Bounty Hunters and Multiagent Task Allocation

Drew Wicke
Department of Computer Science
George Mason University
4400 University Drive MSN 4A5
Fairfax, VA 22030, USA
dwicke@gmu.edu

David Freelan
Department of Computer Science
George Mason University
4400 University Drive MSN 4A5
Fairfax, VA 22030, USA
dfreelan@gmu.edu

Sean Luke
Department of Computer Science
George Mason University
4400 University Drive MSN 4A5
Fairfax, VA 22030, USA
sean@cs.gmu.edu

ABSTRACT

We propose a system for multiagent task allocation inspired by the model used by bounty hunters and bail bondsmen. A bondsman posts tasks for agents to complete, along with bounties to be collected by an agent on completion. Multiple agents, taking the role of the bounty hunters, compete to finish tasks and collect their bounties. While a task remains uncompleted, its bounty gradually rises, making it more and more desirable to pursue. Unlike auctions, this model does not assume rationality in agents' bids (as there are none), and since tasks are not exclusive to given agents, the system is robust to highly noisy environments. We examine how agents may locally develop rational task valuations in such an environment, gradually adapting to dividing tasks according to the agents best suited to them. We compare different methods for building these valuations against approaches which are more "auction-like" in that they permit exclusivity, and we do so under both static environments and ones in which agents, and task details, change dynamically.

Categories and Subject Descriptors

I.2.11 [**Distributed Artificial Intelligence**]: Multiagent Systems

General Terms

Algorithms, Experimentation, Performance

Keywords

Bounty Hunting; Auctions; Task Allocation

INTRODUCTION

The *dynamic multiagent task allocation problem* is one in which one or more agents collectively perform tasks as they appear dynamically in the environment. The agents typically vary in their ability to perform various tasks, and as this is a *multiagent* problem, there is no one agent who decides how to allocate the tasks to them. Rather, the agents decide on their own what tasks they should vie for, with certain limitations on the degree to which they may consult one another to make these decisions. Ideally, out of a morass of individual greedy agent decisions, we would see a task allocation which approaches optimal global efficiency.

Scenarios such as these crop up throughout multiagent systems: they appear in robots divvying up room cleaning needs; or automated multiagent delivery services; or video games where a band of plucky outlaws must defend themselves against hordes of zombies. Considerable study over the years has focused on centralized approaches to globally optimal dynamic task allocation, involving areas such as register coloring and pebbling, multi-armed bandits, and bin packing. When the problem is restricted to the multiagent case, most work has focused on greedy methods, the most popular approach being *auctions*.

A simple auction method might work as follows. There is a single auctioneer who distributes tasks, and multiple agents who bid on the privilege to do them. As tasks come available, agents compete with bids in an auction, and after a clearing time has passed, the winning agent is given the task to complete. Typically the agents will bid their *valuation* of the task, that is, some estimate the agent has of how good the task would be for the agent to do.

This approach is intuitive and appealing at first glance, but it has some curiosities. First, in much of the literature, there is no incentive or reward for agents to complete a task. Second, agents generally have an infinite money supply, and so we must presume they are altruistic and non-strategic. Third, winning agents usually have exclusive ownership of tasks, so some other mechanism (perhaps real options) must be implemented to deal with agents with poor or noisy task performance. Fourth, agents typically must have some built-in ability to produce an accurate valuation of the tasks they are bidding on. Fifth, auction methods commonly assume that tasks are bid on sequentially as they arrive, perhaps necessitating a multi-task mechanism for each agent or a clearing time. For these reasons and others, auctions seem to us to be an odd approach to task allocation.

In this paper we study an alternative model which seems to us to be a closer fit to the dynamic multiagent task allocation problem: *bounty hunting*. Here, the auctioneer is replaced with a bail bondsman, and the agents are no longer bidders, but bounty hunters.

A quick explanation of these terms for those who may be unfamiliar with them. In the United States, an arrested man may be set free temporarily prior to his trial in return for some amount of money (*bail*) as collateral. Bail is often costly, so there is an industry of *bail bondsmen* who will, for a fee, post bail on behalf of the arrested man. If the man does not show for his trial, a bail bondsman can recover his posted bail only if he brings the fugitive back to be tried before the government has re-arrested him. To do this, the bail bondsman will offer some amount of reward (a *bounty*) to anyone who successfully captures the fugitive. The competing agents who try to capture the fugitive in return for this bounty are *bounty hunters*. While the fugitive is still at large, the bounty may be increased so as to make his capture more lucrative. This model appears elsewhere in different guises of course: for example, a reward for a lost cat, a Most Wanted Criminal list, a murder contract, or a software bug bounty.

Appears in: *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2015), Bordini, Elkind, Weiss, Yolum (eds.), May 4–8, 2015, Istanbul, Turkey.*

Bounties are an attractive approach to the dynamic multiagent task allocation problem for several reasons. First, because agents may compete for the same bounty, tasks are no longer exclusive. If an agent cannot successfully do a given task, eventually its bounty will rise and attract other agents to take it from him. Second, there is no bidding involved: agents can be self-interested, though still perhaps not hyperstrategic, and there is no need for a queue or clearing interval. Finally, since agents are not bidding their valuation, they don't need to have (at least initially) an accurate valuation mechanism in the first place.

