To appear in *Proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Database (E*
Antewrp, Belgium,, September 2008.

Online Multiagent Learning against Memory Bounded Adversaries

Doran Chakraborty and Peter Stone

Department of Computer Sciences
University of Texas
Austin, Texas, USA
{chakrado,pstone}@cs.utexas.edu

Abstract. The traditional agenda in Multiagent Learning (MAL) has been to develop learners that guarantee convergence to an equilibrium in self-play or that converge to playing the best response against an opponent using one of a *fixed set* of known targeted strategies. This paper introduces an algorithm called *L*earn *o*r *E*xploit for *A*dversary *I*nduced *M*arkov *D*ecision *P*rocess (*LoE-AIM*) that targets optimality against any learning opponent that can be treated as a memory bounded adversary. *LoE-AIM* makes no prior assumptions about the opponent and is tailored to optimally exploit any adversary which induces a Markov decision process in the state space of joint histories. *LoE-AIM* either explores and gathers new information about the opponent or converges to the best response to the partially learned opponent strategy in repeated play. We further extend *LoE-AIM* to account for online repeated interactions against the same adversary with plays against other adversaries interleaved in between. *LoE-AIM-repeated* stores learned knowledge about an adversary, identifies the adversary in case of repeated interaction, and reuses the stored knowledge about the behavior of the adversary to enhance learning in the current epoch of play. LoE-AIM and LoE-AIM-repeated are fully implemented, with results demonstrating their superiority over other existing MAL algorithms.

1 Introduction

The aim of many adversarial strategic interactions is to learn a model of the opponent(s) and to respond accordingly [1, 3, 14]. If the opponents execute static policies, then the learning agent is faced with a stationary environment, thus reducing the problem to effectively a single-agent decision problem. However when in the presence of other learning agents, there is an inherent non-stationarity in the environment which makes the learning problem for an individual agent much harder [12]. The most popular solution concept in such multiagent settings has been the Nash equilibrium [13] and most multiagent learning (MAL) algorithms proposed to date aim at convergence to such an equilibrium in self-play [5, 8, 15].

Their popularity notwithstanding, the ability to find Nash equilibria does not solve all multiagent problems. For one thing, there can be multiple Nash equilibria in general sum games: MAL algorithms provide no guarantee that the Nash equilibrium attained at convergence will be the one maximizing social welfare. Furthermore, an algorithm that converges to such an equilibrium in self-play may perform poorly when faced with an adversary that behaves differently.

Motivated in part by this observation, Powers and Shoham recently proposed an alternate set of evaluation criteria for MAL algorithms, focusing on *Targeted Optimality*, *Auto Compatibility* and *Safety* [14]. In their setting, the goal is to converge to within ϵ of the best response if the opponent uses one of a set of known targeted strategies, to within ϵ of a Pareto-dominant Nash equilibrium in self-play, and to within ϵ of the safety value against any unknown opponent. The authors further proposed an algorithm that meets these criteria against a set of target opponents [14, 17]. The optimal responses to the stored set of target strategies are pre-computed, such that when an opponent is recognized to be using such a strategy, the matching response can be played. While their approach is effective for a fixed set of opponents, no prior learning algorithm guarantees outcomes greater than the safety value against arbitrary opponents. This paper introduces the first algorithm capable of meeting the Powers and Shoham criteria against adversaries of a finite memory size. We show that a large class of existing algorithms are actually memory bounded and can be exploited by our approach. To the best of our knowledge, this learning algorithm is the first that targets optimality against a mixture of opponents with different properties and goals. Rather than fixing the set of target opponents, we instead focus our algorithm on any adversary that induces a Markov Decision Process (MDP) according to the *A*dversary *I*nduced *M*DP (*AIM*) model [1]. By this model, it can be shown that for a large class of opponents, the learner finds itself in an MDP whose states are determined by bounded histories of joint actions and whose transition function is determined by the opponent's strategy. Specifically, we introduce an algorithm *L*earn *o*r *E*xploit for *AIM* (*LoE-AIM*) that either explores and gathers new information about the opponent or converges to the best response to the partially learned opponent strategy.

To demonstrate *LoE-AIM*'s effectiveness, we first test it against opponents (both deterministic and stochastic) drawn from the literature of MAL research. Our results show that in most cases, *LoE-AIM* converges to playing the optimal policy against the opponent without knowing the opponent's identity.

Unfortunately it is infeasible to develop a learning algorithm that plays optimally against *every possible* memory bounded opponent of a fixed memory size without the ability to restart play (i.e. erase the history and start over), e.g., consider the following opponents in the Prisoner's Dilemma (PD) game (Table 1(a)): (1) one which always plays cooperate, (2) one which starts playing cooperate, but defects forever if the opponent ever defects once (known as "grim-trigger"). It is not possible to develop a learner which can learn to play optimally against both the opponents without having a restart. Just to differentiate between them, the learner must play defect, and once it does so, it loses the chance of attaining the (cooperate, cooperate) payoff against the grim-trigger opponent.

On the other hand, in online learning it is not uncommon to face the same *type* of adversary in multiple well-defined "epochs" of several plays, possibly with epochs against other types of adversaries interleaved in between. In such situations an effective restart is possible: each time a new opponent of the same type appears, the history starts over, but the experience from past epochs remains. Specifically, we consider the case in which the learning agent plays against multiple adversaries that it knows are drawn from the same population and therefore use the same (or similar) strategy. It plays against each individual for a finite time before playing against the next. This scenario is representative of common cases such as online auctions in which an auctioneer repeatedly sells goods to a pool of bidders. Bids in each auction are irrevocable, but the process restarts when the next good is introduced to the market. In such a setting, we propose a mechanism *LoE-AIM-repeated* that leverages such repeated interactions to learn a model of the opponent and store it in its repository of learned models. When playing a new adversary, it tries to map the model of the new adversary to one of the stored models and uses the knowledge it gathered before about the adversary to further enhance learning in the current epoch.

