Scalable Learning in Stochastic Games

Michael Bowling and **Manuela Veloso**
Computer Science Department
Carnegie Mellon University
Pittsburgh PA, 15213-3891

Abstract

Stochastic games are a general model of interaction between multiple agents. They have recently been the focus of a great deal of research in reinforcement learning as they are both descriptive and have a well-defined Nash equilibrium solution. Most of this recent work, although very general, has only been applied to small games with at most hundreds of states. On the other hand, there are landmark results of learning being successfully applied to specific large and complex games such as Checkers and Backgammon. In this paper we describe a scalable learning algorithm for stochastic games, that combines three separate ideas from reinforcement learning into a single algorithm. These ideas are tile coding for generalization, policy gradient ascent as the basic learning method, and our previous work on the WoLF ("Win or Learn Fast") variable learning rate to encourage convergence. We apply this algorithm to the intractably sized game-theoretic card game Goofspiel, showing preliminary results of learning in self-play. We demonstrate that policy gradient ascent can learn even in this highly non-stationary problem with simultaneous learning. We also show that the WoLF principle continues to have a converging effect even in large problems with approximation and generalization.

Introduction

We are interested in the problem of learning in multiagent environments. One of the main challenges with these environments is that other agents in the environment may be learning and adapting as well. These environments are, therefore, no longer stationary. They violate the Markov property that traditional single-agent behavior learning relies upon.

The model of stochastic games captures these problems very well through explicit models of the reward functions of the other agents and their affects on transitions. They are also a natural extension of Markov decision processes (MDPs) to multiple agents and so have attracted interest from the reinforcement learning community. The problem of simultaneously finding optimal policies for stochastic games has been well studied in the field of game theory. The traditional solution concept is that of Nash equilibria, a policy for all the players where each is playing optimally with respect to the others. This concept is a powerful solution for these games even in a learning context, since no agent could learn a better policy when all the agents are playing an equilibria.

It is this foundation that has driven much of the recent work in applying reinforcement learning to stochastic games (Littman 1994; Hu & Wellman 1998; Singh, Kearns, & Mansour 2000; Littman 2001; Bowling & Veloso 2002a; Greenwald & Hall 2002). This work has thus far only been applied to small games with enumerable state and action spaces. Historically, though, a number of landmark results in reinforcement learning have looked at learning in particular stochastic games that are not small nor are the state easily enumerated. Samuel's Checkers playing program (Samuel 1967) and Tesauro's TD-Gammon (Tesauro 1995) are successful applications of learning in games with very large state spaces. Both of these results made generous use of generalization and approximation, which have not been used in the more recent work. On the other hand, both TD-Gammon and Samuel's Checkers player only used deterministic strategies to play competitively, while Nash equilibria often require stochastic strategies.

We are interested in scaling some of the recent techniques based on the Nash equilibria concept to games with intractable state spaces. Such a goal is not new. Singh and colleagues' also described future work of applying their simple gradient techniques to problems with large or infinite state and action spaces (Singh, Kearns, & Mansour 2000). This paper examines some initial results in this direction. We first describe the formal definition of a stochastic game and the notion of equilibria. We then describe one particular very large, two-player, zero-sum stochastic game, Goofspiel. Our learning algorithm is described as the combination of three ideas from reinforcement learning: tile-coding, policy gradients, and the WoLF principle. We then show results of our algorithm learning to play Goofspiel with self-play. Finally, we conclude with some future directions for this work.

Stochastic Games

A *stochastic game* is a tuple $(n, \mathcal{S}, \mathcal{A}_{1\ldots n}, T, R_{1\ldots n})$, where n is the number of agents, $\mathcal{S}$ is a set of states, $\mathcal{A}_i$ is the set of actions available to agent i (and $\mathcal{A}$ is the joint action space $\mathcal{A}_1 \times \ldots \times \mathcal{A}_n$), T is a transition function $\mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$, and R_i is a reward function for the ith agent $\mathcal{S} \times \mathcal{A} \rightarrow \Re$.

This looks very similar to the MDP framework except we have multiple agents selecting actions and the next state and rewards depend on the joint action of the agents. Another important difference is that each agent has its own separate reward function. The goal for each agent is to select actions in order to maximize its discounted future rewards with discount factor γ.

Stochastic games are a very natural extension of MDPs to multiple agents. They are also an extension of matrix games to multiple states. Two example matrix games are in Figure 1. In these games there are two players; one selects a row and the other selects a column of the matrix. The entry of the matrix they jointly select determines the payoffs. The games in Figure 1 are zero-sum games, so the row player would receive the payoff in the matrix, and the column player would receive the negative of that payoff. In the general case (general-sum games), each player would have a separate matrix that determines their payoffs.

$$\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix} \qquad \begin{pmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{pmatrix}$$

Matching Pennies R-P-S

Figure 1: Matching Pennies and Rock-Paper-Scissors matrix games.

