

Multiagent* Gradient Ascent with Predicted Gradients

Asher Lipson

University of British Columbia
Department of Computer Science
201-2366 Main Mall
Vancouver, B.C.
V6T 1Z4
alipson@cs.ubc.ca

Abstract

Learning in multiagent environments is a difficult task that requires each agent to consider both their own action and those of their opponents. Research has generally concentrated on using game theoretic ideas and applying the reinforcement learning techniques of Q-learning or policy search to solve this problem. However, most approaches make assumptions that limit their applicability. We present a new algorithm, ‘Gradient Ascent with Predicted Gradients’, that unlike current gradient ascent algorithms, does not require the full information of the game to be visible. Experiments and comparisons are shown between this algorithm and algorithms from the literature, IGA, WoLF-IGA and minimax-Q. The results are promising, showing that the algorithm is able to predict the strategy of an opponent and learn a best response to that strategy.

Introduction

In single agent learning environments there are a number of well established algorithms, such as Q-learning, that guarantee the learning of optimal actions in explored states. Extending these single agent approaches to multiagent environments is a difficult task due to the utility of an agent being dependant on both the agent’s own actions as well as the actions of other agents. The notion of a single optimal action is no longer as relevant and this has led to the use of game theoretic ideas that formalise the interaction between multiple agents. Though much work has been done on creating algorithms that combine game theory and learning, the algorithms often make crippling assumptions that limit their applicability.

In this paper, we present a short description of two learning approaches, Q-learning and policy learning/search¹, with a focus on gradient ascent algorithms. We then discuss the limitations of these approaches, including diminished applicability and information visibility requirements. A new gradient ascent algorithm is then presented, ‘Gradient Ascent with Predicted Gradients’ (GAPG), that avoids

*The spelling of this term varies between ‘multi agent’, ‘multi-agent’ and ‘multiagent’, depending on the author. This paper takes the latter spelling.

¹The literature often uses the terms policy learning and policy search interchangeably, though they are applied in different contexts. This paper will refer to policy search to cover both.

the requirement in policy search algorithms that the full information of the game is visible. We perform experimental comparisons on the games Battle of the Sexes, Chicken, Matching Pennies and Prisoner’s Dilemma. GAPG is compared against minimax-Q (Littman, 1994), infinitesimal gradient ascent (Singh, Kearns, & Mansour, 2000) and infinitesimal gradient ascent with the WoLF principle, WoLF-IGA (Bowling & Veloso, 2001a). We expect that in 2-agent games, GAPG is able to model the strategies of opponents and learn a best response to these strategies. Lastly we list a number of ideas for future work.

Multiagent Learning

We define multiagent learning as the process by which more than one agent interacts in an environment with both the environment and other agents, learning what actions to take. Agent i learns what action is best in response to what agent j does. If we use the history of what j has done, then we have j teaching i and vice versa. Each agent can thus be seen as both a learner and a teacher (Shoham, Powers, & Grenager, 2003).

In single agent learning, the environment is stationary and is modelled as a Markov Decision Process (MDP). The agent transitions between states with a probability based on the action taken. Reinforcement learning (RL) is a technique used to learn in MDPs. In RL, an agent is rewarded or punished for actions and the agent tries to maximise the reward it receives in each state. There are two variants to RL: Q-learning and policy search.

Moving from the single agent to multiple agent brings us into game theory and stochastic games. The environment is no longer stationary and multiple agents affect transitions between different states or stage games. Each of these stage games is a one-shot game theory game. Unfortunately, the standard reinforcement learning algorithms do not work ‘out of the box’ for all cases of multiagent learning. The value of a state is no longer dependant only on a single agent’s actions, but rather on all agents’ actions.

The lack of a clear definition for an optimal action has meant that most algorithms are judged on their ability to converge to a Nash equilibrium, where every agent’s action is a best response to the current actions of all other agents. There has recently been discussion as to whether converging to a Nash equilibrium is the correct goal. Shoham, Pow-

ers, & Grenager (2003) argue that the best response depends on the type of agent in the environment. For example, one might want to accumulate rewards, guarantee a minimum value, obtain the maximal expected payoff or converge to an equilibrium, depending on the other types of agents. Similar ideas of learning algorithms depending on the class of agents has been articulated elsewhere in the literature (Bowling & Veloso 2001b, Littman 2001). This paper takes the view that converging to a Nash equilibrium is not always the optimal goal and obtaining rewards may be equally important.

