

# Using A\* for Simultaneous Allocation of Multi-Robot Tasks

Fang Tang and Spondon Saha  
Computer Science Department  
California State Polytechnic University, Pomona  
Pomona, CA, 91768  
Email: {ftang|ssaha}@csupomona.edu

**Abstract**—Task allocation is the problem of determining an efficient mapping between robots and tasks. Usually, a complex team-level task can be decomposed into a task tree or a series of subtasks with time constraints, such that each subtask can either be accomplished by a single robot with a weakly-cooperative solution or by a coalition of multiple robots with a strongly-cooperative solution. Our previously developed ASyMTRe architecture is able to autonomously generate task solutions for subtasks by forming corresponding coalitions (assuming a coalition can also include only one robot) depending on the specific task requirement and the current capabilities of the robot team members. We have extended the ASyMTRe architecture and developed an allocation algorithm that simultaneously maps multiple subtasks to robot coalitions. To maximize the overall efficiency of the system, one must compare different solutions based on various factors such as the task completion time, the solution quality and robot costs. However, this winner determination problem is NP-hard, which is equivalent to a Set Partitioning Problem. We propose an A\* algorithm to achieve the efficient allocation, prove its optimality theoretically, and validate the results empirically.

## I. INTRODUCTION

The multi-robot task allocation taxonomy described in [8] divides the task allocation problem into different categories: (1) based on the characteristics of task solutions: single-robot (SR) vs. multi-robot (MR) tasks; and (2) based on the number of tasks being considered for allocation: instantaneous (IA) vs. time-extended (TA) assignment. A single-robot task is the one that can be accomplished independently by a single robot. We call these task solutions *weakly-cooperative*. A multi-robot task typically requires a strongly cooperative solution [4], meaning that the task is not trivially serializable, and cannot be decomposed into subtasks that can be completed by individual robots operating independently. Thus this type of task requires multiple robots to act in concert as in a *coalition* to achieve the task objective. The instantaneous assignment approach considers only one task at a time without planning for future allocation of tasks. The time-extended assignment approach considers multiple tasks concurrently, which means that the allocation process involves task scheduling or planning. This paper addresses the problem of simultaneous allocation of tasks that require either strongly-cooperative or weakly-cooperative solutions, which falls into the MR-TA category that subsumes the SR-TA problems.

To motivate the need for the combination of strongly-cooperative coalitions with task allocation for weakly-

cooperative tasks, let us consider a site clearing task that requires a specific area to be cleared of obstacles. The objective of the application is to clear the site in as little time as possible while minimizing the cost to the robots (e.g., energy consumption or computational requirements). For the purposes of this discussion, we assume that a map is available to enable the robot team to determine the positions of the obstacles in the area. We further assume that a partial-order planner exists to determine the ordering constraints of removing the obstacles, in case certain obstacles need to be removed before other obstacles can be cleared. For example, one obstacle may block the path to another obstacle. Thus, this team-level task can be decomposed into a series of subtasks with ordering constraints. Each subtask is aimed at removing one obstacle from the site. Since only some subtasks have ordering constraints, the system can allocate a subset of the subtasks to the robots for concurrent execution. Obstacles have different weights or sizes, thus a subtask of removing one obstacle may require a varying number of robots to form coalitions to accomplish it in a strongly cooperative manner. Additionally, when multiple coalitions are available, the system must determine which coalitions are the winning coalitions to the current set of subtasks to maximize efficiency of the team as a whole. This is a typical problem of simultaneous allocation of multi-robot tasks, which is equivalent to a Set Partitioning Problem, where we need to partition the robot team into non-overlapping subteams such that each subteam can accomplish a subtask, with the task set being accomplished in an optimal manner.

In prior work [18], Tang and Parker developed an approach that layers our ASyMTRe-D coalition-formation algorithm for autonomous task solutions with a traditional auction-based mechanism for achieving the sequential allocation of multiple weakly-cooperative tasks. Note that from our perspective, an individual task cannot technically be categorized in advance as a MR- or a SR-task. Instead, whether or not the task requires single or multiple robots depends upon the capabilities of the robot team members. Some robots may be able to perform a given task on their own (thus making the task a SR-task), while other robots may require help from teammates to accomplish that same task (thus making that same task a MR-task). Our ASyMTRe-D approach for generating strongly-coupled task solutions is able to find combinations of robot capabilities that can accomplish the task in either the SR- or MR-case, depending upon the team

capabilities. It is challenging to predefine solutions for the team when the team capabilities are unknown at design time or they change during the task execution. However, the above approach also falls into the MR-IA category.

