Advances in Complex Systems, Vol. 12, Nos. 4 & 5 (2009) 455–473

World Scientific
www.worldscientific.com

LEARNING FROM ACTIONS NOT TAKEN IN MULTIAGENT SYSTEMS

KAGAN TUMER* and NEWSHA KHANI†

Oregon State University, 204 Rogers Hall,
Corvallis, Oregon 97331, USA
**kagan.tumer@oregonstate.edu*
†*khanin@onid.orst.edu*

Received 31 January 2009
Revised 11 June 2009

In large cooperative multiagent systems, coordinating the actions of the agents is critical to the overall system achieving its intended goal. Even when the agents aim to cooperate, ensuring that the agent actions lead to good system level behavior becomes increasingly difficult as systems become larger. One of the fundamental difficulties in such multiagent systems is the slow learning process where an agent not only needs to learn how to behave in a complex environment, but also needs to account for the actions of other learning agents. In this paper, we present a multiagent learning approach that significantly improves the learning speed in multiagent systems by allowing an agent to update its estimate of the rewards (e.g. value function in reinforcement learning) for all its available actions, not just the action that was taken. This approach is based on an agent estimating the counterfactual reward it would have received had it taken a particular action. Our results show that the rewards on such "actions not taken" are beneficial early in training, particularly when only particular "key" actions are used. We then present results where agent teams are leveraged to estimate those rewards. Finally, we show that the improved learning speed is critical in dynamic environments where fast learning is critical to tracking the underlying processes.

Keywords: Multiagent learning; counterfactual reward; difference reward.

1. Introduction

Learning in large multiagent systems is a critical area of research with applications ranging from robocup soccer [26, 27], to rover coordination [19], to trading agents [25, 43], to air traffic management [32]. What makes this problem particularly challenging is that the agents in the system provide a constantly changing background in which each agent needs to learn its task. As a consequence, almost by definition, all multiagent learning occurs in complex environments, where the agents need to extract the underlying reward signal from the noise of the other agents acting within the same environment.

Furthermore, typically, two learning problems are coupled where the agent needs to solve both a temporal credit assignment problem (how to assign a reward received

at the end of sequence of actions to each action) and a structural credit assignment problem (how to assign credit to a particular agent at the end of a multiagent task) [1, 15, 16, 28, 38, 41, 44]. The temporal credit assignment problem has been extensively studied [10, 16, 28, 31, 30, 39, 42], and the structural credit assignment problem has recently been investigated as well [4, 8, 11, 20, 23, 35].

Learning sequences of actions for multiagent systems has blended these two areas of research and led to key advances [6, 8, 12, 26, 40]. In these cases, the learning needs of the agents are modified to account for their presence in a larger system [2, 11, 13, 22, 35, 37]. However, though these methods have yielded tremendous advances in multiagent learning, they are principally based on an agent trying an action, receiving an evaluation of that action, and updating its own estimate on the "value" of taking that action in that state. Though effective, such an approach is generally slow to converge, particularly in large and dynamic environments.

In this paper, we explore the concept of agents learning from actions they do not take by estimating the rewards they would have received had they taken those actions. These counterfactual rewards are estimated using the theory developed for structural credit assignment, and prove effective in the congestion games. Furthermore, a team structure can be used to provide the required information for the agents to compute these reward estimates [24, 29]. A key benefit of this approach is that an increase in the number of agents can be leveraged to improve the estimates of actions not taken, turning a potential pitfall (e.g. how to extract useful information from the actions of so many agents) into an asset (e.g. learn from the experiences of other agents). Though the concept of updating rewards for actions not taken is present in learning automata literature, where for example, the probability of taking a particular action may go up down based on similar actions' results [21, 38, 39], in this work we explicitly aim to quantify the counterfactual concept of "*what would my reward have been, had I taken another action.*"

In Sec. 2, we discuss the congestion problem that we use in the reported experiments. In Sec. 3, we summarize the basic agent learning architecture. In Sec. 4, we provide the action-not-taken (ANT) rewards and modify them using team rewards. We also provide experimental results showing the basic behavior of the ANT reward. In Sec. 5, we explore the application of these rewards to dynamic domains where the rapidly changing conditions put a premium on learning quickly. Finally, in Sec. 6, we discuss the results and provide directions for future research.

2. Congestion Problems

Congestion problems where system performance depends on the number of agents taking a particular action provide an interesting domain to study the behavior of cooperative multiagent systems. In congestion problems, agents need to learn how to synchronize (or not synchronize) their actions, rather than learn to take particular actions. This type of problem is ubiquitous in routing domains (e.g. on

a highway, a particular lane is not preferable to any other lane, but what matters is how many others are using a particular lane) [18, 34].

The multi-night bar problem is an abstraction of congestion games (and a variant of the El Farol bar problem [5]) which have been extensively studied [1, 5, 9, 7, 14]. In this version of the congestion problem, each agent has to determine which day in the week to attend a bar. The problem is set up so that if either too few agents attend (boring evening) or too many people attend (crowded evening), the total enjoyment of the attending agents drop.

The system performance is quantified by a system reward function G. This reward is a function of the full system state z (e.g. the joint action of all agents in the system), and is given by:

$$G(z) = \sum_{\text{day}=1}^{n} x_{\text{day}} e^{\frac{-x_{\text{day}}}{C}}, \tag{1}$$

where n is the number of actions (for example $n = 7$ if actions are days); x_{day}: the total attendance on a particular day; and C: a real-valued parameter that represents the capacity of the resource (e.g. the capacity of the bar).

What is interesting about this game is that selfish behavior by the agents tends to lead the system to undesirable states. For example, if all agents predict an empty bar, they will all attend (poor reward) or if they all predict a crowded bar, none will attend (poor reward). This aspect of the bar problem is what makes this a "congestion game" and an abstract model of many real-world problems ranging from lane selection in traffic to job scheduling across servers to data routing.

3. Basic Agent Learning

The agent actions in this problem is to select a resource (day on which to attend the bar). The learning algorithm for each agent is a simple reinforcement learner (action value). Each agent keeps an n-dimensional vector providing its estimates of the reward it would receive for taking each possible action. The system dynamics are given by:

Initialize: week 0
Repeat until week > Max week

1. agents choose actions;
2. agents' joint action leads to an overall system state;
3. the system state results in a system reward;
4. each agent receives a reward;
5. each agent updates its action selection procedure (i.e. learning);
6. week $\leftarrow$ week $+ 1$.

