Multiagent Learning for Black Box System Reward Functions [*]

Kagan Tumer
Oregon State University
Oregon 97331, USA
kagan.tumer@oregonstate.edu

Adrian Agogino
UCSC, NASA Ames Research Center, Mailstop 269-3
Moffett Field, California 94035, USA
adrian@email.arc.nasa.gov

Abstract

In large, distributed systems composed of adaptive and interactive components (agents), ensuring the coordination among the agents so that the system achieves certain performance objectives is a challenging proposition. The key difficulty to overcome in such systems is one of credit assignment: How to apportion credit (or blame) to a particular agent based on the performance of the entire system. In this paper, we show how this problem can be solved in general for a large class of reward functions whose analytical form may be unknown (hence "black box" reward). This method combines the salient features of global solutions (e.g., "team games") which are broadly applicable but provide poor solutions in large problems, with local, but aligned solutions (e.g., "difference rewards") which learn quickly, but can be computationally burdensome. We introduce two estimates for the difference reward for a class of problems where the mapping from the agent actions to system reward functions can be decomposed into a linear combination of nonlinear functions of the agents' actions. We test our method's performance on a distributed marketing problem and an air traffic flow management problem and show a 44% performance improvement over team games and a speedup of order n for difference rewards (for an n agent system).

[*]Appears in *Advances in Complex Systems*, Vol 12, pp. 493–512, 2009.

1 Introduction

The ability of a team of agents to learn distributed policies has been demonstrated successfully in numerous domains such as controlling multiple robots, aggregating information from distributed data sources and distributed system administration [14, 27, 30]. While diverse, each of these domains share two important properties fundamental to interesting distributed learning problems: 1) each agent learns its own set of actions (policy), 2) each policy is trying to maximize a system reward that is a nonlinear function of all the policies, thus coupling the policies together. This type of problem is best described as a multiagent learning problem, where each agent, i, takes an action z_i and tries to maximize a reward function, $G(z)$, that is a function of z, the actions of all the agents [35, 33, 39, 36].

When the agent actions need to be coordinated, this issue becomes particularly challenging due to the structural credit assignment problem [1, 22, 37, 38]. In this problem, credit must be assigned to a particular agent based on the performance of the full system. For example, when an agent takes an action and G improves, the agent needs to determine whether its action was (partly) responsible for that improvement. Though lengthy learning trails can statistically eliminate the impact of other agents on G, such an approach is not practical for large systems. If G is linearly separable in the agents' actions, this credit assignment problem is trivial as each agent can maximize its own separate component of that reward. In contrast if G depends on all the agents' actions directly, such as the parity problem, finding an adequate distributed solution is nearly impossible, and the problem needs to be reformulated. In this paper we focus on problems where moderate numbers of agents need to coordinate their actions with one another to reach satisfactory values of G.

For systems with few agents, this credit assignment problem can be sidestepped, and all agents can use G directly. However, when the number of agents in a system increase, this method breaks down and agents need to receive a reward that accounts for their contribution to the system. The "difference reward" provides such a reward, and has produced good results in many domains [5, 31, 33, 34]. However, as currently expressed, the difference reward requires knowledge of the functional form of the system reward.

In this paper we present an approach that lifts this requirement that uses two estimates of the difference reward that retains its fast learning characteristics, but does not require full knowledge of the functional form of $G(z)$. In the next section, we briefly describe the related work. Section 3 describes the system reward structure and the basic difference used in multiagent learning. Section 4 derives two estimates for the difference reward that allows its application to domains with G of unknown functional form. Section 5 presents experimental results in both a distributed marketing problem, and a complex air traffic flow problem. Section 6 discusses the mathematical implications and the future applications of the estimated difference rewards.

2 Related Work

In general work in multiagent learning can be grouped into one of two broad categories: (i) work leveraging domain knowledge; and (ii) general work applicable to a subset of the domains. Some of the most successful work in multiagent learning fall into the first category. In robotic soccer for example, player specific subtasks, followed by tiling provide good convergence properties [27]. In foraging robot coordination, specific rules induce good division of labor [19]. In a distributed air traffic control domain, a combination of positive rewards and penalty rewards allows a collection of aircrafts to navigate safely [15]. In all cases, the agent coordination is achieved through exploiting knowledge of the system dynamics and accentuating the known desirable interactions among the agents.

The second set of approaches provide general solutions to a subset of the problems. Early work on this topic focused on "team games" where each agent considers itself the only agent in the system and receives the full system reward. An example of this approach is the control of four elevators where a separate reinforcement learner was used to control each elevator, and each learner received the full system reward [10]. While such a "team game" approach is effective, it is restricted to domains with a small number of agents. In problems where groups of agents can be assumed to be independent, the task can be decomposed by learning a set of basis functions used to represent the value function, where each basis only processes a small number of the state variables [14]. Task decomposition has also been used in single agent RL using hierarchical reinforcement learning methods such as MAXQ value function decomposition [12]. In multiagent learning, Partially Observable Markov Decision Precesses (POMDPs) can be simplified through piecewise linear rewards [24]. In other cases agents can be assumed to be locally connected through a graph and can learn efficiently through local rewards [6]. Outside of reinforcement learning, mechanism design has been used with MDPs to address the issue of creating good agent incentives for specific types of rewards [25].

3 Agent and System Rewards

As stated in the introduction, in this paper we present a method to estimate difference rewards that does not require full knowledge of the functional form of the system reward G.

3.1 System Reward

In particular, we focus our study to the class of problems where system reward is in the form:

$$G(z) = G_f(f(z)) = G_f\left(\sum_i f_i(z_i)\right) \ , \tag{1}$$

where G_f is a known non-linear function and the f_is are unknown non-linear functions. Table 1 summarizes the functional form of G and its arguments.

Table 1: Functional Forms for System Objective Function

Function	Form		Argument
G	Unknown	Non-linear	z
G_f	Known	Non-Linear	f
f	Known	Linear	f_i
f_i	Unknown	Non-linear	z_i

The key assumption in this work is that the f_i cannot be sampled from the domain, but that $\sum_i f_i$ can be sampled (potentially at a high cost). This form of $G(z)$ applies to a large number of domains where agents have an unknown effect on their environment (f_i) and these effects are aggregated together. Such domains includer air (or highway) traffic flow management, distributed gating and distributed information gathering. While the agents do not know the f_is they do know how these aggregated effects contribute to the system goal in the form of G_f. Our estimate exploits this structure of G to create local rewards that allow learning to proceed significantly faster than directly using G and be applied to systems where the agent-specific rewards cannot be applied because the form of G is unknown.