But despite their seemingly natural fit, bounties have surprisingly been little studied in multiagent systems. In this paper we will try to remedy this. We first describe one possible formal model for a bounty system, and relax this model in two ways (notably adding exclusive ownership of tasks) which are more "auction like" (the scenarios we study do not lend themselves to classical auctions). In all these approaches agents have no initial valuation of tasks, so we then examine how they may adapt to develop a proper valuation, and thus improve global system performance. We will also examine how well this adaptation performs when the nature of tasks, or the agents, changes radically. The "auction like" models are slightly more efficient in common situations: but we then show that they are problematic in noisy situations. Finally, we study what might happen when agents are permitted to abandon tasks.

RELATED WORK

A taxonomy of solutions to the general multiagent task allocation problem has been offered [10] with three major dimensions: the type of task, the type of agent, and the information available to determine task assignment. This taxonomy has recently been improved to classify more task allocation problems, like those where tasks have dependencies [16]. Current solution methods include centralized approaches using k-armed bandits, integer programming, and combinatorial optimization; and decentralized approaches like auctions, markets, and task swapping [17, 18, 19, 22].

We are primarily interested in the decentralized case. The most common approach, auctions, were popularized by MURDOCH, an auction method which focused on minimizing resource usage, task completion time, and communication overhead [11]. This approach trusted agents to truthfully bid their task valuations, though sensor noise or an unknown environment could cause their bids to be inaccurate. TraderBots, another auction framework, was designed to be more flexible and extensible [8, 13]. Auctions have been augmented to be robust to robot malfunctions [23]. CoMutaR merged multiagent task allocation with coordination through the use of auctions under the assumption of truthful agents [29]. We are aware of only one paper that has considered approaches to extend auctions with non-exclusivity and non-commitment [20].

Markets have also been used in multi-robot task allocation [1, 27] and exploration [30, 32]. One paper describes a market based planning system in which the agents learn to bid their opportunity cost for doing a task, and tasks have rewards associated with them, allowing agents to re-auction them [28].

Another approach is *token-passing* [9]. This method, like auctions and markets, also assumes task exclusivity. But like some market methods, it allows the tasks to be reassigned by passing the token to other agents. Similarly, [7] used *vacancy chains* as a method for allocating tasks. A vacancy chain is a social construct where when one agent receives a task, he releases his previous task, which goes to another agent in the chain, who releases his task, and so on. Finally, L-ALLIANCE adapted threshold parameters determining how robots choose tasks [24, 25].

A few auction methods have agents that use adaptive mechanisms for bidding. For example, [14] use Support Vector Regression to formulate a bid for a task in domains where some tasks can't be completed due to a lack of resources. Another adaptive bidding procedure for task allocation was developed in [4] and was used to bid in a sequential auction with a clearing time. In a method developed by [26], the auctioneer, rather than the bidders, is adaptive and weights the bids for tasks based on each agent's success rate. This is very close to a centralized approach. One non-auction task allocation method [2] uses biologically inspired adaptive agents that have an upper and lower bound for going after tasks with particular difficulties (similar to L-ALLIANCE), where the lower threshold changes based on task difficulty distribution. Because of the strong assumptions these adaptive auctions make, they could be adapted to our target problem case only with radical modification. We have instead taken a different approach for comparison: adapting our proposed bounty mechanism into one which has certain the basic features (notably exclusivity) of an auction.

Bounties are a subset of *contests*, scenarios where agents are pit against one another to obtain a prize [6]. One type of contest, the *tournament*, is very common in sports and competitions [5], and contests are also found throughout economics, and in the managerial and political sciences [15]. Contests have not been applied much in multiagent task allocation to date. Contest theory has explored adjustable incentive mechanisms in terms of promotions within companies and cash prizes based on performance [3, 21, 31]. These incentive mechanisms are mainly used to manipulate the effort of the contestants. In contrast, the purpose of the bounty mechanism is not to increase the contestant effort, but to more effectively pair contestants to specific tasks.

To our knowledge, competitive bounty hunting has not been used as a mechanism for multiagent task allocation. Interestingly, while bounties have been studied in terms of public policy [12], not much work has been done examining them as an economic model either. Bounties share some relationships with *real options* where an agent may negotiate to have the option to exercise something in the future: for example, movie producers may purchase the rights to make a movie from a book; or mining companies may purchase the rights to use a parcel of public land. The option seller in some sense may be viewed as a bondsman and the buyer as a bounty hunter: however in a bounty system, the "option" is not exclusive, guaranteed, nor even negotiated. Options are more similar to market or token methods such as in [9, 28].

MODEL DESCRIPTION

The Bounties task allocation model involves two types of actors: a bondsman and one or more bounty hunter agents. The bondsman generates and maintains a list of available tasks. Tasks belong to *task classes*. Tasks in a given class are considered similar in some sense, and hence related in difficulty. Each task has an associated *bounty* which is posted with the task and which changes (we will assume it rises linearly) over time as long as the task is uncompleted.

A bounty hunter *commits* to at most one task at a time, which allows him to begin work on the task, and which announces to other agents that he has committed to it. Multiple agents may commit to a task, but they may not collaborate on it. For now we will presume that once an agent has committed to a task, he must complete the task, or be beaten out by some other agent, before he may commit to a new task. When a bounty hunter successfully completes a task, he receives a payment equal to the bounty of the task when he had committed to it.

Agents could refuse to commit to any task for some period of time. However in our experiments there is always at least one

available task, and we have set the incentive structure to make refusal irrational. Thus the agents in our experiments by design never wait before committing to tasks.