Each state in a stochastic game can be viewed as a matrix game with the payoffs for each joint action determined by the matrices $R_i(s, a)$. After playing the matrix game and receiving their payoffs the players are transitioned to another state (or matrix game) determined by their joint action. We can see that stochastic games then contain both MDPs and matrix games as subsets of the framework.

Stochastic Policies. Unlike in single-agent settings, deterministic policies in multiagent settings can often be exploited by the other agents. Consider the matching pennies matrix game as shown in Figure 1. If the column player were to play either action deterministically, the row player could win every time. This requires us to consider mixed strategies and stochastic policies. A stochastic policy, $\rho : \mathcal{S} \rightarrow PD(\mathcal{A}_i)$, is a function that maps states to mixed strategies, which are probability distributions over the player's actions.

Nash Equilibria. Even with the concept of mixed strategies there are still no optimal strategies that are independent of the other players' strategies. We can, though, define a notion of best-response. A strategy is a *best-response* to the other players' strategies if it is optimal given their strategies. The major advancement that has driven much of the development of matrix games, game theory, and even stochastic games is the notion of a best-response equilibrium, or *Nash equilibrium* (Nash, Jr. 1950).

A Nash equilibrium is a collection of strategies for each of the players such that each player's strategy is a best-response to the other players' strategies. So, no player can do better by changing strategies given that the other players also don't change strategies. What makes the notion of equilibrium compelling is that all matrix games have such an equilibrium, possibly having multiple equilibria. Zero-sum, two-player games, where one player's payoffs are the negative of the other, have a *single* Nash equilibrium.[1] In the zero-sum examples in Figure 1, both games have an equilibrium consisting of each player playing the mixed strategy where all the actions have equal probability.

The concept of equilibria also extends to stochastic games. This is a non-trivial result, proven by Shapley (Shapley 1953) for zero-sum stochastic games and by Fink (Fink 1964) for general-sum stochastic games.

Learning in Stochastic Games. Stochastic games have been the focus of recent research in the area of reinforcement learning. There are two different approaches being explored. The first is that of algorithms that explicitly learn equilibria through experience, independent of the other players' policy (Littman 1994; Hu & Wellman 1998; Greenwald & Hall 2002). These algorithms iteratively estimate value functions, and use them to compute an equilibrium for the game. A second approach is that of best-response learners (Claus & Boutilier 1998; Singh, Kearns, & Mansour 2000; Bowling & Veloso 2002a). These learners explicitly optimize their reward with respect to the other players' (changing) policies. This approach, too, has a strong connection to equilibria. If these algorithms converge when playing each other, then they must do so to an equilibrium (Bowling & Veloso 2001).

Neither of these approaches, though, have been scaled beyond games with a few hundred states. Games with a very large number of states, or games with continuous state spaces, make state enumeration intractable. Since previous algorithms in their stated form require the enumeration of states either for policies or value functions, this is a major limitation. In this paper we examine learning in a very large stochastic game, using approximation and generalization techniques. Specifically, we will build on the idea of best-response learners using gradient techniques (Singh, Kearns, & Mansour 2000; Bowling & Veloso 2002a). We first describe an interesting game with an intractably large state space.

Goofspiel

Goofspiel (or The Game of Pure Strategy) was invented by Merrill Flood while at Princeton (Flood 1985). The game has numerous variations, but here we focus on the simple two-player, n-card version. Each player receives a suit of cards numbered 1 through n, a third suit of cards is shuffled and placed face down as the deck. Each round the next card is flipped over from the deck, and the two players each select a card placing it face down. They are revealed simultaneously and the player with the highest card wins the card from the deck, which is worth its number in points. If

[1]There can actually be multiple equilibria, but they will all have equal payoffs and are interchangeable (Osborne & Rubinstein 1994).

the players choose the same valued card, then neither player gets any points. Regardless of the winner, both players discard their chosen card. This is repeated until the deck and players hands are exhausted. The winner is the player with the most points.