3.2 Difference Reward

In a multiagent setting, while each agent can try to maximize the system reward directly, such an approach leads to slow/poor learning due to the structural credit assignment problem. An alternative is to have each agent attempt to maximize an agent-specific reward function derived in such a way that if agents succeed in maximizing that reward function, they collectively also maximize G. One such reward function is the **difference** reward function of the form [33]:

$$D_i \equiv G(z) - G(z - z_i + c_i) \ , \tag{2}$$

where z_i is the action of agent i, and c_i is an arbitrary "action" that does not depend on agent i's actions[1]. In the second term of D_i, $z - z_i + c_i$ represents the "counterfactual" states where the action of agent i, z_i, is replaced by a fixed action c_i that is independent of the agent's action.

There are two advantages to using D: First, the second term, $G(z - z_i + c_i)$, differs from the first term, $G(z)$, only in the actions of agent i. If agent i's action is not tightly coupled to the actions of the other agents, then the second term will subtract out much of the impact of the actions of the other agents in the system, therefore providing an agent with a "cleaner" signal than G. For instance if all the other agents choose poor actions, the impact of these actions would appear in both terms of D_i, and would mostly cancel out. This benefit has been dubbed "learnability" (agents have an easier time learning) in previous work [33]. Second, because the second term does not depend on the actions of agent i, any action taken by agent i that improves D, also improves G. Therefore we expect policies that maximize D will also maximize G. This specific form of difference reward has been effective in a number of domains including congestion problems, multi-rover policy evolution and bin-packing [2, 30, 33].

As an example, consider the application of this reward to a multi-robot coordination problem where multiple robots need to gather importance weighted information and maximize the total information collected by all the robots [5, 30]. In such a case, selecting a c_i that removes the robots' observations from the system, the difference reward measures the contribution of that robot to the system. Note, this is not equivalent to having each robot simply maximize the information it collects (which leads to poor system behavior) [5]. Instead, the difference reward leads to robots exploring areas that would not have been explored by other robots. That is, if a second robot would have observed a particular area, then the difference reward provides low values, urging the robot to find information with more value to the full system [5].

4 Estimates of Difference Rewards

Though providing a good compromise between aiming for system performance and removing the impact of other agents from an agent's reward, one issue that may plague D is computational cost. Because it relies on the computation of the counterfactual term $G(z-z_i+c_i)$ (i.e., the system performance without agent i) it may be difficult or impossible to compute, particularly when the exact mathematical form of G is not known.

For reward functions that are of the form given in equation 1 and summarized in Table 1, however, we can derive estimates for D that overcome this limitation. Our premise is that we can sample values from $f(z)$, enabling us to compute G, but that we cannot sample from each $f_i(z_i)$. In addition, we assume we may not be able to even compute $f(z)$ directly and must sample it from a "black box" computation (e.g., a system simulator) or measure it from the environment.

4.1 First Estimate

The key element in the computation of the difference reward is the counterfactual $G(z - z_i + c_i)$:

$$\begin{aligned} G(z - z_i + c_i) &= G_f(f(z - z_i + c_i)) \\ &= G_f\left(\sum_{j \neq i} f_j(z_j) + f_i(c_i)\right) \\ &= G_f\left(f(z) - f_i(z_i) + f_i(c_i)\right) \ . \end{aligned} \tag{3}$$

[1]This notation uses zero padding and vector addition rather than concatenation to form full state vectors from partial state vectors.

Unfortunately, we cannot compute this directly as the values of $f_i(z_i)$ are unknown. However, if agents take actions independently (i.e., they do not observe how other agents act before taking their own actions) we can take advantage of the linear form of $f(z)$ in the f_is with the following equality:

$$E(f_{-i}(z_{-i})|z_i) = E(f_{-i}(z_{-i})|c_i) \tag{4}$$

where $E(f_{-i}(z_{-i})|z_i)$ is the expected value of $f_{j \neq i}$ (all fs other than f_i) given the value of z_i and $E(f_{-i}(z_{-i})|c_i)$ is the expected value of $f_{j \neq i}$ given c_i. We then get the following estimate for $f(z - z_i + c_i)$:

$$\begin{aligned}
f(z - z_i + c_i) &= f(z) - f_i(z_i) + f_i(c_i) \\
&= f(z) - f_i(z_i) - E(f_{-i}(z_{-i})|z_i) \\
&+ f_i(c_i) + E(f_{-i}(z_{-i})|c_i) \\
&= f(z) - E(f_i(z_i)|z_i) - E(f_{-i}(z_{-i})|z_i) \\
&+ E(f_i(c_i)|c_i) + E(f_{-i}(z_{-i})|c_i) \\
&= f(z) - E(f(z)|z_i) + E(f(z)|c_i) \ .
\end{aligned} \tag{5}$$

Therefore we can evaluate $D_i = G(z) - G(z - z_i + c_i)$ as:

$$D_i^{est1} = G_f(f(z)) - G_f(f(z) - E(f(z)|z_i) + E(f(z)|c_i)) \ .$$

The first term of D_i^{est1} is the same as the original difference reward. The second term of D_i^{est1} tries to remove the impact of the other agents, but cannot do this as elegantly as the difference reward since the form of function $f(z)$ is not known. Instead of subtracting out $f_i(z_i)$ and adding $f_i(c_i)$ directly, we estimate this by taking the difference between average impact of action z_i of $f(z)$ and the average impact of action c_i on $f(x)$. This leaves us with the task of estimating the values of $E(f(z)|z_i)$ and $E(f(z)|c_i))$. These estimates can be computed by keeping a table of averages where we average the values of the observed $f(z)$ for each value of z_i that we have seen. Note, this estimate improves as the number of samples increases.