The rest of the paper is structured as followed, firstly Q-learning and policy search are described with examples of algorithms provided. The reader is directed to Fudenberg & Levine (1999) for a general discussion of multiagent learning techniques. While there has been work on learning policies or coordination mechanism efficiently (Brafman & Tenenholz 2003, Bagnell *et al.* 2003), these will not be discussed here.

Q-learning

Q-learning is a technique initially devised for learning in single agent environments. The value of an action in a stage game is encoded in a Q-function, with the agent maximising the function at each stage. The value is dependant on the reward from the current stage and a discounted value of expected future rewards. The choice of actions can also be controlled by a policy defined over the states. This section will discuss algorithms that learn the Q-functions, while the subsequent section will discuss algorithms that learn the policy.

In multiagent versions of Q-learning, one can either take the other agents into account or one can assume that they form part of a stationary environment. If the other agents are taken into account, then each learning agent explicitly models the Q-functions of the other agents. This approach requires a large amount of space and means that all information in the game must be visible, with agents knowing each others payoffs, learning rates and actions. Claus & Boutilier (1997) refer to *joint action learners* (JALs) that keep a belief of what actions the other agents will play. The Hyper-Q algorithm of Tesauro (2003) is very similar to the work by Claus & Boutilier, but with a Bayesian approach. Hyper-Q explicitly models mixed strategies, with policies being greedily chosen based on the probability of the other agent's mixed strategy.

If one assumes that the other agents are part of the environment, then one does not model them and they are ignored. Though this idea appears flawed, it has been proved that if such an algorithm converges, it will converge to a Nash equilibrium. The idea of converging regardless of what the other agents do sounds appealing, but it also runs counter to the idea of multiagent learning and taking into account the existence of other agents. In reality, the agents are trying to learn the best action for a state that is dependant on the actions of other agents, essentially a moving target problem (Vidal & Durfee, 1998).

One of the first multiagent extensions to Q-learning was minimax-Q (Littman, 1994) for 2-agent zero-sum games. The value of a state is defined as the maxmin of the ac-

tions. This provides a guaranteed minimum reward by assuming the opponent will play the action that leads to the worst payoff for the learner. This idea can be extended to general-sum games in order to guarantee a minimum payoff. Hu & Wellman (1998) presented the Nash-Q algorithm in which agents stored a Q-table for each other agent. Actions at each stage game are chosen by solving for a Nash equilibrium. The algorithm is quite restrictive in that convergence is only guaranteed when there is a single unique equilibrium in each stage game, which cannot be predicted during learning (Bowling & Littman, 2003). Both minimax-Q and Nash-Q aim to converge independently of their opponent's actions (Bowling & Veloso, 2000). The algorithms also suffer from only being able to learn pure strategies.

The Friend-or-Foe Q-Learning algorithm (Littman, 1994) is an attempt to converge even in the presence of multiple equilibria. Each agent in the environment is identified as either a friend or a foe to the learning agent and a different learning rule is used accordingly, allowing convergence to a coordination or adversarial equilibrium respectively. The friend portion plays single agent Q-learning, maximising over the action space for all friends, whilst the foe portion plays minimax-Q. Unfortunately this algorithm suffers from a similar limitation to Nash-Q in that convergence is not guaranteed if more than one (or none) of either type of equilibria exists.

Policy learning

There are two forms of policy search, those in which a Q-function is stored and the policy defines the best action for each state and those where the policy space is defined by the probabilities of agents taking an action. These will be referred to as policy hill-climbing (PHC) and gradient ascent (GA) respectively. The 'Win-or-lose-fast' (WoLF) principle (Bowling & Veloso, 2001a; 2001b) can be applied to both PHC and GA algorithms.

WoLF's main feature is the use of a variable learning rate that can be set to a high (fast) or low (slow) rate. The rate changes according to whether the learning agent is currently doing better or worse than an equilibrium strategy. The agent chooses a strategy that forms a Nash equilibrium and if its current strategy receives a higher payoff, then the learning rate is set to the lower value, allowing other agents to 'adapt' their best response. If the agent receives a payoff worse than the equilibrium payoff, then it should learn quickly in order to find a best response and the learning rate is set to the larger value. In cooperative games, one might not want to slow down one's learning while doing well, but rather accelerate it. The use of WoLF can help algorithms converge to a Nash equilibrium, but if that is not the goal, then one may want to use a different learning algorithm. Results of WoLF learning in a variety of different games would be useful to support this claim.

