Fast Adaptive Learning in Repeated Stochastic Games by Game Abstraction

Mohamed Elidrisi, Nicholas Johnson, Maria Gini
Department of Computer Science and Eng.
University of Minnesota
Minneapolis, MN 55455
{elidrisi,njohnson,gini}@cs.umn.edu

Jacob Crandall
Department of Electrical Engineering and Computer Science
Masdar Institute of Science and Technology
Abu Dhabi, United Arab Emirates
jcrandall@masdar.ac.ae

ABSTRACT

An agent must learn and adapt quickly when playing against other agents. This process is challenging in particular when playing in stochastic environments against other learning agents. In this paper, we introduce a fast and adaptive learning algorithm for repeated stochastic games (FAL-SG). FAL-SG utilizes lossy game abstraction to reduce the state space of the game and facilitate learning and adapting rapidly. We analyze FAL-SG's performance by proving bounds on the abstraction loss and prediction mistakes and show that FAL-SG satisfies three criteria prescribed for multiagent learning algorithms. We successfully establish the robustness and scalability of FAL-SG with extensive theoretical and experimental results.

Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Intelligent agents*

Keywords

Multiagent Learning, Game Theory, Adversarial Learning

1. INTRODUCTION

There is a growing need for multiagent learning algorithms that operate in complex environments with other learning algorithms. Many multiagent learning algorithms [1, 5, 23] have been developed for normal-form games, however fewer algorithms [3, 10, 11] have been developed for complex environments (e.g., stochastic games [26]). The difficulty lies in having to not only learn a model of the environment but also learn a model of the opponents. This is even more difficult when the opponents are also learning. An agent that is able to learn and adapt quickly to changes in opponent behavior will have an advantage [13].

The need for agents to learn in multiagent settings has previously been explored from many different perspectives through adaptation of single agent algorithms (e.g., Minimax-Q [16] and Nash-Q [17]), direct policy search [3], and through multiagent criteria [8, 24, 7]. Additionally, agents have been developed with the focus on performing well in environments such as stochastic games (WoLF-PHC [3] and M-Qubed [10]). Crandall recently developed a framework (Pepper [9]) that utilizes a matrix based algorithm to play in repeated stochastic games. However, there has yet to be a focus on the need to learn and adapt quickly in a limited number of interactions in stochastic games.

It is difficult to perform well in stochastic games due to the large state space. State abstractions have been effectively used in reinforcement learning to deal with large states spaces and changes in the environment [20]. Game abstraction has also been used to find effective strategies in large games such as Poker [15, 27]. However, most abstraction methods are game specific. Sandholm *et al.* [25] have recently motivated the need for such general lossy abstraction and set forth a general method for stochastic games. FAL-SG utilizes a general game abstraction method which allows it to focus on learning fast and adapting to opponent changes in any general stochastic game.

Fast adaptive learning is grounded in real-world applications where an agent only has a limited number of interactions with other agents. Many multiagent learning algorithms focus on converging to an equilibrium strategy [3, 8]. However, this requires a large number of interactions, which is a luxury not available in most real-world domains (e.g., auction bidding, search and rescue operations, and stock market investing).

In this paper, we introduce a multiagent learning algorithm, *Fast Adaptive Learning in Stochastic Games (FAL-SG)*, that is able to deal with large state spaces and adaptive environments. The key behind the algorithm design consists of three models: (1) *Meta-game model* to abstract the stochastic game into a lossy matrix game representation, (2) *Prediction model* to predict the opponents' next action, (3) *Reasoning model* to reasons about the next action to play given the abstracted game and the predicted opponents' actions.

We first describe the models and present FAL-SG (Section 2). We then analyze its performance by providing loss bounds on the meta-game model and prediction model and show how FAL-SG satisfies three criterion prescribed for multiagent learning algorithms (Section 3). Finally, we empirically demonstrate the effectiveness of FAL-SG through experiments in two stochastic games (Section 4). The theoretical and experimental results show how FAL-SG outperforms existing multiagent learning algorithms while providing performance guarantees.

Appears in: *Alessio Lomuscio, Paul Scerri, Ana Bazzan, and Michael Huhns (eds.), Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014), May 5-9, 2014, Paris, France.*

2. FAST ADAPTIVE LEARNING

We now present our algorithm for general-sum stochastic games that quickly learns and adapts to changes in the opponents' strategy. We call this algorithm **FAL-SG** (Fast Adaptive Learning in Stochastic Games). The algorithm is built around three models:

1. **Meta-game model**: defines an abstract game based on key target states in the stochastic game by observing the actions and rewards of all players.
2. **Predictive model**: predicts the opponents' action.
3. **Reasoning model**: chooses a suitable strategy given the meta-game and the opponents' predicted action.

We consider stochastic games that are defined as a tuple $(N, \mathcal{A}_{1\ldots n}, \mathcal{S}, \mathcal{T}, \mathcal{R}_{1\ldots n})$, where $N := \{1, \ldots, n\}$ is the set of players, $\mathcal{A}_i$ is the set of actions available to player i where $\mathcal{A} = \bigotimes_{i=1}^{n} \mathcal{A}_i$ is the joint action space, $\mathcal{S} := \{s_1, \ldots, s_D\}$ is the set of states, $\mathcal{T} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ is the transition function, and $\mathcal{R}_i : \mathcal{A} \times \mathcal{S} \to \mathbb{R}$ is the reward function for player i. We assume that the opponents' actions and rewards are observable.

2.1 Meta-game Model

There have been few algorithms that use abstraction in general stochastic games [25]. FAL-SG utilizes the meta-game model to effectively use state abstraction and to reduce the state space of a general stochastic game.

