Learning Task Performance in Market-Based Task Allocation

Charles Pippin
Georgia Tech Research Institute
Georgia Institute of Technology
Atlanta, Georgia 30332
Email: pippin@gatech.edu

Henrik Christensen
Center for Robotics and Intelligent Machines
Georgia Institute of Technology
Atlanta, Georgia 30332

***Abstract*—Auction based algorithms offer effective methods for de-centralized task assignment in multi-agent teams. Typically there is an implicit assumption that agents can be trusted to effectively perform assigned tasks. However, reliable performance of team members may not always be a valid assumption. An approach to learning team member performance is presented, which enables more efficient task assignment. A policy gradient reinforcement learning algorithm is used to learn a cost factor that can be applied individually to auction bids. Experimental results demonstrate that agents that model team member performance using this approach can more efficiently distribute tasks in multi-agent auctions.**

I. Introduction

The area of multi-agent systems has been an active area of research for many years, due in no small part to the ability for a team of robots to operate more efficiently and be more robust to failure than a single robot. However, there are still many challenges related to the interaction between the robots themselves. In traditional multi-agent systems approaches, each team member explicitly operates as part of a team and has the team's goals either explicitly or implicitly encoded. However, future robotic teams may have different internal goals as well as performance capabilities, costs, and owners. As such, robots may need to learn which team members reliably estimate and perform tasks as part of a team.

This work describes approaches for learning task performance of team members as applied to the multi-agent auction domain. A reinforcement learning algorithm is used to learn a cost factor for adjusting each agent's bids, based on the observed performance of that agent in completing tasks as estimated.

Market-based auction methods are a class of decentralized algorithms that solve the multi-robot task allocation problem by splitting computation across multiple nodes and iteratively performing task assignments [1]. The basic auction approaches to the task allocation problem assume that team members can be trusted and have the goal of the team in mind (to reduce the overall cost) [2]. These algorithms serve as a mechanism for distributed task allocation and generally do not explicitly consider individual team members' performance characteristics. However, there are situations in which teams may be formed dynamically, and the individual players within the team may have varying levels of performance and task estimation accuracy. In order for tasks to be allocated efficiently, it is important to be able to reliably trust that robots will perform their assigned tasks with costs that closely approximate their estimated costs. If a robot regularly exceeds their estimated cost for performing a task, the auction algorithm should be able to adjust the estimate to reflect the robot's true expected performance.

This paper will present an approach for learning which team members perform tasks at costs that accurately reflect the estimated costs. This approach can be used to more effectively perform auction based task allocation by using a reinforcement learning algorithm to adjust a cost factor that is applied to each team member's bid estimates.

The rest of this paper is organized as follows. In Section II, we present the background and related work for learning in multi-agent auctions. In Section III, we discuss the use of the reinforcement learning algorithm within an auction framework. In Section IV, we present results of simulated experiments using this approach. Finally, in Section V, we conclude and present future work.

II. Related Work

The ability to determine when a robot is not performing or functioning as expected can be used to re-assign tasks or call attention to an operator. An analysis of approaches to robot performance based metrics are presented by Parker in [3].

Pippin and Christensen considered allocating tasks to robots with different sensors characteristics in [4]. Given the probabilities of detection for each sensor, the expected utility is calculated and applied to each agent's cost bid. In that work, the sensor performance characteristics were known in advance.

An example of learning opportunity costs in auctions was performed in simulation of Martian rovers [5]. The appropriate opportunity costs allow for the specialized robots to avoid becoming underutilized. Over multiple simulations, the different types of robots adjusted their opportunity costs such that neither was underutilized.

Agents learned what valuation to bid by direct observations of similar other agents in [6]. The approach in that work is for the agent to learn to adapt their valuation (and the resulting bid) to market conditions in a simulated, electronic market, using a reinforcement learning algorithm. The market domain

was inspired by real world electronic commerce applications in which physical resources, such as trucks, and workers, competed to win tasks. The related problem of learning whether an agent should submit a bid is useful in domains in which computing a bid can be expensive because communication and computation costs can be considerable [7].

Jones, Dias and Stentz investigated techniques for learning proper task cost estimates in oversubscribed domains, using auction algorithms [8]. In that work, each robot attempted to learn their own bid estimates, and had full knowledge of their own state vectors, including their own schedule. In this paper, we are interested in learning whether bids accurately match their estimated values, but from the viewpoint of the auctioneer. The auctioneer has less visibility into the state features that can be used to estimate a bid and relies on the estimated vs. actual cost to apply a cost adjustment to future bids.