More formally we have:

- A set of bounty hunter agents, denoted $A : \{a_1, a_2, ..., \}$.
- A set of task classes $S : \{S_1, S_2, ...\}$.
- For each task class S_i, there is an associated set of possible tasks $\{I_{i,1}, I_{i,2}, ...\}$.
- At each time step t there is some set $Q^{(t)}$ of uncompleted tasks available for bounty hunters to attempt to complete. We denote these tasks by the integers $1, 2, ..., i, ...$
- Each uncompleted task $i \in Q^{(t)}$ is associated with a *bounty* $b_i \in \mathbb{R}$. This bounty changes over time t according to some (typically monotonically nondecreasing) function $B_i(t)$. In our experiments we will assume that $B_i(t)$ is a simple linear function, and so denote the *rate* at which b_i increases per timestep as r_i.
- At any time t, a mapping $M_t(A) : A \times \{Q^{(t)} \cup \{\square\}\} \rightarrow \mathbb{R}$ from agents and the uncompleted tasks to which each is committed (or to none, denoted $\square$), to the bounty established when each agent committed to that task. The bounty on "none" ($\square$) is immaterial.

  In our model we will assume that an agent may commit to no more than a single task at any given time. Furthermore, in some methods, only a single agent may commit to a task (it is *exclusive*).

EXCLUSIVITY AND CLEARING TIME

Unlike a typical auction, a true bounty system is not exclusive: multiple agents may commit to a task. Exclusivity has an advantage in that agents are never wasting time simultaneously working on the same task. But bounties have a different advantage: they provide a straightforward way for agents to resolve situations where an agent cannot complete a task or complete it efficiently, as ultimately another agent will take it from him.

Similarly, a bounty system also does not require a *clearing time* for bids, since there are no bids at all. An auction-style clearing time is not easy to integrate into scenarios where tasks arrive at arbitrary times, levels of urgency, or rate; and where typically only a single agent is available to bid for them: and so agents would likely need to have multi-task queues. Our simple experimental scenario, defined later, is one such example. We failed to find an auction model from the distributed multiagent task allocation literature which could be straightforwardly adapted to it.

Lacking an easy way to compare against the auction literature, we can still tweak a bounty model into an "auction-like" environment, by adding task exclusivity and optionally a one-task queue of sorts. Such modified models are no longer true bounty systems, but retain much of their flexibility. We consider three possibilities:

- **Bounties** This is the model discussed earlier. Tasks are not exclusive.
- **Bounties with Exclusivity** This is the Bounties model, except that when an agent commits to a task, he owns it exclusively: no other agent may commit to it. If two or more agents simultaneously commit to a task, the winner is chosen at random, and the losers vie for the remaining tasks.
- **A Bounty "Auction"** This is the same as the Bounties with Exclusivity model, except that we change the procedure with which an agent decides to commit. When one or more agents are ready to commit to tasks at time t, all agents (committed or not) are queried for their valuations of the current uncommitted tasks in $Q^{(t)}$. We then iterate as follows: the highest-valued task is paired with the agent who valued it thus (breaking ties randomly). This task and agent are then discarded, and of the remaining tasks and agents, the highest-valued task is again paired with the agent who valued it, and so on. This continues until all tasks or all agents have been paired. An agent who wishes to commit then commits to the task to which he has been paired, if any. We assume that an agent will always commit to a new task immediately upon completion of his current task, if a new task exists.

Note that these three approaches run the gamut from a bounties model to one more closely resembling an exclusive multi-item auction of sorts, albeit one without a clearing time, nor one where agents necessarily receive tasks even if ones are available.

ADAPTIVE VALUATION

Because the bounty model does not presume that agents have an accurate valuation of the environment, nor of the nature of other agents, we have experimented with two methods which enable agents to learn this information in order to adapt to the tasks best suited to them. The *Simple* method only tries to learn the *estimated time to completion* of given task classes, and their corresponding *probability of completion*. The *Complex* method expands on the probability of completion to learn the probability of completion given the specific other agents that have also committed to the task.

These two methods provide us with several parameters to experiment with. ε is the probability that the agent will commit to a randomly-chosen task. α is the learning rate for the estimated time to completion. β is the learning rate for the estimated probability of completion. Finally, γ is a (small) *unlearning* rate for the estimated probability of completion: like ε, this promotes exploration.

Simple Method. Given a posted task i, let the estimated expected time to complete tasks of the class of i be T_i and the estimated probability of completion of tasks of the class of i be P_i. Initially, $\forall i : T_i = 1,\ P_i = 1$, to encourage the agent to explore all tasks. The agent then iterates as follows. First, the agent chooses a task to commit to and complete. With ε probability he will pick an entirely random available task. Otherwise he will choose task I^* as:

$$I^* \leftarrow \underset{i \in \text{Available Tasks}}{\text{argmax}} \frac{b_i}{T_i} P_i$$

While the agent is working on a task, nothing is updated. Then at some point the agent either completes the task, or fails to do so because some other agent has completed it instead. In the first case, T_i and P_i are updated as:

$$T_i \leftarrow (1-\alpha)T_i + \alpha t$$
$$P_i \leftarrow (1-\beta)P_i + \beta$$

...where t is the amount of time it has taken to complete the task since the agent committed to it. In the second case, only P_i is updated as:

$$P_i \leftarrow (1-\beta)P_i$$

Finally upon completion or failure of a task, *all* tasks i are updated:

$$\forall i : P_i \leftarrow (1-\gamma)P_i + \gamma$$

The purpose of γ is to slowly drift probabilities towards 1 to make them more likely to try in the future. This, like ε, promotes exploration of the space.

We note that in the *Bounties with Exclusivity* and *Bounty "Auction"* approaches, since tasks are exclusively committed to, the probability table P serves no purpose. Thus for these approaches we fix $(\forall i)P_i = 1,\ \beta = 0,\ \gamma = 0$.