2.1.1 Framework

For a stochastic game G and the abstracted game $\widetilde{G}$ we can view $\widetilde{G}$ as a layered directed acyclic graph (LDAG) where the root node s_1 is the start state of the original game, the reachable states are children, and each leaf node represents a terminal state, as in Figure 1.

Let $\tau := \{s_1, \ldots, s_d\}$ be a path (i.e., set of states visited while traversing the graph from the root node to a leaf node), $\upsilon := \{a_1, \ldots, a_d\}$ be the corresponding actions, and u be the average of the rewards received by visiting each $s \in \tau$. We form a tuple $(\tau, \upsilon, u)_t$ of the path, actions, and average reward for a single iteration t of the game. Let $\Phi_i := \{(\tau, \upsilon, u)_1, \ldots, (\tau, \upsilon, u)_T\}$ be a set of such tuples for player i. Finally, we define $\widetilde{\Omega}^K$ to be a set that contains K clusters of paths. The objective of the meta-game model is to reduce the number of states in G so we can efficiently compute reasonable strategies.

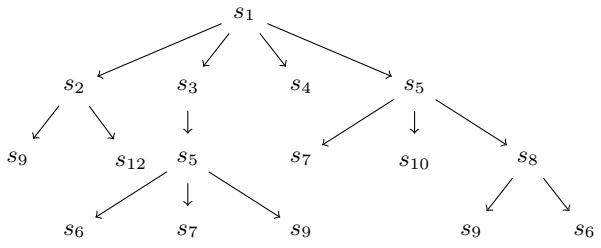

Figure 1: Example of a LDAG of states.

2.1.2 Method

The meta-game model performs the following three steps as outlined in Algorithm 1 for a 2-player game.

Algorithm 1 FAL-SG: `Meta-game`(Φ_{me}, Φ_{opp})

Parameters:
  Φ_{me}, Φ_{opp}: set of tuples (τ, υ, u) for both players.
  $\widetilde{\Omega}^K$: set of K clusters. (Initially $K = 1$)
Initialize:
  $\Phi_{me} \leftarrow (\tau_{me}, \upsilon_{me}, u_{me})_1$, $\Phi_{opp} \leftarrow (\tau_{opp}, \upsilon_{opp}, u_{opp})_1$
  $\widetilde{\Omega}^K \leftarrow$ `PathCluster`(Φ_{me}, Φ_{opp})
  $\widetilde{\mathbf{G}} \leftarrow \widetilde{\Omega}^K$
while $t <$ end of game **do**
  Update Φ_{me}, Φ_{opp} using $V_\pi(s)$ and observed tuples.
  Update $\widetilde{\Omega}^K \leftarrow$ `PathCluster`(Φ_{me}, Φ_{opp})
  Update $\widetilde{\mathbf{G}}$

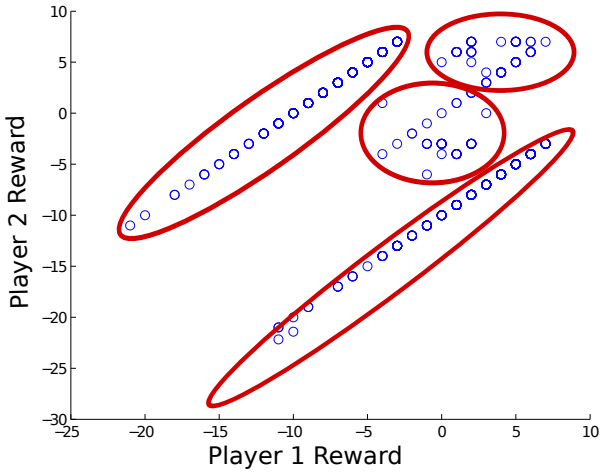

Figure 2: Example of path clustering by rewards.

Discover paths: Initially, Φ_i is empty, so Q-Learning is used at the beginning of the game as a method for focused exploration to discover possible paths from the start state to terminal states. Each path, actions, average reward tuple (τ, υ, u) computed during exploration for player i is added to Φ_i.

Cluster paths: The paths are clustered using the `EM` algorithm which fits a mixture of Gaussian distributions over the observed rewards for all players and assigns a path with the corresponding reward to a cluster $\omega \in \widetilde{\Omega}^K$ following [12].

The number of clusters K is determined by building 1 to $\texttt{MAX}_K$ models. The model with the smallest Bayesian Information Criterion (BIC) [14] is chosen. This process is outlined in Algorithm 2.

We define the value function V for a strategy π in the game G by observing the reward obtained from the joint action of all agents

$$V^\pi = \sum_{i=1}^{m} \pi(a, s)[R(s, a, a') + \sum_{s'} T(s, a, a', s')V^\pi]. \qquad (1)$$

The value function W in the abstract game $\widetilde{G}$ is defined similarly. Note that τ in G represents a strategy in $\widetilde{G}$. Therefore, the expected valuation of cluster ω_i is estimated in an optimistic manner, where we assume the cluster will get the reward of the best path $\tau \in \omega_i$ assigning it the max-

Algorithm 2 FAL-SG: `PathCluster`(Φ_{me}, Φ_{opp})

Parameters:
  Φ_{me}, Φ_{opp}: set of tuples (τ, v, u) for both players.
  $\texttt{MAX}_K$: upper bound on model size.
  $model_{opt}$: optimal mixture model.
for $i \leq \texttt{MAX}_K$ **do**
  $models[i] \leftarrow \texttt{EM}(u_{me}, u_{opp}, i)$
  Compute BIC and store in vector element $BIC[i]$.
$i_{opt} \leftarrow \arg\min_i BIC$
$model_{opt} \leftarrow models[i_{opt}]$
`Return` $model_{opt}$

imum valuation of all paths in the cluster

$$\mathbb{E}(\omega_k) = \max_{\forall \pi \in \omega_k} V^{\pi}(s). \quad (2)$$

Generate meta-game matrix: The final process of the abstraction is to create and update the meta-game matrix as the game is played. There are two parts to be updated: (1) the number of rows (actions) given the updated set of clusters Ω^K, and (2) the estimate of the future reward for cluster ω_k (i.e., row k).

