Counterfactual Exploration for Improving Multiagent Learning

Mitchell Colby
Oregon State University
Corvallis, OR, USA
colbym@engr.orst.edu

Sepideh Kharaghani
Oregon State University
Corvallis, OR, USA
kharaghs@onid.orst.edu

Chris HolmesParker
Parflux LLC
chris@parflux.com

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

ABSTRACT

In any single agent system, exploration is a critical component of learning. It ensures that all possible actions receive *some* degree of attention, allowing an agent to converge to good policies. The same concept has been adopted by multiagent learning systems. However, there is a fundamentally different dynamic in multiagent learning: each agent operates in a non-stationary environment, as a direct result of the evolving policies of other agents in the system. As such, exploratory actions taken by agents bias the policies of other agents, forcing them to perform optimally in the presence of agent exploration. CLEAN rewards address this issue by privatizing exploration (agents take their best action, but internally compute rewards for counterfactual actions). However, CLEAN rewards require each agent to know the mathematical form of the system evaluation function, which is typically unavailable to agents. In this paper, we present an algorithm to approximate CLEAN rewards, eliminating exploratory action noise without the need for expert system knowledge. Results in both coordination and congestion domains demonstrate the approximated CLEAN rewards obtain up to 95% of the performance of directly computed CLEAN rewards, without the need for expert domain knowledge while utilizing 99% less system information.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence —*Multiagent systems*

Keywords

Multiagent learning; CLEAN Rewards

1. INTRODUCTION

As agents in a multiagent system learn policies, they frequently explore by taking random actions in order to determine if these actions improve system performance. Such exploration is critical to finding optimal agent policies. However, learning agents cannot typically differentiate between environmental dynamics and noise caused by the exploratory actions of other agents. This leads to agents learning policies which account for other agents taking random actions, rather than policies which perform well when all agents are following their target, or greedy, policies. Often, system performance can decrease once agents stop exploration, as learned policies depend on the random exploration of other agents [10]. CLEAN rewards address this problem by privatizing agent exploration, removing exploratory action noise from the feedback signals of learning agents.

CLEAN rewards effectively remove exploratory action noise, resulting in superior system performance as compared to more common reward shaping techniques such as difference rewards. When implementing CLEAN rewards, each agent in the system follows their target policy by greedily selecting actions with no exploration. Then, each agent privately computed the system reward if they had taken an exploratory action while all other agents in the system followed their target policies. In this manner, exploratory action noise caused by other agents is removed from an agent's feedback signal, because the exploration for each agent is privatized. This allows for agents to effectively determine if actions are good or not, without the noise caused by other agents in the system taking exploratory actions. Further, agents learn policies which do not depend on other agents in the system exploring, resulting in superior system performance when the learned policies are actually implemented.

Computing CLEAN rewards requires the mathematical form of the system evaluation function which the agents are optimizing, as well as information about the global joint state and joint action of all agents. This information is rarely known to agents in a multiagent system. Thus, although CLEAN rewards address the problem of exploratory action noise, they introduce a new problem of requiring the mathematical form of the system objective function. In order to obtain the benefits of CLEAN rewards while in a realistic setting, a technique to compute CLEAN rewards without the mathematical form of the system evaluation function or global state and action knowledge is needed.

In this paper, we present a generic method for approximating CLEAN rewards using only local state and action knowledge as well as a broadcast value of the system evaluation function (rather than the mathematical form of the

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.*

function itself). We demonstrate how this approximation algorithm is implemented in a congestion and a coordination domain, both of which have extremely high coupling between agents, resulting in high sensitivity to exploratory action noise. Finally, we provide results demonstrating that approximate CLEAN rewards results in better system performance than difference evaluations, and attain up to 95% of the performance of directly computing CLEAN rewards while using 99% less system state and action information.

The specific contributions of this paper are to:

- present a novel algorithm for approximating CLEAN rewards using information easily available to each agent.
- demonstrate empirically that these approximate CLEAN rewards result in performance which is frequently comparable to, and in some cases outperforms, performance attained using directly computed CLEAN rewards.
- demonstrate empirically that these approximate CLEAN rewards provide better performance than rewards which use much more information to compute.
- demonstrate empirically the scalability of these rewards, in large multiagent systems containing up to 5000 agents.

The rest of the paper is organized as follows. Section 2 provides background material and related work. Section 3 details the proposed algorithm to approximate CLEAN rewards. Section 4 details the experimental domains used in this work. Section 5 provides the experimental results. Finally, Section 6 discusses the results and concludes the paper.

2. BACKGROUND

The following sections describe exploratory action noise, CLEAN rewards, difference rewards, and various techniques for approximating evaluation functions.

2.1 Exploratory Action Noise

In practice, agents learn by continuously taking actions in an effort to learn the underlying rewards associated with each action [17]. In order to allow for the expected values of different actions to be learned, exploration strategies are implemented. Agents must balance taking the best known action (exploitation) with taking actions which the agent doesn't know the value of (exploration). The way these two concepts are balanced is known as the exploration-exploitation dilemma [20]. A key problem associated with exploration is that in multiagent systems, this exploration may actually cause noise which leads to agents learning policies which are biased to expect some portion of agents taking random actions.

We present an example to demonstrate the effects of exploratory action noise. Agents learn in the Gaussian Squeeze Domain (Section 4) for 1000 episodes. During learning, agents select actions using ϵ-greedy selection, where $\epsilon = 0.1$. In this case, agents select their best known actions with probability 0.9, and take random exploratory actions with probability 0.1. After 1000 episodes, the agents policies are fixed. We then test performance for varying levels of exploration, by replacing some agents' policies with random exploratory actions. System performance as a function of the exploration level in the system is shown in Figure 1.