4.2 Second Estimate

The discussion above is generally applicable to any selection of c_i. We can improve this estimate if we set $c_i = E(z_i)$ and make the mean squared approximation of $f_i(E(z)) \approx E(f_i(z))$. The last expectation in D_i^{est1} is transformed as follows:

$$\begin{aligned}
E(f(z)|c_i) &= E\left((f_i(z_i) + \sum_{j \neq i} f_j(z_j))|E(z_i)\right) \\
&= E(f_i(z_i)|E(z_i)) + \sum_{j \neq i} E(f_j(z_j)) \\
&= E(f_i(E(z_i))) + \sum_{j \neq i} E(f_j(z_j)) \\
&\approx E(E(f_i(z_i))) + \sum_{j \neq i} E(f_j(z_j)) \\
&= \sum_j E(f_j(z_j)) \\
&= E(f(z)) \ .
\end{aligned} \tag{6}$$

We then we can estimate $G(z) - G(z - z_i + c_i)$ as:

$$D_i^{est2} = G_f(f(z)) - G_f(f(z) - E(f(z)|z_i) + E(f(z))) \ .$$

The estimate D_i^{est2} is the same as D_i^{est1}, except that $E(f(z)|c_i)$ has been replaced with $E(f(z))$. This formulation has two advantages over D_i^{est1}: First, there are more samples at our disposal to estimate $E(f(z))$ than we do to estimate $E(f(z)|c_i))$. Second, this removes the need to select a value for c_i. Since selecting a value of c_i that will lead to high performance can be difficult in some domains, it can be advantageous to have this parameter removed.

5 Experimental Results

To test the effectiveness of the difference reward and its estimates, we conduct a series of experiments in two domains. The first domain is an illustrative example in the form of a distributed marketing problem, where separate marketing agents try to market a common resource to distinct groups of potential customers. The second domain tests the performance of our reward system in a complex air traffic flow domain, where we use the FACET air traffic simulator to test the ability of learning agents to create policies that reduce congestion while minimizing delays [8]. In all experiments we test the performance of five different methods. The first method is Monte Carlo estimation, where random policies are created, with the best policy being chosen. The other four methods are based on reinforcement learning agents where the agents are maximizing one of the following rewards:

1. The system reward, $G(z)$;
2. The actual difference reward, $D_i(z)$;
3. The first difference reward estimate, $D_i^{est1}(z)$; and
4. The second difference reward estimate, $D_i^{est2}(z)$.

In these experiments, the aim of each agent is to learn to take actions that will lead to the best system performance, G. To form policies, each agent uses an agent-specific reward function and tries to maximize it with its own reinforcement learner [20] (though alternatives such as evolving neuro-controllers are also effective [30]). To clearly illustrate the benefit of the reward estimates, in this paper we focus on domains that only need to utilize immediate rewards. As a consequence, simple table-based immediate reward reinforcement learning is used. The reinforcement learner is equivalent to an ϵ-greedy Q-learner with a discount rate of 0 [20]. In all the experiments the learning rate is equal to 0.5 and ϵ is equal to 0.25. Note that in many domains, reinforcement learning needs to look at rewards beyond the immediate reward and address a temporal credit assignment problem of how to reward a current action for a sequence of future rewards. Difference rewards have been shown to address both the structural and temporal credit assignment problems for domains where the functional form of G is known [4].

To make the agent results comparable to the Monte Carlo estimation, the best policies chosen by the agents over a single trial are used in the results. Monte Carlo and similar random approaches are common in complex air traffic problems [21]. All results are an average of thirty independent trials with the differences in the mean $(\sigma/\sqrt{n})$ shown as error bars, though in most cases the error bars are too small to see.

5.1 Distributed Marketing Problem

The first domain we study is a distributed marketing problem where a number of agents need to choose a strategy, and their reward depends on the strategies of all the agents. This is a form of congestion game where particular joint actions lead to desirable or undesirable behavior based on the number of other agents that have selected that particular action [4, 9, 16, 18, 32, 41].

5.1.1 Problem Description

In the "Marketing Problem" there are n agents marketing a constrained resource to n different demographics. Examples include marketing a resort hotel to several different parts of the country, or a public transportation system to different cities in a metropolitan area. In this problem each agent has a finite set of marketing strategies. The action of agent, i, is to choose a strategy z_i. The number of people who use the resource is an unknown nonlinear function of the marketing of all the agents, $f(z)$. After each training episode the value of $f(z)$ is measured. We assume that the marketers are targeting disjoint groups so $f(z)$ has the following form: $f(z) = \sum_i f_i(z_i)$, where $f_i(z_i)$ is a non-linear function of agent i's marketing action. The function $f(z)$ represents the aggregate sum of the effects of all the agents marketing actions.

This form represents situations where marketers do not have a model for the effects of their marketing, so the function $f_i(z_i)$ is unknown. In addition the value of $f_i(z_i)$ is never measured as we only have measurements of the aggregate number of people using the resource, $f(z)$. The system goal is to have the optimal amount of people use the resource. More revenue is gained by having more people use it, but we do not want it to become overused as that will hurt our reputation (e.g. overuse a transportation domain would result in congestion). The system goal is represented by a known non-linear function of the total number of people using the resource: $G(z) = G_f(f(z))$. Note that while the functional form of G_f is known, the form of G (in terms of z) is not, since $f(z)$ is unknown.

5.1.2 Results

We conducted a series of experiments where agents choose one of M marketing actions, where an action deterministically resulted in zero to C customers using the resource, depending on the action. For each agent the mapping from action to customer response $f_i(z)$ was chosen at random at the start of the experiment. The function G_f was set to $ke^{-k/c}$ where $k = f(z)$ is the aggregate number of customers that use the resource and c is the optimal capacity for the resource.