We adopt a mechanism for estimating the future reward from an existing framework called Pepper [9]. In Pepper, future rewards $V^{\pi}(s)$ are estimated with two properties. First, the *realism property*, which states that $V^{\pi}(s)$ should reflect the actual reward received by the agent in an iteration after stage s_i is reached. Second, the *optimism property* which overestimates the value of $V^{\pi}(s)$ in order to ensure that agents avoid premature convergence.

2.1.3 Stochastic Prisoner's Dilemma

To illustrate the abstraction, we walk through a 2-player example using a stochastic version of Prisoner's Dilemma [9].

The game, shown in Figure 3, consists of two players A and B that each start at the bottom opposite corners of the grid. The goal of the game is to enter one of the four terminal gate states labeled 1, 2, 3, and 4 in the smallest number of moves possible. If both players try to enter gate 1 at the same time, gate 1 and 2 become closed and both players have no choice but to enter gate 3 or 4. If one of the players reaches gate 1 before the other then gates 1, 2, and 3 will be closed and the other player will have no choice but to enter gate 4. If both players try to enter gates 2, 3, or 4 at the same time they will both succeed.

The possible moves for each player are *Up, Down,* and *Toward Gate*, where *Toward Gate* is right for player A and left for player B. An attempt to move toward a closed gate, wall (black space), or outside the grid will lead to no transition. A player receives a payoff of -1 for each state it visits except for terminal gate states where the reward is +10. A single iteration of the game ends when both players reach a terminal gate.

For example, a path τ_j for player A represents a set of visited states starting from the bottom left corner and ending in a terminal state. Figure 3 illustrates such paths with the lines labeled τ_1, τ_2, τ_3, and τ_4.

Assume that we observe the game and add the four paths and following average rewards to Φ_i for player i: $u_1 = 1.2$, $u_2 = 3.4$, $u_3 = 6.2$, $u_4 = 5.6$. Clustering is performed resulting in $\omega_1 = \{(\tau_1, v_1, u_1) : 1.2\}$, $\omega_2 = \{(\tau_2, v_2, u_2) : 3.4\}$, and $\omega_3 = \{(\tau_3, v_3, u_3), (\tau_4, v_4, u_4) : 5.9\}$. The initial meta-game matrix is produced as shown in Table 1 where player A is the row player and player B is the column player. The rewards for player A are initialized to be that of a cluster as shown in Eq. 2 for the on diagonal cells and the average of all clusters valuation estimates for the off diagonal cells.

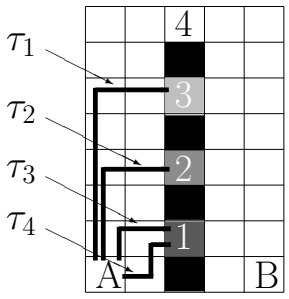

Figure 3: A stochastic Prisoner's Dilemma showing sample paths from start to different terminal state.

	ω_1	ω_2	ω_3
ω_1	1.2, 1.3	2.3, 2.6	3.6, 3.4
ω_2	2.6, 2.7	3.4, 3.9	4.7, 4.7
ω_3	3.6, 3.4	4.7, 4.7	5.9, 5.5

	ω_1	ω_2	ω_3	ω_4
ω_1	3, 3	7, 1	7, 1	7, 1
ω_2	1, 7	5, 5	5, 3	5, 1
ω_3	1, 7	3, 5	3, 3	3, 1
ω_4	1, 7	1, 5	1, 3	1, 1

Table 1: Example of a meta-game matrix in the initial stage of learning (top) and the final meta-game matrix (bottom).

After the initial meta-game matrix is created the agents play more iterations of the game, add or update the paths, and re-cluster according to Algorithm 2. After sufficient exploration, a final meta-game matrix is created and used thereafter, as shown in Table 1.

2.2 Predictive Model

The objective of the predictive model is to determine the opponents' next action from past observed actions. We aim at building a model that can utilize patterns temporally in the joint action space.

We desire the predictive model to have properties that facilitate fast and adaptive learning. The desirable properties include (1) being able to make sequential decisions while giving more weight to recent actions, (2) utilizing joint actions temporally, and (3) being sensitive to changes in the opponent strategy.

The predictive model is constructed using the concept learning method ELPH [18] as a building block. ELPH works by storing each subset of past actions, each called a hypothesis, up to size k in a hypothesis space with a corresponding set of observed outcomes. For instance, if actions a_1 and a_4 are played and each time the following opponent action is a_7, then a hypothesis consists of $\{a_1, a_4\} \Rightarrow a_7$. For each hypothesis an entropy value is calculated, which

is a measure of prediction uncertainty. Hypotheses with entropy values larger than a threshold are pruned away in order to keep the size of the hypothesis space reasonable. In each round of the game, ELPH chooses the hypothesis with the lowest entropy value and uses the corresponding outcome as the opponent's predicted next action.

We use ELPH as a building block because it satisfies the first two desirable properties we listed above. We satisfy the last property by building an ensemble of ELPH models. Each ELPH model is an expert in the ensemble and the final prediction is made through a weighted majority vote amongst the experts. Each expert who makes an incorrect prediction has its weight decreased by a multiplicative factor $\beta_{exp} < 1$.