In policy hill-climbing (PHC) algorithms, agents store Q-functions and update the policy that defines a distribution over the possible actions in each state. The agents do not model or take into account the other agents. This is the application of single-agent policy learning in a multiagent setting and does not guarantee convergence. Bowling & Veloso

(2001b) show that the use of WoLF with PHC encourages convergence in stochastic games. WoLF-PHC does not provide any explicit modelling of the other agents, with the authors referring to the variable learning rate as implicitly modelling the other agents. This technique has been shown to be successful in games with large state spaces (Bowling & Veloso, 2002).

Peshkin *et al.* (2000) describe a distributed policy search algorithm for use in partially observable domains, focusing on cooperative games where each agent receives a common reward. Each agent updates their own policy, regardless of the other agents and searches for a local optima in their own space. The algorithm converges to a local equilibrium, that may not be Nash. The agents learn independently of the others and convergence is primarily due to the cooperative game setting.

Gradient ascent algorithms do not store any Q-function, though they require a full information game, including knowing the other agent’s policies (or strategies). The joint strategies of two agents can be seen as being a \mathbb{R}^2 space in which we can search. The probability of agent i taking their first action and the probability of agent j taking their first action define this space. Areas of zero-gradient are equilibria and can be found by following the path of increasing gradient. Gradient ascent algorithms are local and do not converge to a global maximum. The search space is defined as a unit square, but the space itself is not, meaning that gradient ascent can lead off the edge of this square². This requires gradients on the boundary to be projected back into the valid space. GA algorithms explicitly model mixed strategies due to the definition of the space.

The original work on gradient ascent for multiagents was the infinitesimal gradient ascent (IGA) algorithm by Singh, Kearns, & Mansour (2000), shown in table 1. The algorithm guarantees that the agents’ strategies either converge to a Nash equilibrium or their average payoffs converge to the payoffs of a Nash equilibrium. This is a useful guarantee, though it has been referred to as a weaker notion of convergence (Bowling & Veloso, 2001a). If the average payoffs converge then there will be periods where the payoffs are below the average. Incorporating the WoLF principle (WoLF-IGA) guarantees the convergence for both the strategies and payoffs to a Nash equilibrium (Bowling & Veloso, 2001a). However, this is only shown for self-play and WoLF-IGA vs. IGA in 2-agent, 2-action games. WoLF-IGA changes the update rate η in Table 1 to ηl_i^t , for each agent i at time t with variable learning rate l .

Preliminary testing of the Hyper-Q algorithm (Tesauro, 2003) show that it is able to obtain a higher reward than an IGA or PHC algorithm without any WoLF modifications. The AWESOME algorithm of Conitzer & Sandholm (2003) takes a very different approach to the previous algorithms and does not use any Q-learning or policy search technique. The algorithm computes a Nash equilibrium prior to learning and reverts to playing the Nash equilibrium strategy if it detects the other agents playing their corresponding Nash

²The strategies themselves are probabilities limited to the range [0,1], but the space itself is not bounded.

For the following payoff matrix:

r_{ij} is the payoff to the row agent

c_{ij} is the payoff to the column agent

i is the row agent’s action, j the column agent’s action

α is the probability of the row agent playing their first action

β is the probability of the column agent playing their first action

$$\begin{bmatrix} r_{11}, c_{11} & r_{12}, c_{12} \\ r_{21}, c_{21} & r_{22}, c_{22} \end{bmatrix}$$

We can write the value or expected payoff of the strategy (α, β) as:

$$V_r(\alpha, \beta) = r_{11}(\alpha\beta) + r_{22}(1-\alpha)(1-\beta) + r_{12}(1-\beta)\alpha + r_{21}(1-\alpha)\beta$$

$$V_c(\alpha, \beta) = c_{11}(\alpha\beta) + c_{22}(1-\alpha)(1-\beta) + c_{12}(1-\beta)\alpha + c_{21}(1-\alpha)\beta$$

Letting:

$$u = (r_{11} + r_{22}) - (r_{21} + r_{12}) \text{ and} \\ u' = (c_{11} + c_{22}) - (c_{21} + c_{12})$$