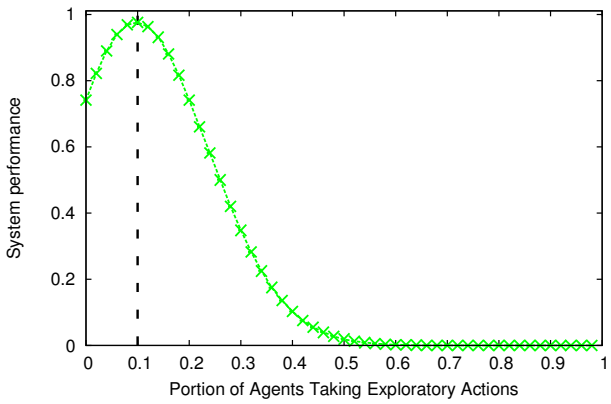

Figure 1: Demonstration of exploratory action noise: agents learned for 1000 episodes with ϵ-greedy action selection where $\epsilon = 0.1$. After learning, the policies are fixed, and we test performance of the learned policies under different amounts of exploration (some agents' policies are replaced with random exploratory policies). Agents perform better in the presence of exploration than they do with no exploration. The highest system performance is when the exploration rate is near ϵ, because agents have learned policies which are biased to include this level of random exploration by other agents in the system.

One of the fundamental difficulties in multiagent systems is the learning process where an agent may not only need to learn how to act in its environment, but may also need to account for the actions of the other learning agents [18]. As seen in Figure 1, exploratory action noise (when agents take random actions with probability 0.1) causes agents to learn policies which perform best when 10% of the agents in the system are taking random actions. This result is not intuitive, but occurs because agents have learned to depend on the exploratory actions of other agents in the system [9]. In particular, this phenomenon is caused by the fact that while learning in a system with agents taking exploratory actions, learning agents are unable to differentiate what parts of their feedback signal are caused by true environmental dynamics, and what parts are caused by exploratory action noise caused by other agents. CLEAN rewards were developed to remove exploratory action noise by privatizing agent exploration.

2.2 CLEAN Rewards

CLEAN rewards are defined as in [10]:

$$C_i = G(z_T - z_{T,i} + c_i) - G(z_T - z_{T,i} + c'_i) \qquad (1)$$

where C_i is the CLEAN reward given to agent i, $G(z)$ is the system evaluation function, z_T is the target state which occurs if all agents act greedily, $z_{T,i}$ is the actual action of agent i, and c_i and c'_i are counterfactual actions of agent i (i.e. alternative actions the agent may have taken instead of following its target policy). CLEAN rewards replace the contribution of the agent's target action with two different counterfactual actions. The agent then is provided with a reward for the action c_i, rather than $z_{T,i}$.

CLEAN rewards remove exploratory action noise of other agents in the system, as all agents are actually following their target policies. Exploration is done offline, by computing the reward an agent would have received for exploring. CLEAN rewards have provided excellent empirical results in a variety of domains, including the Gaussian squeeze domain and defect combination problem [9, 10].

In order to compute them, CLEAN rewards require an accurate model of the underlying system objective $G(z)$, as all exploration is done privately by each agent. In practice, agents never have such information. Thus, although CLEAN rewards can improve learning by removing exploratory action noise, the requirement for an accurate model of $G(z)$ renders them unusable in most real world problems. In this paper, we demonstrate that CLEAN rewards may be approximated such that the performance gains from removing exploratory action noise are retained.

2.3 Difference Rewards

The agent specific Difference Reward $D_i(z)$ is defined as [2]:

$$D_i(z) = G(z) - G(z_{-i} + c_i) \tag{2}$$

where $G(z)$ is the global reward, and $G(z_{-i} + c_i)$ is the global reward when agent i is removed from the system and replaced with some counterfactual agent c_i. Intuitively, the difference reward gives agent i's specific impact on the overall system performance. Note that:

$$\frac{\partial D_i(z)}{\partial a(i)} = \frac{\partial G(z)}{\partial a_i} \tag{3}$$

where $a(i)$ is agent i. Thus, an agent acting to increase the value of the difference reward will also increase the global reward. This property is termed *alignment* [2]. Further, because the second term in the difference reward removes all portions of the system reward not related to agent i, noise in the learning signal caused by the actions of other agents is significantly reduced. This property is termed *sensitivity*. By being aligned with the global reward and sensitive to an individual agent's actions, difference rewards provide extremely effective feedback to each agent, allowing them to learn better policies than when using global rewards. Difference rewards have provided excellent empirical results in a variety of domains, including sensor coordination and mobile robot coordination [2, 19].

2.4 Approximation of Evaluation Functions

In cases where the system evaluation function is unknown, it may be approximated in order to provide better feedback to learning agents [3, 4]. In the case of reinforcement learning, reward functions are modeled; in the case of evolutionary algorithms, fitness functions are modeled. In either case, approximation of system evaluation functions allows for agent-specific evaluation functions to be easily shaped based on the overall system evaluation function. The following sections describe different approaches for approximating system evaluation functions, as well as previous work involving approximating difference evaluations.

Function approximation is commonly seen in reinforcement learning applications, where the value function or system state is approximated due to limited state information [8, 11, 14]. Value functions are often approximated when only partial state information is available, often using neural networks, maps, or some other type of function approximator [1, 5, 16]. The conclusion that only partial state information is necessary to accurately model value functions suggests these types of approaches can be successfully extended to multiagent learning problems, where only partial state information is available to each agent.