Complex Method. In the Complex Method we augment the task selection equation by including the probability that the agent will succeed given that certain other agents have committed to the task. We naively compute this as the product of the estimated probability, for each additional agent committed to the task, that our agent would defeat the additional agent. To do this, instead of P_i, let $P_{i,a}$ be the estimated probability that our agent will defeat another agent a at tasks of the class of i if both agents are committed to it. This (wrongly but conveniently) assumes that success rates against different agents are independent.

Initially we set $\forall i,a : T_i = 1,\ P_{i,a} = 1$. We select using ε as usual, but redefine I^* as:

$$I^* \leftarrow \underset{i \in \text{ Available Tasks}}{\text{argmax}} \frac{b_i}{T_i} \prod_{a \text{ presently committed to } i} P_{i,a}$$

We revised the update equations accordingly. When our agent succeeds at a task, T_i and $P_{i,a}$ are updated as:

$$T_i \leftarrow (1-\alpha)T_i + \alpha t$$
$$\forall a \text{ presently committed to } i : P_{i,a} \leftarrow (1-\beta)P_{i,a} + \beta$$

...where t is the amount of time it has taken to complete the task since the agent committed to it. When our agent fails at a task because a specific other agent a^* has beaten it to the task, we only update: P_{i,a^*}:

$$P_{i,a^*} \leftarrow (1-\beta)P_{i,a^*}$$

In either case, upon completion or failure of a task, all tasks i are then updated for all other agents a as:

$$\forall i,a : P_{i,a} \leftarrow (1-\gamma)P_{i,a} + \gamma$$

We note that the *Bounties with Exclusivity* and *Bounty "Auction"* approaches do not use this method, since it explicitly assumes that other agents may commit to the same task.

EXPERIMENTS

To test the effectiveness of the adaptive bounty methods and different exploration techniques, we performed tests in a simulated environment of agents and balls for them to retrieve. This scenario was derived from similar tasks we encountered in robot soccer experiments. Four agents (1, 2, 3, and 4) are placed at the corners $\langle 0,0\rangle$, $\langle 59,0\rangle$, $\langle 0,39\rangle$, and $\langle 59,39\rangle$ respectively on an infinite discrete grid. We define twenty task classes, each with a *mean location* uniformly randomly chosen somewhere within the 60×40 rectangle between the agents' corners. The simulation starts with twenty balls (the tasks, one task per class), which appear on the field randomly at locations using a gaussian distribution centered at their task class means and with a standard deviation in each direction using $\sigma = 5$.

When an agent commits to a task, he moves towards its ball. Agents can move one grid square (up, down, left, right) per timestep. When an agent reaches the ball, he completes the task, and everybody who was moving toward the ball is instantly teleported back to his corner. Teleportation was meant largely to speed up simulation. When the task is completed, another ball of the task class reappears in $p \sim \text{uniform}(0,19)$ timesteps.

While a task remains uncompleted, its bounty gradually increases at a rate described later. Once an agent commits to a task, he is locked to the current bounty for that task and may not abandon that task until it is completed by some agent (not necessarily himself). It is legal for two agents to occupy the same square. If two agents complete the same task simultaneously, the winner is the agent with the lower agent ID (1, 2, 3, or 4).

We tested six scenarios. First, we used a *static environment* where agents played the game described above. Second, we tried a *dynamic agent environment* where periodically some agents would be removed from the game, then added back. Third, we tried a *dynamic task environment* where task complexities changed for each agent periodically. The fourth and fifth experiments focused on extreme situations. The fourth experiment had unreliable collaborators that were slower in going after tasks than the other agents. In the fifth experiment we randomly modified a task class to be unexpectedly harder to do for a particular agent by making that agent a tenth its normal speed. Finally, in the sixth experiment we examined the effect of allowing agents to abandon going after their chosen task.

Dynamic environments require that agents explore sufficiently in order to understand that the world has changed. We tested the two methods (Simple and Complex) using three different approaches to exploration. First, we simply relied on rising bounty values to trigger more exploration. We called these the *Simple* and *Complex* approaches. Second, we set $\varepsilon = 0.1$, which caused 10% of task selections to be completely random. We called these the *SimpleR* and *ComplexR* methods. Third, we set $\gamma = 0.001$, which gradually caused the agents to become more optimistic about their probabilities of success. We called these the *SimpleP* and *ComplexP* methods. We further fixed $\alpha = 0.1$ and $\beta = 0.2$. The settings of $\alpha, \beta, \varepsilon$, and γ were determined from calibration trials. We also included the *Bounties with Exclusivity* method, hereafter shortened to *Exclusive*, and the *Bounty "Auction"*, hereafter shortened to *Auction*.

We compared these eight methods and exploration techniques against two control methods, largely to serve as rough upper and lower bounds. In *Random*, agents commit to tasks entirely at random. This served as our worst case bound. In *Greedy*, each agent knows the expected time $E(T_i)$ to completion of tasks of the class of task i, and greedily commits to the task $I^* \leftarrow \text{argmax}_{i \in \text{ Uncommitted Tasks}} \frac{b_i}{E(T_i)}$. Only one agent will commit to a given task. This served as a moderate upper bound for comparison.

Simulations ran for 200,000 timesteps, yielding a minimum of 2000 tasks per agent. Each combination of method, exploration technique, and scenario was run for 100 independent trials. Statistical significance was verified using an ANOVA with a Tukey Post-hoc Test at $p = 0.05$. Bounty hunters were required to bring back all prisoners alive: no disintegrations!