In any week, an agent estimates its expected reward for attending a specific night based on action values it has developed in previous weeks. At the beginning of each

training run, each agent has an equal probability of choosing each action in the first week, resulting in a uniformly random distribution across actions. At the beginning of each training week, each agent picks a night to attend based on sampling this probability vector using a Gibbs distribution. Each agent has n actions and a value V_k associated with each action a_k:

$$P_k = \frac{e^{(V_k \cdot \tau)}}{\sum_{\text{agent}} e^{(V_k \cdot \tau)}}, \tag{2}$$

where τ is a temperature term that determines the amount of exploration (low values of τ mean most actions have similar probabilities of being selected, whereas high values of τ increase the probability that the best action will be selected). Each agent receives reward R and updates the action value vector using a value function V_k:

$$V_k = (1 - \alpha) \cdot V_k + \alpha \cdot R. \tag{3}$$

A reasonable option is to provide each agent with the *full system reward* for each week. This leads to each agent receiving the reward given in Eq. (1), and using that reward to update its value estimates for each action. However, this reward is not particularly sensitive to an agent's actions and especially in large systems, leads to particularly slow learning. As a consequence, in this work, we use the *difference reward* as a starting point for the reward an agent receives after each step. Earlier work has shown that the difference reward significantly outperforms both agents receiving a purely local reward and all agents receiving the same system reward [3, 2, 33, 32, 36]. The difference reward is given by:

$$D^i(z) = G(z) - G(z - z_i), \tag{4}$$

where $z - z_i$ specifies the state of the system without agent i.[a] In this instance z is the full attendance profile of the agents, and $z - z_i$ is the attendance profile of all the agents without agent i. Difference rewards are *aligned* with the system reward, in that any action that improves the difference reward will also improve the system reward. This is because the second term on the right-hand side of Eq. (4) does not depend on agent i's actions, meaning any impact agent i has on the difference reward is through the first term (G) [32, 35]. Furthermore, it is more sensitive to the actions of agent i, reflected in the second term of D, which removes the effects of other agents (i.e. noise) from agent i's reward function.

Intuitively, this causes the second term of the difference reward function to evaluate the performance of the system without i, and therefore D measures the agent's contribution to the system reward directly. For the difference reward in the congestion problem, this amounts to having each agent estimate the system reward it would receive were it to take or not take a particular action. In this work, agents

[a]In this paper, we will use zero padded vector addition and subtraction to specify the state dependence on specific components of the system.

do not explicitly communicate with one another, and therefore, the only effect each agent has on the system is to increase the attendance, x_{day}, for night k by 1. This leads to the following difference reward:

$$\begin{aligned} D^i(z) &= G(z) - G(z - z_i) \\ &= x_{\text{day}_i} e^{\frac{-x_{\text{day}_i}}{C}} - (x_{\text{day}_i} - 1) e^{\frac{-(x_{\text{day}_i}-1)}{C}}, \end{aligned} \tag{5}$$

where x_{day_i} is the total attendance on the day selected by agent i.

4. Action-Not-Taken (ANT) Rewards

Though the difference reward given in Eq. (5), provides a reward tuned to an agent's actions, it is still based on an agent sampling each of its actions a (potentially large) number of times. In this work, in order to increase the learning speed, we introduce the concept of ANT rewards.[17] The goal with ANT rewards is to provide estimates of how the system would have turned out had an agent taken a particular action. The mathematics that allow the computation of the difference reward can be used to compute this type of reward.

In this paper, rather than have a separate results section, we provide experimental results directly alongside the reward descriptions to motivate the improvements to the rewards and the derivation of new rewards. All results are based on 20 independent runs with the standard error plotted when large enough to be relevant. Unless otherwise specified (as with the scaling runs or congestion dependent runs) the number of agents in the system was set to 120, with $C = 6$ (capacity), and $n = 5$ (number of actions, or days).

4.1. *Basic action-not-taken reward*

The direct application of this concept is to have agents update their reward estimate based on the reward they would have received had they taken other actions. Therefore, at each time step, agents perform a mathematical operation that simulates their taking a different action and compute the counterfactual reward that would have resulted from that action. For an agent i who selected action a at this step, the counterfactual reward for action b is given by:

$$D^{i \to b}(z) = G(z - z_i^a + z_i^b) - G(z - z_i^a), \tag{6}$$

where $D^{i \to b}$ is the reward for agent i taking action b; z_i^a is the state component where agent i has taken action a; z_i^b is the state component where agent i has taken action b.

The second term of Eq. (6) ($G(z - z_i^a)$) is the same as the second term of Eq. (4). Namely the reward for the state where agent i has not taken the particular action that it took. The first term though is the key to the ANT reward. In this case, we compute the reward that would have resulted had agent i taken action b rather than action a.

Utilizing this structure, D^i_{ANT} can then be formulated as shown in Eq. (7):

$$D^i_{\text{ANT}} = \begin{cases} G(z) - G(z - z^a_i), & \text{for } i \to a, \\ G(z - z^a_i + z^b_i) - G(z - z^a_i), & \text{for } i \to b \neq a, \end{cases} \tag{7}$$

where $i \to a$ means that agent i has taken action a. Note, the removal of the state in which agent i has taken action a in the second term represents the system state without agent i. Because agent i had taken action a, this removal results in a state where agent i has taken neither action a nor action b (which it has never taken). Hence the second term is the same for both conditions of Eq. (7).

Figure 1 shows the learning curves for D and D_{ANT} along with results where agents directly use the system reward G and a local reward L to learn. The local reward L is based on the agents simply receiving the reward for the action they took, which in this instance is the component of Eq. (1) corresponding to the day they attended the bar ($L = x_{\text{day}} e^{\frac{-x_{\text{day}}}{C}}$). This is a "selfish" reward, in that the agent is only concerned with the day on which it decided to attend. Though yielding poor results in this case, this is the naive decomposition of G to its components [45].