The remainder of this paper is organized as follows. Section 2 presents the background necessary for our work. Section 3 summarizes possible adversaries in the existing MAL literature and introduces the class of opponents targeted by *LoE-AIM*. Sections 4 and 5 introduce the *LoE-AIM* algorithm and *LoE-AIM-repeated* respectively, including results achieved against memory bounded adversaries, and Section 6 concludes.

2 Background and Definitions

In this section we introduce the definitions and concepts necessary for our work. We focus on bimatrix stage games because they are general enough to fully explore the concepts we propose and simple enough to implement, study and relate to the existing MAL literature.

Definition 1 (Bimatrix Game:). *A bimatrix game is defined by a pair of matrices* $\{M_i, M_o\}$ *where each* $M_{x|x\in\{i,o\})}$ *is of size* $|A_i| \times |A_o|$ *and* $M_x : A_i \times A_o \mapsto \Re$ *maps every possible joint-action to a reward received by agent* x*.* A_i *and* A_o *are the sets of actions available to agents* i *and* o *respectively.*

For the rest of the paper we consider agent i to be the learner under our control and agent o to be the opponent.

Definition 2 (History(h^k):). *A history* $h^k = (a_i, a_o)^k$ *where* $a_i \in A_i, a_o \in A_o$ *is the sequence of the last* k *joint actions played by the agents. In other words, a history is a vector of length* k *consisting of the past* k *joint actions played by the agents. Often* k *is referred to as the window size or length of the history.* $h^k(j)$ *is the* j*th joint-action in the sequence* h^k *where* $0 \leq j < k, k \in \mathbb{N}$ *with* $h^k(0)$ *being the most recent joint action. Similarly* $h^k_o(j)$ *is the* j*th action played by agent* o *in the sequence* h^k *with* $h^k_o(0)$ *being the most recent action played by* o*. The* *history at time t is denoted $h^{k,t}$; thus the action played by agent o, j steps before time t is denoted $h_o^{k,t}(j)$.*

For the rest of the paper, we refer to the memory size of the adversary as k.

Definition 3 (Policy (π_o):). *A policy π_o of o maps the history to a probability distribution over o's action set, i.e., $\pi_o : h^k \mapsto \Delta A_o$ where k is the memory size of agent o. The probability of playing action j following the policy $\pi_o(\cdot)$ is given by $\pi_o(\cdot)(j)$.*

Definition 4 (Memory Bounded Opponent:). *An opponent is said to be memory bounded if it follows a policy as specified above.*

Now we briefly review some definitions related to Markov Decision Processes (MDPs).

Definition 5 (Markov Decision Process (MDP) :). *An MDP M on a set of states S and with action set $A = \{a_1, \ldots, a_k, \ldots, a_{|A|}\}$ consists of*

Transition Probabilities: For each state-action pair (s, a), a next-state distribution $P_{s,a}(s')$ gives the probability of moving to state s' when action a is taken in state s.

Reward Distribution: For each state-action pair (s, a), a reward distribution $R(s, a)$ specifies the probability distribution on a set of real numbers that can be achieved as reward given action a is taken in state s.

2.1 Adversary Induced Markov Decision Process (AIM)

The key insight enabling this research is that in the setting of a repeated game where the adversary is a memory bounded opponent, the dynamics of the system can be modeled as a MDP whose transition probabilities and reward functions are determined by the model of the opponent. For a history of play (a "state") $h^{k,t}$, the next state $h^{k,t+1}$ and the reward received are determined by the current state $h^{k,t}$, the adversary's policy in that state $\pi_o(h^{k,t})$, and the action a_i chosen by agent i.

Definition 6 (Adversary Induced Markov Decision Process:). *An Adversary Induced MDP (AIM) $\mathcal{M}$ is defined as follows,*

State Space ($\mathcal{S}$) : The state space $\mathcal{S}$ of $\mathcal{M}$ is given by $\boldsymbol{a^k}$ where $a \in A_i \times A_o$, i.e, set of all possible joint histories of length k. From now onwards we will use the word state and history interchangeably.

Action Space ($\mathcal{A}$): The action space $\mathcal{A}$ of $\mathcal{M}$ is given by A_i. The action space is just the set of actions available to agent i.

Transition Probabilities ($\mathcal{P}$): Intuitively, the history is updated as a sliding window. Transitioning from a history $h^{k,t}$ to a history $h^{k,t+1}$ is just keeping the last $k-1$ joint actions (each shifted one time step backwards) and including the latest pair at index 0 of the vector. The transition probability of transitioning from a history $h^{k,t}$ to a history $h^{k,t+1}$ given the action taken being a_i is,

$$\begin{aligned}\mathcal{P}_{h^{k,t},a_i}(h^{k,t+1}) &= \pi_o(h^{k,t})(h_o^{k,t+1}(0)) \text{ where } h_i^{k,t+1}(0) = a_i. \\ &= 0 \text{ o.w}\end{aligned}$$

Note that, there is a non-zero probability to transitioning to only those histories which end in action a_i as they are the possible histories for this transition. For all other histories, the transition probability is 0. If $\pi_o(h^{k,t})$ is stochastic, then $\mathcal{P}_{h^{k,t},a_i}$ is stochastic as well. Whether the AIM is ergodic [1] depends on π_o. For example, against an opponent playing grim-trigger in PD, once learner play a defect action, it can never transition to a state where the opponent has recently played cooperate.