Our most recent work [19] developed an approach for simultaneous allocation with a winner determination algorithm using a breadth-first search. In this continuing work, our objective is to present an improvement of the auction-based mechanism to allow for simultaneous allocation of multiple tasks (these tasks can be either SR- or MR-tasks) using an A\* search algorithm and thus greatly enhances the allocation efficiency. When multiple tasks are considered, combinatorial auction ([15], [2]) is often used for the robots to express their desirability on accomplishing combination of tasks. However, these tasks are often SR-tasks and each robot can win more than one task at a time, which is fundamentally different from our problem, where each coalition can win only one SR- or MR-task at a time. In other words, our problem is equivalent to a combinatorial auction problem where only task bundles with size one are considered [6]. The winner determination process however, is more complicated since each winner is a coalition (subset) of robots instead of a single robot. As a result, our approach needs to guarantee that the winner coalitions do not share the same team members. The proposed A\* search not only saves the computation time for simultaneous allocation, but also guarantees the optimality of the final assignment. By combining the benefits of coalition formation and the improved time-extended task allocation strategy using A\*, the resulting approach provides a flexible mechanism for generating both strongly-cooperative and weakly-cooperative solution strategies.

The remainder of the paper is organized as follows. Section II describes our approach in details, with a focus on the A\* algorithm for simultaneous task allocation. Experiment validation of the approach is discussed in Section III. Section IV describes related work. We briefly provide concluding remarks in Section V.

## II. THE APPROACH

The main idea of our approach to task allocation is illustrated in Figure 1. We assume that there exists a partial order planner that finds a set of subtasks  $T^i$  from the team-level task  $T$  according to the partial order plan. Here,  $T^i$  represents the set of subtasks  $\{t_1, t_2, \dots, t_k\}$  that satisfies the ordering constraints and the precondition requirement, and thus can be allocated concurrently. At the low level, coalitions  $\{C_1, \dots, C_m\}$  from the team of robots  $\{R_1, \dots, R_n\}$  are formed by ASyMTRe to address the given set of subtasks  $T^i$ . Each coalition  $C_i$  may include a varying number of robots. These coalitions are not mutually exclusive and may share the same team members, or could even be identical coalitions. The coalitions then compete for the assignment of subtasks using an auction-based task allocation approach with our proposed A\* algorithm for winner determination. The goal of the winner determination step is to guarantee that: (1) the overall task contribution value for  $T^i$  is maximized (we explain the task contribution later in this section)

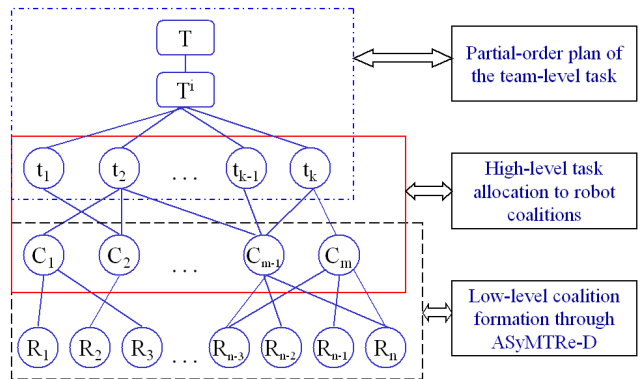


Fig. 1. The relationships between tasks, coalitions and robots.

and (2) each coalition can win at most one subtask  $t_i$  with no winning coalitions sharing the same team member(s). Once the robots start executing the assigned task, they will inform their status and become available for competition for the next round of tasks. The complete formalism of the problem can be found at [19].

### A. Layering Coalition Formation with Auction-Based Task Allocation

To better understand the integrated system, we now describe our previous work on coalition formation, called ASyMTRe-D [14]. The ASyMTRe-D approach has been developed for addressing the formation of heterogeneous robot coalitions that solve a single SR- or MR-task. We share the same motivation behind coalition formation as mentioned in [16]; that is, robots in a coalition should work together to share resources and cooperate on task execution due to their decision that they would benefit more from working together as a coalition than they would from working individually.