In all the experiments, the system performance is measured with respect to G, regardless of how the agents were trained. As previously noted, these results confirm that agents using D significantly outperform agents using G or L in this domain. G learns little, and L learns to do the wrong thing: Because the agent rewards are not aligned, agents aiming to maximize their own reward lead to poor system states. We include the results for agents using G and L here for completeness, but we will omit them in subsequent figures.

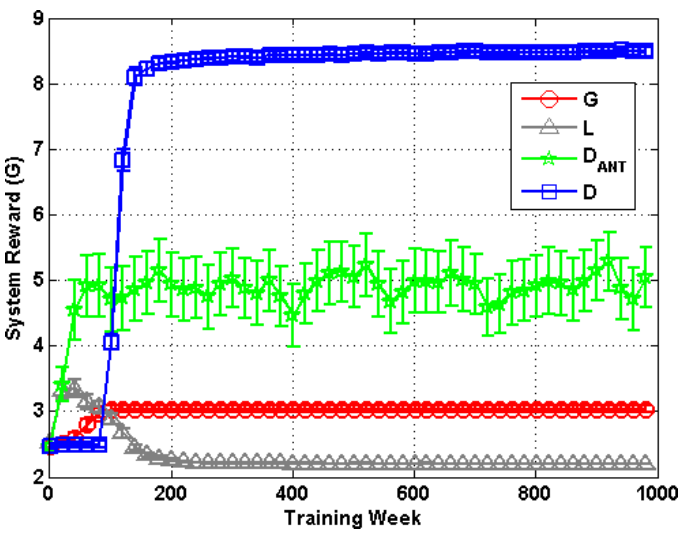

Fig. 1. System performance versus training weeks. In comparison to D, D_{ANT} based on actions not taken learns faster but shows a lower and noisier performance.

The results here show that although D_{ANT} learns faster than D, it struggles to reach good solutions. This shows that the ANT reward has a difficult time estimating the reward for most actions once those actions have been sampled. Even though the agents take advantage of such rewards and learn faster in the first weeks of training, there is a time after which these additional rewards become detrimental to the learning process. This suggests two possible solutions, which we explore in the next two sections:

(1) Use the ANT reward early in the process, but stop and switch to basic D after a "stop week."
(2) Select only a subset of the actions to receive the ANT reward.

4.2. *ANT reward with early stopping*

First, let us consider the early stopping concept to mitigate the noisy feedback agents receive for their actions. This modification is based on the observation that the ANT rewards are better than random rewards, but not as good as rewards that have been updated by actually taking the actions. Figure 2 shows the impact of having agents use ANT rewards for the first 6 weeks and then switch back to using D (the impact of when to stop is discussed in Fig. 3). Results show that this approach significantly speeds up the learning process, though does not result in agents reaching higher performance.

Figure 3 shows the dependence of the system performance on the length of time the ANT reward is used. The learning speed is stable for small values of the stop week, but starts to drop slowly as the actions not taken are used more extensively. There is a steady rightward shift as the stop week moves from 6 to 100, at which

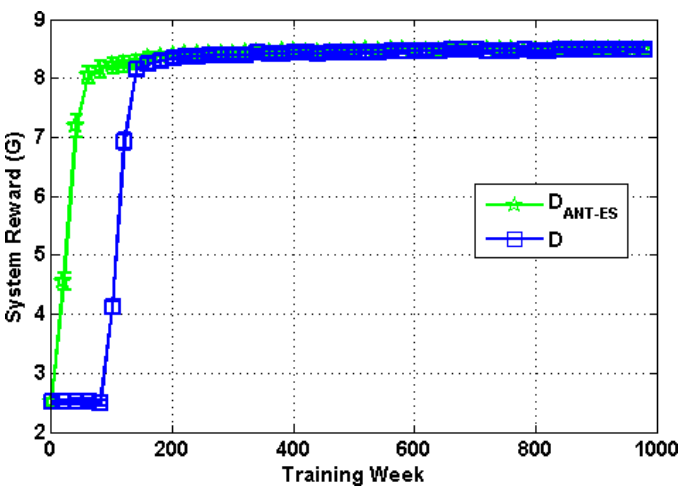

Fig. 2. System performance when actions not taken are stopped after week 6. $D_{\text{ANT-ES}}$ (ANT with Early Stopping) learns faster and reaches the same system rewards as D.

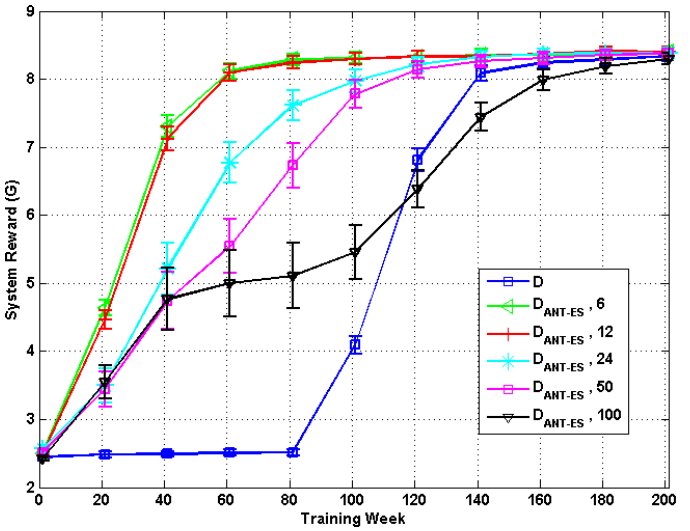

Fig. 3. The impact of the stop week on system performance. The learning speed is directly related to the length of time the action-not-taken reward is used.

point, the system learns more slowly than D alone. Providing a mechanism for selecting the stop week based on either a preset number of ANT rewards, or given performance criteria would provide automation, though in this work, we simply base the stop week on trial and error based on Fig. 3.

4.3. *ANT reward with teams*

The second option we consider is to limit the actions that are updated based on counterfactual rewards to reliable actions sampled by a subset of agents. To that end, we introduce the concept of a team, and denote agent i's team members by T_i. In this context, T_i is a fixed, randomly selected subset of the agents. This formulation gives:

$$D^i_{\text{ANT-}L} = \begin{cases} G(z) - G(z - z^a_i), & \text{for } i \to a, \\ G(z - z^a_i + z^b_i) - G(z - z^a_i), & \text{for } i \to b \in T_i, \\ 0, & \text{otherwise,} \end{cases} \tag{8}$$

where $i \to b \in T_i$ means agent i selects actions b that are sampled by agent's i's teammates T_i. As previously, the removal of agent i in the second term represents the system state without agent i having taken either action a (which it had taken) or action b (which it had not taken), leading to the term being the same in both cases.

