This version of the paper is modified slightly from the version that appeared in the official ICML proceedings. The only substantive change is due to the fact that, based on subsequent discussions with peers, we identified a technical flaw in the ways that our MLeS and CMLeS algorithms were guaranteeing safety.

Specifically, it was possible that MLeS (and also CMLeS) may converge to modeling an arbitrary opponent with a memory size less than or equal to K_{max}, and achieve a return less than its security value. In this version, that flaw is rectified by requiring that the algorithms check whether their average return is less than their security value, periodically at chosen intervals, and if so, requiring that they play their maximin strategy sufficiently many times to bring the return back up to being close to the safety value.

This change is manifest in the text after Theorem 3.2, Step 21 of the main CMLeS algorithm, and the explanation of Step 21 on page 7.

The only other differences from the published version are small textual changes required to make the paper still fit within 8 pages, and fix a few typos.

For any questions of a technical nature, please contact the authors.

Convergence, Targeted Optimality, and Safety in Multiagent Learning

Doran Chakraborty CHAKRADO@CS.UTEXAS.EDU
Peter Stone PSTONE@CS.UTEXAS.EDU
Department of Computer Science, University of Texas, 1 University Station C0500, Austin, Texas 78712, USA

Abstract

This paper introduces a novel multiagent learning algorithm, *C*onvergence with *M*odel *L*earning and *S*afety (or *CMLeS* in short), which achieves convergence, targeted optimality against memory-bounded adversaries, and safety, in arbitrary repeated games. The most novel aspect of *CMLeS* is the manner in which it guarantees (in a PAC sense) targeted optimality against memory-bounded adversaries, via efficient exploration and exploitation. *CMLeS* is fully implemented and we present empirical results demonstrating its effectiveness.

1. Introduction

In recent years, great strides have been made towards creating autonomous agents that can learn via interaction with their environment. When considering just an individual agent, it is often appropriate to model the world as being *stationary*, meaning that the same action from the same state will always yield the same (possibly stochastic) effects. However, in the presence of other independent agents, the environment is not stationary: an action's effects may depend on the actions of the other agents. This non-stationarity poses the primary challenge of *multiagent learning* (MAL) (Buşoniu et al., 2008) and comprises the main reason that it is best considered distinctly from single agent learning.

The simplest, and most often studied, MAL scenario is the stateless scenario in which agents repeatedly interact in the stylized setting of a matrix game (a.k.a. normal form game). In the multiagent literature, various criteria have been proposed to evaluate MAL algorithms, emphasizing what behavior they will converge to against various types of opponents,[1] in such settings. The contribution of this paper is that it proposes a novel MAL algorithm, *CMLeS*, that for a multi-player multi-action (arbitrary) repeated game, achieves the following three goals:

1. Convergence : converges to playing a Nash equilibrium in self-play (other agents are also *CMLeS*);
2. Targeted Optimality : achieves close to the best response with a high probability, against a set of memory-bounded, or *adaptive*,[2] opponents whose memory size is upper bounded by a known value K_{max}. The same guarantee also holds for opponents which eventually become memory-bounded;
3. Safety : achieves very close to at least its security value against other arbitrary opponents, with a high probability;

1.1. Related work

Bowling et al. (Bowling & Veloso, 2001) were the first to put forth a set of criterion for evaluating multiagent learning algorithms. In games with two players and two actions per player, their algorithm WoLF-IGA converges to playing best response against stationary, or *memoryless*, opponents (rationality), and converges to playing the Nash equilibrium in self-play (convergence). Subsequent approaches extended the rationality and convergence criteria to arbitrary (multi-player, multi-action) repeated games (Banerjee & Peng, 2004; Conitzer & Sandholm, 2006). Amongst them, AWESOME (Conitzer & Sandholm, 2006) achieves convergence and rationality in arbitrary repeated games without requiring agents to observe each others' mixed strategies. However, none of the above algorithms have any guarantee about the payoffs achieved when

Appearing in *Proceedings of the 27th International Conference on Machine Learning*, Haifa, Israel, 2010. Copyright 2010 by the author(s)/owner(s).

[1]Although we refer to other agents as opponents, we mean any agent (cooperative, adversarial, or neither)

[2]Consistent with the literature (Powers et al., 2005), we call memory-bounded opponents as *adaptive opponents*.

they face arbitrary non-stationary opponents. More recently, Powers et al. proposed a newer set of evaluation criteria that emphasizes compatibility, targeted optimality and safety (Powers & Shoham, 2005). Compatibility is a stricter criterion than convergence as it requires the learner to converge within ϵ of the payoff achieved by a Pareto optimal Nash equilibrium. Their proposed algorithm, PCM(A) (Powers et al., 2007) is, to the best of our knowledge, the only known MAL algorithm to date that achieves compatibility, safety and targeted optimality against adaptive opponents in arbitrary repeated games.

1.2. Contributions

CMLeS improves on AWESOME by guaranteeing both safety and targeted optimality against adaptive opponents. It improves upon PCM(A) in five ways.