Reward Function ($\mathcal{R}$): The reward function $\mathcal{R}$ of $\mathcal{M}$ is given by $\mathcal{R}(h^{k,t}, a_i) = E_{a_o \sim \pi_o(h^{k,t})} M_i(a_i, a_o)$.

3 A taxonomy of possible Adversaries

The algorithms introduced in this paper target adversaries whose action at time t depend on at most the past k joint actions ($h^{k,t}$). In this section, we show that this apparently restrictive class of adversaries actually captures a large class of opponents from the literature. In order to do so, we present a taxonomy of possible adversaries, along with how several existing strategies can be classified within it. This taxonomy is summarized in Figure 1.

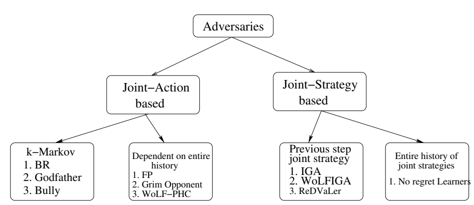


Fig. 1. A taxonomy of possible adversaries

First, an adversary can be broadly classified as either joint-action based or joint-strategy based. A joint-action based adversary bases its current action on the joint-actions played in the past. They can be further classified as k-Markov opponents whose policies depend only on the past k joint-actions, or opponents whose current action depend on the entire history of play. Examples of k-Markov opponents include Bully, Godfather [16] and Best Response (BR), while Fictitious Play (FP) [9], Joint-action learner (JAL) [7], and the family of Q-learners

[1] A MDP is ergodic if there is a non-zero probability of eventually transiting from every state to every other state (possibly via some number of intermediate states).

(e.g, PHC and WoLF-PHC [5]) depend on the entire history. A BR opponent plays the best reponse to the empirical distribution of the opponent's play captured by the current history. If memory size is unbounded, BR is equivalent to FP.

In contrast to a joint-action based adversary, a joint-strategy based adversary bases its current step strategy on the past history of joint *strategies*: not just the actual plays, but the probability distributions from which they were drawn. In practice, it is unnatural to assume that the opponent strategy is ever known. Thus in this paper, the past step opponent strategy is estimated based on the recent history. In this paper we estimate the opponent strategy by the frequency of each action played by the opponent in the captured history at that time instant. As a result, joint-strategy opponents are in effect also joint-action opponents. Nonetheless, we classify them differently since in the literature they are presented and analyzed as acting based on past joint-startegies.

Similar to the joint-action case, joint-strategy based adversaries can be further classified based on whether the current step strategy depends either on just the past step joint strategy or the entire history of joint strategies. Examples of the former are MAL algorithms which converge to a single stage Nash Equilibrium in a repeated setting (e.g. IGA [15], WoLF-IGA [4] and ReDVaLer [2]) while examples of the latter are no-regret learners which attempt to minimze the cost of online learning [10].

As our targeted opponents in this paper, we consider the k-Markov joint-action opponents and single-step joint-strategy adversaries. Though our results are against a sample of such opponents drawn from the literature, our claims hold for any opponent which induces an AIM in a joint-action space of bounded length.

4 LoE-AIM

This section introduces the *LoE-AIM* algorithm which is the heart of our overall learning mechanism. We present two versions of this algorithm, one for opponents which play deterministically (e.g. Bully, Godfather, and BR) and another for opponents who play stochastic stratgies (e.g. MAL algorithms). We start by assuming that the player[2] knows whether the opponent is playing deterministically or stochastically. In Section 5 we present a more general framework which enables the player to learn this attribute of the opponent well.

Algorithm 1 presents the version of the learning algorithm for deterministic opponents. Due to space constraints, we only present the high level algorithm and for all called methods, we give a textual explanation. The algorithm takes as input the current opponent-model ($\hat{\pi_o}$), the current start state (history) and the number of episodes for which it should continue learning. Note $\hat{\pi_o}$ refers to some partially learned model if it exists. If the algorithm has no prior information about the opponent it is playing, opponent-model is null. All the results

[2] From this point onwards we will refer to the learner as the player.

Algorithm 1: LoE-AIM-DETERMINISTIC

begin
  input : episodes, $\hat{\pi_o}$, history
  output: $\hat{\pi_o}$, π_i
  episode $\leftarrow 0$
  $\pi_i \leftarrow$ SOLVE-AIM-MODEL($\hat{\pi_o}$)
  for *episode++ < episodes* **do**
    opponent-action $\leftarrow$ action taken by opponent
    player-action $\leftarrow$ action as per π_i
    if $\{$*history, opponent-action*$\} \notin \hat{\pi_o}$ **then**
      $\hat{\pi_o} \leftarrow \hat{\pi_o} \cup \{$history, opponent-action$\}$
      $\pi_i \leftarrow$ SOLVE-AIM-MODEL($\hat{\pi_o}$)
    history $\leftarrow$ UPDATE-HISTORY(history, $\{$player-action, opponent-action$\}$)
end

presented in this section assume that there exists no such partial model and the learner learns from scratch. In Section 5 when we talk about repeated interactions with an opponent, then the $\hat{\pi_o}$ fed as input can be a partially learned model from past interaction(s) with the same opponent. The algorithm outputs the final $\hat{\pi_o}$ and the solved AIM strategy (π_i) governing the model. π_i is explained below. Since the opponent is deterministic, just one visit is needed to a state to know what the opponent's policy is for that state. The SOLVE-AIM-MODEL function finds a control policy (π_i) for the underlying AIM by assuming that for all known histories h_t of play, the opponent plays $\hat{\pi_o}$ and for all *unknown* histories, the opponent plays the maximax strategy for the player (the strategy that maximizes the maximum pay-off for the player). The assumption for unknown histories causes π_i to explore towards histories of play not visited before. The UPDATE-HISTORY method updates the history by prepending the most recent joint action and removing the oldest joint-action.