Figure 4 shows the results when an agent has 12 randomly selected team members (in this case there are 120 total agents, so the team sizes are 10% of the total agents). Other than at the extremes (e.g. team size of 2 or 110), the experiments were not particularly sensitive to this parameter. By limiting the number of actions that are updated ($D_{\text{ANT-}L}$ in black/dark), the variability of the reward is reduced as

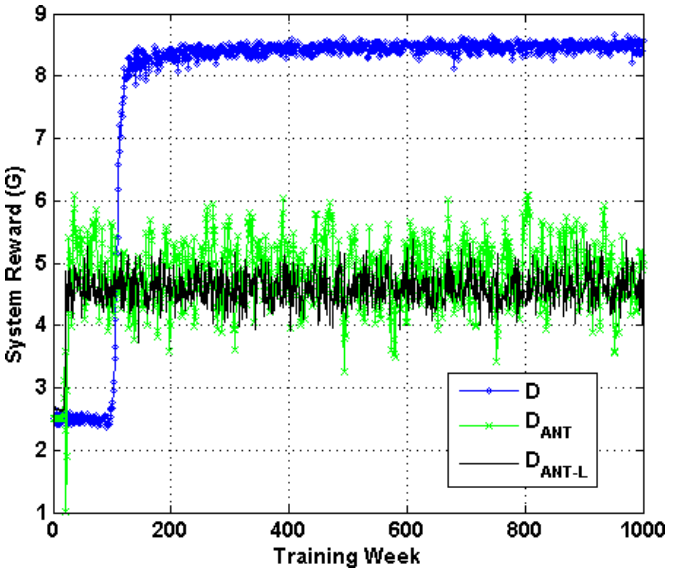

Fig. 4. System performance when only a subset of actions are explored by an agent (120 total agents, team size of 12). D performs well. $D_{\text{ANT-}L}$ based on a limited number of actions not taken performs similarly as D_{ANT} but shows a response with a lower noise level.

compared to the full D_{ANT} (in green/light), but there is no discernible improvement in the quality of the solution. However, from a computational and communication perspective, this is an interesting result, which points to a significant reduction in the need for counterfactual reward computation without loss of convergence speed.

We now combine the two concepts and have agents use teams and early stopping. Furthermore, instead of using the team members as information sources only, we increase the connection among team members by providing them all with the same reward. That is, all team members attending a particular day will receive the same reward. The learning strategy is to use team information only during the first weeks (three in the reported results, but the performance is similar for minor changes to this parameter) of learning and switch to the regular difference reward [Eq. (5)] for the rest of the training period.

The key aspect of this approach is that the team members measure the impact of a team not taking a particular action, rather than an individual agent. As a result, agents learn with their team in a smaller state space defined by the world minus their team space instead of the entire world. This is conceptually similar to the reward described in Eq. (8) but where the impact of the whole team, rather than agent i is removed, leading to:

$$D^i_{\text{Team}} = \begin{cases} G(z) - G(z - z^a_{T_i}), & \text{for } T_i \to a, \\ G(z - z^a_i + z^b_j) - G(z - z^b_{T_i} - z^a_i), & \text{for } i \to b \in T^i, \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

where $z^a_{T_i}$ is the state component of team members of agent i taking action a. In this formulation, the impact of all of agent's i teammates are removed before the

reward is calculated. Note in this case, unlike in Eqs. (7) and (8), the second term is different for the two actions. This is because this term estimates the impact of removing all team members of i that had taken a particular action. When agent i changes its action, this also changes the team members taking the same action as i. For the action a selected by agent i, we only need to remove all its team members who took that action. But to find the counterfactual reward for action b, we need to remove the actual action of agent i (action a) and then remove the team members who had taken action b. Though conceptually similar to previous rewards, the presence of team members leads to this subtle difference in the computation of the team ANT reward.

Now, let us explicitly compute D^i_{Team} for the congestion problem considered in this paper. First, for the action taken by agent i [first line of Eq. (9)], the reward becomes:

$$\begin{aligned} D^{i\to a}_{\text{Team}} &= G(z) - G(z - z^a_{T_i}) \\ &= \sum_{\text{day}} x_{\text{day}} e^{\frac{-x_{\text{day}}}{C}} - \Bigg(\sum_{\text{day}\neq \text{day}_i} x_{\text{day}} e^{\frac{-x_{\text{day}}}{C}} \\ &\quad + (x_{\text{day}_i} - |T^i_{\text{day}_i}|) e^{\frac{-(x_{\text{day}_i} - |T^i_{\text{day}_i}|)}{C}} \Bigg), \end{aligned} \tag{10}$$

where $z - z^a_{T_i}$ is the state component in which agent i and its teammate taking action a have no effect; day_i is the day agent i selects to attend; x_{day_i} is the attendance on the day agent i selects to attend; and $|T^i_{\text{day}_i}|$ is the number of agent i's teammates that choose day_i to attend.

Second, let us focus on the actions not taken by agent i [second line of Eq. (9)]. This is the reward agent i would have received had it taken the actions b chosen by some of its teammates, leading to:

$$\begin{aligned} D^{i\to b}_{\text{Team}} &= G(z - z^a_i + z^b_i) - G(z - z^b_{T_i} - z^a_i) \\ &= \sum_{\text{day}\neq \text{day}_{i\to a,b}}^{\text{day}} x_{\text{day}} e^{\frac{-x_{\text{day}}}{C}} + (x_{\text{day}_{i\to a}} - 1) e^{\frac{-(x_{\text{day}_{i\to a}} - 1)}{C}} \\ &\quad + (x_{\text{day}_{i\to b}} + 1) e^{\frac{-(x_{\text{day}_{i\to b}} + 1)}{C}} \\ &\quad - \Bigg(\sum_{\text{day}\neq \text{day}_{i\to a,b}} x_{\text{day}} e^{\frac{-x_{\text{day}}}{C}} + (x_{\text{day}_{i\to a}} - 1) e^{\frac{-(x_{\text{day}_{i\to a}} - 1)}{C}} \\ &\quad + (x_{\text{day}_{i\to b}} - |T^i_{\text{day}_{i\to b}}|) \cdot e^{\frac{-(x_{\text{day}_{i\to b}} - |T^i_{\text{day}_{i\to b}}|)}{C}} \Bigg) \\ &= (x_{\text{day}_{i\to b}} + 1) e^{\frac{-(x_{\text{day}_{i\to b}} + 1)}{C}} + (x_{\text{day}_{i\to b}} - |T^i_{\text{day}_{i\to b}}|) \cdot e^{\frac{-(x_{\text{day}_{i\to b}} - |T^i_{\text{day}_{i\to b}}|)}{C}} \end{aligned} \tag{11}$$