The fundamental idea of ASyMTRe-D is to change the abstraction that is used to represent robot competences from the typical “task” abstraction to a biologically-inspired “schema” abstraction and provide a mechanism for the automatic reconfiguration of these schemas to address the multi-robot task at hand. To achieve this, we view robot capabilities as a set of environmental sensors that are available for the robot to use, as well as a set of perceptual schemas, motor schemas, and communication schemas that are pre-programmed into the robot at design time. The ASyMTRe-D approach extends prior work on schema theory [1], [12] by autonomously connecting schemas at run time instead of using pre-defined connections. According to the information invariants theory [7], the information needed to activate a certain schema or to accomplish a task remains the same regardless of the way that the robot may obtain or generate it. We can therefore label inputs and outputs of all schemas with a set of information types, such as *laser range data*, *self global positioning data*, etc. Two schemas can be connected if their input and output information labels match. Thus, schemas can be autonomously connected within or across robots based upon the flow of information required to accomplish a task. With the run time connection capabilities, task solutions can



TABLE II

THE A\* SEARCH ALGORITHM FOR WINNER DETERMINATION.

$$h(n) = \sum_{R_i \notin n.bid} Contribute(R_i) \quad (1)$$

$$Contribute(R_i) = \max_{R_i \in t_j.bid_k.robots} \frac{t_j.bid_k.val}{t_j.bid_k.size} \quad (2)$$

Where  $n.bid$  is the set of robots that appear on the current path of  $n$  and  $Contribute(R_i)$  represents the maximum potential contribution of a robot  $R_i$  among all unallocated tasks. To calculate  $Contribute(R_i)$ , we first find the set of unallocated tasks ( $\{\forall_j | t_j \text{ is unallocated on the current path of } n.\}$ ) that robot  $R_i$  participates as a coalition member, and then compute a set of potential contribution values by dividing the bid value ( $t_j.bid_k.val$ ) by the number of robots in that coalition ( $t_j.bid_k.size$ ). The highest value in the set represents the maximum potential contribution for a robot  $R_i$ . This heuristic will never underestimate the legitimate contribution a robot could make on the current partition and thus it is an admissible heuristic for this search problem. The pseudocode in Table II shows how we can apply the A\* to the winner determination problem.

### III. EXPERIMENTS

#### A. Simulation Setup

All tests were conducted on a Dell workstation equipped with a Pentium 2.66 GHz Dual-core processor and 2GB of RAM. The A\* algorithm implementation is in Python and tests were conducted on the Windows XP SP2 operating system. For data creation (bid generation) the following three distributions were used:

- *Random*: This distribution accepts bids of any size up to the maximum allowed coalition size, i.e. 10 robots. So, coalitions ranging from size 1 to size 10 placed bids on 20, 40 and 60 tasks and their respective execution times were obtained using the A\* algorithm.
- *Uniform*: This distribution accepts bids of a fixed size, in our case, 3, 6 and 9 were tested. So bids that were submitted by the above fixed size coalitions were placed against 20, 40 and 60 tasks and their respective execution times were collected using the A\* algorithm.
- *Bounded*: This distribution allows bids from coalitions whose size fell within certain bounds. For example, for a bounded distribution of 4-6, bids coming from coalitions whose sizes ranged from 4 to 6 were tested. For different bounds, different bids were collected and their respective execution times were obtained for 20, 40 and 60 tasks using the A\* algorithm.

In all of our data collection, the execution times recorded were for only one iteration of the A\* algorithm. If there were leftover or unaddressed tasks from the first iteration, we can reinsert the tasks back to the task queue for the next round of allocation.

Similar to our previous work [19], we did a pre-processing step of conditioning the input data which would result in minimizing the number of nodes created in the tree. The idea is to first sort the input data (bids submitted) with respect to

---

#### Algorithm A\* for Winner Determination

**Input:** A set of bids for a set of tasks

**Output:** A set of winning coalitions that maximizes the sum of contributions

- 1)  $R = \text{CreateNode}(0, \{\}, 0)$  //a dummy root
  - 2)  $Q = \{R\}$  //a queue of unexpanded nodes
  - 3)  $n = \text{pop}(Q)$
  - 4) WHILE (True)
    - a)  $children = \text{FindChildren}(n)$
    - b) IF ( $children$  is empty) RETURN  $n.bid$
    - c) Compute f-value  $f(n) = g(n) + h(n)$  for each child of  $n$
    - d) Push all children nodes to  $Q$
    - e) Sort  $Q$  in a decreasing order of f-values
    - f)  $n = \text{pop}(Q)$
- 