Algorithm 2 is similar to Algorithm 1 except now that opponent can play stochastic strategies. In this case, the player maintains a stochastic model of the opponent. UPADATE-OPPONENT-MODEL updates $\hat{\pi_o}$ with the latest decision taken by the opponent. Note, "updating" here means updating the percentage of times an action has been played for that state and then normalizing over all possible actions. The HAS-CHANGED-OPPONENT-MODEL? returns true if for any state the probability of taking an action is η gretear than that of the same action in the previous solved model and the number of visits to that state is at least κ. All results in this paper use values for η and κ that led to the best results in informal preliminary testing, namely $\eta = 0.1$ and $\kappa = 20$.

Lemma 1. *In repeated infinite play LoE-AIM either converges to the optimal policy for the partially learned opponent model or keeps expanding the learned model.*

Algorithm 2: LoE-AIM-STOCHASTIC

begin
  input : episodes, $\hat{\pi_o}$, history
  output: $\hat{\pi_o}$,π_i
  episode $\leftarrow$ 0
  $\pi_i \leftarrow$ SOLVE-AIM-MODEL($\hat{\pi_o}$)
  for *episode++ < episodes* **do**
    opponent-action $\leftarrow$ action taken by opponent
    player-action $\leftarrow$ action as per π_i
    $\hat{\pi_o} \leftarrow$ UPADATE-OPPONENT-MODEL($\hat{\pi_o}$, {history, opponent-action})
    if HAS-CHANGED-OPPONENT-MODEL?*($\hat{\pi_o}$)* **then**
      $\pi_i \leftarrow$ SOLVE-AIM-MODEL($\hat{\pi_o}$)
      history $\leftarrow$ UPDATE-HISTORY(history, {player-action, opponent-action})
end

Proof. Let $\bar{\pi_o}$ be the remainder of π_o that needs to be learnt at a particular time instant. $\hat{\pi_o}$ refers to the part of the opponent strategy that the player knows while $\bar{\pi_o}$ being the part that still needs to be explored. By solving for a control policy for $\hat{\pi_o}$ where for every state in $\bar{\pi_o}$ the player believes that it could get the best possible reward (since it assumes that the opponent playes the maximax strategy for the player at those states and the value of maximum possible achievable reward is known), the algorithm generates π_i that will always promote exploring states in $\bar{\pi_o}$. However if $\hat{\pi_o}$ is non-ergodic, then there are chances that the current state may prohibit transition to newer states, i.e, the strategy of the opponent is such that it prevents further expanding of the model. Then the algorithm converges to the optimal policy given the partially learned model.

Corollary 1. *If π_o is ergodic, then LoE-AIM converges to the optimal policy in infinite repeated play.*

Note that this exploratory aspect of *LoE-AIM* is motivated in part by the R-Max algorithm [6] which also deliberately balances exploitation with exploration of unvisited states. The main difference is that R-Max is designed for single agent MDP's and hence the exploration depends only on the action of the agent, whereas in AIMs, the agent and its adversary jointly determine the state space explored.

4.1 Results against deterministic opponents

This section presents the results achieved by *LoE-AIM* against the deterministic opponents mentioned in Section 3, namely k-Markov adversaries such as Godfather, BR and Bully.

Figure 2 shows the results achieved by *LoE-AIM* in the game of Prisoner's Dilemma (PD) (Table 1(a)) against a couple of variations of the Godfather, Bully [16] and BR strategies.

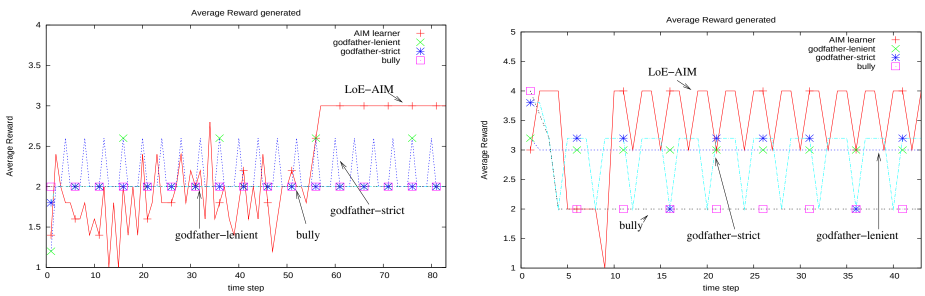
(a) Prisoner's Dilemma

	cooperate	defect
cooperate	(3,3)	(1,4)
defect	(4,1)	(2,2)

(b) Chicken

	$A1$	$A2$
$A1$	(3,3)	(2,4)
$A2$	(4,2)	(1,1)

Table 1. Payoff matrices.