This paper uses a learning approach that is very similar to that used in Kohl and Stone for learning fast gaits on quadrupedal robots [9]. They applied a policy gradient reinforcement learning algorithm to learn control parameters for leg motions. The policy gradient algorithm was also applied by Mitsunaga, et al. to adapt robot behaviors to human partners [10]. This paper will apply the policy gradient learner to the task of learning a cost factor for other robots' cost estimates in market-based auction algorithms.

III. Approach

Robots may fail to perform tasks according to their initial estimate for a number of reasons. For instance, in complex domains, the cost function may be expensive to calculate and the task estimate might be based on a heuristic function. Perhaps, over time a robot's performance might have degraded due to hardware failure or wear. Finally, the robot's internal state may not reflect the true state of the robot, causing errors in estimation.

For this paper, we will assume that a subset of the team members regularly mis-estimate task costs by an unknown factor, due to errors in the robot's internal state model. Therefore, the task is to learn the true cost factor for those robots. A learning algorithm will be used to approximate the cost factor and apply it to the task assignment function used by a market based auction algorithm.

A. Auction Approach

In the basic multi-agent auction algorithm, the problem is to assign tasks to agents. The tasks in this case are to visit a target location and perform an observation. In the auction framework, each robot is a *bidder* and the items to be auctioned are the tasks. Each of the agents in the system also participates as an *auctioneer* and periodically auctions new task requests (it is assumed that the task requests are periodically provided to the agent by an external process, such as a human operator or other event). This approach can easily be used on teams with different robot characteristics: each robot knows their own location and cost function and submits cost based bids to the auctioneer. While costs and rewards use the same basis for calculation, no revenue is actually exchanged. Rather, an agent awards itself a utility value when one of its own tasks is completed.

In this work, the agents each maintain a current task list and locally compute their bid to complete the proposed task. The bid consists of the time-based cost to perform the task. A potential source of error in task estimation is in the use of an insertion heuristic for calculating the marginal cost to perform a task, in addition to those tasks already assigned. In this paper, each robot plans to visit the targets in the order in which they were assigned (using the $O1$ assignment rule from [11]). For each auction announcement received, each robot calculates their bid as the amount of time required to complete the task in addition to those on the current task list. When the winning *bidder* is assigned a new task, the task is appended to the robot's assigned task list.

B. Learning the Cost Factor

The learning method used in this work is the *policy gradient reinforcement learning* (PGRL) algorithm. This is a reinforcement learning method that is used to estimate the policy gradient when the true value function is not known. The PGRL algorithm is presented in detail by Baxter and Bartlett in [12], and it is shown that this approach converges towards a local optimum.

1) The PGRL Algorithm: The pseudocode for the PGRL algorithm, adapted from [9] and [10], is shown in Figure 1. At the beginning of the algorithm, the policy vector, Θ, is initialized. In this paper, we are using the algorithm to learn a single parameter, θ, which is the cost factor to apply to a robot's task estimation. Therefore, we initialize $\theta = 1$, reflecting the belief that each robot perfectly estimates tasks that they will perform. In the main loop, the algorithm generates a set of random permutations for the policy by adding either $+\epsilon$, 0 or $-\epsilon$ to the policy.

Next, each of these permutations is evaluated by the system and the resulting reward is received. Averages of the rewards are maintained for the permutations of each type. After all of the permutations have been evaluated, the gradient is approximated by calculating the adjustment, a_j, related to each parameter in the policy. Each parameter's step distance, ϵ_j, and the global step distance, η, are applied to a_j and the values are normalized. Finally, the policy is updated with the adjustment.

An example of learning a cost factor, θ, using the PGRL algorithm is shown in Figure 4(a). In this example, the unknown cost factor is 2, and θ is initialized to 1. After about 100 evaluations, the learned policy begins to converge near the true value.