1. The only guarantees of optimality against adaptive opponents that PCM(A) provides are against the ones that are drawn from an initially chosen target set. In contrast, *CMLeS* can model every adaptive opponent whose memory is bounded by K_{max}. Thus it does not require a target set as input: its only input is K_{max}.
2. Once convinced that the other agents are not self-play agents, PCM(A) achieves targeted optimality against adaptive opponents by requiring all feasible joint histories of size K_{max} to be visited a sufficient number of times. K_{max} for PCM(A) is the maximum memory size of any opponent from its target set. *CMLeS* significantly improves this by requiring a sufficient number of visits to only all feasible joint histories of size K, the true opponent's memory size.
3. Unlike PCM(A), *CMLeS* promises targeted optimality against opponents which eventually become memory-bounded with $K < K_{max}$.
4. PCM(A) can only guarantee convergence to a payoff within ϵ of the desired Nash equilibrium payoff with a probability δ. In contrast, *CMLeS* guarantees convergence in self-play with probability 1.
5. *CMLeS* is relatively simple in its design, in comparison to PCM(A).

The remainder of the paper is organized as follows. Section 2 presents background and definitions, Section 3 and 4 presents our algorithm, Section 5 presents empirical results and Section 6 concludes.

2. Background and Concepts

This section reviews the definitions and concepts necessary for fully specifying *CMLeS*.

A *matrix game* is defined as an interaction between n agents. Without loss of generality, we assume that the set of actions available to all the agents are same, i.e., $A_1 = \ldots = A_n = A$. The payoff received by agent i during each step of interaction is determined by a utility function over the agents' *joint action*, $u_i : A^n \mapsto \Re$. Without loss of generality, we assume that the payoffs are bounded in the range [0,1]. A *repeated game* is a setting in which the agents play the same matrix game repeatedly and infinitely often. A single stage *Nash equilibrium* is a stationary strategy profile $\{\pi_i^*, \ldots, \pi_n^*\}$ such that for every agent i and for every other possible stationary strategy π_i, the following inequality holds: $E_{(\pi_1^*,\ldots,\pi_i^*,\ldots,\pi_n^*)}u_i(\cdot) \geq E_{(\pi_1^*,\ldots,\pi_i,\ldots,\pi_n^*)}u_i(\cdot)$. It is a strategy profile in which no agent has an incentive to unilaterally deviate from its own share of the strategy. A *maximin* strategy for an agent is a strategy which maximizes its own minimum payoff. Playing it repeatedly, leads to an agent getting its *security value* on every time step, in expectation.

An *adaptive* opponent strategy looks back at the most recent K joint actions played in the current *history* of play to determine its next stochastic action profile. K is referred to as the *memory size* of the opponent (minimum memory size that fully characterizes the opponent strategy). The strategy of such an opponent is then a mapping, $\pi : A^{nK} \mapsto \Delta A$. If we consider opponents whose future behavior depends on the entire history, we lose the ability to (provably) learn anything about them in a single repeated game, since we see a given history only once. The concept of memory-boundedness limits the opponent's ability to condition on history, thereby giving us a chance to learning its policy.

We now specify what we mean by playing optimally against adaptive opponents. For notational clarity, we denote the other agents as a single agent o. It has been shown previously (Chakraborty & Stone, 2008) that the dynamics of playing against such an o can be modeled as a Markov Decision Process (MDP) whose transition probability function and reward function are determined by the opponents' (joint) strategy π. As the MDP is induced by an adversary, this setting is called an *A*dversary *I*nduced *M*DP, or AIM in short.

An AIM is characterized by the K of the opponent which induces it: the AIM's state space is the set of all feasible joint action sequences of length K. By way of example, consider the game of Roshambo or rock-paper-scissors (Figure 1) and assume that o is a single agent and has $K = 1$, meaning that it acts entirely based on the immediately previous joint action. Let the current state be (R, P), meaning that on the previous step, i selected R, and o selected P. Assume that from that state, o plays actions R, P and S with probability $0.25, 0.25$, and 0.5 respectively. When i chooses to take action S in state (R, P), the probabilities of transitioning to states $(S, R), (S, P)$ and (S, S) are then $0.25, 0.25$ and 0.5 respectively. Transitions to states that have a different action for i, such as (R, R), have probability 0. The reward obtained by i when it transitions to state (S, R) is -1 and so on.

The optimal policy of the MDP associated with the AIM is the optimal policy for playing against o. A policy that achieves an expected return within ϵ of the expected return achieved by the optimal policy is called an *ϵ-optimal policy* (the corresponding return is called ϵ-optimal return). If π is known, then we can have computed the optimal policy (and hence ϵ-optimal policy) by doing dynamic programming (Sutton & Barto, 1998). However, we do not assume that π or even K are known in advance: they need to be learned in online play. By return, we mean the time averaged return of running a policy in a MDP.

Finally, it is important to note that there exist opponents in the literature which do not allow convergence to the optimal policy once a certain set of moves have been played. For example, the *grim-trigger* opponent in the well-known *Prisoner's Dilemma* (*PD*) game, an opponent with memory size 1, plays *cooperate* at first, but then plays *defect* forever once the other agent has played *defect* once. Thus, there is no way of detecting its strategy without defecting, after which it is impossible to recover to the optimal strategy of mutual cooperation. In our analysis, we constrain the class of adaptive opponents to include only those which do not negate the possibility of convergence to optimal exploitation, given any arbitrary initial sequence of exploratory moves (Powers & Shoham, 2005).