This game has numerous interesting properties making it a very interesting step between toy problems and more realistic problems. First, notice that this game is zero-sum, and as with many zero-sum games any deterministic strategy can be soundly defeated. In this game, it's by simply playing the card one higher than the other player's deterministically chosen card. Second, notice that the number of states and state-action pairs grows exponentially with the number of cards. The standard size of the game $n = 13$ is so large that just storing one player's policy or Q-table would require approximately 2.5 terabytes of space. Just gathering data on all the state-action transitions would require well over 10^{12} playings of the game. Table 1 shows the number of states and state-action pairs as well as the policy size for three different values of n. This game obviously requires some form of generalization to make learning possible. Another interesting property is that randomly selecting actions is a reasonably good policy. The worst-case values of the random policy along with the worst-case values of the best deterministic policy are also shown in Table 1.

This game can be described using the stochastic game model. The state is the current cards in the players' hands and deck along with the upturned card. The actions for a player are the cards in the player's hand. The transitions follow the rules as described, with an immediate reward going to the player who won the upturned card. Since the game has a finite end and we are interested in maximizing total reward, we can set the discount factor γ to be 1. Although equilibrium learning techniques such as Minimax-Q (Littman 1994) are guaranteed to find the game's equilibrium, it requires maintaining a state-joint-action table of values. This table would require 20.1 terabytes to store for the $n = 13$ card game. We will now describe a best-response learning algorithm using approximation techniques to handle the enormous state space.

Three Ideas – One Algorithm

The algorithm we will use combines three separate ideas from reinforcement learning. The first is the idea of tile coding as a generalization for linear function approximation. The second is the use of a parameterized policy and learning as gradient ascent in the policy's parameter space. The final component is the use of a WoLF variable learning rate to adjust the gradient ascent step size. We will briefly overview these three techniques and then describe how they are combined into a reinforcement learning algorithm for Goofspiel.

Tile Coding.

Tile coding (Sutton & Barto 1998), also known as CMACS, is a popular technique for creating a set of boolean features from a set of continuous features. In reinforcement learning, tile coding has been used extensively to create linear approximators of state-action values (e.g., (Stone & Sutton 2001)).

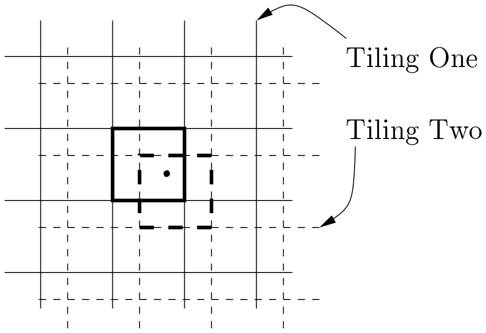

Figure 2: An example of tile coding a two dimensional space with two overlapping tilings.

The basic idea is to lay offset grids or tilings over the multidimensional continuous feature space. A point in the continuous feature space will be in exactly one tile for each of the offset tilings. Each tile has an associated boolean variable, so the continuous feature vector gets mapped into a very high-dimensional boolean vector. In addition, nearby points will fall into the same tile for many of the offset grids, and so share many of the same boolean variables in their resulting vector. This provides the important feature of generalization. An example of tile coding in a two-dimensional continuous space is shown in Figure 2. This example shows two overlapping tilings, and so any given point falls into two different tiles.

Another common trick with tile coding is the use of hashing to keep the number of parameters manageable. Each tile is hashed into a table of fixed size. Collisions are simply ignored, meaning that two unrelated tiles may share the same parameter. Hashing reduces the memory requirements with little loss in performance. This is because only a small fraction of the continuous space is actually needed or visited while learning, and so independent parameters for every tile are often not necessary. Hashing provides a means for using only the number of parameters the problem requires while not knowing in advance which state-action pairs need parameters.

Policy Gradient Ascent

Policy gradient techniques (Sutton *et al.* 2000; Baxter & Bartlett 2000) are a method of reinforcement learning with function approximation. Traditional approaches approximate a state-action value function, and result in a deterministic policy that selects the action with the maximum learned value. Alternatively, policy gradient approaches approximate a policy directly, and then use gradient ascent to adjust the parameters to maximize the policy's value. There are three good reasons for the latter approach. First, there's a whole body of theoretical work describing convergence problems using a variety of value-based learning techniques with a variety of function approximation techniques (See (Gordon 2000) for a summary of these results.) Second, value-based approaches learn deterministic policies, and as we mentioned earlier deterministic policies in multiagent

n	$\|S\|$	$\|S \times A\|$	SIZEOF(π or Q)	VALUE(det)	VALUE(random)
4	692	15150	$\sim$ 59KB	-2	-2.5
8	3×10^6	1×10^7	$\sim$ 47MB	-20	-10.5
13	1×10^{11}	7×10^{11}	$\sim$ 2.5TB	-65	-28

Table 1: The approximate number of states and state-actions, and the size of a stochastic policy or Q table for Goofspiel depending on the number of cards, n. The VALUE columns list the worst-case value of the best deterministic policy and the random policy respectively.

settings are often easily exploitable. Third, gradient techniques have been shown to be successful for simultaneous learning in matrix games (Singh, Kearns, & Mansour 2000; Bowling & Veloso 2002a).

