to the traveling salesman problem, except the salesman is required to follow a path, rather than a tour (*i.e.* the salesman does not have to return to the starting city).

[4]The data is courtesy of Omead Amidi and Ryan Miller.

auctions against "single-level" task auctions, meant to represent existing task allocation algorithms that model tasks as atomic entities and do not consider the structure or complexity of the tasks. In addition to task tree auctions, we look at three particular types of single-level allocation mechanisms.

***Goal point-level allocation:*** Here the mission is initially decomposed by the *OpTrader* into a set of observation points. All auctions that follow are for tasks in this set of goal points only (*i.e.* there is no notion of the NAIs; no further decompositions occur).

***Area-level allocation:*** The *OpTrader* decomposes the mission into a list of NAIs. Subsequent auctions are for these area coverage tasks only. When a *RoboTrader* bids on a task, it can compute its own decomposition but it can never reallocate any of the subtasks (thus there can be no cooperation between robots in covering a single NAI).

***Mission-level allocation:*** The mission is not pre-decomposed: the auctions are for the mission task as a whole. *RoboTraders* can decompose the mission locally, but cannot reallocate subcomponents. This means that in any solution only one robot is executing the mission.

***Task tree allocation:*** Tasks are represented as task trees, and are traded at multiple levels of abstraction. Task tree auctions are expected to outperform area (and mission-level) auctions because the use of task tree auctions makes it possible for traders to share coverage of NAIs when such an allocation is beneficial. In addition, task tree auctions are expected to be superior to goal point auctions because they permit the traders to redecompose the complex tasks and come up with more efficient plans.

We consider also two general types of task allocation mechanisms: centralized, and distributed. In centralized allocation, a single agent is responsible for assigning the tasks to the robots. With distributed allocation algorithms, the robots have an initial allocation (possibly coming from an centralized allocation phase) and can then reallocate tasks by engaging in peer-to-peer negotiations. In our experiments we consider both of these settings.

***Centralized auctions:*** The *OpTrader* holds a series of auctions to allocate the tasks to the robots. In the case of task tree auctions, algorithm 3 of section III-E is used. For single-level auctions, we use a greedy clearing algorithm, similar to one used in *TraderBots*: the robots send bids for all available tasks to the *OpTrader*; the *OpTrader* then greedily selects the best bids and awards those tasks to the robots (one task per robot); repeat until all tasks have been awarded.

***Distributed (peer-to-peer) auctions:*** The initial allocation is obtained via a centralized auction (as described above). The *RoboTraders* then hold further auctions in rounds, in which each trader, sequentially and in random order, calls and clears one auction. Rounds are held repeatedly until a stable solution is reached (*i.e.* no more awards are being given out). For peer-to-peer task tree auctions, each robot randomly chooses one of the trees to which it has committed for auction, and algorithm 2 is used to clear. For single-level auctions, each *RoboTrader* offers all of its available tasks; the other robots submit bids and a single award is given out for the one task that results in the greatest profit for the auctioneer.

The majority of existing multirobot task allocation algorithms fall roughly into the single-level centralized allocation category [2], [4], [7], [11], [12], although a small number of approaches use some form of single-level distributed allocation (*e.g.* [8], [10]). *TraderBots* [5], and, to some extent, M+ [2], [3] include both single-level centralized and single-level distributed task allocation.

### *E. Results*

For each type of allocation mechanism (centralized, distributed) we ran the three single-level allocation algorithms and compared the solutions with that produced by the task tree auction mechanism. To determine how the algorithms are affected by problem complexity, in a first set of experiments we vary the number of areas (from 1 to 10), and in a second set we vary the number of robots (between 2 and 10).

The quality of a solution is quantified as the total distance traveled by the team. In order to evaluate each single-level approach for a given problem instance, we look at the ratio of the solution cost of that approach to the task tree solution cost (*i.e.* a result greater than one indicates that the task tree solution is superior). Results are shown in figures 4-7. In the plots, each data point is averaged over 40 trials, and 95% confidence intervals are displayed as error bars.