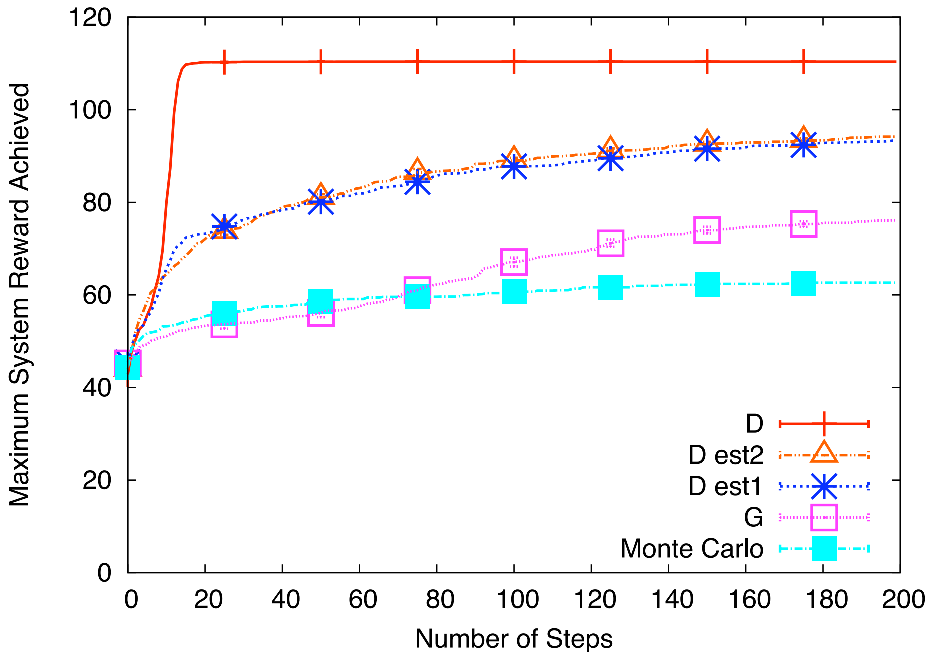


Figure 1: Marketing experiment with 100 agents. The estimates for D perform better than G, though not as well as the full D, which is not "computable" in many real world domains.

Figure 1 shows the performance of the five different methods in a marketing problem with one hundred agents, where $M = 10$ and $C = 18$. Monte Carlo (MC) optimization provides a baseline solution. Agents using G directly as their reward perform slightly better than MC. But in this case, each agent's reward is affected by the actions of the other 99 other agents, making it hard for an agent to discern the effects of its action on its reward. In contrast agents using the true difference reward learn fast and learn well. However, this reward is not directly computable when agents do not know the functional form of $f(z)$ needed to compute the counterfactual $G_f(f(z - z_i + c_i))$. This therefore is a theoretical result that cannot be implemented in many real domains. The results for the estimates to the difference rewards (described in Section 4), where an agent only needs to know the value of $f(z)$, are promising as they outperform agents using G.

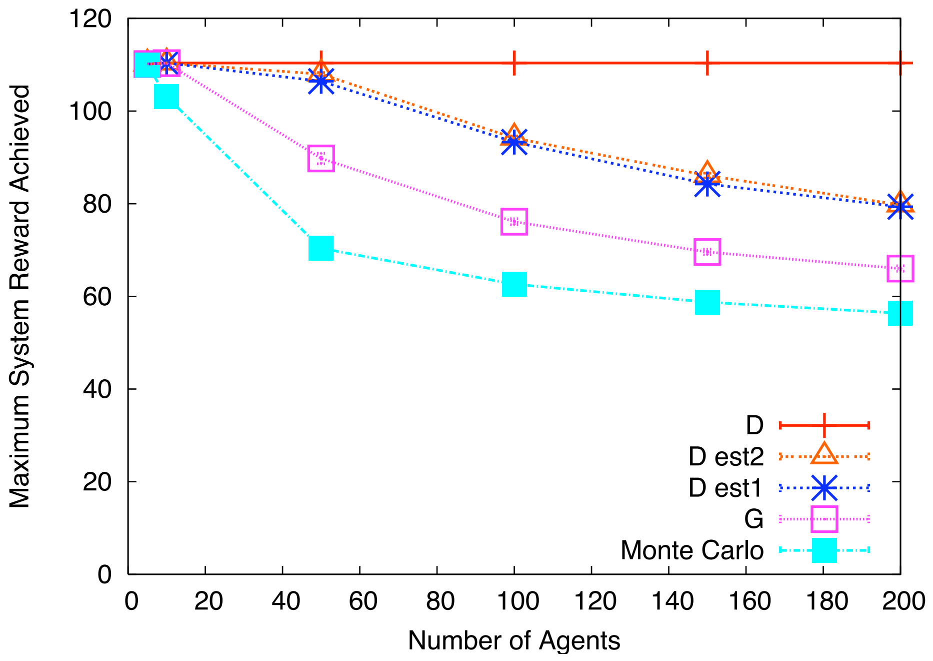


Figure 2: Marketing experiment scaling after 200 episodes. The estimates for D degrade more gracefully than G.

One interesting question that arises concerns the convergence properties of agents using the different rewards. Unfortunately analysis of convergence in a multiagent problem is difficult given non-linear interactions between the agent learning algorithms and the reward function. In theory, the difference rewards are shown to converge to (potentially local) minima as long as the system reward converges [33]. In practice, the performance of the agents to not change significantly after 200 learning steps, in these experiments.

Figure 2 shows the scaling results for the number of agents ranging from 1 to 200. These results show that the relative performance of the algorithms is not affected by the number of agents. Note that the true difference reward has remarkably good scaling characteristics as its performance does not degrade as the number of agents is increased from five agents to two hundred agents, making it a good choice for large domains where the functional form of G is known.

5.1.3 Computational Cost of D and D^{est}

The results above show the performance of the different algorithms after a specific number of episodes, demonstrating that D performs significantly better than the other algorithms. In domains where the functional form of G is known, D can often be computed without explicit calls to G [2]. However, if the agents are unable to streamline their computation of D, agents using the difference reward may be forced to make many computations of G. In general, for n agents, that means D gets n times as many G function calls. Table 2 shows the relative performance for a given number of G evaluations. The reward D performs best when used over the full two hundred episodes, but requires 4000 computations of G. The two estimates to D provide the best compromise between performance and computational cost, outperforming both D and G for a given number of G evaluations. Note that in cases where G is only computed by sampling the environment, it may not even be possible to compute D at any computational cost and the estimates will have to be used as discussed in Section 6.

Table 2: Marketing Experiment with 100 Agents, after 200 G evaluations (except for D^{4000} which has 4000 G evaluations at episode 200).