3. Model learning with Safety (*MLeS*)

In this section, we introduce a novel algorithm, *M*odel *Le*arning with *S*afety (*MLeS*), that ensures targeted optimality against adaptive opponents and safety.

3.1. Overview

MLeS begins with the hypothesis that the opponent is an adaptive opponent (denoted as o) with an unknown memory size K, that is bounded above by a known value K_{max}. *MLeS* maintains a model for each possible value of o's memory size, from $k = 0$ to K_{max}. Each model $\hat{\pi}_k$ is a mapping $(A_i \times A_o)^k \mapsto \Delta A$ representing a possible o strategy. $\hat{\pi}_k$ is the maximum likelihood distribution based on the observed actions played by o for each joint history of size k encountered. Henceforth we will refer to a joint history of size k as

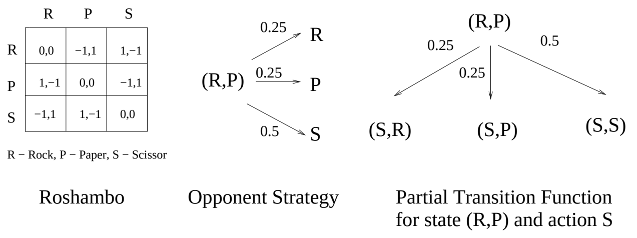
R
P
S
0,0
−1,1
1,−1
R − Rock, P − Paper, S − Scissor
Roshambo
(R,P)
0.25
0.25
0.5
Opponent Strategy
(S,R)
(S,P)
(S,S)
Partial Transition Function for state (R,P) and action S

Figure 1. Example of AIM

s_k and the empirical distribution captured by $\hat{\pi}_k$ for s_k as $\hat{\pi}_k(s_k)$. $\hat{\pi}_k(s_k, a_o)$ will denote the probability assigned to action a_o, by $\hat{\pi}_k(s_k)$. When a particular s_k is encountered and the respective o's action in the next step is observed, the empirical distribution $\hat{\pi}_k(s_k)$ is updated. Such updates happen for every $\hat{\pi}_k$, on every step. For every s_k, *MLeS* maintains a count value $v(s_k)$, which is the number of times s_k has been visited.

The operations performed by *MLeS* on each step can be summarized as follows:
1. Update all models based on the past step.
2. Determine k_{best} which is the best memory size that describes o's behavior. In order to do so, it makes a call to the FIND-K algorithm.
3. Take a step towards solving the reinforcement learning (RL) problem for the AIM induced by k_{best}.
Of these three steps, Step 2 is by far the most complex. We present how *MLeS* addresses it next.

3.2. Find-K algorithm

The objective of FIND-K is to return the best estimate of o's memory size (k_{best}) on every time step. From a high level, it does so by comparing models of increasing size to determine at which point the larger models cease to become more predictive of o's behavior.

We begin by proposing a metric called Δ_k, which is an estimate of how much models $\hat{\pi}_k$ and $\hat{\pi}_{k+1}$ differ from each other. But, first, we introduce two notations that will be instrumental in explaining the metric. We denote $(a_i, a_o) \cdot s_k$ to be a joint history of size k+1, that has s_k as its last k joint actions and (a_i, a_o) as the last k+1'th joint action. For any s_k, we define a set $Aug(s_k) = \cup_{\forall a_i, a_o \in A_i \times A_o}((a_i, a_o) \cdot s_k | v((a_i, a_o) \cdot s_k) > 0)$. In other words $Aug(s_k)$ contains all joint histories of size k+1 which have s_k as their last k joint actions and have been visited at least once. Δ_k is then defined as $max_{s_k, s_{k+1} \in Aug(s_k), a_o \in A} |\hat{\pi}_k(s_k, a_o) - \hat{\pi}_{k+1}(s_{k+1}, a_o)|$. Δ_k is thus the maximum difference in prediction of the models $\hat{\pi}_k$ and $\hat{\pi}_{k+1}$.

Based, on the concept of Δ_k, we make a couple of crucial observations that will come in handy for our theoretical claims made later in this subsection.

Observation 1. *For all $K \leq k < K_{max}$ and for any k sized joint history s_k and any $s_{k+1} \in Aug(s_k)$, $E(\hat{\pi}_k(s_k)) = E(\hat{\pi}_{k+1}(s_{k+1}))$. Hence $E(\Delta_k) = 0$.*

Let, s_K be the last K joint actions in s_k and s_{k+1}. $\hat{\pi}_k(s_k)$ and $\hat{\pi}_{k+1}(s_{k+1})$ represent draws from the same fixed distribution $\pi(s_K)$. So, their expectations will always be equal to $\pi(s_K)$. This is because o just looks at the most recent K joint actions in its history, to decide on its next step action.