Fitness functions have also been approximated for use in evolutionary algorithms [6, 13]. This fitness approximation is typically used because the number of fitness function evaluations dominates the optimization cost in evolutionary algorithms, and fitness approximation typically decreases time to convergence [12, 15]. However, these approaches can easily be extended to cases in which the mathematical form of the fitness function is unknown, as in cases where the difference evaluation must be approximated. All of the methods described above are single agent cases, but there has also been research investigating approximating system evaluation functions in multiagent learning settings.

As seen above, there is a wide range of research involving approximation in reinforcement learning and evolutionary algorithms. However, to date, there has been no research involving approximating CLEAN rewards. This paper presents the first approximation of CLEAN rewards, demonstrating how they may be implemented in systems when agents do not have global knowledge about the system state or the system evaluation function.

3. CLEAN APPROXIMATION ALGORITHM

In this section, we introduce a general algorithm for approximating CLEAN rewards, and then demonstrate how this algorithm is implemented in a stateless discrete action domain. For the purpose of this analysis, we assume a cooperative multiagent system aims to coordinate in order to optimize some system level objective function $G(z)$, and that the true value of $G(z_T)$ is broadcast to each agent after a target joint action z_T is selected based on each agent's greedy policy. Recall that the CLEAN reward is defined as:

$$C_i = G(z_T - z_{T,i} + c_i) - G(z_T - z_{T,i} + c'_i) \tag{4}$$

As we assume that the value of $G(z_T)$ is broadcast to each agent, approximating the CLEAN reward requires approximating $G(z)$ based on an agent's local information. Given the particular domain, a suitable function approximator is chosen. For a stateless domain with discrete actions, this approximator may simply be a tabular function approximator. For a stateful domain with continuous actions, this approximator may be a neural network. Note that these are not the only possible options for these function approximators, but are potential choices which match the domains they will be used in.

The CLEAN approximation algorithm is presented in Algorithm 1. At each time step, each agent takes a greedy action, and $G(z_T)$ is computed and broadcast to each agent, where z_T is the system state when each agent follows its target policy (greedily selects actions). Each agent i maintains a private approximation for $G(z)$, denoted as $\hat{G}_i(z_i)$. This is not a full approximation of the system evaluation function; rather, it is an approximation of $G(z)$ as a function of only one agent's state and action. The states and actions of other agents in the system are embedded into the function approximation. Note that the only information available to the agent includes the agent's state, action choice, and the

Initialize N agents
foreach *Agent* **do**
  Initialize private function approximator $\hat{G}_i(z_i)$
end
foreach *Learning Step* **do**
  foreach *Agent* **do**
    collect local state information s_i
    greedily select action a_i using agent's policy
    $z_{T,i} = \{s_i, a_i\}$
    $z_T = \{z_{T,1}, z_{T,2}, ..., z_{T,n}\}$
  end
  Calculate $G(z_T)$ based on joint action
  Broadcast $G(z_T)$ to each agent
  foreach *Agent* **do**
    Update $\hat{G}_i(z_i)$ using s_i, a_i, and $G(z_T)$
    $\hat{C}_i = \hat{G}(c_i) - \hat{G}(c_i')$
    Update policy/value table based on $\hat{C}_i$
  end
end

Algorithm 1: Approximation of CLEAN rewards. The functional form of $\hat{G}$ depends on the specific domain. For example, a stateless discrete action domain may use a tabular function approximator, while a continuous state and action domain may use a neural network.

broadcast value of $G(z_T)$. Each agent's approximation of $G(z)$ is therefore a mapping from that agent's state and action to the value of the system objective function. It is of note that this approximation is initially very noisy, because the state and action information used to approximate $G(z)$ is limited only to one agent's state and action. However, as learning progresses, the policies of each agent begin to converge; as the policies of other agents converge, approximating $G(z)$ using only one agent's state and action becomes less noisy, because the variance in the joint action for a particular system-level state is reduced. Given an agent's approximation $\hat{G}_i(z_i)$, the CLEAN reward can be estimated as:

$$\hat{C}_i(z_i) = \hat{G}_i(c_i) - \hat{C}_i(c_i') \tag{5}$$

where $\hat{C}_i(z_i)$ is agent i's approximation of the CLEAN reward. In order to evaluate $\hat{C}_i(c_i)$, a default action c_i' is chosen for each agent at the beginning of each episode for evaluation, and a random action c_i is chosen to be evaluated at each time step. In other words, the approximation of the CLEAN reward determines *the difference between the system objective function for a random action c_i and a default action c_i', assuming all other agents in the system are following their target policies.*

The implementation of this approximation approach in a stateless discrete action domain is detailed in the following sections. We demonstrate how a multiagent reinforcement learning algorithm with an approximation of CLEAN rewards may be implemented in the stateless discrete action domain. The general algorithm presented in Algorithm 1 can be implemented in any domain where a broadcast value of $G(z)$ is available, whether the states or actions are continuous or discrete. For simplicity in demonstrating the CLEAN approximation algorithm, we choose to demonstrate this approximation approach in a stateless discrete action setting.