(a) Against Godfather-strict (b) Against Godfather-lenient

Fig. 2. Results against Godfather opponents in PD.

Godfather is a finite-state strategy that makes its opponent an offer that it cannot refuse. Godfather chooses a targetable pair [3]. From then on, if the opponent keeps playing its half of targetable pair in one stage, Godfather plays its half in the next stage. Otherwise it plays a strategy (threat) that forces the opponent to achieve at most its safety value. Hence Godfather is a memory-bounded adversary with $k = 1$. We now introduce a couple of variations of the Godfather strategy that are tailored for $k > 1$.

– Godfather-lenient plays its part of a targetable pair if the opponent at least once played its own half of the pair (within the last k actions). Otherwise Godfather-lenient punishes its opponent by playing the threat strategy that reduces the opponent's best outcome to its safety value.
– Godfather-strict is a stricter version that always punishes its opponent if the opponent ever deviated from the targetable pair during the observable history.

Note that in case of PD, the Godfather players target the {cooperate, cooperate} pair and use defect as the threat strategy.

Bully is a deterministic strategy given by $\text{argmax}_{a_o \in A_o} M_o(a_i^*, a_o)$ where $a_i^* = \text{argmax}_{a_i \in A_i} M_i(a_i, a_o)$. The opponent optimizes its payoff by assuming that the Bully remains fixed while the Bully optimzes its payoff by assumimg that the opponent is the follower and would adapt accordingly.

[3] A pair of deterministic policies is a targetable pair if playing them results in each player getting more than the safety value and plays its half of the pair

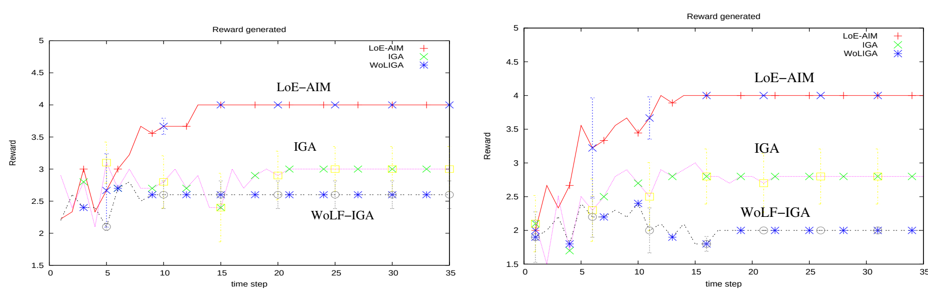
Now we present results which show that *LoE-AIM* exploits all of the above opponents without knowing their identity. For benchmarking purposes we also present results had the player chosen any of the deterministic strategies as its strategy instead of *LoE-AIM*. The results presented in Figure 2 are for $k = 3$ and averaged over 10 random instantiations of the start state (e.g. the assumed "history" of the opponent when it makes its first decision). However, each run is independent and the learner starts learning from scratch with each restart. In the spirit of online learning, *LoE-AIM* converges to the optimal policy in each of the occasions without requiring a restart. Against Godfather-strict, the *LoE-AIM* algorithm eventually learns (after about 55 episodes of learning) that it should play cooperate (its half of the targetable pair) and hence converges to a payoff of 3 (Figure 2(a)). The results show that none except the Godfather-strict [4] strategy converge to the optimal payoff. For Godfather-lenient, *LoE-AIM* learns to optimally exploit by playing cooperate frequently enough so that the history always contains one cooperate action for the player. At convergence, the *LoE-AIM* player plays defect twice followed by a cooperate ensuring two consecutive payoffs of 4 followed by a payoff of 3 (Figure 2(b) shows that the average converged payoff (after about 10 episodes) oscillates between 3 and 4). In case of PD, both the Bully strategy and BR strategy is to play defect deterministically. Against both of these opponents, the learner eventually learns to play defect and converges to a payoff of 2 (for space constraints, we omit the graphs).

4.2 Results against stochastic opponents

We now present results of *LoE-AIM* learning against popular MAL algorithms that converge to single-stage Nash equilibrium in repeated play. Due to space constraints we only present results against IGA [15] and WoLF-IGA [3], but the algorithm also works against all other MAL algorithms that decide their next step strategy based on the past step joint-strategy (e.g. ReDVaLer [2]). We assume that the opponent cannot observe the player's past step strategy and hence approximates it by the proportion of each action played by the player in the current state (history). A point to note is that the opponent knows its own strategy for sure and uses it to compute its next step strategy. This makes the process non-Markovian in the space of the k-history. However if k is large enough, the proportion of each action played by the opponent will be close to its real strategy and hence will make the process approximately Markovian. Though it seems that larger the value of k the better, our results show that even for $k = 4$, *LoE-AIM* can efficiently model the opponent and exploit it to the optimum. Once again all our results are averaged over 10 different instantiations of the start state and learning at each restart starts from scratch. The learning rate used for IGA is 0.1 and the learning rates for WoLF-IGA are 0.1 and 0.2. Figure 3 gives evidence that the *LoE-AIM* learner was successful in reaching its optimal payoff in the game of chicken(Table 1(b)) by exploiting the MAL opponents on both the occasions. The reason we choose Chicken game is because the game

[4] Note, Godfather-strict strategy in self play always converge to the targetable pair.

has three Nash equilibria: two in pure strategies, sustaining the outcomes (4,2) and (2,4), and one in mixed strategies where the players play each of their actions with equal probability with the corresponding expected payoff of 2.5 for each agent. Neither IGA, nor WoLF-IGA guarantees the possible final converged Nash pay-off in self-play, e.g, in both Figure 3(a) and Figure 3(b), self-play generates outcomes much less than 4 showing that on numerous occasions the final converged Nash payoff was not (4,2), the one most coveted by the player. In contrast, in all of its runs, *LoE-AIM* converged to the outcome (4,2).