From the results in figures 4-7, we can see that the task tree allocation mechanism outperforms all of the single-level task allocation algorithms. Furthermore, in most cases, adding peer-to-peer auctions following the centralized auctions results in even greater improvement of the task tree algorithm over the others. Having the flexibility both to replan and to cooperate on complex tasks gives the task tree algorithm an advantage over all of the single-level approaches.

As expected, in the peer-to-peer allocation scenarios, the mission-level allocation scheme produces the worst results. This is due to the fact that the mission-level algorithm assigns the entire mission to one robot, and does not take advantage of the presence of multiple robots. Perhaps surprisingly, in the centralized auctions, mission-level allocation is often better than allocations at other levels; another general trend in the centralized algorithm results, is that area-level allocations always perform better than goal point-level allocations. Both of these effects demonstrate a weakness of greedy single-task allocation algorithms that are typically used in practice – that is, because each robot is awarded one task in every auction round (excepting the last), bad allocations occur when some of those assignments would have been better-suited for other robots. This is more pronounced when there are more tasks to assign, as is the case with the more primitive-level allocations. As evidenced in figures 6 and 7, peer-to-peer

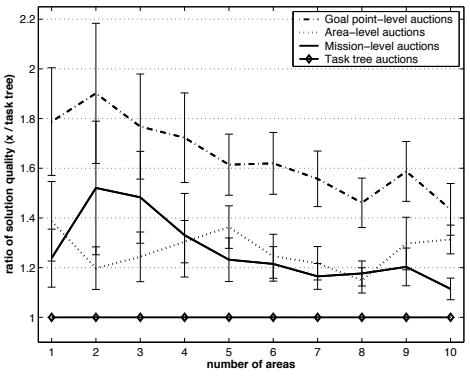



Fig. 4. A comparison of solution quality of task tree auctions vs. single-level auctions varying the number of areas. The number of robots is held fixed at five. All auctions are centralized (*OpTrader*) auctions.

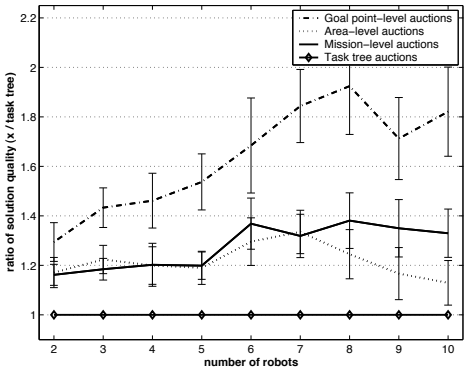



Fig. 5. A comparison of solution quality of task tree auctions vs. single-level auctions varying the number of robots. The number of areas is held fixed at five. All auctions are centralized (*OpTrader*) auctions.

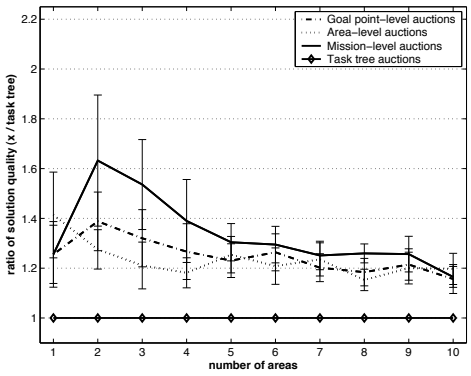



Fig. 6. A comparison of solution quality of task tree auctions vs. single-level auctions varying the number of areas. The number of robots is held fixed at five. All auctions are centralized (*OpTrader*) auctions followed by reallocation through peer-to-peer (*RoboTrader*) auctions.

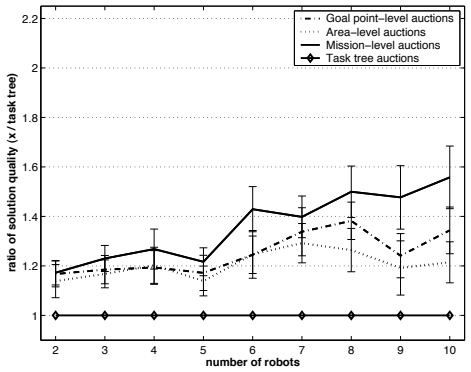



Fig. 7. A comparison of solution quality of task tree auctions vs. single-level auctions varying the number of robots. The number of areas is held fixed at five. All auctions are centralized (*OpTrader*) auctions followed by reallocation through peer-to-peer (*RoboTrader*) auctions.

reallocations (as used in *TraderBots*) tend to repair many of these bad allocations.