Observation 2. *Once every joint history of size K has been visited at least once, $E(\Delta_{K-1}) \geq \psi > 0$, where ψ is the lower bound of the extent to which $\hat{\pi}_{K-1}$ can approximate π.*

Intuitively, Observation 2 follows from the reasoning that $\hat{\pi}_{K-1}$ cannot fully represent π without losing some information. We illustrate this with a simple example. Assume that o is of memory size 1 ($K = 1$) and let $A = \{a, b\}$, i.e., each player has just two actions, a and b. Let the probabilities assigned by o to action a for the possible 1 step joint histories (a, a), (a, b), (b, a) and (b, b) be 0.2, 0.3, 0.3 and 0.7 respectively. Now the probability assigned to action a by the 0 step model $\hat{\pi}_0(, a)$ can only be a linear combination of these values, where the coefficients come from the number of visits made to each of these joint histories. Once every 1-step joint history has been visited once, $E(\hat{\pi}_0(, a))$ can lie anywhere between 0.2 and 0.7. Since $E(\hat{\pi}_1((a, a), a)) = 0.2, \ldots, E(\hat{\pi}_1((b, b), a)) = 0.7$, it is evident that $E(\Delta_0)$ is lowest when $E(\hat{\pi}_0(, a))$ is $\frac{0.2+0.7}{2} = 0.45$. Hence $\psi = 0.25$ in this case.

High-level idea of Find-K: We denote the current values of $\hat{\pi}_k$ and Δ_k at time t, as $\hat{\pi}_k^t$ and Δ_k^t respectively. The approach taken by FIND-K (Alg. 1) can be broadly divided into two steps:

line 2: For each time step t, compute values Δ_k^t and σ_k^t, for all $0 \leq k < K_{max}$. For the time being, assume that the σ_k^t's computed always satisfy the condition:

$$\forall K \leq k < K_{max}\colon\ Pr(\Delta_k^t \geq \sigma_k^t) \leq \rho \qquad (1)$$

where ρ is a very small probability value. In other words, even without the knowledge of K, we want the difference between two consecutive models of size k and k+1 where $k \geq K$ to be less than σ_k^t with a high probability of at least $1 - \rho$. Note that although we compute a σ_k^t for every $0 \leq k < K_{max}$, the guarantee from Inequality 1 only holds for $K \leq k < K_{max}$. We will soon show how we compute the σ_k^t's.

lines 3 -12 : Then, iterate over values of k starting from 0 to K_{max} and choose the minimum k s.t. for all $k \leq k' < K_{max}$, the condition $\Delta_{k'}^t < \sigma_{k'}^t$ is satisfied. Finally return that value as k_{best}.

Next we show that eventually the k_{best} returned by FIND-K is K with a high probability. We start by providing an intuitive justification for it, we will prove it later when we specify Lemma 3.1.

We begin by showing that eventually FIND-K will reject a $k < K$ as a possible value for k_{best}, with a high probability. With more samples, Δ_{K-1}^t will tend to a positive value $\geq \psi$ with a high probability (from Observation 2). This coupled with the fact that σ_{K-1}^t assumes a value lower than ψ eventually (once the condition stated in Lemma 3.1 is met), makes $\sigma_{K-1}^t < \Delta_{K-1}^t$. This is a sufficient condition for FIND-K to keep rejecting all $k < K$ as a possible candidate for k_{best} (Steps 6-8).

Next we show that, once every $k < K$ keeps getting rejected consistently, FIND-K will select K as a possible value for k_{best} with a high probability of at least $1 - (K_{max} - K)\rho$ (the proof follows from Inequality 1 and using Union bound over all $K \leq k < K_{max}$). $k > K$ is only considered for selection once K gets rejected. The latter can only happen with a probability of at most $(K_{max} - K)\rho$ which is a very small value. Thus FIND-K will converge to selecting K as k_{best} with a high probability eventually.

We now address the final part of FIND-K that we have yet to specify: setting the σ_k^t's (Step 2).

Choosing σ_k^t: In the computation of Δ_k^t, *MLeS* chooses a specific s_k^t from the set of all possible joint histories of size k, a specific s_{k+1}^t from $Aug(s_k^t)$ and an action a_o^t, for which the models $\hat{\pi}_k^t$ and $\hat{\pi}_{k+1}^t$ differ maximally on that particular time step. So,

$$\Delta_k^t < \sigma_k^t \equiv |\hat{\pi}_k^t(s_k^t, a_o^t) - \hat{\pi}_{k+1}^t(s_{k+1}^t, a_o^t)| < \sigma_k^t \qquad (2)$$

The goal will be to select a value for σ_k^t s.t. Inequality 1 is always satisfied. Hence the rest of the derivation will focus on the range $K \leq k < K_{max}$. We can then rewrite Inequality 2 as,

$$\begin{aligned} \equiv \quad & |(\hat{\pi}_k^t(s_k^t, a_o^t) - E(\hat{\pi}_k^t(s_k^t, a_o^t)) - (\hat{\pi}_{k+1}^t(s_{k+1}^t, a_o^t) \\ & -E(\hat{\pi}_{k+1}^t(s_{k+1}^t, a_o^t))| < \sigma_k^t \qquad (3) \end{aligned}$$

The above step follows from using $E(\hat{\pi}_k^t(s_k^t, a_o^t)) = E(\hat{\pi}_{k+1}^t(s_{k+1}^t, a_o^t))$ (Observation 1). One way to satisfy Inequality 3 is to have both $|\hat{\pi}_k^t(s_k^t, a_o^t) - E(\hat{\pi}_k^t(s_k^t, a_o^t))|$ and $|\hat{\pi}_{k+1}^t(s_{k+1}^t, a_o^t) - E(\hat{\pi}_{k+1}^t(s_{k+1}^t, a_o^t))|$ be $< \frac{\sigma_k^t}{2}$. By upper bounding the probabilities of failure of the above 2 events by $\frac{\rho}{2}$ and then using Union bound, we get $Pr(\Delta_k^t < \sigma_k^t) > 1 - \rho$.