#### Algorithm FindChildren( $n$ )

**Input:** A node  $n$ , where

- $n.depth$  is the depth of  $n$ ; any bid for a task  $t_d$  is always inserted at depth  $d$
- $n.bid$  is the set of robots that appear on the path from the root to the node  $n$
- $n.gval$  is the cumulative path contribution value from the root to the node  $n$

**Output:** A set of children nodes of  $n$

- 1)  $d = n.depth$
  - 2) IF ( $d == \text{numOfTasks}$ ) RETURN  $\{\}$
  - 3)  $m = \text{CreateNode}(d+1, n.bid, n.gval)$  //a dummy child
  - 4) Insert  $m$  into  $children$
  - 5) FOR each  $bid$  submitted for task  $t_d$ 
    - a) IF ( $bid.robots \cap n.bid$  is empty)
      - i)  $m = \text{CreateNode}(d+1, n.bid \cup bid.robots, n.gval+bid.val)$
      - ii) Insert  $m$  into  $children$
    - b) RETURN  $children$
- 

each task that was addressed in the auction - each task would have its own collection of bids. Next, sort the tasks depending on the size of their bid collection in increasing/ascending order. So the task with the least number of bids would be sitting in front of the list of our data input set and the task with the maximum number of bids would be sitting at the end of our data input set. The objective of such a step is to minimize the number of dummy nodes created, without affecting solution quality. For an in depth explanation with pictures, please refer to our previous paper.

#### B. Results and Discussion

There are two suites of tests that we conducted to evaluate the A\* algorithm. The first suite aims at evaluating the performance gain that was achieved from using the A\* algorithm against the BFS algorithm [19] with respect to the execution time. The objective was to see how quickly the A\* approach generated solutions with an increasing number of bids and tasks, without affecting the solution quality.

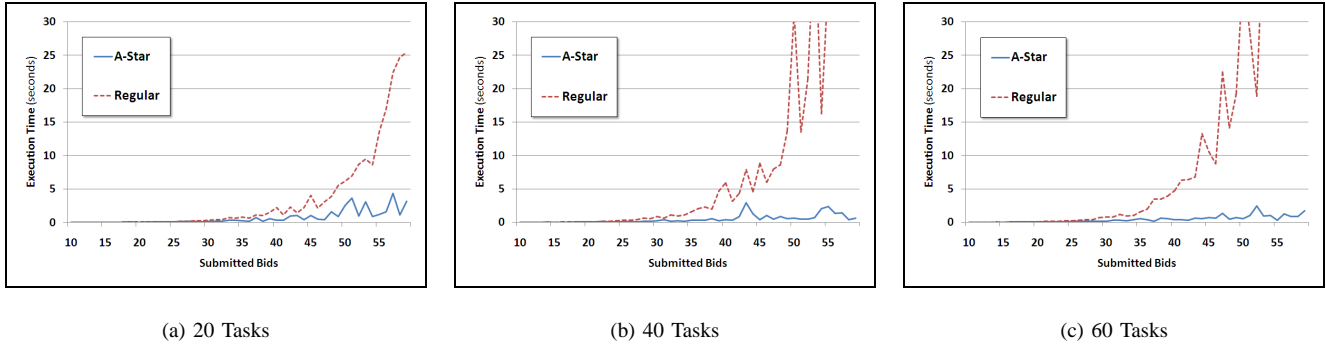


Fig. 3. A\* and Regular BFS comparison.

The second approach aims at measuring the effectiveness of the A\* algorithm alone. This approach therefore utilizes the Uniform and Bounded data distributions for isolating any performance quirks that would otherwise be hidden in a standard execution of the A\* algorithm.

The first suite of tests resulted in the following graphs displayed in Figure 3. Each graph corresponds to the number of tasks that were used (20, 40 and 60 tasks) and plots for an increasing number of bids and their corresponding execution times. The Random distribution was used to generate the data and create these plots. For each of these graphs, the dashed (red) line is the execution times for the regular algorithm, while the solid (blue) line is the execution times of the A\* algorithm. As can be observed, the A\* algorithm clearly outperforms the Regular BFS algorithm in all 3 cases. As the number of tasks increase, the Regular BFS algorithm's rate of climb in execution time increases exponentially while the A\* algorithm shows to have a linear increase. Even for 60 tasks, the A\* approach performs extremely well for increasing bids: maximum recorded execution time of A\* is approximately 3 seconds, while the Regular peaks at more than 30 seconds, for approximately 59 submitted bids. When the bid size is less than 25 - both algorithms perform at about the same rate. Performance boosts could therefore be introduced into the A\* approach if we decide to use a parallel approach in the future. But overall, we have proved with these three graphs that the A\* approach is a functional improvement over the Regular approach introduced in our previous paper.