In the above experiments, we also measure the computation time required for each type of auction. The experiments were all run on a 733 MHz Pentium®III processor with 256 MB of memory. Since all of the trader agents ran sequentially, rather than as separate processes on separate machines, auction times are estimated: for any section that would have occurred in parallel (*e.g.* bidding), we use the maximum time taken by any robot. When auctions occur in rounds, the time reported is the time taken to reach a stable local minimum solution (*i.e.* once all trading ceases). Table I gives a subset of the computation time results for the previously described experiments. The data shows that goal point- and mission-level auctions are typically faster than task tree and area-level auctions, mainly due to the fact that less task decompositions occur. To put the time values in perspective, we also estimate the time required for the robots to execute the solutions, assuming that the robots travel at $1m/s$ (a typical speed for our outdoor robots) and that the world size is $1km^2$ (a reasonable size for an area reconnaissance mission). The execution time that we use is the makespan – the maximum time it takes any one robot to complete its part of the mission. When the computation times are compared to the execution times, we find that typically the computation times are on the order of a tenth of a percent of the execution times for moderate-sized problems, and in the most complex problems that we run, about one percent. Additionally, these numbers can be further reduced since trading can occur during execution.

TABLE I

COMPUTATION TIMES FOR A SUBSET OF THE EXPERIMENTS, IN SECONDS. TT, G, A, AND M STAND FOR *task tree*, *goal point-*, *area-*, AND *mission*-LEVEL ALLOCATION RESPECTIVELY. THE SUBSCRIPTS $c$ AND $d$ REPRESENT *centralized* AND *distributed* AUCTIONS.

| Robots | Areas | $TT_c$ | $G_c$ | $A_c$ | $TT_d$ | $G_d$ | $A_d$ | M |
|---|---|---|---|---|---|---|---|---|
| 5 | 5 | 7.6 | 1.8 | 2.3 | 11.0 | 2.5 | 4.0 | 2.6 |
| 10 | 5 | 6.7 | 1.6 | 2.5 | 12.7 | 2.6 | 5.6 | 2.8 |
| 5 | 10 | 21.3 | 12.5 | 13.2 | 32.5 | 15.3 | 29.6 | 5.1 |

### F. Robot experiments

An implementation on a robot platform was also tested on a team of autonomous E-Gators. The E-Gators are outdoor electric utility vehicles manufactured by John Deere, which have been fitted with computing and sensing including a tilting laser range scanner, GPS, and gyroscopes. The software architecture is similar to one we have

used on a team of Pioneer robots [6], and consists of a *RoboTrader* process that communicates with lower-level processes responsible for executing tasks, robot control, inter-robot communications, and data management and mapping. Two modified E-Gators are shown in figure 8.

Fig. 8. The autonomous E-Gator platform.

Figure 9 is a map created by a two E-Gator team tasked with an area reconnaissance mission consisting of three NAIs. The experiments are carried out on a grassy field with some sparse obstacles, such as trees and trash bins. The robots initially acquire tasks by task tree-trading with the *OpTrader*, and subsequently hold peer-to-peer task tree auctions with one another. In the final allocation, one robot handles two NAIs and the other one NAI, using their own task decompositions to determine a set of OPs for each area.

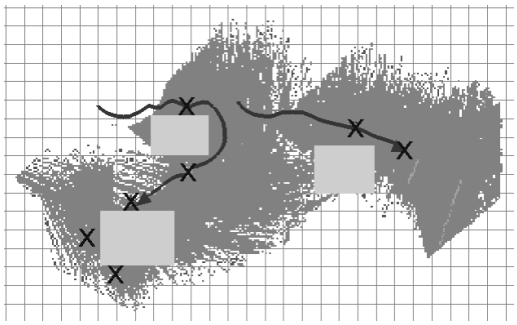


Fig. 9. Snapshot of a map created by two E-Gators on an area reconnaissance mission. Three NAIs are marked as lightly shaded rectangles, and seven OPs are shown as black X's. The dark-shaded region is space mapped out by the laser rangefinders. Robot paths are also displayed. Grid cells are $2m$x$2m$.