Also, we observe that the following holds :

$$Pr(|\hat{\pi}_{k+1}^t(s_{k+1}^t, a_o^t) - E(\hat{\pi}_{k+1}^t(s_{k+1}^t, a_o^t))| \geq \frac{\sigma_k^t}{2}) \leq \frac{\rho}{2} \quad (4)$$

$$\implies Pr(|\hat{\pi}_k^t(s_k^t, a_o^t) - E(\hat{\pi}_k^t(s_k^t, a_o^t))| \geq \frac{\sigma_k^t}{2}) \leq \frac{\rho}{2} \quad (5)$$

Algorithm 1: Find-K

```
   output : $k_{best}$
1  $k_{best} \leftarrow K_{max}$
2  for all $0 \leq k < K_{max}$, compute $\Delta^t_k$ and $\sigma^t_k$
3  for $0 \leq k < K_{max}$ do
4      $flag \leftarrow$ true
5      for $k \leq k' < K_{max}$ do
6          if $\Delta^t_{k'} \geq \sigma^t_{k'}$ then
7              $flag \leftarrow$ false
8              break
9      if $flag$ then
10         $k_{best} \leftarrow k$
11         break
12 return $k_{best}$
```

This can be derived by applying Hoeffding's inequality and using $v(s^t_k) \geq v(s^t_{k+1})$. $v(s^t_k) \geq v(s^t_{k+1})$ because the number of visits to a joint history s_k must be at least the number of visits to any member from $Aug(s_k)$. Intuitively, if we are confident that we have learned a bigger model to a reasonable approximation, then we are also confident with at least the same confidence that we have learned a smaller model to the same approximation. So, Inequality 4 $\implies Pr(\Delta^t_k < \sigma^t_k) > 1 - \rho$

The problem now boils down to selecting a suitable σ^t_k s.t. Inequality 4 is satisfied. By applying Hoeffding's inequality and solving for σ^t_k, we get,

$$\sigma^t_k = \sqrt{(\frac{2}{v(s^t_{k+1})} ln(\frac{4}{\rho}))} \tag{6}$$

So in general, for each $k \in [0, K_{max} - 1]$, the σ^t_k value is set as above. Note that, $v(s^t_{k+1})$ is the number of visits to the specific s^t_{k+1} chosen for the computation of Δ^t_k.

Theoretical underpinnings: Now, we state our main theoretical result regarding Find-K.

Lemma 3.1. *After all feasible joint histories of size K have been visited $\frac{8}{\psi^2} ln(\frac{4}{\rho})$ times, then with probability at least $1 - (K_{max} + 1)\rho$, the k_{best} returned by* Find-K *is K. ψ is the lower bound on the degree to which $\hat{\pi}^t_{K-1}$ can approximate π, and ρ is the small probability value from Inequality 1.*

Proof Sketch: We have already shown that (i) once the choice boils down to selecting a k_{best} from the range $[K, K_{max}]$, K is selected with a high probability of at least 1-$(K_{max} - K)\rho$. Now, we show that after all feasible joint histories of size K have been visited the number of times specified in the definition of the lemma, the probability of rejecting a $k_{best} < K$ is at least $1 - \rho$.

To reject a $k < K$, it is sufficient to have $\Delta^t_{K-1} \geq \sigma^t_{K-1}$. By using Hoeffding's inequality, we can show that $\Delta^t_{K-1} \geq E(\Delta^t_{K-1}) - \sigma^t_{K-1} \geq \psi - \sigma^t_{K-1}$, with probability of error at most ρ. Therefore $\Delta^t_{K-1} \geq \sigma^t_{K-1} \Leftarrow \sigma^t_{K-1} \leq \frac{\psi}{2} \Leftrightarrow v(s^t_K) \geq \frac{8}{\psi^2} ln(\frac{4}{\rho})$ (the last step follows from using Equation 6). Hence (ii) when all joint histories of size K are visited $\geq \frac{8}{\psi^2} ln(\frac{4}{\rho})$ times, the probability of rejecting a $k < K$ is at least $1 - \rho$. By combining (i) and (ii) we have the proof. □

The onus now lies on the action selection mechanism (Step 3 of *MLeS*) to ensure that every feasible K sized history gets visited the number of times specified in Lemma 3.1, which will enable Find-K to keep returning K consistently.

3.3. Action selection

On each time step, the action selection mechanism decides on what action to take for the ensuing time step. It picks the AIM associated with opponent memory k_{best} and takes the next step in the reinforcement learning problem of computing a near-optimal policy for that AIM. In order to solve this RL problem, we use the model based RL algorithm R-Max (Brafman & Tennenholtz, 2003). We maintain a separate instantiation of the R-Max algorithm for each of the possible K_{max}+1 AIMs pertaining to the possible memory sizes of o, i.e, $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_{K_{max}}$. On each step, based on the k_{best} returned, the R-Max instance for the AIM $\mathcal{M}_{k_{best}}$ is selected to take an action. The two conditions that ensure targeted optimality against adaptive opponents are then:

1. With probability at least $1 - \frac{\delta}{3}$, ensure that all histories of size K are visited $\frac{8}{\psi^2} ln(\frac{12(K_{max}+1)}{\delta})$ times. Once the above criterion is satisfied, Find-K keeps returning $k_{best} = K$ with probability at least $1 - \frac{\delta}{3}$ (from Lemma 3.1 and by setting $\rho = \frac{\delta}{3(K_{max}+1)}$).
2. With probability at least $1 - \frac{\delta}{3}$, allow R-Max instance of $\mathcal{M}_K$ to converge to achieving an ϵ-optimal return.