3.1 Stateless Discrete Action Domains

Initialize N agents
foreach *Agent* **do**
  Initialize private value table $V_i(a)$
  Initialize private function approximator $\hat{G}_i(a_i)$
end
foreach *Learning Step* **do**
  foreach *Agent* **do**
    greedily select action a_i using agent's policy
  end
  Calculate $G(a)$ based on joint action
  Broadcast $G(a)$ to each agent
  foreach *Agent* **do**
    $\hat{G}_i(a_i) \leftarrow \alpha_1 \cdot \hat{G}_i(a_i) + (1 - \alpha_1) \cdot G(a)$
    $\hat{C}_i = \hat{G}_i(c_i) - \hat{G}_i(c_i')$
    $V_i(c_i) \leftarrow \alpha_2 \cdot V_i(c_i) + (1 - \alpha_2) \cdot \hat{C}_i$
  end
end

Algorithm 2: Stateless Discrete-Action Multiagent Reinforcement Learning using $D_i(s, a)$ Approximation

We now demonstrate how Algorithm 1 may be implemented for a multiagent reinforcement learning algorithm in a stateless discrete action domain. The implementation of Algorithm 1 in a multiagent reinforcement learning problem with a stateless discrete action domain is given in Algorithm 2. Note that such problems may also be solved with coevolutionary algorithms. In a stateless discrete action domain, each agent maintains a value vector $V_i(a)$ containing the expected value of each possible action. Each agent also maintains an approximation of $G(a)$, which is simply a vector containing the estimated value of the system evaluation function corresponding to each action the agent may take. At each learning step, each agent i selects an action a_i. The system evaluation function $G(a)$ is then calculated based on the joint action $a = \{a_1, a_2, ..., a_n\}$, and this value is broadcast to each agent in the system. Each agent then updates its approximation $\hat{G}_i(a_i)$ according to:

$$\hat{G}_i(a_i) \leftarrow (1 - \alpha_1) \cdot \hat{G}_i(a_i) + \alpha_1 \cdot G(a) \tag{6}$$

Once each agent has updated its approximation of $G(a)$, the CLEAN for each agent is estimated as:

$$\hat{C}_i = \hat{G}_i(c_i) - \hat{C}_i(c_i') \tag{7}$$

where c_i' is a randomly chosen action for evaluation, and c_i' is some default action. The value table is then updated using the estimate of the CLEAN reward:

$$Q_i(c_i) \leftarrow (1 - \alpha_2) \cdot Q_i(c_i) + \alpha_2 \cdot \hat{C}_i \tag{8}$$

4. EXPERIMENTAL DOMAINS

In the following sections, we detail the two experimental domains used in this work: the Defect Combination Problem (DCP) and the Gaussian Squeeze Domain (GSD). These domains were chosen for two reasons. First, the DCP is a coordination domain, while the GSD is a congestion domain. The inclusion of both of these types of domains illustrates how approximate CLEAN rewards perform in varying types of problems. Second, both of these domains are extremely

sensitive to exploratory action noise; even in very large systems (e.g. 1000 agents), a single agent changing its policy can dramatically affect the overall system performance. As both of these domains are sensitive to exploratory action noise, they provide insight on how CLEAN rewards can improve system performance by privatizing agent exploration.

4.1 Defect Combination Problem

The Defect Combination Problem (DCP) assumes that there exists a set of imperfect sensors X which have constant measurement errors due to manufacturing defects or imperfections [7, 19]. Each sensor $x_i \in X$ has an attenuation a_i in its measurement. Thus, if sensor x_i is measuring some value A, its measurement is $A + a_i$. The individual sensor attenuation values are drawn from a Gaussian distribution with a mean of zero and some variance σ. Solving the DCP involves choosing a subset of the sensors such that the aggregate attenuation of the combined readings is minimized, which is equivalent to minimizing:

$$G(z) = \frac{\left|\sum_{i=1}^{N} n_i a_i\right|}{\sum_{i=1}^{N} n_i} \tag{9}$$

where N is the number of sensors in the system, $n_i \in \{0, 1\}$ is an indicator function based on whether the sensor is "on" or "off," and $G(z)$ is based on the aggregated attenuation of the combined sensor readings. The DCP is a large distributed agent coordination problem where each agent (device) needs to determine whether or not to be part of the aggregate device (determine whether to turn "on" or "off").

4.2 Gaussian Squeeze Domain

The Gaussian Squeeze Domain (GSD) is a problem involving agents which must coordinate their actions in order to optimize for a "capacity" in a Gaussian objective function [10]. The objective function to be maximized is:

$$G = xe^{\frac{-(x-\mu)^2}{\sigma^2}} \tag{10}$$

where $x = \sum_i x_i$ is the cumulative sum of the actions of agents (where x_i is the contribution of agent i), μ is the mean of the system objective's Gaussian (the target x the agents must coordinate to achieve), and σ is the standard deviation of the system objective's Gaussian. The goal of the agents is to choose their individual actions x_i such that the sum of their individual actions optimizes Equation 10. Each agent has n actions, ranging in value from zero to $(n-1)$. The Gaussian squeeze domain is a congestion domain, where each agent determines their allocation of resources, with the goal being that a set number of resources (defined by μ) is not over- or under-utilized. Adjusting the standard deviation σ affects the coordination complexity of the problem. For low values of σ, coordination complexity is high as the gradient around the optimal point is large. For high values of σ, coordination complexity is lower as the gradient around the optimal point is more gradual.

5. EMPIRICAL RESULTS

The following sections provide the results detailing the performance of the CLEAN approximation algorithm in both a coordination domain (Defect Combination Problem) and a congestion domain (Gaussian Squeeze Domain).