## V. Conclusions

In this paper, we identify a previously unexplored problem in multirobot task allocation: that is, the allocation of complex tasks. To address this problem, we introduce a market-based solution that uses novel task tree auctions. Empirical evidence from an area reconnaissance scenario demonstrates that our approach improves upon what we call "single-level allocation" algorithms, which represent the current state of the art in multirobot complex task allocation. An implementation on a team of outdoor robots verifies the feasibility of our approach.

In the future, we look to expand the existing task tree market in several ways, including: how to decide which tasks should be decomposed in the case of computationally costly decomposition algorithms; when to decompose tasks already being executed when working under uncertainty or dynamic conditions; and optimizing other solution metrics, such as makespan. Additionally, we would like to compare our approach to other single-level task allocation mechanisms, such as combinatorial auctions.

## Acknowledgements

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 authors would like to thank Bernardine Dias, Marc Zinck, Juan Pablo Gonzalez, Boris Sofman, Joseph Carsten, and Herman Herman for their important contributions in developing the robot hardware and software, and assisting with field testing.

## References

[1] R. Aylett and D. Barnes. A multi-robot architecture for planetary rovers. In *Proceedings of the 5th ESA Workshop on Advanced Space Technologies for Robotics and Automation*, 1998.

[2] S. S. C. Botelho and R. Alami. M+: A scheme for multi-robot cooperation through negotiated task allocation and achievement. In *Proceedings of the International Conference on Robotics and Automation*, 1999.

[3] S. S. C. Botelho and R. Alami. Robots that cooperatively enhance their plans. In *Proceedings of the 5th International Symposium on Distributed Autonomous Robotic Systems (DARS)*, 2000.

[4] P. Caloud, W. Choi, J.-C. Latombe, C. L. Pape, and M. Yim. Indoor automation with many mobile robots. In *Proceedings of the International Workshop on Intelligent Robotics and Systems (IROS)*, 1990.

[5] M. B. Dias. *TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments*. PhD thesis, Robotics Institute, Carnegie Mellon University, January 2004.

[6] M. B. Dias, R. Zlot, M. Zinck, J. P. Gonzalez, and A. Stentz. A versatile implementation of the *TraderBots* approach to multirobot coordination. In *the 8th International Conference on Intelligent Autonomous Systems (IAS-8)*, 2004.

[7] B. P. Gerkey and M. J. Matarić. Sold!: Auction methods for multi-robot control. *IEEE Transactions on Robotics and Automation Special Issue on Multi-Robot Systems*, 18(5), 2002.

[8] M. Golfarelli, D. Maio, and S. Rizzi. A task-swap negotiation protocol based on the contract net paradigm. Technical Report 005-97, CSITE (Research Center for Informatics and Telecommunication Systems), University of Bologna, 1997.

[9] J. A. Hoogeveen. Analysis of christofides' heuristic: Some paths are more difficult than cycles. *Operations Research Letters*, 10, 1991.

[10] T. Sandholm. An implementation of the contract net protocol based on marginal cost calculations. In *Proceedings of the 12th International Workshop on Distributed Artificial Intelligence*, 1993.

[11] R. Simmons, D. Apfelbaum, W. Burgard, D. Fox, S. Thrun, and H. Younes. Coordination for multi-robot exploration and mapping. In *Proceedings of the National Conference on Artificial Intelligence*, 2000.

[12] R. Simmons, D. Apfelbaum, D. Fox, R. P. Goldman, K. Z. Haigh, D. J. Musliner, M. Pelican, and S. Thrun. Coordinated deployment of multiple heterogeneous robots. In *Proceedings of the Conference on Intelligent Robotics and Systems (IROS)*, 2000.

[13] A. Stentz. Optimal and efficient path planning for partially-known environments. In *Proceedings of the International Conference on Robotics and Automation*, volume 4. IEEE, May 1994.

[14] R. Zlot and A. Stentz. Multirobot control using task abstraction in a market framework. In *Collaborative Technology Alliances Conference*, 2003.

[15] 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.