We have gradients:

$$\frac{\partial V_r(\alpha, \beta)}{\partial \alpha} = \beta u - (r_{22} - r_{12})$$

$$\frac{\partial V_c(\alpha, \beta)}{\partial \beta} = \alpha u' - (c_{22} - c_{12})$$

giving update rules:

$$\alpha_{t+1} = \alpha_t + \eta \frac{\partial V_r(\alpha_t, \beta_t)}{\partial \alpha}$$

$$\beta_{t+1} = \beta_t + \eta \frac{\partial V_c(\alpha_t, \beta_t)}{\partial \beta}$$

Table 1: The Infinitesimal Gradient Ascent algorithm

equilibrium strategy (convergence in self-play). If AWESOME detects the other agents playing a stationary strategy, then it will play a best response to that strategy. The key assumption is that all agents compute the same Nash equilibrium. This is the same problem as Nash-Q, where if the agents learn different Nash equilibria, there is no convergence. Conitzer & Sandholm state that since the agents use the same algorithm, this is a reasonable assumption. This author disagrees with this statement.

Limitations of current approaches

There are a number of limitations that occur in algorithms for multiagent learning. A partial list is provided below, along with references to work that suffer from them.

Many of the algorithms lose their convergence guarantees in the presence of multiple equilibria (Hu & Wellman 1998, Littman 2001). Convergence should be dependant on the strategies or actions of other agents, rather than independent of them (Peshkin *et al.* 2000, Hu & Wellman 1998, Littman 1994). Strategy convergence should also not be limited to pure strategies, a problem that many of the Q-learning algorithms suffer because actions are chosen that provide the maximum value. This leads to a deterministic pure strategy that can be exploited by other algorithms. All gradient ascent algorithms require the full information of the game,

including payoffs and mixed strategies, to be visible. We want to avoid this. There is some debate as to whether the actions of agents are visible or not. This author takes the view that they are.

A multiagent learning algorithm should take into account the actions of other agents and have the ability to learn a mixed strategy. The goal should be to learn a best response to the strategies of other agents and the current environment. The best response may not always be a Nash equilibrium. In addition, we want to avoid the requirement that the full information of the game be visible.

Learning with unknown information

We now make an attempt to fix one of the limitations of gradient ascent algorithms, the need for the full information of the game to be visible. We apply the ideas of Claus & Boutilier (1997) where each agent maintains beliefs about the strategies of other agents. A similar idea is alluded to by Singh, Kearns, & Mansour (2000) where they state that a stochastic gradient ascent algorithm would be possible if only the previous action of the other agent was visible. We assume that only an agent’s action is visible, not their mixed strategy.

The new algorithm, ‘Gradient Ascent with Predicted Gradients’ (GAPG) is described below for the two-agent, two-action case.

Let α be the probability of agent 1 taking their first action and β be the probability that agent 2 takes their first action. If both agents are playing with GAPG, then we have that agent 1 keeps a belief, $\hat{\beta}$, over agent 2’s mixed strategy and agent 2 keeps a belief, $\hat{\alpha}$, over agent 1’s mixed strategy. The update equations are:

<p>At time $t+1$:</p> <p>For agent 1: $\hat{\beta}_{t+1} = \gamma * \hat{\beta}_t + (1 - \gamma) * \frac{actCount_1}{\# of games}$</p> <p>For agent 2: $\hat{\alpha}_{t+1} = \gamma * \hat{\alpha}_t + (1 - \gamma) * \frac{actCount_2}{\# of games}$</p>
--

Table 2: Belief update equations for GAPG

Where γ is the decreasing update rate, $actCount_i$ is a count of how many times the first action has been played by agent i and “# of games” is the total number of games that have been played. After each stage game, $actCount_i$ is incremented if agent i played their first action. This update is used instead of the simpler $\hat{\beta}_{t+1} = \frac{actCount_1}{\# of games}$, as it allows us to set $\hat{\beta}_0$ based on any knowledge that we have. This is essentially putting a prior on the predicted strategy of the agent. The update equations also allow us to control the effect of observed actions through our update rate. The form of the update equations means that an agent must view a large amount of evidence that the opponent has changed their strategy for it to affect the beliefs of the agent. If an agent has played 5000 games with a pure strategy and then plays a pure strat-

egy with another action, it will take a large number of games of this new strategy before the the beliefs reflect this.