We use the policy gradient technique presented by Sutton and colleagues (Sutton *et al.* 2000). Specifically, we will define a policy as a Gibbs distribution over a linear combination of features, such as those taken from a tile coding representation of state-actions. Let θ be a vector of the policy's parameters and ϕ_{sa} be a feature vector for state s and action a then this defines a stochastic policy according to,

$$\pi(s,a) = \frac{e^{\theta \cdot \phi_{sa}}}{\sum_b e^{\theta \cdot \phi_{sb}}}.$$

Their main result was a convergence proof for the following policy iteration rule that updates a policy's parameters,

$$\theta_{k+1} = \theta_k + \alpha_k \sum_s d^{\pi_k(s)} \sum_a \frac{\partial \pi_k(s,a)}{\partial \theta} f_{w_k}(s,a). \quad (1)$$

For the Gibbs distribution this is just,

$$\theta_{k+1} = \theta_k + \alpha_k \sum_s d^{\pi_k(s)} \sum_a \phi_{sa} \cdot \pi(s,a) f_{w_k}(s,a) \quad (2)$$

Here α_k is an appropriately decayed learning rate and $d^{\pi_k}(s)$ is state s's contribution to the policy's overall value. This contribution is defined differently depending on whether average or discounted start state reward criterion is used. $f_{w_k}(s,a)$ is an independent approximation of $Q^{\pi_k}(s,a)$ with parameters w, which is the expected value of taking action a from state s and then following the policy π_k. For a Gibbs distribution, Sutton and colleagues showed that for convergence this approximation should have the following form,

$$f_w(s,a) = w \cdot \left[\phi_{sa} - \sum_b \pi(s,b)\phi_{sb} \right].$$

As they point out, this amounts to f_w being an approximation of the advantage function, $A^\pi(s,a) = Q^\pi(s,a) - V^\pi(s)$, where $V^\pi(s)$ is the value of following policy π from state s. It is this advantage function that we will estimate and use for gradient ascent.

Using this basic formulation we derive an on-line version of the learning rule, where the policy's weights are updated with each state visited. The total reward criterion for Goofspiel is identical to having $\gamma = 1$ in the discounted setting. So, $d^\pi(s)$ is just the probability of visiting state s when following policy π. Since we will be visiting states on-policy, this amounts to updating weights in proportion to how often the state is visited. By doing updates on-line as states are visited we can simply drop this term from equation 2, resulting in,

$$\theta_{k+1} = \theta_k + \alpha_k \sum_a \phi_{sa} \cdot \pi(s,a) f_{w_k}(s,a). \quad (3)$$

Lastly, we will do the policy improvement step (updating θ) simultaneously with the value estimation step (updating w). We will do value estimation using gradient-descent Sarsa(0) (Sutton & Barto 1998) over the same feature space as the policy. Specifically, if at time k the system is in state s and takes action a transitioning to state s' and then taking action a', we update the weight vector,

$$w_{k+1} = w_k + \beta_k \left(r + \gamma Q_{w_k}(s',a') - Q_{w_k}(s,a)\right) \quad (4)$$

The policy improvement step uses equation 3 where s is the state of the system at time k and the action-value estimates from Sarsa Q_{w_k} are used to compute the advantage term,

$$f_{w_k}(s,a) = Q_{w_k}(s,a) - \sum_a \pi(s,a|\theta_k) Q_{w_k}(s,a).$$

Win or Learn Fast

WoLF ("Win or Learn Fast") is a method for changing the learning rate to encourage convergence in a multiagent reinforcement learning scenario (Bowling & Veloso 2002a). Notice that the gradient ascent algorithm described does not account for a non-stationary environment that arises with simultaneous learning in stochastic games. All of the other agents actions are simply assumed to be part of the environment and unchanging. WoLF provides a simple way to account for other agents through adjusting how quickly or slowly the agent changes its policy.

Since only the rate of learning is changed, algorithms that are guaranteed to find (locally) optimal policies in non-stationary environments retain this property even when using WoLF. In stochastic games with simultaneous learning, WoLF has both theoretical evidence (limited to two-player, two-action matrix games), and empirical evidence (experiments in matrix games, as well as smaller zero-sum and general-sum stochastic games) that it encourages convergence in algorithms that don't otherwise converge (Bowling & Veloso 2002a). The intuition for this technique is that a learner should adapt quickly when it is doing more poorly than expected. When it is doing better than expected, it should be cautious, since the other players are likely to change their policy. This implicitly accounts for other players that are learning, rather than other techniques that try to explicitly reason about their action choices.