A new expert is added if we observe a change in the opponent strategy. A change in opponent strategy is estimated using the Jensen-Shannon divergence (JSD) which is a symmetric and smooth version of the Kullback-Leibler (KL) divergence defined between two distributions P and Q as:

$$JSD(P,Q) = \frac{1}{2}KL(P||D) + \frac{1}{2}KL(Q||D), \quad (3)$$

where $D = \frac{1}{2}(P+Q)$, P and Q represent the distributions of the opponent's play on different time horizons. When $JSD > \delta$, a new expert is added to the ensemble at time t. The new expert's weight is set to a multiplicative factor of the total weight of the existing experts as $w_{t,|\mathfrak{E}|+1} = \gamma W_t$, where $0 \leq \gamma \leq 1$, $W_t = \sum_{i=1}^{|\mathfrak{E}|} w_{t,i}$, and $\mathfrak{E}$ is the set of experts.

Algorithm 3 FAL-SG: `Predictive Model`

Parameters:
  δ: threshold for adding a new expert.
Initialize:
  $\mathfrak{E} = ELPH_1$ (Initial expert)
while $t <$ end of game **do**
  Predict based on weighted majority vote from $\mathfrak{E}$.
  Update incorrect experts' weights, $w_{t+1,i} = \beta_{exp} w_{t,i}$
  Compute JSD.
  if JSD $> \delta$ **then**
    $\mathfrak{E} \leftarrow \mathfrak{E} \bigcup ELPH_{|\mathfrak{E}|+1}$
    Set weight of $ELPH_{|\mathfrak{E}|+1}, w_{t,|\mathfrak{E}|+1} = \gamma W_t$

Adding new experts helps the model to be sensitive to changes in the opponent strategy by allowing the new experts to concentrate only on recent strategies, while existing experts may be diluted by previous strategies. However, existing experts are kept since these experts are already trained and can quickly detect if an opponent switches back to a previous strategy. Algorithm 3 outlines this process.

2.3 Reasoning Model

The reasoning model combines information from both the meta-game model and the predictive model to decide the next action. The main objective of the reasoning model is to choose a pair of strategies, known as a targetable pair, in the meta-game that yield expected rewards that are larger than the meta-game minimax value for all players. The reasoning model computes a discounted prediction accuracy of the predictive model to measure discounted regret [6] as:

$$\rho_t = \sum_{i=1}^{t} \beta_{t-i} \ell(p_i, a_{-i}), \quad (4)$$

Algorithm 4 FAL-SG

Parameters:
  a_t: my action at time t.
  p_t: predicted opponent action at time t.
  ρ_t: prediction accuracy after t rounds.
  σ: prediction accuracy threshold.
  R^{max}: optimistic reward.
  num_{teach}: number of rounds to teach for.
  $num_{explore}$: number of rounds to explore for.
Initialize:
  $\widetilde{G}$ = null, $\forall s \in S$, $V^{\pi}(s) = R^{max}$, π = Q-Learning
while $t <$ end of game **do**
  $\widetilde{G} \leftarrow$ Algorithm 1 (Meta-game model)
  $p_t \leftarrow$ Algorithm 3 (Predictive model)
  $\rho_t \leftarrow$ Equation 4
  if $t < num_{explore}$ **then**
    $a_t \leftarrow \pi$
  else if $t - num_{explore} < num_{teach}$ **then**
    $a_t \leftarrow$ my half of targetable pair in $\widetilde{G}$
  else if $\rho_t < \sigma$ **then**
    Play Minimax in $\widetilde{G}$
  else if p_t is a targetable pair in $\widetilde{G}$ **then**
    $a_t \leftarrow$ my half of targetable pair in $\widetilde{G}$
  else
    $a_t \leftarrow$ Best Respond to p_t in $\widetilde{G}$

where $\beta_{t-i} = te^{\frac{i}{\lambda}}$, t is the number of rounds played, β_t are the discounted weights, e is Euler's constant, and ℓ is the 0-1 loss function that yields 1 if the predicted action p_i does not match the actual observed action a_{-i} and 0 otherwise.

The reasoning model starts by exploring to learn the game structure for $num_{explore}$ number of iterations. Afterwords, it will attempt to teach the opponents to cooperate. In the teaching phase, it will continue to play its half of the same targetable pair for num_{teach} iterations of the game. After that it will start to choose actions based on its prediction of the opponents' next action. It will only base its next action on the prediction when the past predictions have been accurate. If the prediction accuracy is high it will play its half of a targetable pair if the prediction is for the opponents to play their half, otherwise it will play a best response action. However, if the prediction accuracy is low it will play it safe by playing the minimax strategy.

3. THEORETICAL ANALYSIS

3.1 Abstraction Loss Bound

FAL-SG will not perform well if the abstraction does not accurately represent the underlying game. It has been shown in [15, 27] that abstraction is lossy for some general classes of games. However, a bound on a lossy abstraction is necessary as raised in [25]. As such, we establish an upper bound on the loss between the original game and our abstraction following the methodology in [25].