We modify the gradient ascent updates of Singh, Kearns, & Mansour (2000) (see Table 1) to move with step size η in the direction of the believed gradient. GAPG can also be run with WoLF-IGA, in which case η is replaced by a variable update, η_t^i .

$$\alpha_{t+1} = \alpha_t + \eta \frac{\partial V_r(\alpha_t, \hat{\beta}_t)}{\partial \alpha}$$

$$\beta_{t+1} = \beta_t + \eta \frac{\partial V_c(\hat{\alpha}_t, \beta_t)}{\partial \beta}$$

Experiments

Four algorithms are used in testing, minimax-Q, Infinitesimal Gradient Ascent (IGA), IGA with WoLF (IGA-WoLF) and GAPG with WoLF-IGA. Each algorithm is run against all the other algorithms (including itself) for a total of ten tests. Each test consists of a game being played 50000 times and each test is run ten times, with the results averaged between them. Two games were tested thoroughly, the zero-sum Matching Pennies and Prisoner’s Dilemma, shown in Figure 1. We also provide a small set of results for the games Chicken and Battle of the Sexes. The payoff matrices are generated using the GAMUT game generator (Nudelman *et al.*, 2004). Each algorithm begins with the probability of choosing the first action set to 0.5. Minimax-Q is given a high exploration probability in an attempt to prevent a deterministic strategy from being played.

$\begin{bmatrix} 1, & -1 & -1, & 1 \\ -1, & 1 & 1, & -1 \end{bmatrix}$	$\begin{bmatrix} 1, & -1 & -4, & 0 \\ 0, & -4 & -3, & -3 \end{bmatrix}$
Matching Pennies	Prisoner’s Dilemma

Figure 1: Matching Pennies and Prisoner’s Dilemma

For the gradient ascent algorithms, if the gradient step goes outside of the unit square, then the strategy is set to the boundary point. The minimax-Q algorithm plays the pure strategy that returns the maxmin value of the payoff matrix, which guarantees a minimum payoff to the agent.

The parameters are set as follows: for minimax-Q, the exploration rate is set to 0.8, the discount factor of future states to 0.9, the learning rate to 0.1 and the learning rate decay to 0.1. For IGA, the step size is set to 0.16 with the decay set to 0.999. In WoLF-IGA, the slow learning rate is 0.008, the fast learning rate to 0.16 and the decay rate to 0.9999954. For GAPG, we use the same learning rates as WoLF for the learning algorithm, while for the belief updates we use $\gamma = 0.16$ and reduce this by 0.9999954 each step.

The majority of figures concentrate on results involving GAPG, with Figure 2 providing an overview of the number of games won by each algorithm in the four different games. This figure shows what percentage of stage games each algorithm wins against the other for each game. The graph shows how varied the results are for different games. For the majority of games, GAPG wins at least half the games

played against the algorithms. The exception being against WoLF-IGA in Battle of the Sexes and against minimax-Q in Chicken.

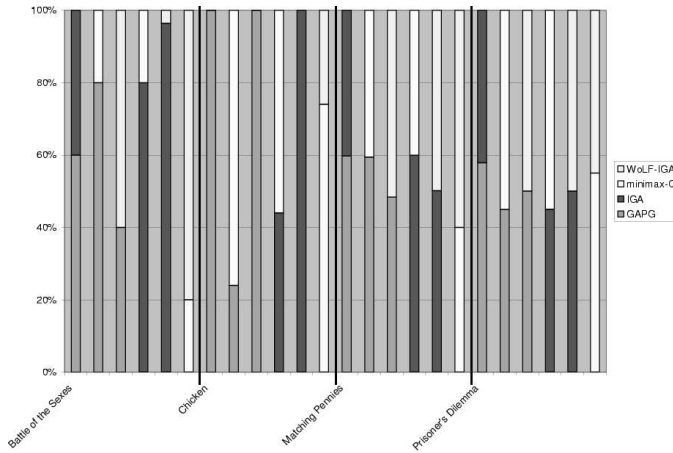


Figure 2: Comparison of algorithms. Height of each bar refers to the percentage of games in which the algorithm received a higher payoff than the other.

The primary motivation for using GAPG is its use of predicted strategies and this is shown in Figure 3. The difference in prediction against minimax-Q is due to the high exploration rate of minimax-Q, meaning that it often chooses its action randomly. Against IGA the strategy is predicted exactly (the two plots are indistinguishable in the figure), with similar results for GAPG predicting against itself and against WoLF-IGA. In other comparisons, GAPG effectively tracks the strategies of an opponent even if the opponent is constantly changing their strategy.