Metric. The metric we used for global system efficiency was the sum total bounty $h(t)$ over outstanding tasks at a given time t, that is, $h(t) = \sum_{i \in Q^{(t)}} b_i$. This metric nicely reflects two objectives at once. First, as tasks pile up, $h(t)$ will grow. Second, if tasks are left stagnant in the queue, their bounty increases and again $h(t)$ grows. In many scenarios, the sum total bounty can also be considered an indication of urgency: essentially, it is a measure of the number of fires to put out, and their importance.

How Fast Should Bounties Increase? If the bail bondsman increases the bounty too rapidly, agents might reasonably choose to wait rather than commit to a task. With multiple competing agents, such task speculation is less common, but with a single agent, speculation is easy to demonstrate.

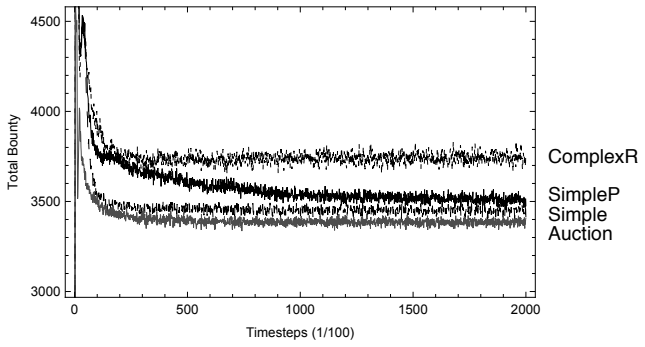

Figure 1: Experiment 1, Static Environment (Selected Results), 200,000 timesteps. Lower values are better.

Equivalence Classes	Method	γ	ε	Mean
+	Random	-	-	6139.72
+	ComplexR	0	0.1	3739.18
+	SimpleR	0	0.1	3641.62
+	ComplexP	0.001	0	3476.67
+	SimpleP	0.001	0	3475.81
+ +	Complex	0	0	3434.75
+ +	Simple	0	0	3408.04
+ +	Auction	-	-	3407.77
+ +	Exclusive	-	-	3403.4
+	Greedy	-	-	3372.64

Table 1: Experiment 1 results, Static Environment, at time=200,000. Lower mean values are better. Equivalence Classes show statistically insignificant differences between methods.

Imagine a scenario where a single agent is given the same task i over and over again. His only choice, then, is when to commit to the task. The longer he waits, the higher the bounty he receives on task completion. Let $b_{i,0}$ be the bounty for task i when it is first offered. Suppose the task takes $m_i > 0$ time to complete, and the bounty rises at a rate of $r_i \geq 0$ per timestep. If the agent waited for $n > 0$ time to begin the task, it would take $n + m_i$ time to finish it for a bounty of $b_{i,0} + nr_i$, yielding a bounty per timestep of $\frac{b_{i,0}+nr_i}{n+m_i}$. If he had started the task immediately, his bounty would be $b_{i,0}$, for a bounty per timestep of $\frac{b_{i,0}}{m_i}$. Thus we want $\frac{b_{i,0}+nr_i}{n+m_i} \leq \frac{b_{i,0}}{m_i}$ to discourage procrastination. Rearranging, we get $r_i \leq \frac{b_{i,0}}{m_i}$.

We have chosen to adopt this conservative rule. In our experiments, the maximum expected task length was approximately 100, and so the initial bounty on all task classes was set to 100 and increased at a rate of 1.

First Experiment: A Static Environment

The first experiment verified that the discussed methods could learn to adapt to the best task choices per-agent. Results are shown in Table 1, and selected results in Figure 1.

Results. The results made very clear that all techniques will converge to values near to our *Greedy* reference. We note an initial spike in bounty as the techniques spend time learning their best options.

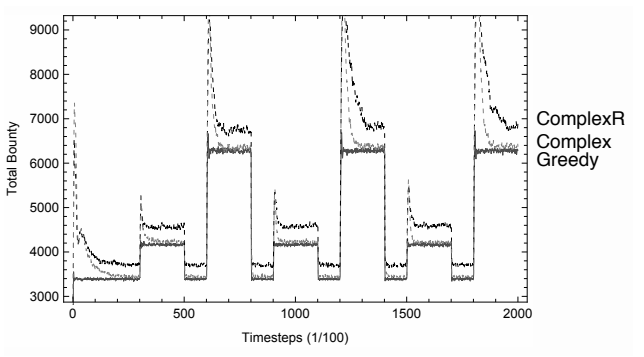

Figure 2: Experiment 2, Dynamic Agents (Selected Results), 200,000 timesteps. Lower values are better. Complex peaks exceed 9500, 10500, and 11500 respectively.

Equivalence Classes	Method	γ	ε	Mean
+	Random	-	-	11255.4
+	ComplexR	0	0.1	6904.13
+ +	SimpleR	0	0.1	6808.35
+ +	SimpleP	0.001	0	6572.01
+ +	ComplexP	0.001	0	6495.58
+ +	Simple	0	0	6437.8
+ +	Exclusive	-	-	6412.1
+ +	Complex	0	0	6383.45
+	Auction	-	-	6326.7
+	Greedy	-	-	6289.88

Table 2: Experiment 2 results, Dynamic Agents, at time=200,000. Lower values are better. Equivalence Classes show statistically insignificant differences between methods.

The best five methods were (in order but with statistically insignificant differences) *Greedy*, *Exclusive*, *Auction*, *Simple*, and *Complex*. While *Auction* and *Exclusive* outperformed our methods, the difference was statistically insignificant. We note that *Simple* (and to a lesser extent *Complex*) performed this well despite having no explicit exploration strategy. We also note here that *SimpleR* and *ComplexR* — which use a 10% random task exploration strategy — did poorly. This will be a continuing theme.