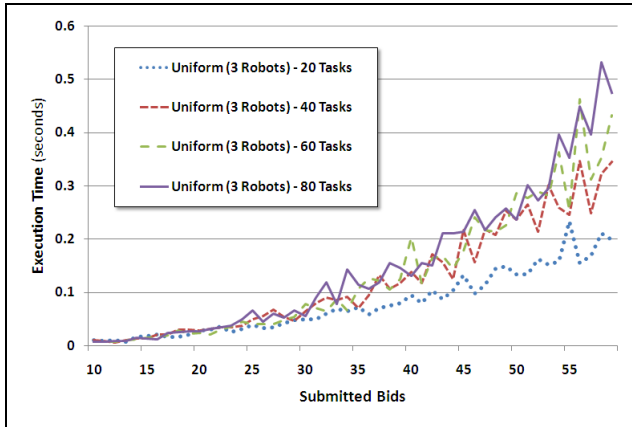
The second suite of tests aims at evaluating the A\* algorithm for different data distributions and observing any performance quirks that may help us evaluate the design of the A\* approach. Figure 4 displays the results of using the Uniform and the Bounded data distributions. We have excluded the results obtained from the Random distribution because it did not help us generate any observable trend that could help us evaluate the design of the A\* algorithm. As a result, our observations were mostly centered on the Uniform and Bounded distributions. For each of the 6 graphs, we plotted the execution times for an increasing number of bids and increasing task sizes of 20, 40, 60 and 80 tasks. These 4 separate plots can be seen as different colored lines with their

individual dots, dashes and solid line textures. As expected of each graph, for increasing task size, the execution time scales up slightly.

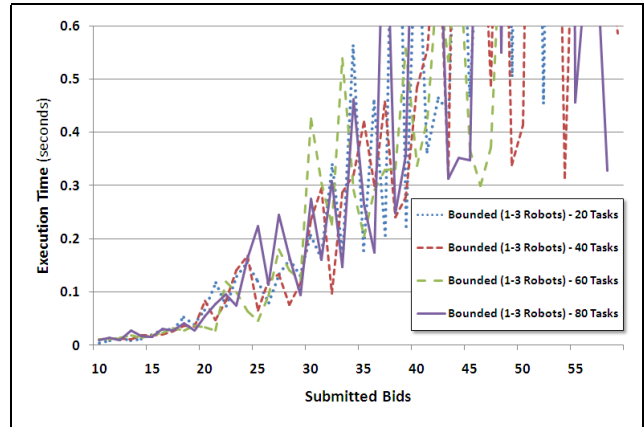
For the Uniform data distribution, Figures 4(a), 4(c) and 4(e) are the corresponding graphs. For Figure 4(a), only bids coming from coalitions of size 3 were considered and used for generating the solution tree. Figure 4(c) and 4(e) correspond to coalition size of 6 and 9 respectively.

For Figure 4(a), notice that the execution time is longer than its 2 counterpart graphs: Figures 4(c) and 4(e). This trend can be attributed to the following reason: solution/bid trees consisting of coalitions of size 3 (Uniform-3) are much denser than solution trees containing coalitions of size 6 (Uniform-6) and 9 (Uniform-9), since we have less number of robots populating a certain path from each coalition (smaller coalitions result in more nodes being added to a certain path, bigger coalitions exhaust the path and prevents other coalitions from being added to the path). If more robots were being represented by a coalition on a certain path, as in the case of Uniform-6 and Uniform-9, then we would have sparsely populated trees resulting in a lesser execution time. Mathematically, the execution time can then be expressed as being inversely proportional to the coalition size. Therefore, we can positively confirm that the greater execution time is a result of denser trees created in Uniform-3 (smaller coalition size), compared to the lesser execution times created as a result of the sparsely populated trees in Uniform-6 and Uniform-9 (larger coalition sizes).