The WoLF principle naturally lends itself to policy gradient techniques where there is a well-defined learning rate, α_k. With WoLF we replace the original learning rate with two learning rates $\alpha_k^w < \alpha_k^l$ to be used when winning or losing, respectively. One determination of winning and losing that has been successful is to compare the value of the current policy $V^\pi(s)$ to the value of the average policy over time $V^{\bar{\pi}}(s)$. With the policy gradient technique above we can define a similar rule that examines the approximate value, using Q_w, of the current weight vector θ with the average weight vector over time $\bar{\theta}$. Specifically, we are "winning" if and only if,

$$\sum_a \pi(s,a|\theta)Q_w(s,a) > \sum_a \pi(s,a|\bar{\theta})Q_w(s,a). \qquad (5)$$

When winning in a particular state, we update the parameters for that state using α_k^w, otherwise α_k^l.

Learning in Goofspiel

We combine these three techniques in the obvious way. Tile coding provides a large boolean feature vector for any state-action pair. This is used both for the parameterization of the policy and for the approximation of the policy's value, which is used to compute the policy's gradient. Gradient updates are then performed on both the policy using equation 3 and the value estimate using equation 4. WoLF is used to vary the learning rate α_k in the policy update according to the rule in inequality 5. This composition can be essentially thought of as an actor-critic method (Sutton & Barto 1998). Here the Gibbs distribution over the set of parameters is the *actor*, and the gradient-descent Sarsa(0) is the *critic*. Tile-coding provides the necessary parameterization of the state. The WoLF principle is adjusting how the actor changes policies based on response from the critic.

The main detail yet to be explained and where the algorithm is specifically adapted to Goofspiel is in the tile coding. The method of tiling is extremely important to the overall performance of learning as it is a powerful bias on what policies can and will be learned. The major decision to be made is how to represent the state as a vector of numbers and which of these numbers are tiled together. The first decision determines what states are distinguishable, and the second determines how generalization works across distinguishable states. Despite the importance of the tiling we essentially selected what seemed like a reasonable tiling, and used it throughout our results.

We represent a set of cards, either a player's hand or the deck, by five numbers, corresponding to the value of the card that is the minimum, lower quartile, median, upper quartile, and maximum. This provides information as to the general shape of the set, which is what is important in Goofspiel. The other values used in the tiling are the value of the card that is being bid on and the card corresponding to the agent's action. An example of this process in the 13-card game is shown in Table 2. These values are combined together into three tilings. The first tiles together the quartiles describing the players' hands. The second tiles together the quartiles of the deck with the card available and player's action. The last tiles together the quartiles of the opponent's hand with the card available and player's action. The tilings use tile sizes equal to roughly half the number of cards in the game with the number of tilings greater than the tile sizes to distinguish between any integer state values. Finally, these tiles were all then hashed into a table of size one million in order to keep the parameter space manageable. We don't suggest that this is a perfect or even good tiling for this domain, but as we will show the results are still interesting.

Results

One of the difficult and open issues in multiagent reinforcement learning is that of evaluation. Before presenting learning results we first need to look at how one evaluates learning success.

Evaluation

One straightforward evaluation technique is to have two learning algorithms learn against each other and simply examine the expected reward over time. This technique is not useful if one's interested in learning in self-play, where both players use an identical algorithm. In this case with a symmetric zero-sum game like Goofspiel, the expected reward of the two agents is necessarily zero, providing no information.

Another common evaluation criterion is that of convergence. This is true in single-agent learning as well as multiagent learning. One strong motivation for considering this criterion in multiagent domains is the connection of convergence to Nash equilibrium. If algorithms that are guaranteed to converge to optimal policies in stationary environments, converge in a multiagent learning environment, then the resulting joint policy must be a Nash equilibrium of the stochastic game (Bowling & Veloso 2002a).

Although, convergence to an equilibrium is an ideal criterion for small problems, there are a number of reasons why this is unlikely to be possible for large problems. First, optimality in large (even stationary) environments is not generally feasible. This is exactly the motivation for exploring function approximation and policy parameterizations. Second, when we account for the limitations that approximation imposes on a player's policy then equilibria may cease to exist, making convergence of policies impossible (Bowling & Veloso 2002b). Third, policy gradient techniques learn only locally optimal policies. They may converge to policies that are not globally optimal and therefore necessarily not equilibria.