Reward	G	$\sigma/\sqrt{n}$	steps
D^{est2}	94.2	0.5	200
D^{est1}	93.3	0.5	200
D	52.6	0.8	2
D^{4000}	110.4	0.0003	200
G	76.1	0.7	200
MC	62.6	0.6	200

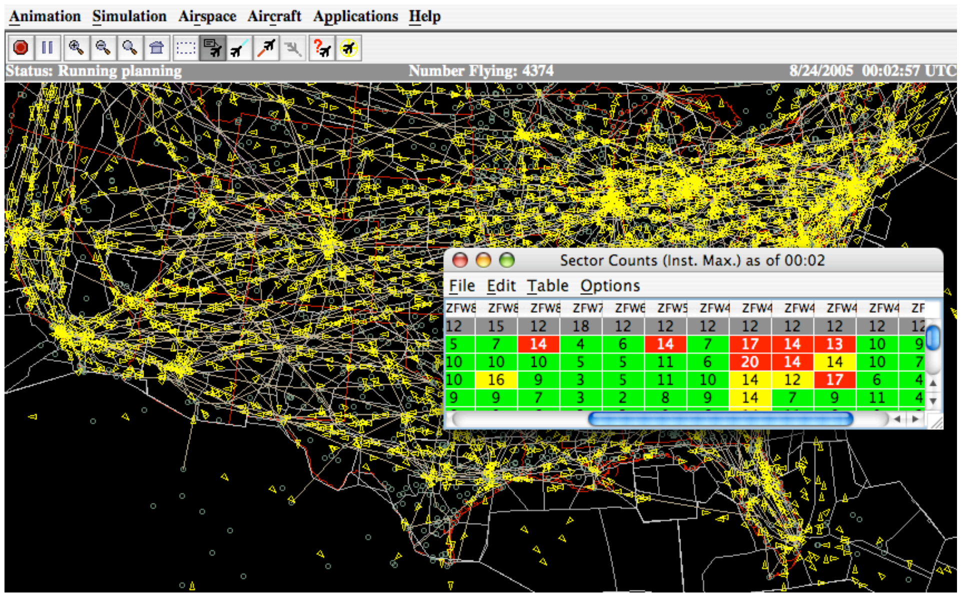
5.2 Air Traffic Flow Problem

The second domain we study is the complex domain of air traffic flow management [3, 23, 26, 29, 31]. This is a complex real world problem, where the agent actions cannot be directly be mapped to a system reward in analytical form, creating a "black box" reward for the learning system.

5.2.1 Problem Description

In this section we summarize how distributed learning agents can learn to manage air traffic flow [31]. First, we will assign agents to airspace locations called "fixes" to map the air traffic problem to a multiagent problem. Each agent is responsible for any aircraft going through its fix [31, 3]. The action of an agent is to determine the separation (distance between aircraft) that aircraft have to maintain, when going through the agent's fix (though aircraft will always keep a safe distance, d_s, if d is set too low). The effect of issuing higher separation values is to slow down the rate of aircraft that go through the fix. By increasing the value of d, an agent can limit the amount of air traffic downstream of its fix, reducing congestion at the expense of increasing the delays upstream.

Second, we will use FACET (Future ATM Concepts Evaluation Tool, where ATM stands for Air Traffic Management) to simulate air traffic and determine the impact of the agents' actions [8]. FACET simulates air traffic based on flight plans and through a graphical user interface allows the user to analyze congestion patterns of different sectors and centers (Figure 3). FACET also allows the user to change the flow patterns of the aircraft through a number of mechanisms, including "metering" aircraft. Metering is performed by choosing a "Miles in Trail" (MIT) value, which specifies the minimum distance that aircraft may be spaced from each other when passing through a particular location. Larger MIT values cause aircraft to be spaced further apart. In this paper, agents send scripts to FACET asking it to simulate air traffic based on metering orders imposed by the agents. The agents then produce their rewards based on received feedback from FACET about the impact of these meterings.



Status: Running planning Number Flying: 4374 8/24/2005 00:02:57 UTC

Sector Counts (Inst. Max.) as of 00:02

ZFW8	ZFW8	ZFW8	ZFW7	ZFW6	ZFW5	ZFW4	ZFW4	ZFW4	ZFW4	ZFW4	ZF
12	15	12	18	12	12	12	12	12	12	12	12
5	7	14	4	6	14	7	17	14	13	10	9
10	10	10	5	5	11	6	20	14	14	10	7
10	16	9	3	5	11	10	14	12	17	6	4
9	9	7	3	2	8	9	14	7	9	11	4

Figure 3: FACET screen-shot displaying traffic routes.

Finally, we will define a system reward function that focuses on the amount of congestion in a particular sector and on the amount of measured air traffic delay. This is measured as a function of the agents' action vector z, specifying the MIT values chosen by the agents. More precisely, we have:

$$G(z) = -((1-\alpha)B(z) + \alpha C(z)) \ , \tag{7}$$

where $B(z)$ is the total delay penalty for all aircraft in the system, and $C(z)$ is the total congestion penalty, and α determines the relative importance of these two. Neither $B(z)$, nor $C(z)$ can be analytically computed by an agent. Rather, they are computed after the number of aircraft in a sector are computed.

With $\alpha = 0.5$, for the two-congestion problem in our experiments we used an instance of this reward function described in detail in [31] and summarized as follows:

$$\begin{aligned} G(z) = \quad & - \quad A_1 \sum_i \sum_t u(t - T_i) k_{t,i}(t - T_i) \\ & - \quad A_2 \sum_i \sum_t u(k_{t,1} - C_i) e^{\beta(k_{t,1} - C_i)} \end{aligned} \tag{8}$$

where t is time, $k_{t,i}$ is the number of aircraft in congestion i, and $u(t)$ is the unit step function. T_i is the delay penalty constant ($T_1 = 200$ and $T_2 = 175$ here) and C_i is the congestion penalty constant ($C_1 = 18$, and $C_2 = 15$ here). A_1 and A_2 are scaling factors for the delay and congestion terms ($A_1 = \frac{1}{2}$ and $A_2 = 50$), and $\beta = 0.3$. The values of $k_{t,i}$ are computed by FACET and are affected by the actions of the agents as described in the following section. The term β is a user defined constant controlling the penalty curve for congestion. Note that G cannot be expressed in closed form in terms of the actions of the agents, since the effect of those actions of the congestion ($k_{t,j}$) isn't known in closed form.