Second Experiment: Dynamic Agents

In the real world, agents are prone to failure. We tested each method's ability to adapt to a situation where agents were periodically removed from the game, then later reinstated. In this experiment, agent 1 was removed every 30,000 timesteps, and agent 2 was removed every 60,000 timesteps. Agents were reinserted 20,000 timesteps after removal. Ideally while agents were gone, teammates would adapt to cover for them.

Results. We discovered that all the methods would adapt quickly, as illustrated in Figure 2. The *Complex* method converged to the *Greedy* performance regardless of the number of agents in the game. This is verified in Table 2, which reflects the final timestep 200,000, when two agents were missing from the game.

We note in Figure 2 that *Complex* and *ComplexR* (shown Figure 2) had very high temporary peaks of poor performance compared to other methods whenever an agent would disappear: they had a larger

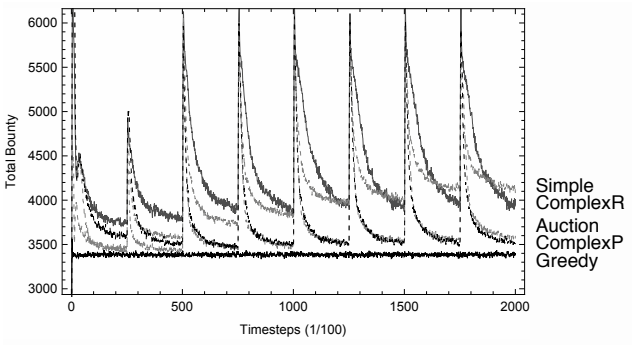

Figure 3: Experiment 3, Dynamic Tasks (Selected Results), 200,000 timesteps. Lower values are better.

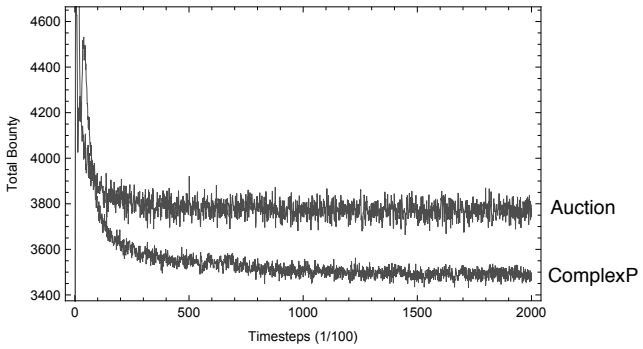

Figure 4: Experiment 4, Unreliable Collaborators (Selected Results), 200,000 timesteps. Lower values are better. *Exclusive* is omitted as its results are very similar to *Auction*.

Equivalence Classes	Method	γ	ε	Mean
+	Random	-	-	6035.74
+	Complex	0	0	4150.56
+	Simple	0	0	4086.08
+	ComplexR	0	0.1	3934.94
+	SimpleR	0	0.1	3928.61
+	Auction	-	-	3591.57
+	SimpleP	0.001	0	3578.96
+	Exclusive	-	-	3529.17
+	ComplexP	0.001	0	3509.76
+	Greedy	-	-	3394.73

Table 3: Experiment 3 results, Dynamic Tasks, at time=200,000. Lower values are better. Equivalence Classes show statistically insignificant differences between methods.

state table and would be expected to take longer to adapt. The 10% random task exploration strategy once again did poorly compared to other methods. Note that *Auction* outperformed *SimpleP* and *SimpleR*, but not *Complex*.

Third Experiment: Dynamic Tasks

If the distribution of tasks suddenly changes, we want the bounty system to recover. To test this, we occasionally rotated the corners among the four agents: that is, agent 1's corner would become agent 2's corner, whose old corner would now belong to agent 3, and so on. We did this every 25,000 timesteps, with a second rotation performed every 50,000 timesteps (a worst case scenario for task distribution, as the closest balls became the furthest and vice versa).

Results. This experiment, as shown in Figure 3 and Table 3, shows the weakness of relying solely on bounty for exploration: the *Simple* and *Complex* methods performed poorly. Following them were the remaining random exploration methods (*SimpleR*, and *ComplexR*). Finally, the best adaptive methods were *SimpleP*, *Auction*, *ComplexP*, and *Exclusive*. We note that, *SimpleP* and *ComplexP*, performed just as well as the *Exclusive* and *Auction* methods. Unfortunately, no adaptive method could consistently converge to *Greedy*'s performance.

Equivalence Classes	Method	γ	ε	Mean
+	Exclusive	-	-	7652.40
+	Auction	-	-	7334.31
+	ComplexP	0.001	0	5625.17

Table 4: Experiment 4 results, Unreliable Collaborators, at time=200,000. Lower values are better. Equivalence Classes show statistically insignificant differences between methods.

We also note that after several iterations of rotations, the agents were unable to converge to the same (lower) value. This is because the rate of rotating was too fast for the adaptive methods to catch up and so the total bounty would gradually pile up. This was especially true for the *Auction* strategy. In the first few iterations *Auctions* started out better than *ComplexP*, but by the last iteration, they were equivalent.

Fourth Experiment: Unreliable Collaborators

The purpose of Experiments 4 and 5 is to show when exclusivity can fail. In the previous experiments exclusivity was favorable, since non-exclusive approaches potentially wasted time on tasks another agent was completing. However, in these next experiments we show there are situations where non-exclusivity is desirable.