Although convergence to equilibria and therefore convergence in general is not a reasonable criterion we would still expect self-play learning agents to learn something. In this paper we use the evaluation technique used by Littman with Minimax-Q (Littman 1994). We train an agent in self-play, and then freeze its policy, and train a challenger to find that policy's worst-case performance. This challenger is trained using just gradient-descent Sarsa and chooses the action with maximum estimated value with ϵ-greedy exploration. Notice that the possible policies playable by the challenger are the deterministic policies (modulo exploration) playable by the learning algorithm being evaluated. Since Goofspiel

My Hand	1	3	4	5	6	8	11	13
Quartiles	*		*		*	*		*
Opp Hand	4	5	8	9	10	11	12	13
Quartiles	*		*		*	*		*
Deck	1	2	3	5	9	10	11	12
Quartiles	*		*		*	*		*
Card	11							
Action	3							

$$\begin{array}{c} \langle 1,4,6,8,13\rangle, \\ \langle 4,8,10,11,13\rangle, \\ \langle 1,3,9,10,12\rangle, \\ 11,3 \end{array}$$

$$\Downarrow \text{(Tile Coding)}$$

$$\textsc{Tiles} \in \{0,1\}^{10^6}$$

Table 2: An example state-action representation using quartiles to describe the players' hands and the deck. These numbers are then tiled and hashed with the resulting tiles representing a boolean vector of size 10^6.

is a symmetric zero-sum game, we know that the equilibrium policy, if one exists, would have value zero against its challenger. So, this provides some measure of how close the policy is to the equilibrium by examining its value against its challenger.

A second related criterion will also help to understand the performance of the algorithm. Although policy convergence might not be possible, convergence of the expected value of the agents' policies may be possible. Since the real desirability of policy convergence is the convergence of the policy's value, this is in fact often just as good. This is also one of the strengths of the WoLF variable learning rate, as it has been shown to make learning algorithms with cycling policies and expected values converge both in expected value and policy.

Experiments

Throughout our experiments, we examined three different learning algorithms in self-play. The first two did not use the WoLF variable learning rate, and instead followed a static step size. "Fast" used a large step size $\alpha_k = 0.16$; "Slow" used a small step size $\alpha_k = 0.008$; "WoLF" switched between these learning rates based on inequality 5. In all experiments, the value estimation update used a fixed learning rate of $\beta = 0.2$. These rates were not decayed, in order to better isolate the effectiveness apart from appropriate selection of decay schedules. In addition, throughout training and evaluation runs, all agents followed an ϵ-greedy exploration strategy with $\epsilon = 0.05$. The initial policies and values all begin with zero weight vectors, which with a Gibbs distribution corresponds to the random policy, which as we have noted is reasonably good.

In our first experiment we trained the learner in self-play for 40,000 games. After every 5,000 games we stopped the training and trained a challenger against the agent's current policy. The challenger was trained on 10,000 games using Sarsa(0) gradient ascent with the learning rate parameters described above. The two policies, the agent's and its challenger, were then evaluated on 1,000 games to estimate the policy's worst-case expected value. This experiment was repeated thirty times for each algorithm.

The learning results averaged over the thirty runs are shown in Figure 3 for card sizes of 4, 8, and 13. The baseline comparison is with that of the random policy, a very competitive policy for this game. All three learners improve on this policy while training in self-play. The initial dips in the 8 and 13 card games are due to the fact that value estimates are initially very poor making the initial policy gradients not in the direction of increasing the overall value of the policy. It takes a number of training games for the delayed reward of winning cards later to overcome the initial immediate reward of winning cards now. Lastly, notice the affect of the WoLF principle. It consistently outperforms the two static step size learners. This is identical to affects shown in non-approximated stochastic games (Bowling & Veloso 2002a).

The second experiment was to further examine the issue of convergence and the affect of the WoLF principle on the learning process. Instead of examining worst-case performance against some fictitious challenger, we now examine the expected value of the player's policy *while learning* in self-play. Again the algorithm was trained in self-play for 40,000 games. After 50 games both players' policies were frozen and evaluated over 1,000 games to find the expected value to the players at that moment. We ran each algorithm once on just the 13 card game and plotted its expected value over time while learning.

The results are shown in Figure 4. Notice that expected value of all the learning algorithms seem to have some oscillation around zero. We would expect this with identical learners in a symmetric zero-sum game. The point of interest though is how close these oscillations stay to zero over time. The WoLF principle causes the policies to have a more constant expected value with lower amplitude oscillations. This again shows that the WoLF principle continues to have converging affects even in stochastic games with approximation techniques.