5.2.2 Results

We tested the performance of the different rewards on an air traffic domain with 300 aircraft. The aircraft go through two points of congestion over a four hour simulation, with 200 going over one point of congestion and 100 going over the other point of congestion. The second congestion is less severe than the first one, so agents have to form different policies depending which point of congestion they are influencing. The points of congestion are created by setting up a series of flight plans that cause the number of aircraft in the sectors of interest to be significantly more than the number allowed by the FAA.

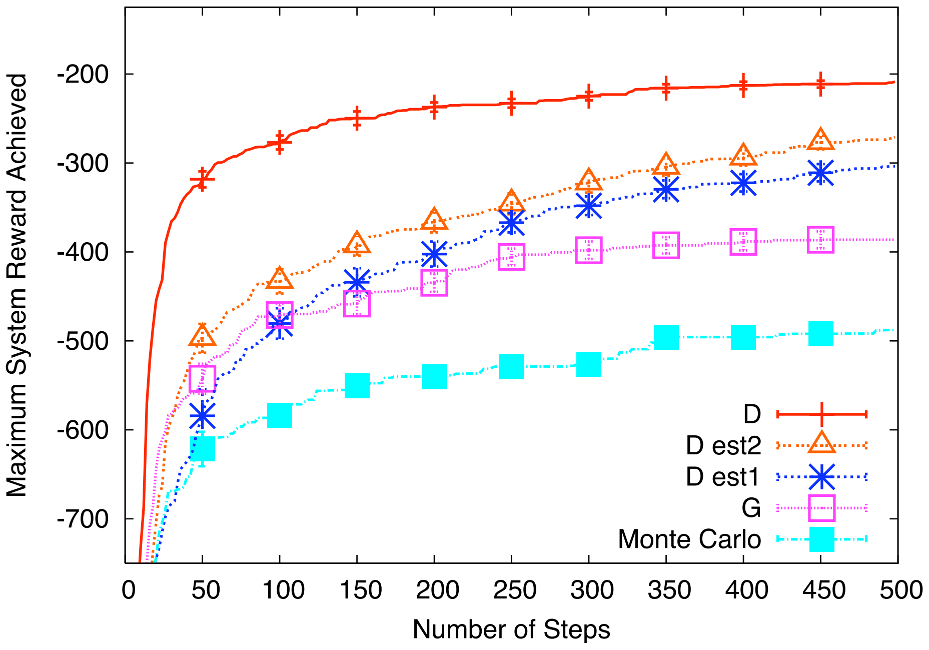


Figure 4: Performance with 300 Aircraft, 20 Agents. The estimates for D perform better than G, though not as well as the full D, which is computationally expensive in this domain.

The results displayed in Figure 4 show that the relative performance of the five methods is similar to the Marketing Problem. However, in this case D^{est2} performs better than D^{est1}. This is caused by the limited amount of data available in this domain and that D^{est2} draws from a larger sample to estimate D, resulting in a cleaner signal. Figure 5 shows scaling results for the number of agents varying from 10 to 50 and shows that the conclusions are not sensitive to the number of agents. Agents using D^{est2} perform slightly better than agents using D^{est1} in all cases but for 40 and 50 agents where they are statistically equivalent. While adding more fixes increases the amount of control the agents have over the system, this increase does not necessarily improve performance. The main issue is that when the number of fixes grows in this problem, the number of aircraft going through each fix decreases. This could result in certain fixes in superior positions to control less aircraft, causing a reduction in performance.

5.2.3 Computational Cost of D and D^{est}

As was the case for the Marketing domain, the results above show that D is superior to the other algorithms. However, in the air traffic domain, D can only be computed with additional calls to the FACET simulator, which come at significant computational cost. The computation cost of the system reward, G (Equation 7) is almost entirely dependent on the computation of the airplane counts for the congestions k_t, which need to be computed using FACET[2]. Except when D is used, the values of k are

[2]In our simulations a computation from FACET took 900 milliseconds, while all the other computation for all 20 agents in an episode took a combined 5 milliseconds.

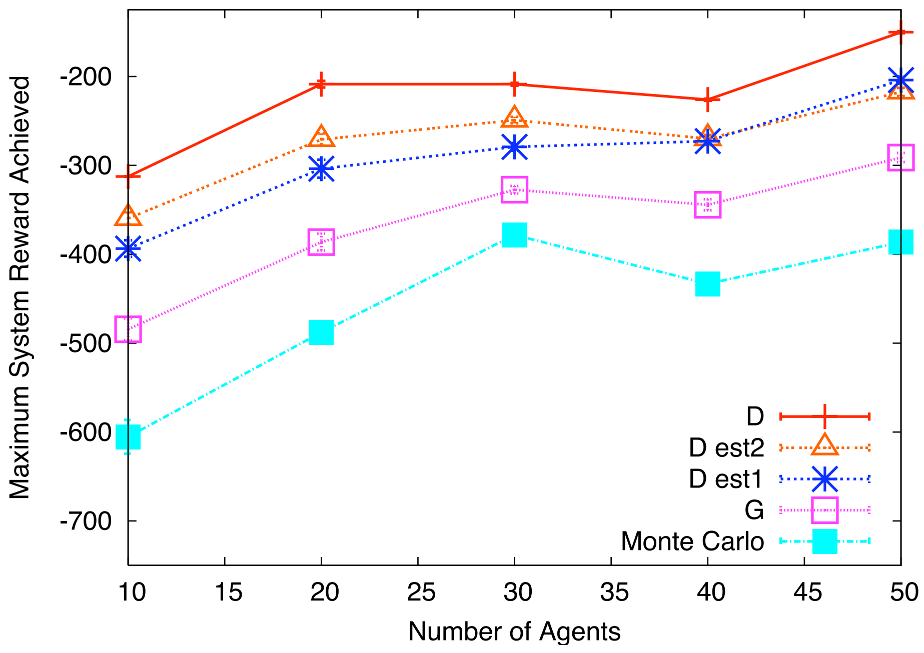


Figure 5: Impact of number of agents on system performance with 300 Aircraft. Performance improves with higher number of agents, but only if the algorithms and agents rewards can "extract" the extra information.

computed once per episode. However, to compute the counterfactual term in D, if FACET is treated as a "black box", each agent has to compute its own values of k for their counterfactual resulting in $n+1$ computations of k per episode.