PROPOSITION 1. *Let the reward loss between the original and abstract game be defined as, $\epsilon^R = |V^{\pi} - W^{\pi}|$, then the valuation error is upper bounded by the largest distance between any two clusters valuation, $\forall s \in S, \forall R(s) \in R$*

$$\epsilon^R \leq max_{i,j \in k} | \max_{\forall \pi \in \omega_i} V^{\pi}(s) - \max_{\forall \pi \in \omega_j} V^{\pi}(s)|.$$

PROOF SKETCH. The reward for state s is a function of the clustering process. We have two potential cases,
Case 1: s is in its optimal cluster ω^*. in this case s will get assigned a reward that is initially optimistic, in accordance with the optimism criterion but has been shown to converge to the expected reward in [4, 9].
Case 2: s is not in the optimal cluster ω. In this case the loss of state s is the distance between the expected reward of s in the optimal cluster and the expected reward of s in a non-optimal cluster.

$$\begin{aligned}\epsilon^R &\leq |\mathbb{E}(\omega^*) - \mathbb{E}(\omega)| \\ &\leq max_{i,j\in K}|\mathbb{E}(\omega_i) - \mathbb{E}(\omega_j)| \\ &\leq max_{i,j\in K}|\max_{\forall\pi\in\omega_i} V^\pi(s) - \max_{\forall\pi\in\omega_j} V^\pi(s)| \quad (5)\end{aligned}$$

Assume we increase the number of clusters K as the game is played, then as the number of clusters approaches the number of states, s will be its own cluster and its reward will be $\widetilde{R}(h(s), g(a)) = R(s, a)$. □

3.2 Prediction Mistake Bound

We sequentially choose an action a_t based on the prediction of the opponents' action p_t using Algorithm 4. One goal is to minimize the number of prediction mistakes. We establish a typical mistake bound in online learning literature [21] for FAL-SG specifically following [19] using Theorem 1.

THEOREM 1. *Between any time i and j where $i < j$, if $\beta_{exp} + 2\gamma < 1$ then the number of mistakes between i and j is bounded by*

$$M_j - M_i \leq \frac{m\log(1/\beta_{exp}) + \log(1/\gamma)}{\log(2/(1+\beta_{exp}+2\gamma))},$$

where M_i is the number of mistakes made up to time i and m is the number of mistakes the new expert makes between time i and j.

PROOF. To prove Theorem 1 we need to show two things: (1) an upper bound on the total weight W_j at time j and (2) a lower bound on W_j.

(1) Whenever FAL-SG makes a mistake we know that at least half of the experts in the ensemble made a mistake. Each time an expert makes a mistake we multiply its weight by β_{exp}. If the Jensen-Shannon divergence $> \delta$ then we add an expert to the ensemble and set its weight to a fraction of the total weight of the ensemble γW_t. We can bound the total weight of the ensemble at time j with

$$W_j \leq \frac{1}{2}W_i + \frac{\beta_{exp}}{2}W_i + \gamma W_i = \frac{1+\beta_{exp}+2\gamma}{2}W_i.$$

Recursively this leads to

$$W_j \leq \frac{1+\beta_{exp}+2\gamma}{2}^{M_j - M_i} W_i.$$

Rearranging the terms and taking the logarithm, we get

$$\begin{aligned}&\Rightarrow \log(W_j/W_i) \leq (M_j - M_i)\log((1+\beta_{exp}+2\gamma)/2) \\ &\Rightarrow M_j - M_i \leq \frac{\log(W_i/W_j)}{\log(2/(1+\beta_{exp}+2\gamma))}. \quad (6)\end{aligned}$$

(2) When $JSD > \delta$, we add a new expert with initial weight of γW_i. The weight of this expert is decreased each time it makes a mistake by a factor of β_{exp}. Let m be the number of mistakes the expert makes by time j then

$$W_j \geq \gamma W_i \beta_{exp}^m.$$

Substituting this into Equation 6 we obtain the desired bound,

$$\begin{aligned}&\Rightarrow M_j - M_i \leq \frac{\log(W_i/\gamma W_i \beta_{exp}^m)}{\log(2/(1+\beta_{exp}+2\gamma))} \\ &\Rightarrow M_j - M_i \leq \frac{m\log(1/\beta_{exp}) + \log(1/\gamma)}{\log(2/(1+\beta_{exp}+2\gamma))} \quad \square\end{aligned}$$

3.3 Multiagent Learning Criteria

The multiagent learning literature has proposed multiple criteria that lay the theoretical groundwork on which to build learning agents [3, 24]. We believe the criteria presented in [24], Safety, Auto-Compatibility, and Targeted Optimality, are appropriate for FAL-SG.

For these criteria, we assume a n-player, m-action meta-game, that all players are able to calculate the targetable pair strategies and minimax value, and in self-play the players choose the best targetable pair. Additionally, we assume the period of exploration is finished so a complete meta-game has been built.

Safety: Safety requires an agent to guarantee a lower bound on the average reward it receives against any opponent. This provides a minimum performance regardless if everything else fails.

THEOREM 2. *The average reward against any opponent is at least $SV - \epsilon$, where SV is the minimax value of the meta-game and ϵ is some error.*

PROOF SKETCH. We show that FAL-SG satisfies the safety criterion in a worst-case analysis using Theorem 1. FAL-SG will only receive a reward less than SV if it plays the targetable pair or best response strategies and its prediction is incorrect. We already proved an upper bound on the number of mistakes FAL-SG makes, which we can use to establish a lower bound on the safety by utilizing a worst-case scenario and assume that FAL-SG will always trust its prediction (i.e., the prediction accuracy threshold $\sigma = 0$).

The total number of mistakes made by FAL-SG in a repeated game of T rounds is bounded by

$$M_{total} \leq \frac{m\log(1/\beta_{exp}) + \log(1/\gamma)}{\log(2/(1+\beta_{exp}+2\gamma))}.$$

If the minimum reward R_{min} is received after each mistake then the total reward received is

$$\begin{aligned}R_{total} &= (T - M_{total}) \times SV + M_{total} \times R_{min} \\ &= T \times SV - M_{total} \times (SV - R_{min}).\end{aligned}$$

Let $\Delta = SV - R_{min}$, then

$$= T \times SV - M_{total} \times \Delta.$$

The average reward is

$$\begin{aligned}R_{avg} &= \frac{1}{T}(T \times SV - M_{total} \times \Delta) \\ &= SV - \frac{M_{total}}{T} \times \Delta.\end{aligned}$$

If we let $\epsilon = \frac{M_{total}}{T} \times \Delta$ then we obtain the desired bound. □

Auto-Compatibility: An agent, when playing against itself, should learn a Pareto optimal strategy.