Conclusion

We have described a scalable learning algorithm for stochastic games, composed of three reinforcement learning ideas. We showed preliminary results of this algorithm learning in the game Goofspiel. These results demonstrate that the pol-

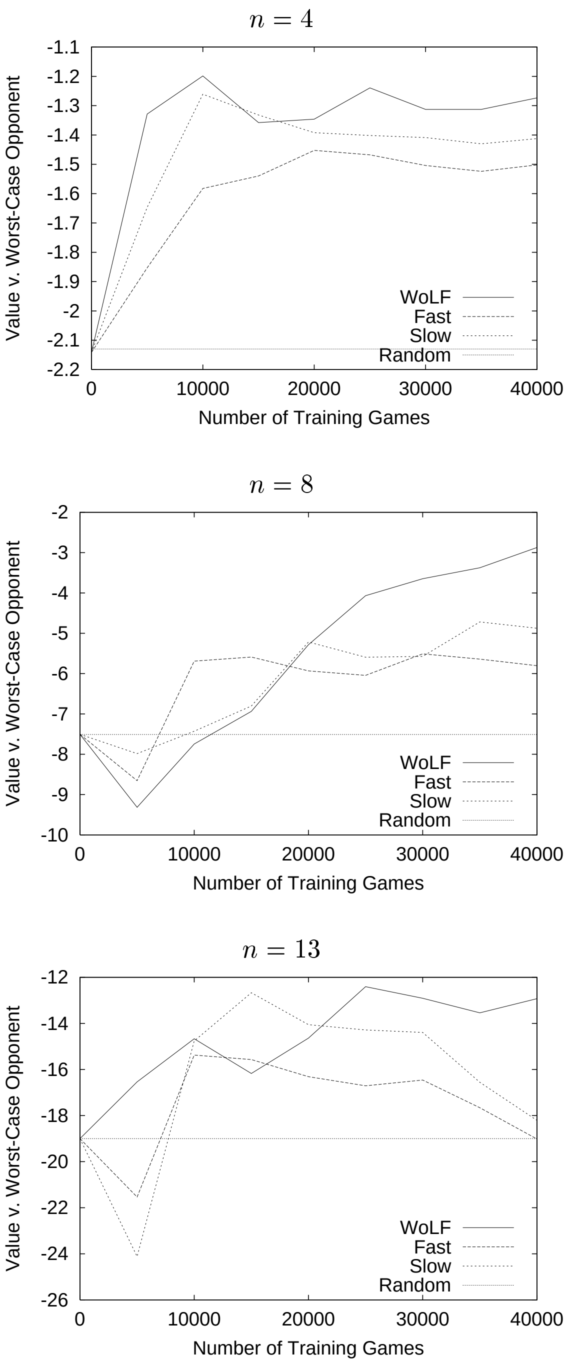

Figure 3: Worst-case expected value of the policy learned in self-play.