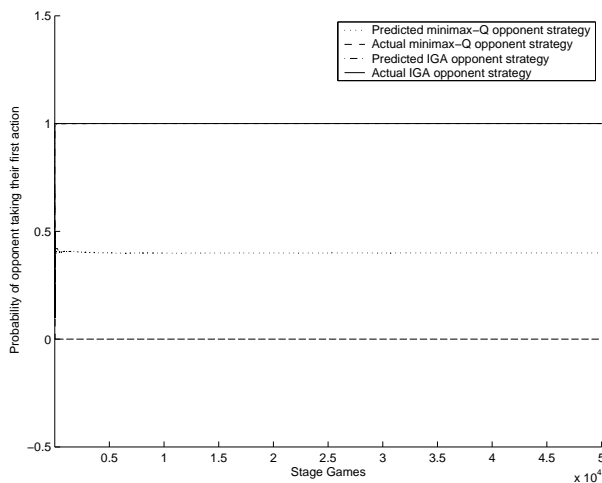


Figure 3: Predicated vs Actual strategy in Prisoner's Dilemma

Figure 4 shows the expected value of the GAPG algorithm against the other algorithms in Matching Pennies. If a Nash

equilibrium strategy is played, the expected value of the game is 0. When playing against minimax-Q, the expected value constantly oscillates near 1 (the graph makes it somewhat difficult to view this) and this shows how minimax-Q plays a pure strategy and GAPG learns a best response strategy to this. Playing against itself, the oscillation can be interpreted as the learner changes its strategy, then after a while, the opponent changes theirs and this sequence is repeated. This is due to GAPG adapting its strategy based on what it believes the opponents is playing. The oscillation occurs around the Nash equilibrium value. The positive expected value against IGA shows that GAPG is able to take a small advantage of the strategy played by IGA. Against WoLF-IGA, the expected value is around zero, showing how both algorithms learn a Nash equilibrium strategy.

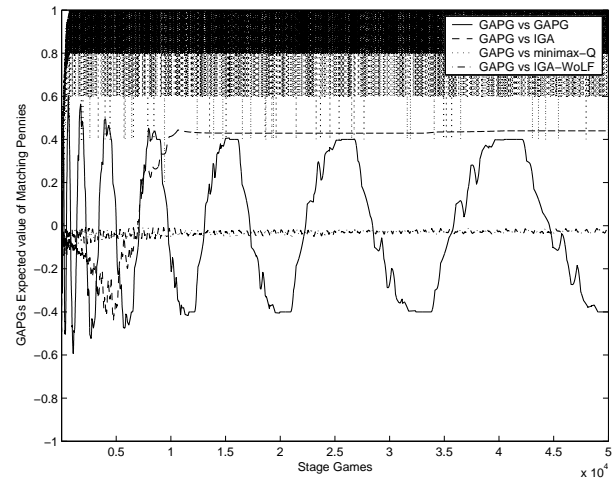


Figure 4: Expected value of Matching Pennies

One of the goals of the learning algorithm is to learn a best response to the strategy of the other agents. Figure 5 shows the strategy learned by GAPG against the other algorithms in Matching Pennies. Against IGA and IGA-WoLF, the Nash equilibrium strategy of 0.5 is learned, while against itself, it oscillates around the equilibrium strategy. However, it does not show signs of convergence to the Nash equilibrium strategy. The strategy against minimax-Q is very different due to minimax-Q playing a pure strategy and GAPG exploiting this for higher reward.

Looking at Figures 4 and 5 one can see the expected value changing as GAPG's strategy changes.

Concluding remarks

We have presented a new algorithm, Gradient Ascent with Predicted Gradients, that uses the predicted strategy of an opponent to learn with a gradient ascent algorithm. Preliminary results of this algorithm against three algorithms in a repeated game setting, show promising results. GAPG is able to effectively predict the strategy of opponents, often doing so exactly. In the learning of a best response, the algorithm learns a strategy that returns the Nash equilibrium

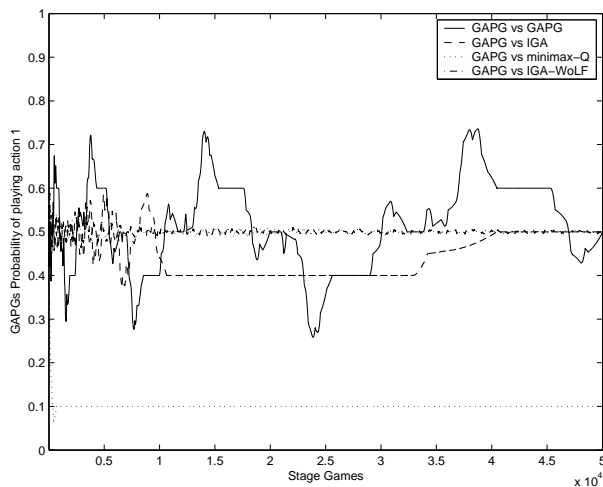


Figure 5: GAPG strategy in Matching Pennies

value of the game or that exploits the strategy of an opponent. However, as was shown in Figure 2, the results can vary between different games.