Our ultimate goal is to have our action selection mechanism implicitly satisfy both the conditions in sample complexity polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, $\frac{1}{\psi}$, $\frac{1}{\lambda}$, K_{max}, N_K, $|A|$ and T^*. N_K is the number of joint histories of size K and T^* is the mixing time for the average return (Brafman & Tennenholtz, 2003). λ is the minimum positive probability that o assigns to any action. Note, we do not have the ability to take samples at will from different histories, but may need to follow a chain of different histories to get a sample pertaining to one history. In the worst case, the chain can be the full set of all K sized histories, with each transition occurring with

λ. Hence the unavoidable dependence on $\frac{1}{\lambda}$ in sample complexity. By the sample complexity property of R-Max, condition 2 will always be satisfied in sample complexity polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, N_K, $|A|$, and T^*. Hence, all we need to ensure is that condition 1 remains satisfied in similar sample complexity with an additional dependence on $\frac{1}{\psi}$, $\frac{1}{\lambda}$, and K_{max}.

RMax requires knowledge of the mixing time (T^*) of the MDP. If T^* is unknown, then RMax has to be run in phases where in each phase the mixing time is assigned to a value T, T being incremented with each phase. Each phase is essentially a restart and it ends when RMax has run for a sufficient number of time steps (see (Brafman & Tennenholtz, 2003) for an exact value) with that value of T as T^*. The number of time steps for each phase is the number required by RMax to ensure an ϵ-optimal return for that whole phase, with a high probability, assuming T is the correct value for T^*. Since from some phase onwards T equals or exceeds T^*, the return is always ϵ-optimal from then onwards, with a high probability.

On a similar note to facilitate our theoretical claims, we run each RMax instance in phases by incrementing T. A phase for an instance ends when that instance has been selected by the action selection mechanism to decide on an action a sufficient number of times, as required by RMax. It can be shown that due to these repeated restarts, eventually in a particular phase, all feasible joint hsitories of size K do get visited a sufficient number of times allowing FIND-K to keep returning K consistently from then onwards, with a high probability. As a consequence, the RMax instance associated with $\mathcal{M}_K$ is chosen by the action selection mechanism to decide on its action from then onwards, with a high probability. This in turn leads to *MLeS* eventually achieving a near-optimal return for that AIM, with a high probability.

This brings us to our main theorem regarding *MLeS*.

Theorem 3.2. *For any arbitrary $\epsilon > 0$ and $\delta > 0$, with probability at least 1-δ, MLeS achieves at least within ϵ of the best response against any adaptive opponent, in number of time steps polynomial in $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, $\frac{1}{\psi}$, $\frac{1}{\lambda}$, K_{max}, N_K, $|A|$ and T^*.*

If there is an arbitrary opponent (neither *MLeS* nor adaptive) in the population, then *MLeS* models it by assigning it a memory size from 0 to K_{max} that best describes its behavior. Depending on the nature of the arbitrary opponent, that might turn out to be a good strategy leading to a good return for the *MLeS* agent. However, in the worst case, this may lead to a return that is less than the agent's security value due to the inadequate model. To counteract this possibility, *MLeS* always checks its return before an RMax instance restarts. If the return is more than ϵ less than its security value, it plays its maximin strategy a sufficient number of time steps following it to compensate for the loss and brings the return back to within ϵ of its security value. The number of times it requires to play its maximin strategy depends on how much less is the return from security value minus ϵ, and can be computed using Hoeffding's inequality. Thus, *MLeS* always achieves a return which is at least very close to its security value, with a high probability.

4. Convergence and Model learning with Safety (*CMLeS*)

In this section we build on *MLeS* to introduce a novel MAL algorithm for an arbitrary repeated game which achieves safety, targeted optimality, and convergence, as defined in Section 1. We call our algorithm, *C*onvergence with *M*odel *L*earning and *S*afety: (*CMLeS*). *CMLeS* begins by testing the opponents to see if they are also running *CMLeS* (self-play); when not, it uses *MLeS* as a subroutine.

4.1. Overview

CMLeS (Alg. 2) can be tuned to converge to any Nash equilibrium (NE) of the repeated game in self-play. Here, for the sake of clarity, we present a variant which converges to the single stage NE. This equilibrium also has the advantage of being the easiest of all Nash equilibria to compute and hence has historically been the preferred solution concept in multiagent learning (Bowling & Veloso, 2001; Conitzer & Sandholm, 2006).