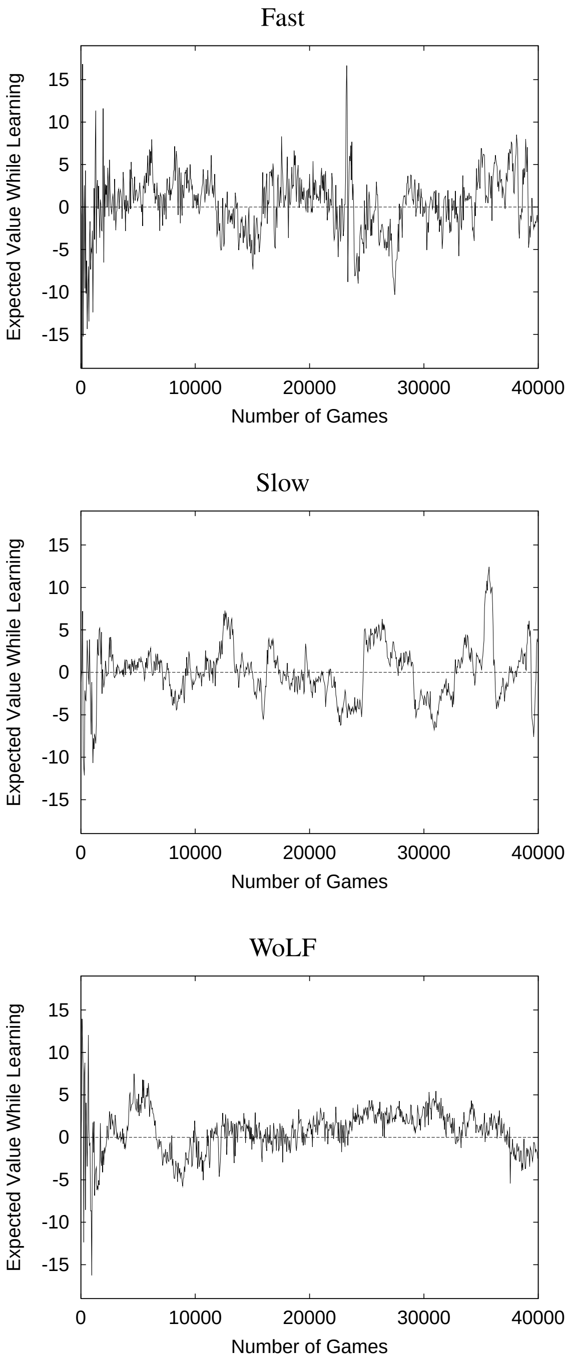

Figure 4: Expected value of the game while learning.

icy gradient approach using an actor-critic model can learn in this domain. In addition, the WoLF principle for encouraging convergence also seems to hold even when using approximation and generalization techniques.

There are a number of directions for future work. Within the game of Goofspiel, it would be interesting to explore alternative ways of tiling the state-action space. This could likely increase the overall performance of the learned policy, but would also examine how generalization might affect the convergence of learning. Might certain generalization techniques retain the existence of equilibrium, and is the equilibrium learnable? Another important direction is to examine these techniques on more domains, with possibly continuous state and action spaces. Also, it would be interesting to vary some of the components of the system. Can we use a different approximator than tile-coding? Do we achieve similar results with different policy gradient techniques (e.g. GPOMDP (Baxter & Bartlett 2000)). These initial results, though, show promise that gradient ascent and the WoLF principle can scale to large state spaces.

References

Baxter, J., and Bartlett, P. L. 2000. Reinforcement learning in POMDP's via direct gradient ascent. In *Proceedings of the Seventeenth International Conference on Machine Learning*, 41–48. Stanford University: Morgan Kaufman.

Bowling, M., and Veloso, M. 2001. Rational and convergent learning in stochastic games. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, 1021–1026.

Bowling, M., and Veloso, M. 2002a. Multiagent learning using a variable learning rate. *Artificial Intelligence*. In Press.

Bowling, M., and Veloso, M. M. 2002b. Existence of multiagent equilibria with limited agents. Technical report CMU-CS-02-104, Computer Science Department, Carnegie Mellon University.

Claus, C., and Boutilier, C. 1998. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*. Menlo Park, CA: AAAI Press.

Fink, A. M. 1964. Equilibrium in a stochastic n-person game. *Journal of Science in Hiroshima University, Series A-I* 28:89–93.

Flood, M. 1985. Interview by Albert Tucker. The Princeton Mathematics Community in the 1930s, Transcript Number 11.

Gordon, G. 2000. Reinforcement learning with function approximation converges to a region. In *Advances in Neural Information Processing Systems 12*. MIT Press.

Greenwald, A., and Hall, K. 2002. Correlated Q-learning. In *Proceedings of the AAAI Spring Symposium Workshop on Collaborative Learning Agents*. In Press.

Hu, J., and Wellman, M. P. 1998. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proceedings of the Fifteenth International Conference on Machine Learning*, 242–250. San Francisco: Morgan Kaufman.

Kuhn, H. W., ed. 1997. *Classics in Game Theory*. Princeton University Press.

Littman, M. L. 1994. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the Eleventh International Conference on Machine Learning*, 157–163. Morgan Kaufman.

Littman, M. 2001. Friend-or-foe Q-learning in general-sum games. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 322–328. Williams College: Morgan Kaufman.

Nash, Jr., J. F. 1950. Equilibrium points in n-person games. *PNAS* 36:48–49. Reprinted in (Kuhn 1997).

Osborne, M. J., and Rubinstein, A. 1994. *A Course in Game Theory*. The MIT Press.

Samuel, A. L. 1967. Some studies in machine learning using the game of checkers. *IBM Journal on Research and Development* 11:601–617.

Shapley, L. S. 1953. Stochastic games. *PNAS* 39:1095–1100. Reprinted in (Kuhn 1997).

Singh, S.; Kearns, M.; and Mansour, Y. 2000. Nash convergence of gradient dynamics in general-sum games. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, 541–548. Morgan Kaufman.

Stone, P., and Sutton, R. 2001. Scaling reinforcement learning toward Robocup soccer. In *Proceedings of the Eighteenth International Conference on Machine Learning*, 537–534. Williams College: Morgan Kaufman.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning*. MIT Press.

Sutton, R. S.; McAllester, D.; Singh, S.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*. MIT Press.

Tesauro, G. J. 1995. Temporal difference learning and TD–Gammon. *Communications of the ACM* 38:48–68.