2) Learning in Auctions: The PGRL is applied to multi-robot auctions by applying the learned cost factor to each robot's bid, as shown in Figure 2. Each auctioneer maintains a PGRL learner for each known team member. After auctioning a task, the auctioneer will receive a set of bids from team members, where each bid represents the time-based cost for the bidder to complete that task. The auctioneer then queries

1: $\Theta \leftarrow$ initial policy vector of size N
2: **while** NOT done **do**
3: $\Theta^T \leftarrow$ t random permuations of Θ
4: **for** $t = 1 \rightarrow T$ **do**
5: Run system using parameter set Θ^t
6: Evaluate reward
7: **end for**
8: **for** $n = 1 \rightarrow N$ **do**
9: $Avg_{+\epsilon,n} \leftarrow$ average reward for all Θ^t that have a positive perturbation in dimension n.
10: $Avg_{0,n} \leftarrow$ average reward for all Θ^t that have zero perturbation in dimension n.
11: $Avg_{-\epsilon,n} \leftarrow$ average reward for all Θ^t that have a negative perturbation in dimension n.
12: **if** ($Avg_{0,n} > Avg_{+\epsilon,n}$ AND $Avg_{0,n} > Avg_{-\epsilon,n}$) **then**
13: $a_j \leftarrow 0$
14: **else**
15: $a_j \leftarrow (Avg_{+\epsilon,n} - Avg_{-\epsilon,n})$
16: **end if**
17: **end for**
18: $A \leftarrow \frac{A}{|A|} * \eta$
19: $a_j \leftarrow a_j * \epsilon_j, \forall j$
20: $\Theta \leftarrow \Theta + A$
21: **end while**

Fig. 1. The Policy gradient reinforcement learning algorithm pseudocode, with N policy parameters. During each iteration, t random policies are sampled near the current policy for evaluation. The resulting reward from each sample is used to estimate the gradient and move by a small amount in the correct direction.

the learner to get the cost factor, θ, related to that agent. This cost factor is multiplied by the original bid in line 3 to get an updated estimate for the agent to perform the task. The resulting bid in the set with the minimum cost is then awarded the task.

When a task is completed by an agent, the auctioneer that assigned the task is sent a message with the completed task information. The auctioneer can then compare the updated estimated cost for the task with the actual cost for completion, as shown in Figure 3. The ratio of these costs is used to determine the reward signal used by the reinforcement learner. The PGRL reward function, shown in Figure 4(b), calculates the scalar reward signal from this value. The reward is used by the PGRL algorithm, as described above, to calculate adjustments to the cost factor.

IV. Experimental Results

A. Experimental Setup

A set of experiments were performed in simulation to test the cost factor learning approach in a multi-agent auction environment. In these experiments, each robot has 50 tasks that arrive at regular intervals and are sequentially auctioned by that robot's auctioneer. As part of the auction process, they

Input: The set of posted bids, B.
Input: The set of bidders, A.

1: **for all** $a : A$ **do**
2: $\theta \leftarrow GetTheta(Learner_a)$
3: $B_a^* \leftarrow \theta * B_a$
4: $EstCost_{B_a} \leftarrow Cost_{B_a^*}$
5: **end for**
6: $winner \leftarrow Min(B_a^*)$
7: $AnnounceWinner(winner, a)$

Fig. 2. HandleBids() pseudocode. The policy gradient reinforcement learning method learns the cost parameter, θ, for each team member. This cost factor is applied to future bids for that agent.

1: $ActualCost \leftarrow (CompleteTime - StartTime)$
2: $CostFactor_{task} \leftarrow ActualCost/EstCost_{B_a}$
3: $UpdateRL(Learner_a, CostFactor_{task})$

Fig. 3. TaskComplete() pseudocode. The estimated vs. actual task completion time is used to evaluate the value of θ in the reinforcement learning algorithm.

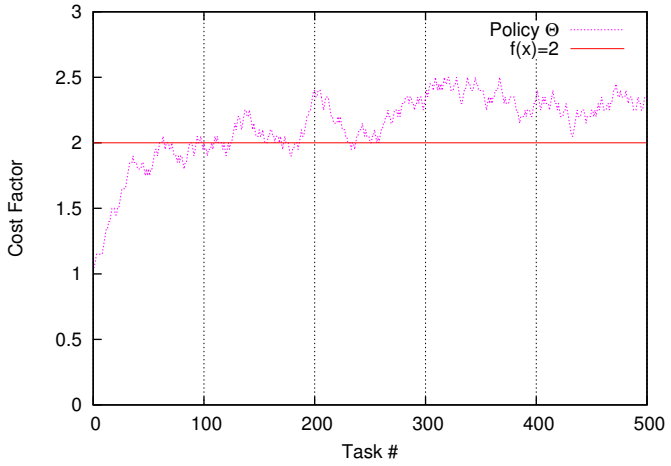
also bid on their own tasks. The robots in the simulation have a limited communications range and can therefore only perform auctions with a subset of the other team members at a given time.