5.1 Defect Combination Problem

Approximate CLEAN rewards are tested in the DCP and compared to global rewards, difference rewards, and directly computed CLEAN rewards. For each experiment analyzed, 400 independent runs are conducted and we report the average performance over all runs. The first experiment involves analyzing the performance of 1000 agents, when the exploration rate ϵ is 0.01, the standard deviation for the sensor noise distribution σ is 1, the learning rate α_1 is 0.005, and the update rate for the agents' function approximators α_2 is 0.001. The results for this experiment are shown in Figure 2, where lower values (corresponding to lower sensor error) are preferable.

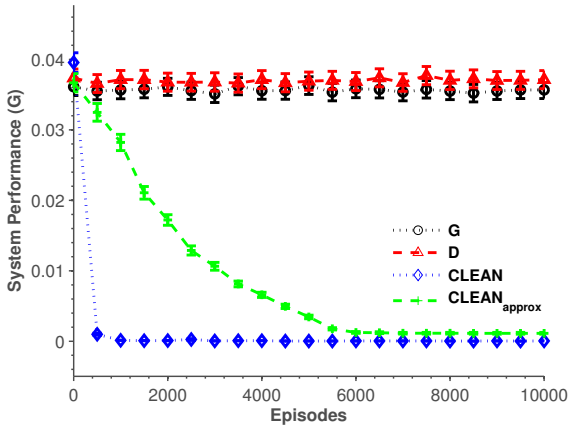

Figure 2: DCP results with 1000 agents, where $\epsilon = 0.01$, $\sigma = 1$, $\alpha_1 = 0.005$, and $\alpha_2 = 0.001$. Approximated CLEAN rewards outperform difference rewards and global rewards, and achieve 98% of the improvement over difference evaluations that the analytically computed CLEAN rewards attain.

As seen in Figure 2, approximate CLEAN rewards outperform global and difference rewards, and attain 98% of the performance of directly computed CLEAN rewards. It is of note that this 2% performance drop is accompanied by a 99.9% drop in system information used to compute the reward, as CLEAN rewards utilize the entire system joint action, while approximate CLEAN rewards only utilize an individual agent's action. This experiment demonstrates not only the ability of CLEAN rewards to effectively filter out exploration noise to improve performance, but that the benefits of CLEAN rewards can be attained with a small fraction of the overall information available within the system.

For the next experiment, the learning rate α_1 is changed to 0.01 rather than 0.005. This results in agents placing a higher emphasis on recently received rewards. The results for this experiment are shown in Figure 3.

As seen in Figure 3, approximate CLEAN rewards outperform not only global and difference rewards, but they actually outperform directly computed CLEAN rewards. This counter-intuitive result requires closer analysis. As agent exploration is privatized with CLEAN rewards, multiple agents may simultaneously change their policies while assuming that all other agents in the system will retain their current policies. For example, consider a system with one hundred agents. Suppose that during their private exploration,

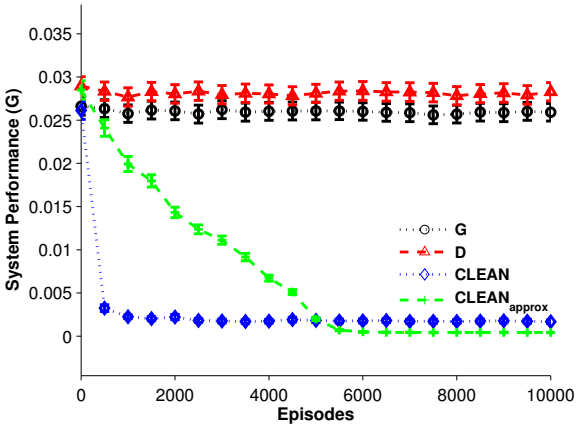

Figure 3: DCP results with 1000 agents, where $\epsilon = 0.01$, $\sigma = 1$, $\alpha_1 = 0.01$, and $\alpha_2 = 0.001$. Approximated CLEAN rewards outperform difference rewards and global rewards, as well as directly computed CLEAN rewards.

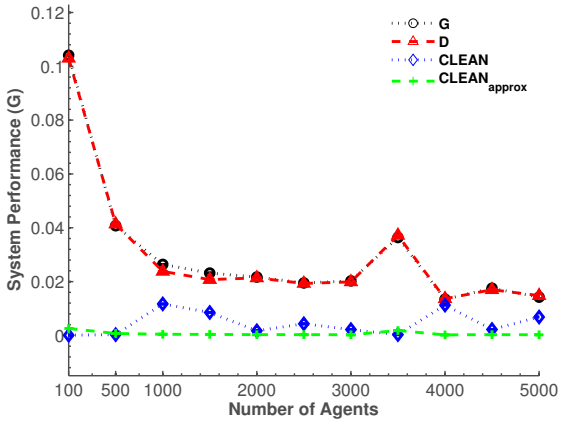

Figure 4: DCP results: Scaling. Regardless of system size, approximate CLEAN rewards outperform global and difference rewards, and outperform directly computed CLEAN rewards in some cases.

twenty of these agents determine that they should change their policies based on information derived from CLEAN rewards. When using CLEAN rewards, each agent in the system determines if an action they did not take would result in higher system performance, *assuming that every other agent in the system retains their current policy.* When all twenty of these agents change their actions simultaneously, the result may ultimately be a drop in system performance, as the assumption that other agents in the system remains static does not hold. Agents which approximate CLEAN rewards each have different local approximations of the system evaluation function, all of which focus on that particular agent's impact on the system evaluation function. This results in the number of agents altering their policies in any given time step to be lower than when directly computing CLEAN rewards. Intuitively, approximated CLEAN rewards slow down the process of agent's altering their policies, resulting in a less dynamic learning environment. This allows for individual agents to change their policies asynchronously, resulting in a lower frequency of decreased system performance as a result of widespread changes in the target policies of each agent. This counter-intuitive result only holds for cases where the approximation of the system evaluation function is sufficiently accurate, such that the benefit of breaking the symmetry of CLEAN actions overcomes the inaccuracies caused by the approximation.