Steps 1 - 2: Like AWESOME, we assume that all agents have access to a NE solver and they compute the same Nash equilibrium profile.
Steps 3 - 4: The algorithm maintains a null hypothesis that all agents are playing equilibrium ($AAPE$). The hypothesis is not rejected unless the algorithm is certain with probability 1 that the other agents are not playing *CMLeS*. τ keeps count of the number of times the algorithm reaches Step 4.
Steps 5 - 8 (Same as AWESOME): Whenever the algorithm reaches Step 5, it plays the equilibrium strategy for a fixed number of episodes, N_τ. It keeps a running estimate of the empirical distribution of actions played by all agents, including itself, during this run. At Step 8, if for any agent j, the empirical distribution ϕ_j^τ differs from π_j^* by at least ϵ_e^τ, $AAPE$ is set to false. The *CMLeS* agent has reason to believe that j may not be playing the same algorithm. The ϵ_e^τ and N_τ values for each τ are assigned in a similar fashion to AWESOME

(Definition 4 of (Conitzer & Sandholm, 2006)).

Steps 10 - 20: Once $AAPE$ is set to false, the algorithm goes through a series of steps in which it checks whether the other agents are adaptive with memory size at most K_{max}. The details are explained below in Theorem 4.1.

Step 21: Before the *CMLeS* agents enter a new equilibrium coordination phase, they check whether their return is more than ϵ less than their security value. If so, then they play their maximin strategy for a sufficient number of time steps to compensate for it. Akin to *MLeS*, the number of times it requires to play its maximin strategy depends on how much less is the return from security value minus ϵ, and can be computed using Hoeffding's inequality. To keep every *CMLeS* agent in sync, once a *CMLeS* agent switches to playing its maximin strategy to compensate for any loss, every other agent also does so, and waits for the process to complete. Once that is over, they go back and start a new NE coordination phase (Step 4).

Step 24: When the algorithm reaches here, it is sure (probability 1) that the other agents are not *CMLeS* agents. Hence it switches to playing *MLeS*.

Algorithm 2: *CMLeS*

```
   input : $n, \tau = 0, \tau' = 0$
1  for $\forall j \in \{1, 2, \ldots, n\}$ do
2      $\pi_j^* \leftarrow$ ComputeNashEquilibriumStrategy()
3  $AAPE \leftarrow true$
4  while $AAPE$ do
5      for $N_\tau$ rounds do
6          Play $\pi_{self}^*$
7          For each agent $j$ update $\phi_j^\tau$
8      recompute $AAPE$ using the $\phi_j^\tau$'s and $\pi_j^*$'s
9      if $AAPE$ is false then
10         if $\tau' = 0$ then
11             Play $a_o$, $K_{max}$+1 times
12         else if $\tau' = 1$ then
13             Play $a_o$, $K_{max}$ times followed by a
14             random action other than $a_o$
15         else
16             Play $a_o$, $K_{max}$+1 times
17         if any other agent plays differently then
18             $AAPE \leftarrow false$
19         else
20             $AAPE \leftarrow true$
21             If return < security value - $\epsilon$, then play
               maximin strategy enough times to compensate
22             $\tau' \leftarrow \tau' + 1$
23     $\tau \leftarrow \tau + 1$
24 Play MLeS
```

4.2. Theoretical underpinnings

CMLeS cannot distinguish between a CMLeS agent and an adaptive agent if the latter plays the computed NE strategy from the beginning, and hence may coordinate with it to converge to the NE. Note, this might not strictly be the best response against such an adaptive opponent, but we believe it to be a reasonable solution concept for such cases. Henceforth, our analysis on adaptive opponents will exclude this special case.

Theorem 4.1. *CMLeS achieves targeted optimality against adaptive opponents.*

Proof. To prove the theorem, we need to prove that for adaptive opponents of memory size at most K_{max}, *CMLeS* reaches Step 24 with some probability. We utilize the property that a K adaptive opponent is also a K_{max} adaptive opponent (see Observation 1). The first time $AAPE$ is set to false, it selects a random action a_o and then plays it K_{max}+1 times in a row. The second time when $AAPE$ is set to false, it plays a_o, K_{max} times followed by a different action. If the other agents have behaved identically in both of the above situations, then *CMLeS* knows : 1) either the rest of the agents are playing *CMLeS*, or, 2) if they are adaptive, they play stochastically for a K_{max} bounded memory where all agents play a_o. The latter observation comes in handy below. Henceforth, whenever $AAPE$ is set to false, *CMLeS* always plays a_o, K_{max}+1 times in a row. Since an adaptive opponent must be stochastic (from the above observation), at some point of time, it will play a different action on the K_{max}+1'th step with a non-zero probability. *CMLeS* then rejects the null hypothesis that all other agents are *CMLeS* agents and jumps to Step 24. □

Theorem 4.2. *In self-play, CMLeS converges to playing the Nash equilibrium of the repeated game.*

The proof follows from the corresponding proof for Awesome (Theorem 3 of Conitzer et al., 2006).

All that remains to be shown is that *CMLeS* achieves safety against arbitrary opponents. If *CMLeS* converges to playing *MLeS*, then by virtue of *MLeS*, it achieves safety. If *CMLeS* never converges to playing *MLeS*, then Step 21 ensures that it never gets exploited by an arbitrary opponent, and safety is ensured.