where $z - z_i^a + z_i^b$ is the state component in which agent i takes action b rather than action a; $z - z_{T_i}^b - z_i^a$ is the state component on which agent i (taking action a) and its teammates taking action b are removed from the state; $x_{i \to b}$ is the attendance resulting from agent i taking action b; $|T^i_{\text{day}_{i \to b}}|$ is the number of agent i's teammates that choose to attend on day resulting from action b.

In this formulation, if the agent i's team members have taken all the possible actions, each action that agent i had not taken will still be updated. Otherwise, only actions taken by i's teammates will be available for reward information and therefore updated.

Figure 5 shows the learning curves for D , $D_{\text{ANT-ES}}$ and D_{TEAM}. Agents using D_{TEAM} not only learn faster, but also reach higher system rewards than agents using the baseline D or previous variants of D_{ANT}. In this instance, not only information from team members was used, but also the reward of each team member was the same, resulting in a larger "block" of agents receiving a reward, and removing a significant amount of noise from the rewards.

4.4. *ANT reward with weighted teams*

The use of team rewards provided tangible benefits, though it treated all information received from team members equally. Yet, one can consider that the more team members take a particular action, the more reliable the estimate for the reward of

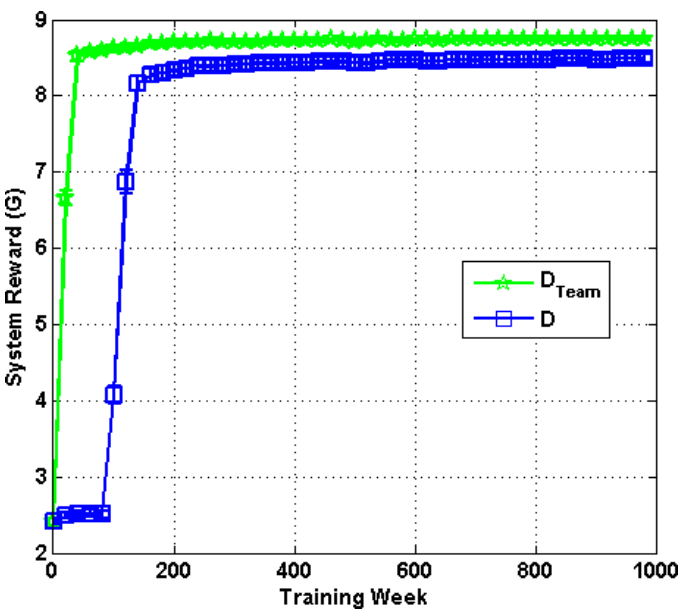

Fig. 5. System performance versus training weeks. D performs well, but D_{Team} based on updating only actions that were taken by team members both learns faster and reaches higher system rewards than D or $D_{\text{ANT-ES}}$.

that action would become. This becomes particularly relevant when the congestion in the system increases.

A simple solution to this problem is to use a weighting factor for the second term of the counterfactual reward function. In this work, we use the average number of team members selecting particular actions, though more sophisticated methods can also be used. This leads to modifying Eq. (9), that for agent i and action b leads to a weighted team reward D_{WT}:

$$D_{\mathrm{WT}}^{i\to b} = G(z - z_i^a + z_i^b) - \mu_{|T^i_{\mathrm{day}_{i\to b}}|} \cdot G(z - z^b_{T_i} - z_i^a), \tag{12}$$

where $\mu_{|T^i_{\mathrm{day}_{i\to b}}|}$ is the average number of team members taking action b.

Figure 6 explores this idea for 460 agents in a system with seven actions and a capacity of 4. Because the optimal capacity in this case is $7 \times 4 = 28$, this creates significant congestion. The results show that traditional D starts to suffer in this case, and that the weighted D_{WT} outperforms D_{TEAM}. Figure 7 shows the impact of congestion directly as the number of agents in the system increases from 120 to 460. D_{WT} handles the increased congestion better than either D_{TEAM} or D.

5. Tracking Dynamic Environments

One of the key advantages to learning rapidly is the ability to adapt to dynamic environments where the conditions may change faster than a traditional learner can adapt. In this section, we test the performance of the ANT rewards with weighted team reward on two types of dynamic environments. First, we explore seemingly

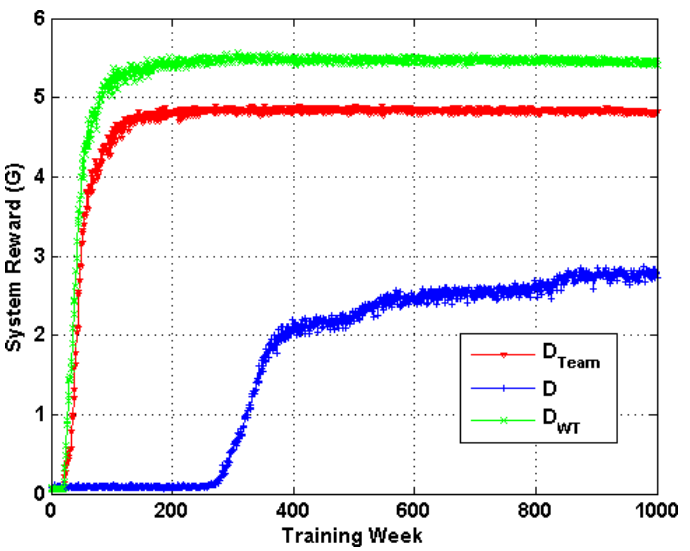

Fig. 6. System performance for the weighted team rewards. There are 460 agents in the system with only seven actions of capacity 4 leading to significant congestion. The performance of D_{WT} is significantly higher than either the base D or D_{TEAM}.