(a) Against IGA (b) Against WoLF-IGA

Fig. 3. Results against MAL opponents in Chicken game

Figure 5 gives a summary of a head to head comparison among the various opponents discussed in this section together with results achieved by *LoE-AIM*. There are 78 structurally distinct 2×2 strict ordinal games in which the two players can strictly rank the four payoffs from best to worst. Of the 78 games, only 6 games (shown in Figure 4) have multiple Nash equilibria with each player favoring a different one. We present results from these games because in self-play none of the MAL algorithms guarantee the final converged Nash pay-off (the algorithms can converge to any one of the Nash equilibria depending on the learning rates and start states). Each point in the plot has been averaged over results achieved from all the 6 games, with the results in each game first averaged over 10 runs with different initial start states. For benchmark comparisons, we show head to head results achieved by various other algorithms that the player could have used as its default strategy instead of *LoE-AIM*. Figure 5 shows that against the MAL algorithms (IGA, WoLF-IGA) and BR, *LoE-AIM* successfully converged to its best outcome of 4 on all occasions thereby demonstrating its ability to exploit its opponent to the optimum. All the benchmarks generate lower average payoffs when played against these opponents. Against the other opponents, *LoE-AIM* still did better than all other benchmarks though the average outcome was lower than 4 in these cases. Note, that against certain opponents it is never possible to achieve the 4 outcome because the opponent won't allow that, e.g, against Godfather-strict in PD (see Figure 2(a)).

	A1	A2
A1	(3,4)	(2,1)
A2	(1,2)	(4,3)

((1,1) (0,0) (0.25,0.5))

	A1	A2
A1	(2,4)	(3,1)
A2	(1,2)	(4,3)

((0,0), (1,1) (0.25,0.5))

	A1	A2
A1	(2,3)	(3,4)
A2	(4,2)	(1,1)

((1,0) (0,1) (0.5,0.5))

	A1	A2
A1	(2,2)	(4,3)
A2	(3,4)	(1,1)

((1,0) (0,1) (0.75,0.75))

	A1	A2
A1	(3,3)	(2,4)
A2	(4,2)	(1,1)

((1,0) (0,1) (0.5,0.5))

	A1	A2
A1	(2,2)	(3,4)
A2	(4,3)	(1,1)

((1,0) (0,1) (0.5,0.5))

Fig. 4. Payoff matrices of 6 games with multiple Nash Equilibria. ((1,1) (0,0) (0.25,0.5)) means that the game has 3 Nash equilibria where the probabilities of playing action A1 by both players are respectively (1,1), (0,0) and (0.25,0.5).)

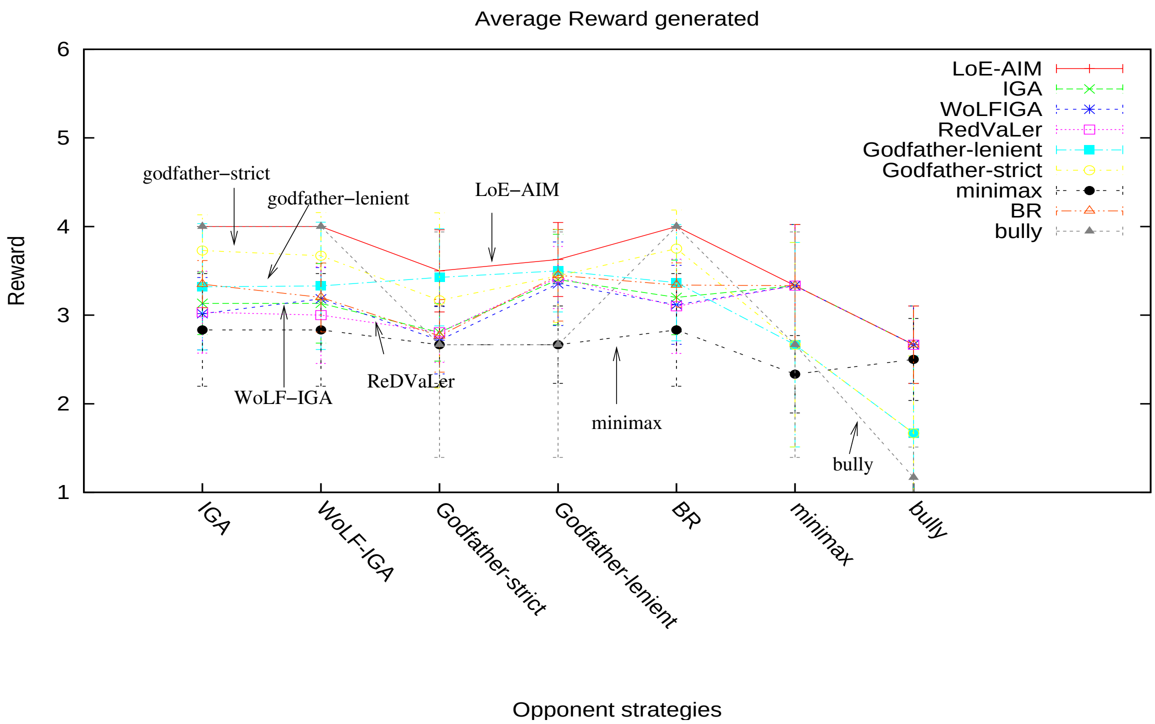


Fig. 5. Result against the 6 games with multiple Nash equilibria

5 LoE-AIM-repeated

In online learning, repeated interaction with multiple opponents is quite common. For example, the player may plays opponent 1 for 10 rounds, opponent 2 for the next 10 rounds, and then again opponent 1 for another 10 rounds. One such scenario is a market with multiple sellers where the buyer is interested in learning an optimal negotiation strategy for buying items. The buyer negotiates in turn with different sellers and learns from these experiences.