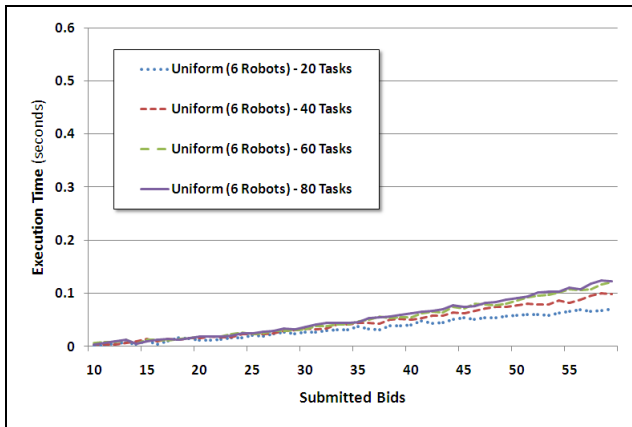
Another observation that can be made regarding Uniform-3 (Figure 4(a)) is the fluctuations in the execution time, compared to its counterparts Uniform-6 (Figure 4(c)) and Uniform-9 (Figure 4(e)). These fluctuations can be attributed to the following reason: the random data introduces a certain degree of unpredictability that affects the execution time. Depending on the data, the solution can be generated faster or slower. We will call this the "degree of randomness" in the data generation. Now, as previously discussed, since coalition size also affects the execution time of the algorithm (coalition-size is inversely proportional to the execution time), the "degree of randomness" in the input data factors into this trend, thereby affecting the fluctuations.



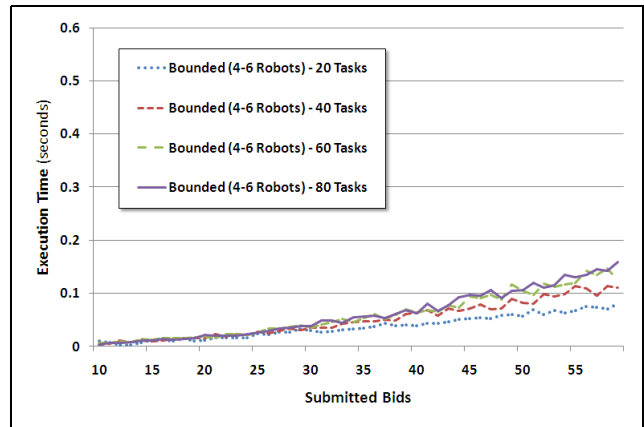
(a) Uniform Distribution - 3 Robots (Uniform-3)



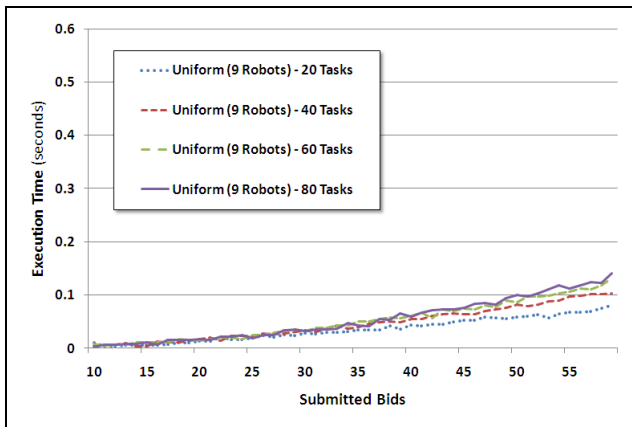
(b) Bounded Distribution - 1 to 3 Robots (Bounded-1-3)



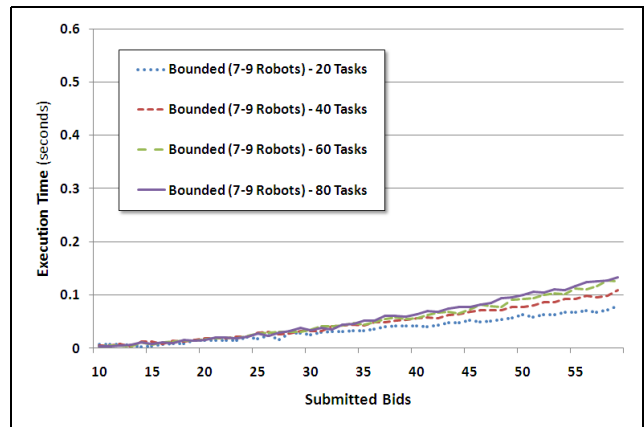
(c) Uniform Distribution - 6 Robots (Uniform-6)