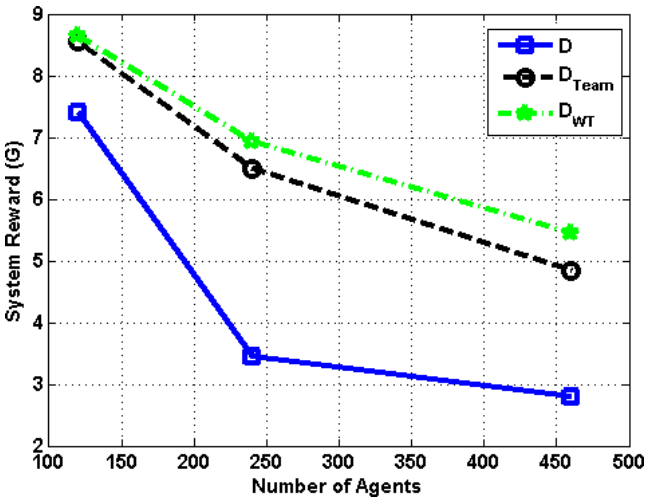

Fig. 7. The impact of congestion on system performance for the weighted team rewards. The number of agents increases, but the capacity of each day stays the same ($C = 4$). The performances of both D_{TEAM} and D_{WT} are significantly higher than D, and D_{WT} handles the congestion the best.

random changes in agent numbers and capacities, and then we explore faster, but periodic changes of both types.

5.1. *Unpredictable changes to the environment*

In this section, we explore the ability of D_{WT} to adjust to unexpected changes in the system. Figure 8 shows the system response to changes in the number of agents. In this case, the number of agents changed every 40 weeks from 280, to 140, to 180, to 100. D_{WT} not only recovers rapidly, but also learns to exploit the new condition, as demonstrated at week 120: after the initial drop caused by the change, agents using D return to their previous state, but agents using D_{WT} reach a higher system reward value.

Figure 9 shows the system response to the capacity changing from 3 to 7 every 70 weeks. D_{WT} learns faster early on and reaches slightly higher performance, but this experiment shows that D can track slow changes in the environment.

5.2. *Periodic changes to the environment*

In this section, we explore periodic and rapid changes to the environment. Figure 10 explores the performance of D_{WT} versus difference reward D when the number of agents is changing rapidly. Unlike in the results of the previous section (Fig. 8), D has a hard time tracking these changes. D_{WT} on the other hand converges to a good solution despite the number of agents in the system changing the optimal solutions for each agent. (For this experiment, we modified the value update function to account for the periodicity of the system, and allowed the value update to

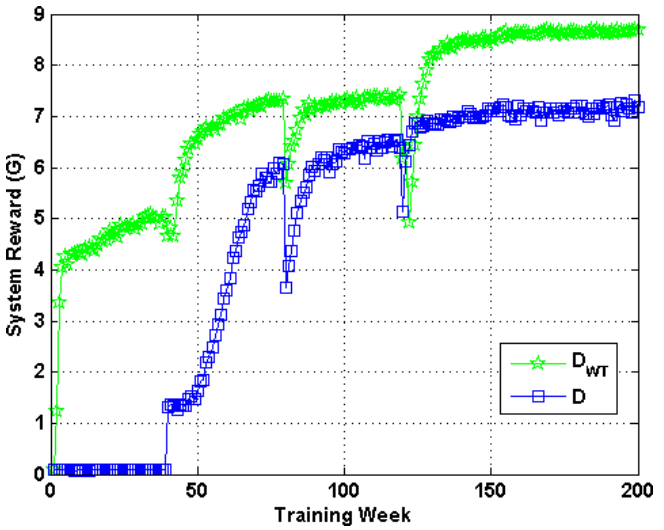

Fig. 8. System performance when the number of agents in the system changed from 280, 140, 180, 100 each 40 time steps, for seven actions and a capacity of 4. D_{WT} outperforms D both in response time and final solution quality.

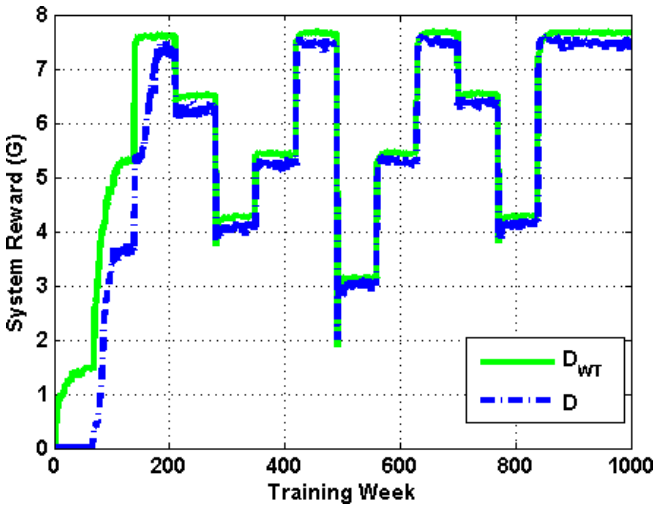

Fig. 9. System performance when the capacity of the system changes from 3 to 7 and back every 70 time steps for four actions and 120 agents.

be: $V_k = (1-\alpha)\cdot(\tau \cdot V_k^{t-1} + (1-\tau)\cdot V_k^{t'}) + \alpha \cdot R$ where t' corresponds to the last time in which the capacity was the same. This value can be estimated in practice, though in this instance, in order to remove the impact of such estimation on the reward analysis, we provided it to both reward functions.)