Table 3: System Performance for 20 Agents, 300 aircraft, after 2100 G evaluations (except for D^{44K} which has 44100 G evaluations at step 2100).

Reward	G	$\sigma/\sqrt{n}$	steps
D^{est2}	-232.5	7.55	2100
D^{est1}	-234.4	6.83	2100
D	-277.0	7.8	100
D^{44K}	-219.9	4.48	2100
G	-412.6	13.6	2100
MC	-639.0	16.4	2100

Table 3 shows the performance of the algorithms after 2100 G computations for each of the algorithms for the simulations presented in Figure 4 where there were 20 agents, 2 congestions. All the algorithms except the fully computed D reach 2100 k computations at time step 2100. D however computes k once for the system, and then once for each agent, leading to 21 computations per time step. It therefore reaches 2100 computations at time step 100. We also show the results of the full D computation at t=2100, which needs 44,100 computations of k as D^{44K}. Although D^{44K} provides the best result by a slight margin, it is achieved at a considerable computational cost. Indeed, the performance of the two D estimates is remarkable in this case as they were obtained with about twenty times fewer computations of k. Furthermore, the two D estimates, significantly outperform the full D computation for a given number of computations of k and validate the assumptions made in Section 4. This shows that for this domain, in practice it is more fruitful to perform more learning steps and approximate D, than few learning steps with full D computation when we treat FACET as a black box.

6 Discussion

Learning multiagent policies is difficult due to the structural credit assignment problem of how to credit an action's contribution to a system reward, which is a function of many actions. Furthermore, the mapping from agent actions to system reward cannot always be computed in closed form. This paper proposes to address this issue using an estimate to a "difference reward" where agents learn using an agent-centric reward that promotes coordination. On a marketing problem and an air traffic flow problem, experimental results show that our method provides an improvement in performance by up to 44% over team games and difference rewards (when computational cost is taken into account).

Whether the difference reward or its estimate should be used depends on what is known about the functional form of the system reward, and how much it costs to compute. We are interested in three main types of system reward:
1) The system reward has a functional form that is completely known;
2) The system reward is a black box with high computational costs; or
3) The system reward is sampled from the environment, where we cannot demand samples for arbitrary actions.

The air traffic flow management problem is an instance of the second type of problem, since the FACET simulator can be used as a black box to retrieve values of $f(z)$, but at an extremely high computational cost. In this case agents should use the estimate of the difference reward to save computational costs. The marketing problem is an instance of the third type because agents do not know f_i (i.e. they do not know how their actions affect their target audience). In this case the agents can only count the aggregate number of people affected by all the agents, so they must use the estimate to the difference reward, since the true difference reward cannot be computed. (Note that the marketing problem would be the first type of problem if the values of f_i were known ahead of time and the difference reward could be computed in closed form. In this case the true difference reward can be used.)

This work provided the groundwork for multiagent learning in domains where the system reward is not known in closed form. There are three promising extensions of this work: First, the manner in which the estimate for $f(z)$ used in the difference rewards is computed can be improved. Currently we use simple averaging, though using data aging or similarity measure to provide a weighted average can improve the estimate. Second, the functional form of the system rewards can be extended beyond that given in Equation 1, and use more general machine learning methods to estimate the difference reward. Third, the difference reward estimates are now restricted by the form of G. Blending imperfect models of the environment with true samples in order to compute the difference reward would increase both the speed and the accuracy of the estimates. We are currently investigating all three avenues of research and extending the application domains to include robotic exploration and more realistic forms of the air traffic flow problem (including the role of human air traffic controllers).

Acknowledgments: The authors thank Dr. Banavar Sridhar and Shon Grabbe for their help with air traffic flow management and the FACET simulator. This work was partially supported by AFOSR grant FA9550-08-1-0187 and NSF grant 0931591.

References

[1] Agogino, A. and Tumer, K., Unifying temporal and structural credit assignment problems, in *Proceedings of the Third International Joint Conference on Autonomous Agents and Multi-Agent Systems* (New York, NY, 2004).

[2] Agogino, A. and Tumer, K., QUICR-learning for multiagent coordination, in *Proceedings of the 21st National Conference on Artificial Intelligence* (Boston, MA, 2006).

[3] Agogino, A. and Tumer, K., Regulating air traffic flow with coupled agents, in *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems* (Estoril,Portugal, 2008).

[4] Agogino, A. K. and Tumer, K., Handling communication restrictions and team formation in congestion games, *Journal of Autonomous Agents and Multi Agent Systems* **13** (2006) 97–115.

[5] Agogino, A. K. and Tumer, K., Efficient evaluation functions for evolving coordination, *Evolutionary Computation* **16** (2008) 257–288.

[6] Bagnell, J. A. and Ng, A. Y., On local rewards and the scalability of distributed reinforcement learning, in *NIPS 18* (2006).

[7] Bayen, A. M., Grieder, P., Meyer, G., and Tomlin, C. J., Lagrangian delay predictive model for sector-based air traffic flow, *AIAA Journal of Guidance, Control, and Dynamics* **28** (2005) 1015–1026.

[8] Bilimoria, K. D., Sridhar, B., Chatterji, G. B., Shethand, K. S., and Grabbe, S. R., Facet: Future atm concepts evaluation tool, *Air Traffic Control Quarterly* **9** (2001).

[9] Challet, D. and Zhang, Y. C., On the minority game: Analytical and numerical studies, *Physica A* **256** (1998) 514.

[10] Crites, R. H. and Barto, A. G., Improving elevator performance using reinforcement learning, in *Advances in Neural Information Processing Systems - 8*, eds. Touretzky, D. S., Mozer, M. C., and Hasselmo, M. E. (MIT Press, 1996), pp. 1017–1023.

[11] de Oliveira, D., Jr., P. R. F., and Bazzan, A. L. C., A swarm based approach for task allocation in dynamic agents organizations, in *AAMAS '04: Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems* (IEEE Computer Society, Washington, DC, USA, 2004), pp. 1252–1253.