(d) Bounded Distribution - 4 to 6 Robots (Bounded-4-6)



(e) Uniform Distribution - 9 Robots (Uniform-9)



(f) Bounded Distribution - 7 to 9 Robots (Bounded-7-9)

Fig. 4. A\* Performance for Uniform and Bounded Distributions

So, mathematically, randomness affects the execution time and therefore is directly proportional to it; coalition size is inversely proportional to the execution time. This results in the following relationship below:

$$ExecutionTime \propto \frac{DegreeOfRandomness}{CoalitionSize} \quad (3)$$

So, as the coalition size increases, the effect on the execution time by the degree of randomness decreases, and therefore we start to have less fluctuations in the reported execution time (Figure 4(c) and Figure 4(e)). If the coalition size is small, the degree of randomness has more effect on the execution time, and hence we will notice higher fluctuations (Figure 4(a)).

Moving on to the graphs shown in Figures 4(b), 4(d) and 4(f) using the Bounded distribution data, we see that the trends are very similar to that of the graphs generated using the Uniform distribution data. Their only difference is that the Bounded distribution is slightly more flexible in accepting bids from certain coalitions whose size falls within certain bounds, while the Uniform distribution only accepts bids from coalitions of a specific size. This does not affect the overall solution quality however, but does expose some trends that we can observe and make deductions about the performance of the algorithm.

Figure 4(b) shows the graph for the Bounded distribution data with coalition sizes ranging from 1 to 3 (Bounded-1-3). This graph is very similar to Uniform-3 but is noisier because of the degree of randomness having greater influence on the execution time. For example, when bids from coalitions of size 1 are submitted, the degree of randomness has full influence on the execution time, which on the graph Figure 4(b) refers to the high peaks. However, for bids submitted by coalitions of size 2 or 3, the randomness factor diminishes in the execution time and we start to see close relevance to Uniform-3's plot. If we notice closely, for bid-size of 33 in Bounded-1-3, the execution time is 0.1 seconds, which is very close to the execution time using bid-size of 33 for Uniform-3. This then implies that the bid was submitted by a coalition of size 3 for Bounded-1-3. Therefore, these two graphs are similar except that Bounded-1-3 has greater vacillation due to the greater influence by the degree of randomness from the input data (effect of equation (3)).

Regarding the rest of the Bounded distribution graphs (Bounded-4-6 and Bounded-7-9) in Figures 4(d) and 4(f), the traits are predictable based on equation (3): for large coalitions, there are less fluctuations in execution time. However, we would still expect some noticeable differences in fluctuations between Bounded-4-6 and Uniform-6 versus Bounded-7-9 and Uniform-9, since the latter group of graphs should exhibit greater stability. But then, we have to remember that equation (3) is an exponential relation and therefore, we would only notice these differences if we sampled for much larger coalition sizes than those covered in this paper.

#### IV. RELATED WORK

In past work, most task allocation approaches deal with the SR-IA ([13], [3], [17], [9]) or SR-TA ([5], [21]) problems.

Typically, a task is decomposed into independent subtasks [13], hierarchical task trees [21], or roles [17] either by a general autonomous planner or by the human designer. Independent subtasks or roles can be achieved concurrently, while subtasks in task trees are achieved in order according to their precedence constraints. The work of [21] also addresses "tightly-coupled" multi-robot tasks; however, their tasks can be decomposed into multiple single-robot tasks and thus falls into the category of weakly-cooperative tasks. To achieve time-extended allocation, an additional layer of planning is introduced such that robots can opportunistically exchange tasks over time based on a market economy ([5], [21]).

Some recent approaches [11], [20], [10] are beginning to deal with the instantaneous assignment of multi-robot tasks (MR-IA). In these approaches, bid incorporating joint revenue [11], mapping tasks to robot capabilities [20], or direct play script [10] are used to initiate the cooperation between robots. However, the above approaches only deal with instantaneous allocation of these tasks. In other approaches, combinatorial auction is been applied to time-extended assignment such that a bundle of closely-related subtasks is been allocated together [15], [2]. Here, the tasks being allocated are independent tasks that can be accomplished by single robots. A robot may also win multiple bundles if its bids are high [15]. Limited work has been done to attack both issues in a single approach: the time-extended allocation of tightly-coupled multi-robot tasks. In this scenario, multiple SR- or MR- tasks are being allocated simultaneously, and thus both coalition formation and task scheduling are required to achieve an appropriate mapping so that the task pool can be accomplished efficiently.