Finally, we explore the impact of rapid changes to the system capacity. Figure 11 shows the system performance when the capacity oscillates between 2 and 5. Unlike

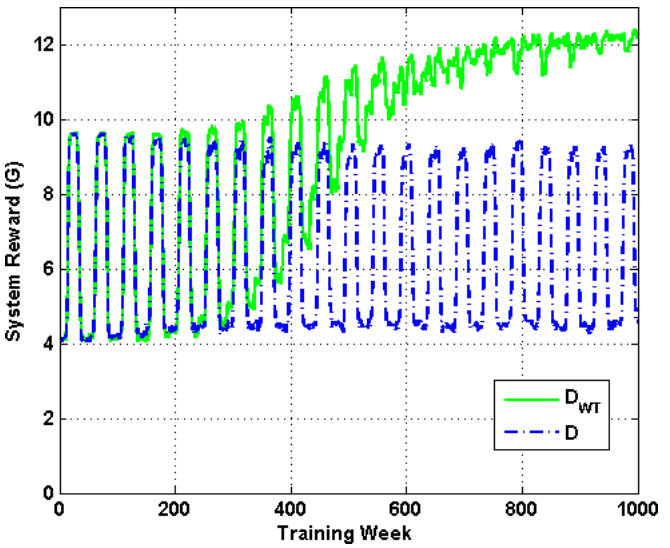

Fig. 10. System performance versus training weeks. There were eight actions with a capacity of 5. The standard difference reward D is plotted D_{WT} with variations of 60–120 in the number of agents. D cannot converge to a good solution, but D_{WT} not only converges to a good solution but does so rapidly after each capacity change.

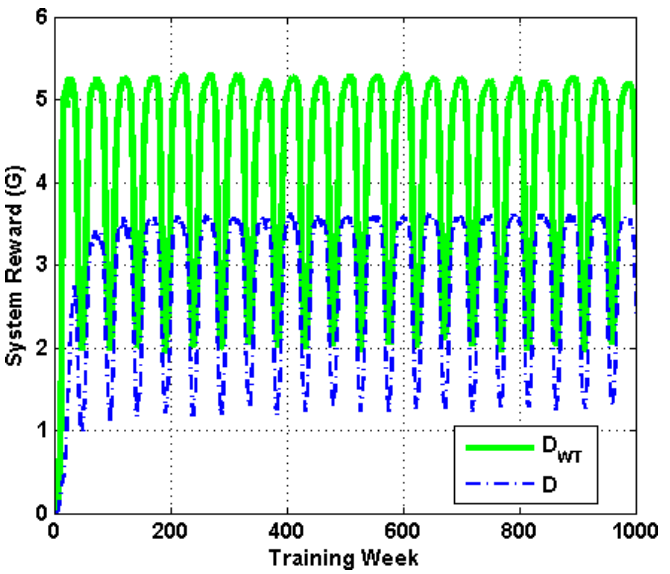

Fig. 11. System performance when the number of agents changes periodically. There were eight actions and 120 agents and capacity changed from 2 to 5 every 50 weeks. D performs poorly, but D_{WT} learns faster and reaches higher system rewards than D for both capacities.

in Fig. 9, D cannot track this continuous change as it does not get sufficient time to learn the system before the environment changes. D_{WT}, however, tracks the changes. Even though it has difficulties with the rapid changes, it both reaches higher system level performance for both $C = 2$ and $C = 5$.

6. Discussion

In large multiagent systems, the agents face a difficult learning problem where their actions are filtered through the "group action" before leading to a reward. As a consequence, an agent has a lengthy learning period where the actions need to be sampled a large number of times to extract the "signal" from the "noise." The use of the difference reward provides an improvement over directly using the system reward. However, a standard difference reward function still relies on each action being sampled before the cleaned up reward can be obtained. In this work, we present a modification to previously used difference reward, called ANT reward that provides agents with rewards on actions that were not taken by the agent.

We then provide modified versions of the ANT reward that through early stopping and team structures provides improvements in both the learning speed and the quality of the solution reached. The increase in speed of learning is the direct result of an agent receiving a counterfactual reward that estimates the reward that agent would have received had it taken a particular action. Furthermore, we show that the performance improvements are significantly more pronounced in dynamic environments where the conditions change either randomly or with high periodicity. In both cases, the rapid learning allows the agents to track a highly dynamic environment.

Though these results are encouraging, there are multiple areas for further investigation in this domain. First, the communication and observation requirements of the agents can be explicitly explored and connected to the system performance. Second, having agents adopt particular roles within a team can potentially provide further improvements in the learning speed. Finally, modifying the way in which agents estimate their ANT rewards can lead to substantial computational gains in addition to the already achieved speed up in the number of iterations required for convergence. We are currently investigating all three extensions of this work.

Acknowledgments

The authors would like to thank Matt Knudson for his insightful comments as well as his help with the preparation of this paper. This work was partially supported by AFOSR grant number FA9550-08-1-0187.

References

[1] Agogino, A. K. and Tumer, K., Handling communication restrictions and team formation in congestion games, *J. Auton. Agents Multi Agent Syst.* **13** (2006) 97–115.