Figure 6 presents *LoE-AIM-repeated*, which such a buyer can employ to maximize her return. We assume that the buyer has a fixed set of interactions

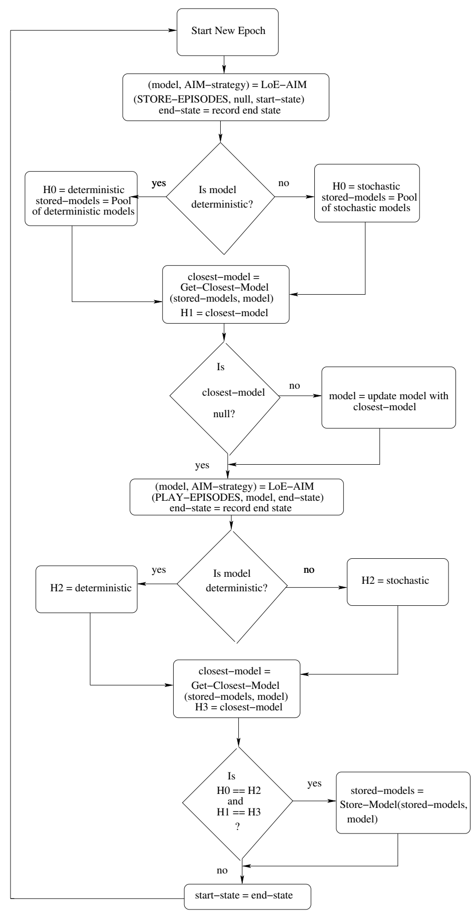
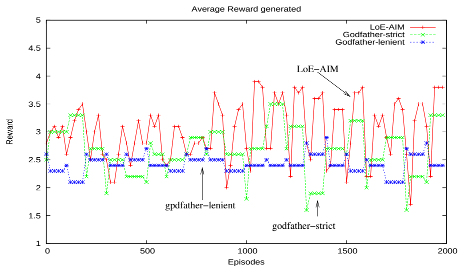


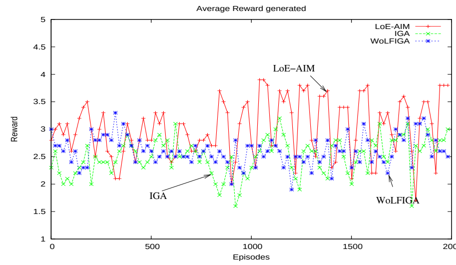
Fig. 6. *LoE-AIM-repeated*

(episodes) with an opponent in one run (EPOCH). For the first STORE-EPISODES number of plays, the buyer tries to build an approximate model of the opponent. The *LoE-AIM* method called with opponent-model set to null, outputs an approximate model and an AIM-strategy for that model. In these STORE-EPISODES number of plays, if ever the opponent took different actions for the same state, H_0 is set as stochastic, else H_0 is set as deterministic. Once the model has been built, the framework searches for the closest-model in the pool of stored models. The method Get-Closest-Model takes as arguments the pool of stored models and model, and returns the closest model that matches the model. In the deterministic case, closest-model is computed by iterating over all the stored deterministic models and returning the one which has the maximum number of states such that the same decision would be taken. In the stochastic case, closest-model is selected by iterating over all the stored stochastic models and returning the one which has the minimum Max Norm distance from model. If a convincing closest-model exists (if the distance is smaller than a fixed threshold for the stochastic case), the model is updated with the closest-model. The player then calls the *LoE-AIM* method with the updated opponent model and runs it for PLAY-EPISODES. Next, the framework recomputes the closest-model (H_3) based on the newly updated model returned by the earlier call to the *LoE-AIM* method. In these PLAY-EPISODES number of plays, if ever the opponent took different actions for the same state, H_2 is set as stochastic, else H_2 is set as deterministic. Finally the framework makes a conservative check to see whether the assumptions it made after the first STORE-EPISODES number of plays also hold after the next PLAY-EPISODES number of plays. If the assumptions hold, it stores (replaces, if it updated a former stored model) the newly generated model in the pool. The whole process repeats with every new EPOCH of play.