THEOREM 3. *The average reward in self-play is Pareto optimal.*

PROOF SKETCH. FAL-SG will initially teach its opponents to be cooperative and will play its half of the best targetable pair. Each player will play this targetable pair. After the teaching phase in Algorithm 4 the prediction accuracy for all players will be perfect. Since $\rho_t >= \sigma$, all players will predict that the others will play this targetable pair, and all players will continue to play the same targetable pair *ad infinitum*. □

Targeted Optimality: In many situations, we design an agent with a particular class of opponents in mind. We desire the performance against this target class of opponents to be optimal.

THEOREM 4. *The average reward is at least $V_{BR} - \epsilon$ where V_{BR} is the expected value of the best response against a target class of opponents and ϵ is some error.*

PROOF SKETCH. The target class of opponents we consider is memory bounded stationary opponents. The memory size of this class of opponents is upper bounded by k, where k is at most the hypothesis size of ELPH. When an opponent has a memory bound of k we mean that the opponent can only keep the last k rounds of joint actions in memory.

When the opponents' memory is bound to the last k joint actions then the most sophisticated strategy they may play is a strategy that has at most $n^{(k+1)}$ possible action sequences where n is the number of actions available. ELPH will be able to fit every possible action sequence into its hypothesis space. Since the opponents are stationary these action sequences are deterministic and the entropy ELPH calculates will converge to zero and FAL-SG will be able to predict with certainty the next action the opponents will play and best respond. Therefore, the average reward will be V_{BR} and $\epsilon = 0$. □

4. EXPERIMENTAL RESULTS

4.1 Stochastic Prisoner's Dilemma

To illustrate the performance of FAL-SG to learn and adapt, we experimented in two 2-player, m-action stochastic games: stochastic Prisoner's Dilemma and stochastic Chicken. Our focus in these experiments was to measure the speed and adaptability of algorithms in a limited number of interactions. To this end, our experiments only consisted of a few thousand iterations per game.

We present results of the performance of FAL-SG, WoLF-PHC, Giga-WoLF, Fictitious Play (FP), and Q-Learning in self-play. We also present some results of these algorithms in pairwise competition with FAL-SG. All the algorithms are extended with Pepper [9] so they can be used in stochastic games. Each experiment was run 100 times and the results are the average amongst these runs.

WoLF-PHC and Giga-WoLF [3, 2] utilize the "Win or Learn Fast" concept by using a variable learning rate to learn fast when losing and more slowly when winning.

Fictitious Play [5] is one of the first multiagent learning algorithms developed for normal form games. It works by assuming the opponents are playing a stationary strategy and then plays a best response strategy to the empirical distribution of opponent actions.

M-Qubed [10] is a reinforcement learning algorithm for stochastic games which balances best response, cautious learning to bound losses, and optimistic learning by looking for strategies with potentially high returns even if risky.

Q-Learning [22] is a single agent reinforcement learning algorithm adapted for multiagent environments.

Self-play: Figure 4 shows the average reward in self-play over time. In this game, converging to the defection strategy leads to an expected reward of 2, which occurs because both players are choosing gate 1 and eventually going to gate 3 for a total reward of 2. Converging to the cooperation strategy leads to an expected reward of 5 which occurs when both choose to go to gate 2. In self-play, WoLF-PHC, Giga-WoLF, Q-Learning, and FP all converge asymptotically to defection with a reward of 2. Q-Learning converges fast while the rest converge at a slower rate. Overall, most of algorithms performed reasonably well, primarily due to the extension provided by Pepper which transforms the stochastic game into a modified normal-form game,

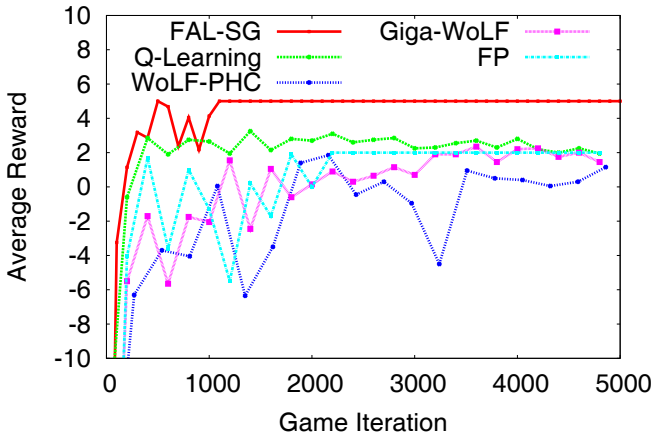

Figure 4: Average reward obtained in self-play in stochastic Prisoner's Dilemma.

FAL-SG computes an abstraction of the game in 200 iterations of exploration and starts exploring abstract states other than gate 1. FAL-SG attempts to teach the opponent early on to cooperate by playing its half of the targetable pair. Eventually, both agents in self-play realize the willingness of the other to cooperate, leading to convergence to cooperation in less than 1,000 games. For the rest of the game, FAL-SG continues to cooperate without ever deviating. The speed of FAL-SG's convergence in self-play is surprising, but it is explained by the nature of the game and the reward model structure. In this game, it is not important whether the agent learns the true reward of the meta-game as long as it learns the relative benefit of going to one gate versus the others.