Rewards are given for task completion to the robot that originated the task. Each robot submits bids that represent the time-based cost for completing a task. Specifically, the bid represents the number of time steps until the task could be completed. Once a robot finishes all tasks in their list, they no longer accumulate costs in the simulation. The initial locations of the robots and the tasks are randomly chosen for each iteration.

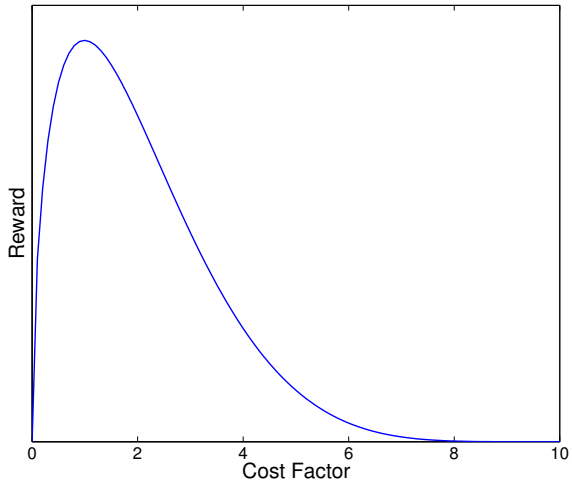
1) Task Estimation: In this paper, the source for estimation error is assumed to be due to poor performing robot having an incorrect model of its own performance capabilities. To simulate robots that bid poorly, a percentage of robots on the team are modeled as *poor performers* by randomly assigning a cost factor at the start of the experiment, using a normal distribution with $\mu = 2$ or $\mu = 3$ and $\sigma = 0.1$. When a *poor performer* bids on a task, the unknown cost factor is drawn from this distribution and is applied to the robot's task performance to simulate error in estimation and execution.

2) Learning and Applying the Policy: During the learning phase, 1000 auctions were performed with the PGRL algorithm running with a varying number of poor performers on the team in order to learn the cost factors. These experiments used a centralized learner to share the results across each agent's auctioneer.

After the cost factors were learned, they were loaded and a set of experiments were performed to perform auctions using the learned cost factor as the PGRL initial policy. The algorithm continued to learn online, but the step distance, ϵ_j was reduced to minimize exploration. This *policy learner* method is compared against a *naive* auction method which does not consider performance; a *known state* method that



(a) Learning the Cost Factor



(b) Cost Factor Reward Function

Fig. 4. The PGRL approach to learning the cost factor: (a) The policy gradient reinforcement learning method learns the cost parameter, θ. In this example, the agent's unknown true cost factor is 2. (b) The reward function used by the PGRL algorithm converts an observed cost factor to a reward value.

has access to the true cost factor for each team member; and a *no cooperation* method in which each team member performs all of their tasks without the benefits of cooperation. For each experiment, the average global score represents the score (*average task reward / average cost*) for the team. For each set of experiments, results were averaged over 100 runs.

B. Results and Discussion

The results of these experiments are shown in Figure 5. The average global score using each strategy is plotted against teams with 2, 3 and 4 *poor performers* on a team of 6 robots. The error bars represent 1 standard deviation.

The *known state* strategy represents the best score on average that a team could achieve given the number of *poor performers* on the team. This strategy has access to an oracle

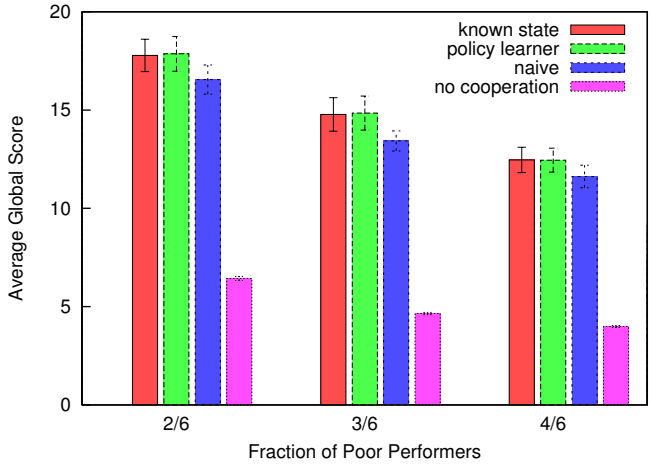


Fig. 5. Task Performance: The learning method is compared with a method that knows the state of each agent in advance, a naive auction method that doesn't consider performance and with the case of no cooperation.

with knowledge of each team member's hidden state and can calculate the true cost factor to apply to each bid. The scores for the *policy learner* strategy approach the scores of the *known state* strategy. The *policy learner* also scores better than the *naive* strategy which only uses a basic auction and does not consider bid estimation accuracy of each team member. Finally, the *no cooperation* strategy scores the worst, demonstrating that it is better to have poor performers on the team than to work on tasks in isolation.