Suppose a system has agents which do not perform nearly as well as other agents. In this experiment, there were 2 agents, placed in the top left corner of the grid world, who moved 10x slower than other agents, and 4 agents, placed in the corners (like normal), who moved at a normal speed. We chose to test *ComplexP* as our main bounty mechanism, and compared it to our two auction-like algorithms (*Auction* and *Exclusive*).

Results. The results are shown in Table 4 and illustrated in Figure 4. *ComplexP* was clearly the best in this test. The obvious underlying reason was that agents at normal speed would win a race to a ball (a task) against a slow, unreliable collaborator. The auction-like methods simply had no way to prevent the unreliable collaborator from holding things up. Also, *ComplexP* was naturally suited to this task due to the added information it learned about the other agents in the environment.

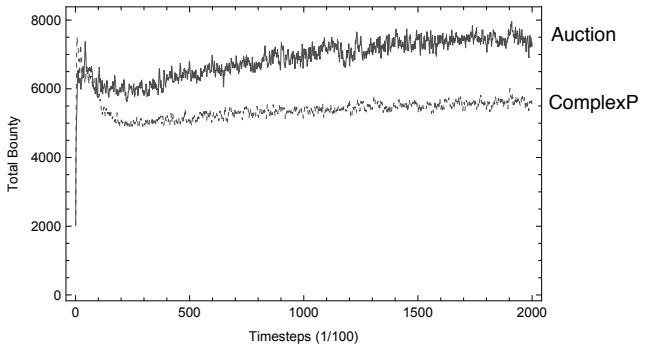

Figure 5: Experiment 5, Unexpectedly Bad Tasks (Selected Results), 200,000 timesteps. Lower values are better. *Exclusive* is omitted as its results are very similar to *Auction*.

Equivalence Classes	Method	γ	ε	Mean
+	Exclusive	-	-	3779.05
+	Auction	-	-	3762.39
+	ComplexP	0.001	0	3497.28

Table 5: Experiment 5 results, Unexpectedly Bad Tasks, at time=200,000. Lower values are better. Equivalence Classes show statistically insignificant differences between methods.

Fifth Experiment: Unexpectedly Bad Tasks

We now consider an experiment where some tasks suddenly change in difficulty for particular agents. We modified the scenario such that, every time a task class reappeared on the board, there was a 10% probability it would become "difficult" for a randomly-selected agent to complete. By "difficult" we mean that this agent, upon choosing this task, would move at one-tenth his normal speed.

We once again compared *ComplexP*, *Exclusive*, and *Auction*.

Results. The results are shown in Table 5 and illustrated in Figure 5. *ComplexP*, again, was the clear winner. Once again, the exclusivity of the auction-like methods prevented agents from taking over "difficult" tasks which have trapped helpless agents.

Sixth Experiment: Jumping Ship

We finally experimented with the effect of *jumping ship* on bounties, by which we mean that an agent may, at any time, abandon his current task and recommit to a different one, perhaps because its bounty is higher, or because a better agent has committed to the old one. If an agent abandons a task, he is teleported back to his corner, so as to counter speculation.

To discourage excess jumping, we modified each agent's estimate of remaining time left to complete his *current* task as $T_i' \leftarrow |T_i - t|$, where t is the amount of time the agent has spent on the task so far. Thus the closer the agent is to his expectation of completion, the more likely he would be to stay on the task. For the current task, T_i' substituted for T_i in the calculation of I^* (potential new tasks just used T_i).

We did not test jumping ship during learning, but rather after learning was completed. Thus we ran the *Complex* method in the Static Environment for 200,000 timesteps, and then turned off learning ($\alpha = 0,\ \beta = 0$) and turned *on* the ability to jump ship. We then ran the agents for 100,000 more timesteps.

Results. The results were not good. Jumping ship resulted in a total bounty of 4178.08 at timestep 300,000, compared to 3557.72 prior. This result was statistically significant.

CONCLUSIONS

We have demonstrated methods for agents to adapt to their best tasks in a bounty hunter system, thus improving the efficiency of the whole system. We have compared various approaches to adaptation, as well as both exclusive and non-exclusive task allocation strategies.

As we had imagined, the "complex" non-exclusive methods performed better in dynamic situations due to their additional per-agent information, since they retained knowledge of missing agents and could immediately re-use that information when the agents were re-introduced. While the exclusive and "auction-like" methods generally performed well in the static and basic dynamic scenarios, Experiments 4 and 5 showed that they would perform very poorly in situations where task exclusivity could saddle agents with leaden tasks. Without any additional information or communication, non-exclusive bounty agents naturally adapted to these situations. In the future, we will look at different scenarios which take advantage of this natural adaptation.

A bounty hunter system is a promising method for task allocation: it seems robust to loss of agents, changes in task difficulty, and other kinds of noise. The key properties of a bounty hunter system which distinguish it from other methods, and permits this robustness, is its non-exclusivity of task assignment, lack of a bidding structure, and the increasing attractiveness of outstanding tasks as the bounty on them rises. For this reason, it is surprising bounties have not been studied more: we hope this trend will change in the future.

ACKNOWLEDGMENTS

The work presented in this paper is supported by NSF NRI Grant 1317813.

REFERENCES

[1] S. C. Botelho and R. Alami. M+: a scheme for multi-robot cooperation through negotiated task allocation and achievement. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 1234–1239, 1999.

[2] Adam Campbell, Annie S. Wu, and Randall Shumaker. Multi-agent task allocation: Learning when to say no. In *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation*, pages 201–208, 2008.