Pairwise Comparison. A pairwise comparison between FAL-SG and Q-Learning, WoLF-PHC, and M-Qubed is presented in Figure 5 for 5,000 game iterations. The relatively small amount of iterations is crucial as it shows the relative quickness of each agent. The agents are acting in unknown environments so we do not expect them to perform optimally but we are evaluating their relative performance. This might be viewed as an inadequate comparison between the models to analyze their performance when they are most likely randomizing and exploring to learn the environment model, but the major goal of this work is to motivate and highlight

the advantages of fast learning. Therefore, this comparison is suitable keeping in mind our goal.

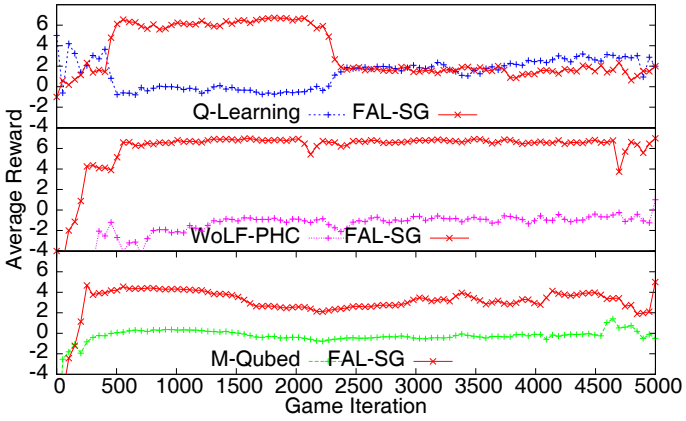

Figure 5: Average reward obtained by FAL-SG vs other agents in stochastic Prisoner's Dilemma.

Figure 5 shows the performance of FAL-SG against three other agents. It is important to note that FAL-SG was able to compute a preliminary meta-game model and base its actions on that. Most of the other agents were still exploring to build an exact game model during most of the 5,000 iterations. This exploration phase of the opponents did not allow FAL-SG to predict their next action accurately, so it reverted to playing the minimax strategy.

Q-Learning versus FAL-SG has an interesting result where Q-Learning explored until around iteration 2,250 when it decided to switch to the defecting strategy and FAL-SG was able to adapt and defected accordingly. WoLF-PHC versus FAL-SG has the widest margin of performance and slower convergence rate than Q-Learning. This is consistent with results shown in self-play as it took longer for WoLF-PHC to converge to the defection strategy which is explained by the variability in the learning rate that WoLF-PHC utilizes in learning. FAL-SG versus M-Qubed has a smaller margin of performance but FAL-SG is still able to obtain a higher average reward by obtaining an asymptotic reward of 4 while M-Qubed obtains an asymptotic reward of 0. We can speculate that this could be due to the fact that FAL-SG might be picking up some signal of cooperation from M-Qubed.

4.2 Stochastic Chicken

A stochastic version of Chicken is shown in Figure 6.

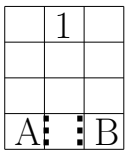

Figure 6: A stochastic game of Chicken.

If one player goes to the state marked 1 via the dotted line the player gets a reward of 7. However, if both go via the dotted line a conflict occurs, neither moves and each gets a -1. If both go directly (not via dotted line) to the state marked 1 both move forward with a 50% probability (meaning that half of the time they will get an additional -1 and not move forward). If both try the dotted line in the worst case they are back to start each with -1. If they go directly each has a 50% chance of going forward which is better than the reward when both go via the dotted line.

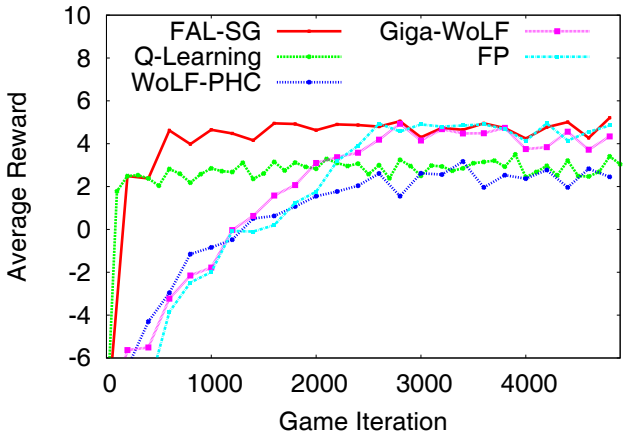

Figure 7: Average reward obtained in self-play in stochastic Chicken.

From Figure 7 we can see that FAL-SG, Giga-WoLF, and Fictitious Play converge to an average reward a little higher than 4. However, FAL-SG converges much faster (about 500 iterations) while Giga-Wolf and Fictitious Play converge after about 2,500 iterations. FAL-SG's fast convergence corresponds to when it is done exploring the game and starts taking advantage of the meta-game it created. Q-Learning and WoLF-PHC converge to a lower average reward of 3.

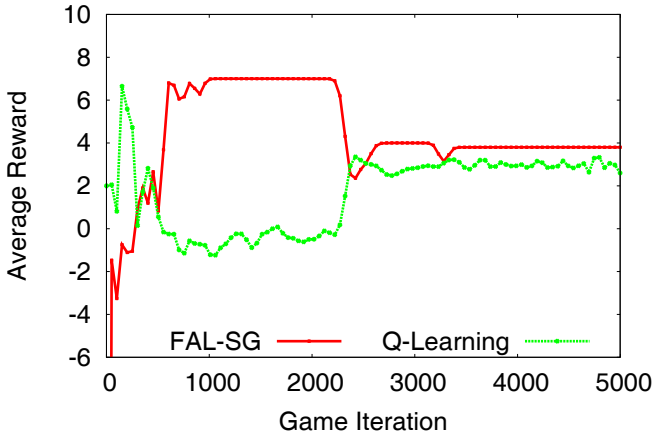

Figure 8: Average reward obtained by FAL-SG vs Q-Learning in stochastic Chicken.