[2] Agogino, A. K. and Tumer, K., Analyzing and visualizing multiagent rewards in dynamic and stochastic environments, *J. Auton. Agents Multi Agent Syst.* **17** (2008) 320–338.
[3] Agogino, A. K. and Tumer, K., Efficient evaluation functions for evolving coordination, *Evol. Comput.* **16** (2008) 257–288.
[4] Arai, S., Sycara, K. and Payne, T., Multi-agent reinforcement learning for planning and scheduling multiple goals, in *Proc. Fourth Int. Conf. on Multiagent Syst.* (2000), pp. 359–360.
[5] Arthur, W. B., Complexity in economic theory: Inductive reasoning and bounded rationality, *Am. Econ. Rev.* **84** (1994) 406–411.
[6] Chalkiadakis, G. and Boutilier, C., Coordination in multiagent reinforcement learning: A bayesian approach, in *Proc. Second Int. Joint Conf. on Autonomous Agents and Multiagent Systems (AAMAS-03)* (Melbourne, Australia, 2003).
[7] Challet, D. and Zhang, Y. C., On the minority game: Analytical and numerical studies, *Physica A* **256** (1998) 514.
[8] Claus, C. and Boutilier, C., The dynamics of reinforcement learning cooperative multiagent systems, in *Proc. Fifteenth National Conf. on Artificial Intelligence* (Madison, WI, 1998), pp. 746–752.
[9] de Cara, M. A. R., Pla, O. and Guinea, F., Competition, efficiency and collective behavior in the "El Farol" bar model, *Eur. Phys. J. B* **10** (1999) 187.
[10] Dietterich, T. G., Hierarchical reinforcement learning with the MAXQ value function decomposition, *J. Artif. Intell.* **13** (2000) 227–303.
[11] Guestrin, C., Lagoudakis, M. and Parr, R., Coordinated reinforcement learning, in *Proc. 19th Int. Conf. on Machine Learning* (2002).
[12] Hu, J. and Wellman, M. P., Multiagent reinforcement learning: Theoretical framework and an algorithm, in *Proc. Fifteenth Int. Conf. on Machine Learning* (1998), pp. 242–250.
[13] Hu, J. and Wellman, M. P., Online learning about other agents in a dynamic multiagent system, in *Proc. Second Int. Conf. on Autonomous Agents* (1998), pp. 239–246.
[14] Jefferies, P., Hart, M. L. and Johnson, N. F., Deterministic dynamics in the minority game, *Phys. Rev. E* **65**(016105) (2002).
[15] Jennings, N. R., Sycara, K. and Wooldridge, M., A roadmap of agent research and development, *Auton. Agents Multi-Agent Syst.* **1** (1998) 7–38.
[16] Kaelbling, L. P., Littman, M. L. and Moore, A. W., Reinforcement learning: A survey, *J. Artif. Intell. Res.* **4** (1996) 237–285.
[17] Khani, N. and Tumer, K., Fast multiagent learning: Cashing in on team knowledge, in *Artificial Neural Networks in Engineering* (ASME, St. Louis, 2008), pp. 3–10.
[18] Klügl, F., Bazzan, A. and Ossowski, S. (eds.), *Applications of Agent Technology in Traffic and Transportation* (Springer, 2005).
[19] Mataric, M. J., Coordination and learning in multi-robot systems, in *IEEE Intelligent Systems* (1998), pp. 6–8.
[20] McGlohon, M. and Sen, S., Learning to cooperate in multi-agent systems by combining Q-learning and evolutionary strategy, *Int. J. Lateral Comput.* **1** (2005) 58–64.
[21] Narendra, K. S. and Thathachar, M. A. L., *Learning Automata: An Introduction* (Prentice Hall, 1989).
[22] Panait, L., Tuyls, K. and Luke, S., Theoretical advantages of lenient learners: An evolutionary game theoretic perspective, *J. Mach. Learn. Res.* **9** (2008) 423–457.
[23] Parkes, D., On learnable mechanism design, in *Collectives and the Design of Complex Systems* (Springer, 2004).

[24] Pynadath, D. and Tambe, M., The communicative multiagent team decision problem: Analyzing teamwork theories and models, *J. Artif. Intell. Res.* **16** (2002) 389–423.
[25] Sherstov, A. and Stone, P., Three automated stock-trading agents: A comparative study, in *Agent Mediated Electronic Commerce VI: Theories for and Engineering of Distributed Mechanisms and Systems (AMEC 2004)*, Lecture Notes in Artificial Intelligence (Springer Verlag, Berlin, 2005), pp. 173–187.
[26] Stone, P., *Layered Learning in Multi-Agent Systems: A Winning Approach to Robotic Soccer* (MIT Press, Cambridge, MA, 2000).
[27] Stone, P., Sutton, R. S. and Kuhlmann, G., Reinforcement learning for RoboCup-soccer keepaway, *Adapt. Behav.* (2005).
[28] Sutton, R. S. and Barto, A. G., *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, MA, 1998).
[29] Tambe, M., Towards flexible teamwork, *J. Artif. Intell. Res.* **7** (1997) 83–124.
[30] Taylor, M. E., Whiteson, S. and Stone, P., Comparing evolutionary and temporal difference methods for reinforcement learning, in *Proc. Genetic and Evolutionary Computation Conf.* (Seattle, WA, 2006), pp. 1321–1328.
[31] Tesauro, G., Practical issues in temporal difference learning, in *Advances in Neural Information Processing Systems*, Vol. 4, eds. Moody, J., Hanson, S. and Lippmann, R. (Morgan Kaufmann, 1992), pp. 259–266.
[32] Tumer, K. and Agogino, A., Distributed agent-based air traffic flow management, in *Proc. Sixth Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems* (Honolulu, HI, 2007), pp. 330–337.
[33] Tumer, K., Agogino, A. and Wolpert, D., Learning sequences of actions in collectives of autonomous agents, in *Proc. First Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems* (Bologna, Italy, 2002), pp. 378–385.
[34] Tumer, K., Welch, Z. T. and Agogino, A., Aligning social welfare and agent preferences to alleviate traffic congestion, in *Proc. Seventh Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems* (Estoril, Portugal, 2008).
[35] Tumer, K. and Wolpert, D. (eds.), *Collectives and the Design of Complex Systems* (Springer, New York, 2004).
[36] Tumer, K. and Wolpert, D. H., Collective intelligence and Braess' paradox, in *Proc. Seventeenth National Conf. on Artificial Intelligence* (Austin, TX, 2000), pp. 104–109.
[37] Tuyls, K. and Parsons, S., What evolutionary game theory tells us about multiagent learning, *Artif. Intell.* **171** (2007) 406–416.
[38] Verbeeck, K., Nowe, A. and Tuyls, K., Coordinated exploration in multi-agent reinforcement learning: An application to load balancing, in *Proc. Fourth Int. Joint Conf. on Autonomous Agents and Multi-Agent Systems* (Utrecht, The Netherlands, 2005).
[39] 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.
[40] Vidal, J. M., Multiagent coordination using a distributed combinatorial auction, in *AAAI Workshop on Auction Mechanism for Robot Coordination* (2006).
[41] Vidal, J. M. and Durfee, E. H., The moving target function problem in multi-agent learning, in *Proc. Third Int. Conf. on Multi-Agent Systems* (AAAI/MIT press, 1998), pp. 317–324.
[42] Watkins, C. and Dayan, P., Q-learning, *Mach. Learn.* **8** (1992) 279–292.

[43] Wellman, M. P., Cheng, S.-F., Reeves, D. M. and Lochne, K. M., Trading agents competing: Performance, progress, and market effectiveness, *IEEE Intell. Syst.* **18** (2003) 48–53.
[44] Whiteson, S., Taylor, M. E. and Stone, P., Empirical studies in action selection for reinforcement learning, *Adapt. Behav.* **15** (2007).
[45] Wolpert, D. H. and Tumer, K., Optimal reward functions for members of collectives, *Adv. Complex Syst.* **4** (2001) 265–279.