The final experiment in the DCP involves analyzing the scalability of the system. For system sizes varying from 100 to 5000 agents, agents learn for 10000 episodes, using either difference, global, CLEAN, or approximate CLEAN rewards. Figure 4 shows converged system performance as a function of the number of agents in the system.

As seen in Figure 4, regardless of system size, approximate CLEAN rewards outperform both global and difference rewards. For some system sizes, approximate CLEAN rewards perform similarly to directly computed system rewards. For other system sizes (1000-1500 agents and 4000-5000 agents), approximate CLEAN rewards actually outperform directly computed CLEAN rewards. This result demonstrates that approximating CLEAN rewards provides the performance advantages of CLEAN rewards, without the restrictive requirement for global system information. In particular, global knowledge about the joint action becomes increasingly difficult to collect as the number of agents in the system grows. By approximating CLEAN rewards without requiring the global joint action, we have provided a technique to implement CLEAN rewards in large multiagent systems.

5.2 Gaussian Squeeze Domain

Approximate CLEAN rewards are tested in the GSD, and are compared to global rewards, difference rewards, and directly computed CLEAN rewards. All results are the average performance over 50 statistical runs. For all experiments, the standard deviation of the objective function σ is 400, the mean of the objective function μ is 200, the learning rate α_1 is 0.1, the update rate for agents' function approximators is 0.1, and the exploration rate ϵ is 0.1. The first experiment involves training 1000 agents, where each agent can select from 15 available actions. The results of this experiment are shown in Figure 5, where higher values are preferable.

As seen in Figure 5, approximate CLEAN rewards significantly outperform both global and difference rewards, and result in 94% of the performance of directly computed CLEAN rewards. In this case, the approximation of CLEAN rewards results in a 6% decrease in system performance while using 99.9% less information about the global joint action to compute. Further, approximate CLEAN rewards converge to their final policy faster than directly computed CLEAN rewards, due to the fact that the learning environment is less dynamic while approximating CLEAN rewards (see Section 5.1. The next experiment involves 1000 agents with 10 actions to choose from, with all other experimental parameters being held constant. These results are shown in Figure 6.

As seen in Figure 6, approximate CLEAN rewards significantly outperform global rewards, but only attain 75% of the performance of directly computed CLEAN rewards,

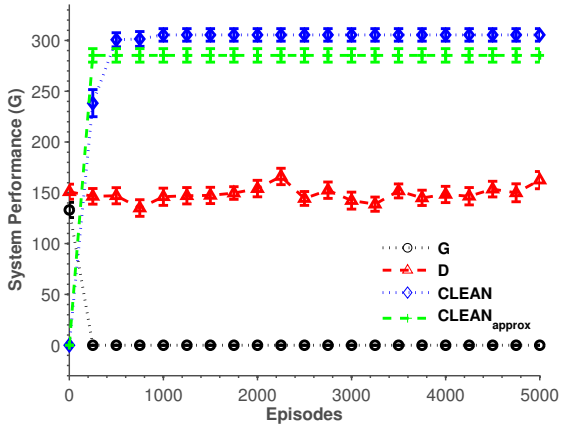

Figure 5: GSD results with 1000 agents, where agents can select from 15 available actions, the standard deviation of the objective function σ is 400, the mean of the objective function μ is 200, the learning rate α_1 is 0.1, the update rate for agents' function approximators is 0.1, and the exploration rate ϵ is 0.1. Approximate CLEAN rewards significantly outperform both global and difference rewards, and provide 94% of the performance of directly computed CLEAN rewards.

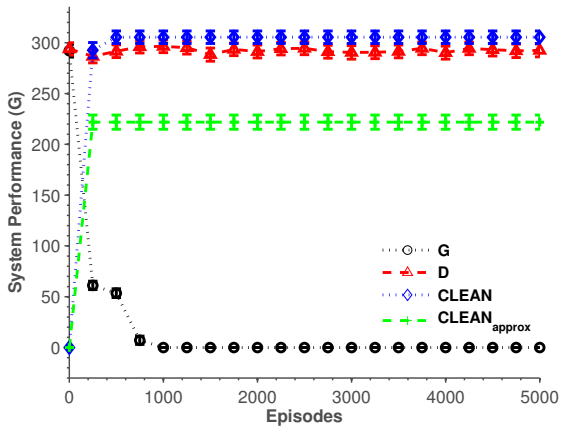

Figure 6: GSD results with 1000 agents, where agents can select from 10 available actions, the standard deviation of the objective function σ is 400, the mean of the objective function μ is 200, the learning rate α_1 is 0.1, the update rate for agents' function approximators is 0.1, and the exploration rate ϵ is 0.1. Approximate CLEAN rewards significantly outperform global rewards, but only achieve 75% of the performance of directly computed CLEAN rewards, and 78% of the performance of difference rewards.

and 78% of the performance of difference rewards. It is important to note that both CLEAN rewards and difference rewards require the global joint action of the system to compute, while approximated CLEAN rewards only depend on a single agent's action. As the size of this system is large (2000 agents), each agent would have difficulty learning the global joint action of the system. Thus, although approximate CLEAN rewards do not provide the performance of directly computed CLEAN rewards or difference rewards, they do not rely on information that is difficult or impossible for a single agent to obtain.