[12] Dietterich, T. G., Hierarchical reinforcement learning with the maxq value function decomposition, *Journal of Artificial Intelligence* **13** (2000) 227–303.

[13] Donohue, G. L. and Shaver III, R. D., *TERMINAL CHAOS: Why U.S. Air Travel Is Broken and How to Fix It* (Amer Inst of Aeronautics and Astronautics, 2008).

[14] Guestrin, C., Lagoudakis, M., and Parr, R., Coordinated reinforcement learning, in *Proceedings of the 19th International Conference on Machine Learning* (2002).

[15] Hill, J. C., Johnson, F. R., Archibald, J. K., Frost, R. L., and Stirling, W. C., A cooperative multiagent approach to free flight, in *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems* (ACM Press, New York, NY, USA, 2005), ISBN 1-59593-093-0, pp. 1083–1090.

[16] Jefferies, P., Hart, M. L., and Johnson, N. F., Deterministic dynamics in the minority game, *Physical Review E* **65 (016105)** (2002).

[17] Jennings, N. R., On agent-based software engineering, *Artificial Intelligence* **177** (2000) 277–296.

[18] Johnson, N. F., Jarvis, S., Jonson, R., Cheung, P., Kwong, Y. R., and Hui, P. M., Volatility and agent adaptability in a self-organizing market (1998), preprint cond-mat/9802177.

[19] Jones, C. and Mataric, M. J., Adaptive division of labor in large-scale multi-robot systems, in *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS-03)* (Las Vegas, NV, 2003), pp. 1969–1974.

[20] Kaelbling, L. P., Littman, M. L., and Moore, A. W., Reinforcement learning: A survey, *Journal of Artificial Intelligence Research* **4** (1996) 237–285.

[21] Lecchini A., Glover W., Lygeros J., and Maciejowskia J., , Monte Carlo Optimization Strategies for Air-Tra?c Control, *AIAA Guidance, Navigation, and Control Conference* (San Francisco, CA, 2005), 470–482.

[22] McGlohon, M. and Sen, S., Learning to cooperate in multiagent systems by combining Q-learning and evolutionary strategy, *International Journal on Lateral Computing* **1** (2005) 58–64.

[23] Menon, P. K., Sweriduk, G. D., and Sridhar, B., Optimal strategies for free flight air traffic conflict resolution, *Journal of Guidance, Control, and Dynamics* **22** (1999) 202–211.

[24] Nair, R., Tambe, M., Yokoo, M., Pynadath, D., and Marsella, S., Taming decentralized POMDPs: Towards efficient policy computation for multiagent settings, in *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence* (Acapulco, Mexico, 2003).

[25] Parkes, D. and Singh, S., An MDP-based approach to online mechanism design, in *NIPS 16* (2004), pp. 791–798.

[26] Pechoucek, M., Sislak, D., Pavlicek, D., and Uller, M., Autonomous agents for air-traffic deconfliction, in *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multi-Agent Systems* (Hakodate, Japan, 2006).

[27] Stone, P., Sutton, R. S., and Kuhlmann, G., Reinforcement learning for RoboCup-soccer keepaway, *Adaptive Behavior* (2005).

[28] Taylor, M. E., Whiteson, S., and Stone, P., Comparing evolutionary and temporal difference methods for reinforcement learning, in *Proceedings of the Genetic and Evolutionary Computation Conference* (Seattle, WA, 2006), pp. 1321–1328, **Best Paper Award**.

[29] Tomlin, C., Pappas, G., and Sastry, S., Conflict resolution for air traffic management: A study in multiagent hybrid systems, *IEEE Transaction on Automatic Control* **43** (1998) 509–521.

[30] Tumer, K. and Agogino, A., Coordinating multi-rover systems: Evaluation functions for dynamic and noisy environments, in *The Genetic and Evolutionary Computation Conference* (Washington, DC, 2005).

[31] Tumer, K. and Agogino, A., Distributed agent-based air traffic flow management, in *Proceedings of the Sixth International Joint Conference on Autonomous Agents and Multi-Agent Systems* (Honolulu,HI, 2007), pp. 330–337, **Best Paper Award**.

[32] Tumer, K., Welch, Z. T., and Agogino, A., Aligning social welfare and agent preferences to alleviate traffic congestion, in *Proceedings of the Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems* (Estoril,Portugal, 2008).

[33] Tumer, K. and Wolpert, D. (eds.), *Collectives and the Design of Complex Systems* (Springer, New York, 2004).

[34] Tumer, K. and Wolpert, D. H., Collective intelligence and Braess' paradox, in *Proceedings of the Seventeenth National Conference on Artificial Intelligence* (Austin, TX, 2000), pp. 104–109.

[35] Tuyls, K. and Parsons, S., What evolutionary game theory tells us about multiagent learning, *Artificial Inelligence* **171**.

[36] Verbeeck, K., Nowe, A., and Tuyls, K., Coordinated exploration in multiagent reinforcement learning: An application to loadbalancing, in *Proceedings of the Fourth International Joint Conference on Autonomous Agents and Multi-Agent Systems* (Utrecht, The Netherlands, 2005).

[37] Verbeeck, K., Peeters, M., Nowe, A., and Tuyls, K., Reinforcement learning in stochastic single and multi-stage games, in *Adaptive Agents and Multi-Agent Systems II, Lecture Notes in Artificial Intelligence* (Springer Verlag, Berlin, 2005), pp. 275–294.

[38] Vidal, J. M., Multiagent coordination using a distributed combinatorial auction, in *AAAI Workshop on Auction Mechanism for Robot Coordination* (2006).

[39] Vidal, J. M. and Durfee, E. H., The moving target function problem in multiagent learning, in *Proceedings of the Third International Conference on Multi-Agent Systems* (AAAI/MIT press, 1998), pp. 317–324.

[40] Whiteson, S., Taylor, M. E., and Stone, P., Empirical studies in action selection for reinforcement learning, *Adaptive Behavior* **15** (2007).

[41] Wolpert, D. H. and Tumer, K., Optimal payoff functions for members of collectives, *Advances in Complex Systems* **4** (2001) 265–279.