#### V. CONCLUSION

Time-extended multi-robot task allocation can lead to more efficient allocations than traditional instantaneous allocations. We have described our approach for layering the ASyMTRe approach for generating task solutions and forming coalitions, with an auction-based time-extended task allocation approach for assigning tasks to coalitions. The proposed A\* search for winner determination guarantees the optimality of the current allocation and generates a minimum search tree for efficiency. Our future work includes demonstrating the complete approach with a set of real-world robot applications.

#### REFERENCES

- [1] R. C. Arkin. Motor schema based navigation for a mobile robot: an approach to programming by behavior. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 264–271, 1987.
- [2] M. Berhault, H. Huang, P. Keskinocak, S. Koenig, W. Elmaghraby, P. Griffin, and A. Kleywegt. Robot exploration with combinatorial auction. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.
- [3] S. Botelho and R. Alami. M+: A scheme for multi-robot cooperation through negotiated task allocation and achievement. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1234–1239, 1999.
- [4] Russell G. Brown and James S. Jennings. A pusher/steerer model for strongly cooperative mobile robot manipulation. In *Proceedings of IEEE International Conference on Intelligent Robots and Systems*, volume 3, pages 562–568, 1995.

- [5] M. B. Dias. *TraderBots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments*. PhD thesis, Robotics Institute, Carnegie Mellon University, 2004.
- [6] M. B. Dias, R. Zlot, N. Kalra, and A. Stentz. Market-based multirobot coordination: a survey and analysis. *Proceedings of IEEE*.
- [7] B. R. Donald. Information invariants in robotics. *Artificial Intelligence*, 72:217–304, 1995.
- [8] B. Gerkey and M. J. Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, 23(9):939–954, 2004.
- [9] B. P. Gerkey and M. J. Mataric. Sold! auction methods for multi-robot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, 2002.
- [10] E. Jones, B. Browning, M. B. Dias, B. Argall, M. Veloso, and A. Stentz. Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks. In *Proceedings of IEEE International Conference on Robotics and Automation*, May 2006.
- [11] N. Kalra, D. Ferguson, and A. Stentz. Hoplitest: a market-based framework for complex tight coordination in multi-robot teams. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2005.
- [12] D. M. Lyons and M. A. Arbib. A formal model of computation for sensory-based robotics. *IEEE Transactions on Robotics and Automation*, 5(3):280–293, 1989.
- [13] L. E. Parker. ALLIANCE: An architecture for fault tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2), 1998.
- [14] L. E. Parker and F. Tang. Building multi-robot coalitions through automated task solution synthesis. *Proceedings of IEEE*, 2006. Special Issue on Multi-Robot Systems.
- [15] T. Sandholm. Algorithm for optimal winner determination in combinatorial auctions. *Artificial Intelligence*, 135:1–54, 2002.
- [16] O. Shehory and S. Kraus. Coalition formation among autonomous agents: strategies and complexity. In C. Castelfranchi and J. P. Muller, editors, *Lecture Notes in Artificial Intelligence no. 957*. 1995. From Reaction to Cognition.
- [17] R. Simmons, S. Singh, D. Hershberger, J. Ramos, and T. Smith. First results in the coordination of heterogeneous robots for large-scale assembly. In *International Symposium on Experimental Robotics*, December 2000.
- [18] F. Tang and L. E. Parker. A complete methodology for generating multi-robot task solutions using ASyMTRe-D and market-based task allocation. In *Proceedings of IEEE International Conference on Robotics and Automation*, 2007.
- [19] F. Tang and S. Saha. An anytime winner determination algorithm for time-extended multi-robot task allocation. In *Proceedings of the International Conference on Automation, Robotics, and Control Systems*, 2008.
- [20] L. Vig and J. A. Adams. Issues in multi-robot coalition formation. In L. E. Parker, A. Schultz, and F. Schneider, editors, *Multi-Robot Systems Volume III: From Swarms to Intelligent Automata*. Kluwer, 2005.
- [21] R. Zlot and A. Stentz. Market-based multirobot coordination for complex tasks. *International Journal of Robotics Research*, 25(1), January 2006. Special Issue on the 4th International Conference on Field and Service Robotics.