We now analyze the differences in results between Figures 5 and 6. The only difference in experimental setups is the number of actions which each agent may take, which determines how drastically exploratory action noise can affect system performance. In the case of the GSD, increasing from 10 to 15 actions per agents dramatically increases the effects of exploratory action noise. As seen in Figures 5 and 6, when exploratory action noise is not expected to have a large impact on the system (agents only pick from 10 actions), then difference rewards outperform approximate CLEAN rewards. When exploratory action noise does have a large impact on the system, then approximate CLEAN rewards outperform difference rewards. This result illustrates the tradeoff between exploratory action noise and the noise caused by the approximation of the system evaluation function in approximate CLEAN rewards. To further investigate this tradeoff, we analyzed converged system performance for each reward structure as a function of the number of actions each agent in the GSD could take for a 1000 agent system. These results are shown in Figure 7.

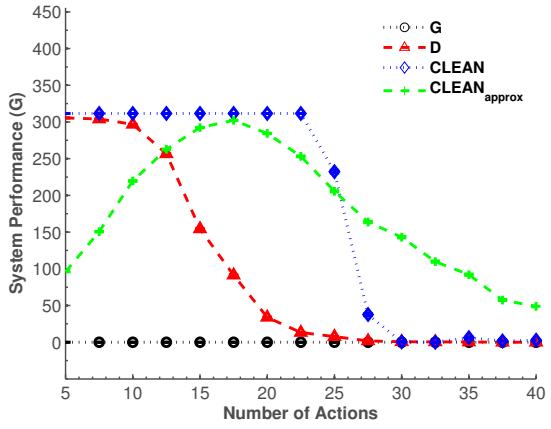

Figure 7: GSD results: scaling the number of actions.

As seen in Figure 7, as the level of potential exploratory action noise rises with the number of actions in the system, approximate CLEAN rewards overtake difference rewards and provide better performance. As the system complexity is increased further to 25 actions for each agent, approximate CLEAN rewards overtake CLEAN rewards. This is due to the less dynamic learning environment created by approximate CLEAN rewards; with CLEAN rewards, too many agents change their target policies on the assumption that the other agents in the system are remaining static. Approximate CLEAN rewards overcome this difficulty because the nature of their approximations results in a more asynchronous process of agents modifying their policies. To demonstrate the scalability of approximate CLEAN rewards, we conduct the final experiment in the GSD with 2000 agents and 15 actions. These results are shown in Figure 8.

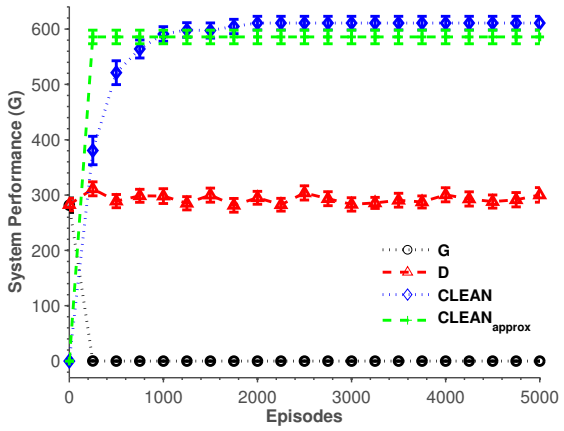

Figure 8: GSD results with 2000 agents, where agents can select from 15 available actions, the standard deviation of the objective function σ is 400, the mean of the objective function μ is 200, the learning rate α_1 is 0.1, the update rate for agents' function approximators is 0.1, and the exploration rate ϵ is 0.1. Approximate CLEAN rewards significantly outperform both global and difference rewards, and provide 96% of the performance of directly computed CLEAN rewards.

As seen in Figure 8, approximate CLEAN rewards now significantly outperform difference rewards, and result in 96% of the performance of directly computed CLEAN rewards. This result in addition to the results from Figure 6 demonstrate a tradeoff in approximating CLEAN rewards. While approximating CLEAN rewards, exploratory action noise is removed, but noise due to the approximation of the system evaluation function is added. In the case of 10 actions, a single agent has a smaller ability to drastically impact system performance, meaning that exploratory action noise has a smaller effect on system performance than in the case of 15 actions. As approximate CLEAN rewards result in superior system performance compared to difference rewards in the 15 action system, but perform worse in the 10 action system, we see that in the case where agents can only select 10 actions, exploratory action noise is less detrimental to system performance than noise caused by the approximation of the system evaluation function. However, in the 15 action case, exploratory action noise is more detrimental to system performance than noise caused by the approximation. This demonstrates that approximate CLEAN rewards should be used in cases where exploratory action noise can significantly impact system performance; these cases are characterized by the ability of a single agent to drastically impact the value of the system evaluation function.

6. DISCUSSION AND CONCLUSION

Agents in multiagent learning systems must often take exploratory actions in order to discover the mapping between actions and the expected value associated with those actions. However, agents are often unable to determine what portions of their feedback signal are caused by environmental dynamics, and what portions are caused by exploratory action noise from other agents. As a result, agents are biased to learn policies which depend on other agents taking exploratory actions, rather than policies which perform optimally when all agents are following their target policies.

CLEAN rewards address the problem of exploratory action noise by privatizing exploration, removing all noise from agent feedback signals caused by the exploratory actions of other agents. In systems where agent exploration can have a large impact on system performance, it is difficult for agents to evaluate the effectiveness of their selected actions. By privatizing exploration, CLEAN rewards allow agents to determine the value associated with the actions they select without being biased by noise caused by agent exploration. This results in policies which can outperform traditional reward signals such as difference rewards or global rewards.