The results indicate that the *policy learner* strategy can result in up to a 10% improvement in team performance than over the basic auction approach alone. Intuitively, we expected that the gap between the *policy learner* and the *naive* approaches would be larger. However, the auction method distributes task effectively as the *poor performers* get behind. That is, as the number of tasks begins to back up for the *poor performers*, the costs to add a task to the end of their schedule increases, and they win fewer auction assignments. This occurs even though their costs are underestimated. We intend to explore additional bidding and task insertion heuristics using this technique in future work. Nevertheless, these experiments demonstrate that the application of a learner to the cost estimates can be more effective than methods that do not consider estimation accuracy.

V. Conclusion

This paper presents a reinforcement learning method for recognizing which agents are more likely to submit bids that accurately reflect the true cost for performing tasks. The above experiments showed that a learning mechanism can be effective for detecting poorly-performing team members in auctions, when compared to the naive approach. This may prove useful in situations in which auction based teams are dynamically formed and not all team members are likely to estimate costs correctly. The algorithm learned the cost factor to apply to each team member's bid estimate in a multi-robot

auction. The results show that by learning the performance characteristics of individual robots, tasks can be allocated more efficiently.

Future work will consider additional learning mechanisms relevant to task performance. This is related to the problem of determining how to recognize when tasks that were assigned to another agent were not only completed according to the initial cost estimate, but completed within stated quality parameters. In addition, we would like to determine how well this method performs with the use of additional bidding and task insertion heuristics. Finally, we hope to validate this approach with a set of experiments using a team of indoor mobile robots.

ACKNOWLEDGMENT

This work was funded internally by the Georgia Tech Research Institute.

REFERENCES

[1] D. P. Bertsekas, "The auction algorithm for assignment and other network flow problems: A tutorial," *Interfaces*, vol. 20, no. 4, pp. pp. 133–149, 1990.

[2] S. Koenig, P. Keskinocak, and C. Tovey, "Progress on agent coordination with cooperative auctions," in *AAAI Conference on Artificial Intelligence*, 2010.

[3] L. E. Parker, "Reliability and fault tolerance in collective robot systems," in *Handbook on Collective Robotics*, S. Kernbach, Ed. Pan Stanford Publishing, 2012.

[4] C. E. Pippin and H. Christensen, "A Bayesian formulation for auction-based task allocation in heterogeneous multi-agent teams," M. A. Kolodny, T. Pham, and K. L. Priddy, Eds., vol. 8047, no. 1. SPIE, 2011, p. 804710.

[5] J. Schneider, D. Apfelbaum, D. Bagnell, and R. Simmons, "Learning opportunity costs in multi-robot market based planners," in *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2005.

[6] E. Oliveira, J. M. Fonseca, J. Manuel, F. Nicholas, and N. R. Jennings, "Learning to be competitive in the market," in *In Proceedings of the AAAI Workshop on Negotiation: Settling Conflicts and Identifying Opportunities*, 1999.

[7] D. Busquets and R. Simmons, *Learning when to Auction and when to Bid*. Springer, 2006, vol. Distributed Autonomous Robotic Systems 7., pp. 21–30.

[8] E. Jones, M. B. Dias, and A. T. Stentz, "Learning-enhanced market-based task allocation for disaster response," in *Intelligent Robots and Systems, 2007. (IROS)*, October 2007.

[9] N. Kohl and P. Stone, "Policy gradient reinforcement learning for fast quadrupedal locomotion," *In Proceedings of the IEEE International Conference on Robotics and Automation*, May 2004.

[10] N. Mitsunaga, C. Smith, T. Kanda, H. Ishiguro, and N. Hagita, "Robot behavior adaptation for human-robot interaction based on policy gradient reinforcement learning," in *Intelligent Robots and Systems, 2005. (IROS)*, Aug. 2005, pp. 218 – 225.

[11] A. Ekici, P. Keskinocak, and S. Koenig, "Multi-robot routing with linear decreasing rewards over time," in *Proceedings of the 2009 IEEE international conference on Robotics and Automation*, ser. ICRA'09. Piscataway, NJ, USA: IEEE Press, 2009, pp. 3944–3949.

[12] J. Baxter and P. L. Bartlett, "Infinite-horizon policy-gradient estimation," *Journal of Artificial Intelligence Research*, 2001.