[3] Pablo Casas-Arce and F. Asís Martínez-Jerez. Relative performance compensation, contests, and dynamic incentives. *Management Science*, 55(8):1306–1320, 2009.

[4] S.H. Choi and W.K. Zhu. A closed-loop bid adjustment method of dynamic task allocation of robots. In *Electrical Engineering and Intelligent Systems*, volume 130 of *Lecture Notes in Electrical Engineering*, pages 81–94. Springer, 2013.

[5] Brian L. Connelly, Laszlo Tihanyi, T. Russell Crook, and K. Ashley Gangloff. Tournament theory thirty years of contests and competitions. *Journal of Management*, 40(1):16–47, 2014.

[6] Luis C. Corchón. The theory of contests: a survey. *Review of Economic Design*, 11(2):69–100, 2007.

[7] T. S. Dahl, M. J. Matarić, and G. Sukhatme. Multi-robot task-allocation through vacancy chains. In *IEEE International Conference on Robotics and Automation*, volume 2, pages 2293–2298, 2003.

[8] M. Bernardine Dias. *Traderbots: A New Paradigm for Robust and Efficient Multirobot Coordination in Dynamic Environments*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, 2004.

[9] Alessandro Farinelli, Luca Iocchi, Daniele Nardi, and Fabio Patrizi. Task assignment with dynamic token generation. In *Monitoring, Security, and Rescue Techniques in Multiagent Systems*, pages 467–477. Springer, 2005.

[10] B. P. Gerkey. A formal analysis and taxonomy of task allocation in multi-robot systems. *International Journal of Robotics Research*, 23(9):939–954, 2004.

[11] B. P. Gerkey and M. J. Matarić. Sold!: auction methods for multirobot coordination. *IEEE Transactions on Robotics and Automation*, 18(5):758–768, 2002.

[12] Eric Helland and Alexander Tabarrok. The fugitive: Evidence on public versus private law enforcement from bail jumping. *Journal of Law and Economics*, 47(1):93–122, 2004.

[13] E. Gil Jones, B. Browning, M. Bernardine Dias, B. Argall, M. Veloso, and A. Stentz. Dynamically formed heterogeneous robot teams performing tightly-coordinated tasks. In *IEEE International Conference on Robotics and Automation*, pages 570–575, 2006.

[14] E Gil Jones, M Bernardine Dias, and Anthony Stentz. Learning-enhanced market-based task allocation for oversubscribed domains. In *International Conference on Intelligent Robots and Systems*, pages 2308–2313, 2007.

[15] K.A. Konrad. *Strategy and dynamics in contests*. LSE perspectives in economic analysis. Oxford University Press, 2009.

[16] G. Ayorkor Korsah, A. Stentz, and M. Bernardine Dias. A comprehensive taxonomy for multi-robot task allocation. *International Journal of Robotics Research*, 32(12):1495–1512, 2013.

[17] M. G. Lagoudakis, M. Berhault, S. Koenigt, P. Keskinocak, and A. J. Kleywegt. Simple auctions with performance guarantees for multi-robot task allocation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, volume 1, pages 698–705, 2004.

[18] Lantao Liu, Nathan Michael, and Dylan Shell. Fully decentralized task swaps with optimized local searching. In *Proceedings of Robotics: Science and Systems*, 2014.

[19] Lantao Liu and Dylan Shell. A distributable and computation-flexible assignment algorithm: From local task swapping to global optimality. In *Proceedings of Robotics: Science and Systems*, 2012.

[20] Maja J. Matarić, Gaurav S. Sukhatme, and Esben H. Østergaard. Multi-robot task allocation in uncertain environments. *Autonomous Robots*, 14(2-3):255–263, 2003.

[21] Barry J. Nalebuff and Joseph E. Stiglitz. Prizes and incentives: towards a general theory of compensation and competition. *The Bell Journal of Economics*, pages 21–43, 1983.

[22] M. Nanjanath and M. Gini. Dynamic task allocation for robots via auctions. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 2781–2786, 2006.

[23] Maitreyi Nanjanath and Maria Gini. Repeated auctions for robust task execution by a robot team. *Robotics and Autonomous Systems*, 58(7):900–909, 2010.

[24] L. E. Parker. L-ALLIANCE: a mechanism for adaptive action selection in heterogeneous multi-robot teams. Technical Report ORNL/TM–13000, Oak Ridge National Laboratory, 1995.

[25] L. E. Parker. ALLIANCE: an architecture for fault tolerant multi-robot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.

[26] Charles E. Pippin and Henrik Christensen. Learning task performance in market-based task allocation. In *Proceedings of the 12th International Conference on Intelligent Autonomous Systems*, volume 2, pages 613–621, 2012.

[27] A Pustowka and E.F. Caicedo. Market-based task allocation in a multi-robot surveillance system. In *Robotics Symposium and Latin American Robotics Symposium*, pages 185–189, 2012.

[28] Jeff Schneider, David Apfelbaum, J. Andrew (Drew) Bagnell, and Reid Simmons. Learning opportunity costs in multi-robot market based planners. In *IEEE International Conference on Robotics and Automation*, pages 1151–1156, 2005.

[29] P. M. Shiroma and M. F M Campos. Comutar: A framework for multi-robot coordination and task allocation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4817–4824, 2009.

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

[31] Christian Terwiesch and Yi Xu. Innovation contests, open innovation, and multiagent problem solving. *Management science*, 54(9):1529–1543, 2008.

[32] Robert Zlot, A. Stentz, M. Bernardine Dias, and Scott Thayer. Multi-robot exploration controlled by a market economy. In *IEEE International Conference on Robotics and Automation*, 2002.