However, CLEAN rewards require global state information, as well as the mathematical form of the system evaluation function in order to compute them. This information is typically unavailable to agents; thus, even though CLEAN rewards solve the problem of exploratory action noise, they introduce a problem of reward computability.

In this work, we demonstrate how CLEAN rewards may be approximated by learning agents using only local state and action information, as well as a broadcast value of the system evaluation function. This approximation approach does not require any global state or action knowledge, nor does it require expert knowledge about the system evaluation function. By approximating the global evaluation function in order to evaluate CLEAN rewards, the benefits of removing exploratory action noise are retained, while the restrictions of requiring expert system knowledge is removed.

Our results demonstrate that CLEAN rewards can be successfully approximated, and that these approximations typically outperform both global and difference rewards in both a congestion domain and coordination domain, with large systems containing up to 5000 agents. These results demonstrate that approximated CLEAN rewards can attain similar levels of performance as directly computed CLEAN rewards, and in some cases may attain even better performance. Results also demonstrate that approximate CLEAN rewards do not always result in superior system performance compared to difference rewards. In particular, in cases where exploratory action noise is relatively small compared to noise in the approximation of the system evaluation function, then approximate CLEAN rewards actually result in worse performance than difference rewards. This demonstrates that approximate CLEAN rewards should only be used in systems where exploratory action noise has a strong detrimental effect on system performance. In particular, approximate CLEAN rewards are most beneficial in systems where a single agent can dramatically impact system performance.

Future work on this research has multiple avenues. First, we will investigate the tradeoff between exploratory action noise and noise caused by the approximation, to determine if we can characterize systems to predict when approximate CLEAN rewards will be beneficial. Second, we will investigate CLEAN approximations when both terms (rather than one) are approximated, which could potentially improve sensitivity, but decrease alignment.

Acknowledgements

This work was partially supported by NETL DE-FE0011403.

REFERENCES

[1] J. Abounadi, D. Bertsekas, and V. Borkar. Stochastic Approximation for Non-Expansive Maps: Application to Q-Learning Algorithms. Technical report, SIAM Journal on Control and Optimization, 1998.

[2] A. K. Agogino and K. Tumer. Analyzing and visualizing multiagent rewards in dynamic and stochastic environments. *Journal of Autonomous Agents and Multi-Agent Systems*, 17(2):320–338, 2008.

[3] K. Anderson and Y. Hsu. Genetic crossover strategy using an approximation concept. In *In IEEE Congress on Evolutionary Computation*, pages 527–533.

[4] J. Branke, C. Schmidt, and H. Schmeck. Efficient Fitness Estimation in Noisy Environment. In *Proceedings of Genetic and Evolutionary Computation*, pages 243–250.

[5] D. Brillinger. Learning a Potential Function From a Trajectory. *Signal Processing Letters, IEEE*, 14(11):867–870, 2007.

[6] D. Buche, N. Schraudolph, and P. Koumoutsakos. Accelerating Evolutionary Algorithms Using Fitness Function Models. In *Proc. Workshops Genetic and Evolutionary Computation Conference*, 2003.

[7] D. Challet and N. Johnson. Optimal combinations of imperfect objects. *Physical Review Letters*, 89(2):028701, 2002.

[8] C. Gaskett, L. Fletcher, and A. Zelinsky. Reinforcement Learning for Visual Servoing of a Mobile Robot. In *In Proceedings of the Australian Conference on Robotics and Automation (ACRA 2000)*, 2000.

[9] C. HolmesParker. Clean learning to improve coordination and scalability in multiagent systems. 2013.

[10] C. HolmesParker, M. Taylor, A. Agogino, and K. Tumer. Clean rewards to improve coordination by removing exploratory action noise. 2014.

[11] Y. Isukapalli. An Analytically Tractable Approximation for the Gaussian Q-Function. In *IEEE Communications Letters*, volume 12, pages 669 – 671.

[12] Y. Jin. A Comprehensive Survey of Fitness Approximation in Evolutionary Computation. *Soft Comput.*, 9(1):3–12, Jan. 2005.

[13] J. Martikainen and S. Ovaska. Fitness Function Approximation by Neural Networks in the Optimization of MGP-FIR Filters. In *Adaptive and Learning Systems, 2006 IEEE Mountain Workshop on*, pages 7–12, 2006.

[14] H. Murao and S. Kitamura. Incremental State Acquisition for Q-Learning by Adaptive Gaussian Soft-Max Neural Network. In *Proceedings of the 1998 IEEE ISIC/CIRA/ISAS Joint Conference*, 1999.

[15] V. Oduguwa and R. Roy. Multiobjective Optimization of Rolling Rod Product Design Using Metamodeling Approach. pages 1164–1171, 2002.

[16] P. Pandey and D. Pandey. Reduct based q-learning: an introduction. In *Proceedings of the 2011 International Conference on Communication*, ICCCS '11, pages 285–288, New York, NY, USA, 2011. ACM.

[17] D. Ray, A. Mandal, S. Mazumder, and S. Mukhopadhay. Application of single-agent q-learning for light exploration. 2010.

[18] R. Sutton and A. Barto. *Reinforcement Learning, an Introduction*. MIT Press, 1998.

[19] K. Tumer. Designing agent utilities for coordinated, scalable and robust multiagent systems. In P. Scerri, R. Mailler, and R. Vincent, editors, *Challenges in the Coordination of Large Scale Multiagent Systems*, pages 173–188. Springer, 2005.

[20] K. Zhang and W. Pan. The two facets of the exploration-exploitation dilemma. 2006.