5. Results

We now present empirical results that supplement the theoretical claims. We focus on how efficiently *CMLeS* models adaptive opponents in comparison to existing algorithms, PCM(A) and Awesome. For *CMLeS*, we set $\epsilon = 0.1$, $\delta = 0.01$ and $K_{max} = 10$. To make the comparison fair with PCM(A), we use the same values of ϵ and δ and always include the respective opponent in the target set of PCM(A). We also add an adaptive strategy with $K = 10$ to the target set of PCM(A), so that it needs to explore joint histories of size 10.

We use the 3-player Prisoner's Dilemma (PD) game as our representative matrix game. The game is a 3 player version of the N-player PD present in GAMUT (`http://gamut.stanford.edu`).The adaptive opponent strategies we test against are :
1. Type 1: every other player plays *defect* if in the last 5 steps *CMLeS* played *defect* even once. Otherwise, they play *cooperate*. The opponents are thus deterministic adaptive strategies with $K = 5$.
2. Type 2: every other player behaves as type-1 with 0.5 probability, or else plays completely randomly. In this case, the opponents are stochastic with $K = 5$.
The total number of joint histories of size 10 in this case is 8^{10}, which makes PCM(A) highly inefficient. However, *CMLeS* quickly figures out the true K and converges to optimal behavior in tractable number of steps. Figure 2 shows our results against these two types of opponents. The Y-axis shows the payoff of each algorithm as a fraction of the optimal payoff achievable against the respective opponent. Each plot has been averaged over 30 runs to increase robustness. Against type-1 opponents (Figure 2(i)), *CMLeS* figures out the true memory size in about 2000 steps and converges to playing near optimally by 16000 episodes. Against type-2 opponents (Figure 2(ii)), it takes a little longer to converge to playing near optimally (about 30000 episodes) because in this case, the number of feasible joint histories of size 5 are much more. Both AWESOME and PCM(A) perform much worse. PCM(A) plays a random exploration strategy until it has visited every possible joint history of size K_{max}, hence it keeps getting a constant payoff during this whole exploration phase.

Due to space constraints we skip the results for convergence and safety. The convergence part of *CMLeS* uses the framework of AWESOME and the results are exactly similar to it.

6. Conclusion and Future Work

In this paper, we introduced a novel MAL algorithm, *CMLeS*, which in an arbitrary repeated game, achieves convergence, targeted-optimality against adaptive opponents, and safety. One key contribution of *CMLeS* is in the manner it handles adaptive opponents: it requires only a loose upper bound on the opponent's memory size. Second, and more importantly, *CMLeS* improves upon the state of the art algorithm, by promising targeted optimality against adaptive opponents by requiring sufficient number of visits to only

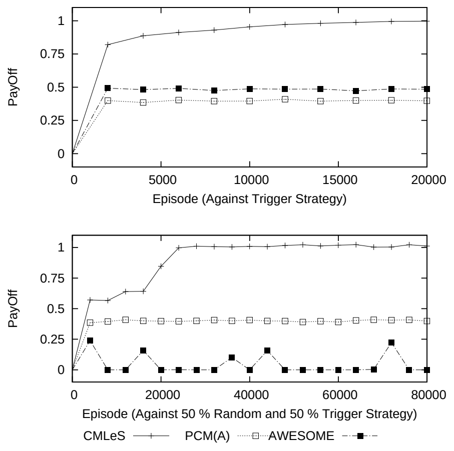

Figure 2. Against adaptive opponents

all feasible joint histories of size K, where K is the opponent's memory size. Right now, the guarantees of *CMLeS* are only in self-play or when all other agents are adaptive. Our ongoing research agenda includes improving *CMLeS* to have better performance guarantees against arbitrary mixes of agents, i.e., some adaptive, some self-play, and the rest arbitrary.

Acknowledgments: This work took place at the Learning Agents Research Group (LARG) at UT, Austin. LARG research is supported in part by grants from the National Science Foundation (IIS-0917122), ONR (N00014-09-1-0658), DARPA (FA8650-08-C-7812), and the Federal Highway Administration (DTFH61-07-H-00030).

References

Banerjee, Bikramjit and Peng, Jing. Performance bounded reinforcement learning in strategic interactions. In *AAAI*, pp. 2–7, 2004.

Bowling, Michael and Veloso, Manuela. Convergence of gradient dynamics with a variable learning rate. In *ICML*, pp. 27–34, 2001.

Brafman, Ronen I. and Tennenholtz, Moshe. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.*, pp. 213–231, 2003.

Buşoniu, L., Babuška, R., and De Schutter, B. A comprehensive survey of multi-agent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, pp. 156–172, 2008.

Chakraborty, Doran and Stone, Peter. Online multiagent learning against memory bounded adversaries. In *ECML*, pp. 211–226, 2008.

Conitzer, Vincent and Sandholm, Tuomas. Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. In *J. Mach. Learn. Res.*, pp. 23–43, 2006.

Powers, Rob and Shoham, Yoav. Learning against opponents with bounded memory. In *IJCAI*, pp. 817–822, 2005.

Powers, Rob, Shoham, Yoav, and Vu, Thuc. A general criterion and an algorithmic framework for learning in multi-agent systems. *Mach. Learn.*, pp. 45–76, 2007.

Sutton, Richard S. and Barto, Andrew G. *Reinforcement Learning*. MIT Press, 1998.