When FAL-SG plays other agents in a game of Chicken, it starts with a period of exploration (500 iterations) after which it again utilizes the meta-game abstraction that it found as shown in Figure 8 against Q-Learning with Pepper. The meta-game abstraction allowed FAL-SG to quickly discover a strategy that yielded a reward above 5 while Q-Learning is too slow in discovering this strategy. There is

no doubt that Q-Learning in the long term will converge as shown in earlier self-play results. However, the motivation of fast learning is exactly these scenarios of learning in limited interactions and in this case the limit was 1,500 iterations.

5. CONCLUSIONS AND FUTURE WORK

We introduced a multiagent learning algorithm, FAL-SG, that is able to learn fast and adapt to opponents in repeated stochastic games through the use of game abstraction. Our analysis shows that the abstraction accurately represents the underlying game and the number of incorrect predictions FAL-SG makes is bounded. We show that FAL-SG satisfies criteria which enable it to perform desirably against any opponent in complicated environments. Our experimental results demonstrate that FAL-SG is able to learn and adapt fast and outperform other learning agents.

In the future we plan to compare different predictive models in order to improve the interplay between online prediction and opponent modeling for better performance and stronger theoretical guarantees. Additionally, we wish to explore how the use of abstractions may allow an agent to be more competitive against a larger target class of opponents.

Acknowledgment: Work supported in part by NSF IIS-1208413 and the Safety, Security, and Rescue Research Center at the University of Minnesota.

6. REFERENCES

[1] B. Banerjee and J. Peng. Performance bounded reinforcement learning in strategic interactions. In *Proc. AAAI Conference*, pages 2–7, 2004.

[2] M. Bowling. Convergence and no-regret in multiagent learning. In *Advances in Neural Information Processing Systems*, page 209. The MIT Press, 2005.

[3] M. Bowling and M. Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136(2):215–250, 2002.

[4] R. Brafman and M. Tennenholtz. R-max-a general polynomial time algorithm for near-optimal reinforcement learning. *The Journal of Machine Learning Research*, 3:213–231, 2003.

[5] G. Brown. Iterative solution of games by fictitious play. *Activity analysis of production and allocation*, 13(1):374–376, 1951.

[6] N. Cesa-Bianchi and G. Lugosi. *Prediction, learning, and games*. Cambridge University Press, 2006.

[7] D. Chakraborty and P. Stone. Convergence, targeted optimality and safety in multiagent learning. In *Proc. Int'l Conf. on Machine Learning*, June 2010.

[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. *Machine Learning*, 67(1):23–43, 2007.

[9] J. Crandall. Just add Pepper: Extending learning algorithms for repeated matrix games to repeated Markov games. In *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, 2012.

[10] J. Crandall and M. Goodrich. Learning to compete, coordinate, and cooperate in repeated games using reinforcement learning. *Machine Learning*, 82(3):281–314, 2011.

[11] E. de Cote and M. Littman. A polynomial-time Nash equilibrium algorithm for repeated stochastic games. In *Proc. Conf. on Uncertainty in AI*, 2008.

[12] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 1–38, 1977.

[13] M. Elidrisi and M. Gini. When speed matters in learning against adversarial opponents (Extended Abstract). In *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, pages 1289–1290, 2012.

[14] C. Fraley and A. E. Raftery. Model-based clustering, discriminant analysis, and density estimation. *Journal of the American Statistical Association*, 97(458):611–631, 2002.

[15] A. Gilpin and T. Sandholm. Better automated abstraction techniques for imperfect information games, with application to Texas Hold'em poker. In *Proc. Int'l Conf. on Autonomous Agents and Multi-Agent Systems*, page 192. ACM, 2007.

[16] A. Greenwald, M. Zinkevich, and P. Kaelbling. Correlated Q-learning. In *Proc. Int'l Conf. on Machine Learning*, pages 242–249, 2003.

[17] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proc. Int'l Conf. on Machine Learning*, pages 242–250, San Francisco, CA, USA, 1998.

[18] S. Jensen, D. Boley, M. Gini, and P. Schrater. Non-stationary policy learning in 2-player zero sum games. In *Proc. AAAI Conference*, pages 789–794. AAAI Press, 2005.

[19] J. Z. Kolter and M. A. Maloof. Using additive expert ensembles to cope with concept drift. In *Proc. Int'l Conf. on Machine Learning*, pages 449–456, 2005.

[20] L. Li, T. Walsh, and M. Littman. Towards a unified theory of state abstraction for MDPs. In *Proc. 9th Int'l Symposium on Artificial Intelligence and Mathematics*, pages 531–539, 2006.

[21] N. Littlestone and M. K. Warmuth. The weighted majority algorithm. *Inf. Comput.*, 108(2):212–261, Feb. 1994.

[22] M. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proc. Int'l Conf. on Machine Learning*, 1994.

[23] M. Littman and P. Stone. Leading best-response strategies in repeated games. In *Int'l Joint Conf. on Artificial Intelligence Workshop on Economic Agents, Models, and Mechanisms*, 2001.

[24] R. Powers, Y. Shoham, and T. Vu. A general criterion and an algorithmic framework for learning in multi-agent systems. *Machine Learning*, 67(1):45–76, 2007.

[25] T. Sandholm and S. Singh. Lossy stochastic game abstraction with bounds. In *Prod. of the 13th ACM Conf. on Electronic Commerce*, pages 880–897, 2012.

[26] L. Shapley. Stochastic games. *Proceedings of the National Academy of Sciences of the United States of America*, 39(10):1095, 1953.

[27] J. Shi and M. L. Littman. Abstraction methods for game theoretic poker. In *Computers and Games*, pages 333–345. Springer, 2001.