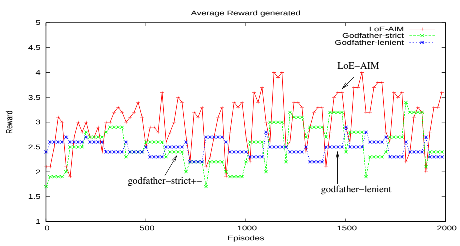
For experimental evalauation of *LoE-AIM-repeated*, we restrict the set of opponents to the two versions of the Godfather together with IGA and WoLF-IGA. The opponents we choose give a fair representative mix of the targeted class that *LoE-AIM-repeated* is designed to exploit. Figure 7 provides a comparative picture of the results achieved by the *LoE-AIM-repeated* in the game of PD and Chicken respectively. As base case results, we also provide the results achieved by each of the opponent approaches had they been the approach employed by the player. We break the results in each game in two individual plots for clarity of expression, one comparing the performance of *LoE-AIM-repeated* with the deterministic opponents (two versions of Godfather) and the other comparing the same with the MAL opponents (IGA nd WoLF-IGA). We tested our approach for different values of STORE-EPISODES and PLAY-EPISODES, and finally decided to fix them at 20 and 80 respectively. As part of our future work, we would like to have a theoritical bound on the number of episodes we need to explore (STORE-EPISODES) to get a reasonable approximation of the model. The simulation has been run for 20 EPOCHs thereby resulting in a run of 2000 episodes in total. After every EPOCH a new opponent is chosen randomly. The results have been averaged over 10 instantiations of the start state. In both the plots, *LoE-AIM* does better than the benchmark opponents. An interesting thing to note is that the *LoE-AIM* learning plot has spikes after every 100 episodes. After every 100 episodes, the learner explores for 20 episodes to build an approximate model of the new opponent. But once it builds the model, it matches it with a stored model and starts using the knowledge it learned from past interactions with the opponent.



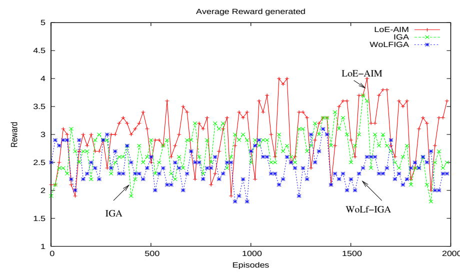
(a) Comparison with the two version of Godfather in PD



(b) Comparison with IGA and WoLF-IGA in PD



(c) Comparison with the two version of Godfather in Chicken



(d) Comparison with IGA and WoLF-IGA in Chicken

Fig. 7. *LoE-AIM-repeated* Results

6 Conclusion

In this paper we introduced a general mechanism for learning against memory-bounded adversaries. Our algorithm *LoE-AIM* either explores to gather new information about the opponent or converges to the best response to the partially learned opponent strategy. We showed detailed results in the games of PD and Chicken and further backed our claims with results averaged over 6 games with ordinal payoffs and multiple Nash equilibria and each player favoring a different Nash equilibrium. Our results show that *LoE-AIM* generates higher average rewards than existing MAL approaches against the same set of opponents. We then introduced a mechanism that enables online learning based on epochs of play against similar opponents by mining of learned knowledge about the opponent and using it to seed learning when faced against the same opponent in future interactions.

This research suggests several possible directions for future work. First, the algorithms presented are limited to targeting opponents with bounded memory. It would be natural to try to extend the results to opponents that fall in other parts of the taxonomy shown in Figure 1. For example, it would be interesting to see how *LoE-AIM* can be generalized to account for opponents whose next step strategy depends on the entire history of play (not just k-Markov as assumed in this paper). An important challenge in that direction would be to devise a compact finite state representation that captures the history of play.

References

1. B. Banerjee and J. Peng. Efficient learning of multi-step best response. In *AAMAS '05: Proceedings of the fourth international joint conference on Autonomous agents and multiagent systems*, pages 60–66, New York, NY, USA, 2005. ACM Press.
2. B. Banerjee and J. Peng. Rvσ(t): a unifying approach to performance and convergence in online multiagent learning. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 798–800, New York, NY, USA, 2006. ACM Press.
3. M. Bowling. Convergence and no-regret in multiagent learning. In *Neural Information Processing Systems 17*. MIT Press, 2005.
4. M. Bowling and M. Veloso. Convergence of gradient dynamics with a variable learning rate. In *Proc. 18th International Conf. on Machine Learning*, pages 27–34. Morgan Kaufmann, San Francisco, CA, 2001.
5. M. H. Bowling and M. M. Veloso. Rational and convergent learning in stochastic games. In *IJCAI*, pages 1021–1026, 2001.
6. R. I. Brafman and M. Tennenholtz. R-max - a general polynomial time algorithm for near-optimal reinforcement learning. *J. Mach. Learn. Res.*, 3:213–231, 2003.
7. C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI/IAAI*, pages 746–752, 1998.
8. V. Conitzer and T. Sandholm. Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. pages 83–90, 2003.
9. F. D. and L. D. K. In *The theory of learning in games*. MIT Press, 1999.
10. A. Greenwald, A. Jafari, G. Ercal, and D. Gondek. On no-regret learning, fictitious play, and nash equilibrium.
11. M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning (ML-94)*, pages 157–163, New Brunswick, NJ, 1994. Morgan Kaufmann.
12. J. F. Nash, Jr. Equilibrium points in n-person games. In *Classics in game theory*, 1997.
13. R. Powers and Y. Shoham. Learning against opponents with bounded memory. In *IJCAI*, pages 817–822, 2005.
14. S. Singh, M. Kearns, and Y. Mansour. Nash convergence of gradient dynamics in general-sum games. pages 541–548.
15. P. Stone and M. L. Littman. Implicit negotiation in repeated games. In J.-J. Meyer and M. Tambe, editors, *Pre-proceedings of the Eighth International Workshop on Agent Theories, Architectures, and Languages (ATAL-2001)*, pages 96–105, 2001.
16. T. Vu, R. Powers, and Y. Shoham. Learning against multiple opponents. In *AAMAS '06: Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*, pages 752–759, New York, NY, USA, 2006. ACM Press.