Future experimentation will test GAPG in games with more than 2 agents and 2 actions per agent. At the time of this report, apart from results in Bowling & Veloso (2001b) and claims made in Conitzer & Sandholm (2003), there has been very little testing of these larger sized games. Other possible future goals include making gradient ascent a globally optimal technique, possibly through preprocessing of the strategy space and applying gradient ascent to stochastic games where different strategies would be used in different stages.

Acknowledgements

Many thanks to Kevin Leyton-Brown for his helpful comments, feedback and for providing access to the GAMUT game generator. Thanks to Jennifer Wortman for her help with the GAMUT game generator. Thanks to Sarah Manske for her comments and suggestions on an earlier version of the paper.

References

Bagnell, J.; Kakade, S.; Ng, A.; and Schneider, J. 2003. Policy search by dynamic programming. In *NIPS '03, Neural Information Processing 16*.

Bowling, M., and Littman, M. 2003. Multiagent learning: A game theoretic perspective. Slides for Tutorial at IJCAI 2003, 18th Int. Joint Conf. on AI.

Bowling, M., and Veloso, M. 2000. An analysis of stochastic game theory for multiagent reinforcement learning. CMU-CS 00-165, Carnegie Mellon University.

Bowling, M., and Veloso, M. 2001a. Convergence of gradient dynamics with a variable learning rate. In *ICML '01, 18th Int. Conf. on Machine Learning*.

Bowling, M., and Veloso, M. 2001b. Rational and convergent learning in stochastic games. In *IJCAI '01, Int. Joint Conf. on Artificial Intelligence*.

Bowling, M., and Veloso, M. 2002. Scalable learning in stochastic games. In *AAAI Workshop on Game Theoretic and Decision Theoretic Agents*.

Brafman, R., and Tennenholtz, M. 2003. Learning to coordinate efficiently: A model-based approach. *Journal of Artificial Intelligence Research* 19:11 – 23.

Claus, C., and Boutilier, C. 1997. The dynamics of reinforcement learning in cooperative multiagent systems. In *AAAI '97, American Association of Artificial Intelligence Workshop on Multiagent Learning*, 746 – 752.

Conitzer, V., and Sandholm, T. 2003. AWESOME: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. In *ICML '03, 20th Int. Conf. on Machine Learning*, 83–90.

Fudenberg, D., and Levine, D. 1999. *The Theory of Learning in Games*. Cambridge, Massachusetts: MIT Press.

Hu, J., and Wellman, M. 1998. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *ICML '98, 15th Int. Conf. on Machine Learning*, 242 – 250.

Littman, M. 1994. Markov games as a framework for multi-agent reinforcement learning. In *ICML '94, 11th Int. Conf. on Machine Learning*, 157 – 163.

Littman, M. 2001. Friend-or-foe Q-learning in general-sum games. In *ICML '01, 18th Int. Conf. on Machine Learning*, 322 – 328.

Nudelman, E.; Wortman, J.; Leyton-Brown, K.; and Shoham, Y. 2004. Run the GAMUT: A comprehensive approach to evaluating game-theoretic algorithms. In *AAMAS '04, 3rd Int. Joint Conf. on Autonomous Agents and Multi Agent Systems*.

Peshkin, L.; Kim, K.; Meuleau, N.; and Kaelbling, L. 2000. Learning to cooperate via policy search. In *UAI '00, 16th Conf. on Uncertainty in Artificial Intelligence*.

Shoham, Y.; Powers, R.; and Grenager, T. 2003. Multi-agent reinforcement learning: a critical survey. Unpublished survey. <http://robotics.stanford.edu/~shoham/>.

Singh, S.; Kearns, M.; and Mansour, Y. 2000. Nash convergence of gradient dynamics in general-sum games. In *UAI '00, 16th Conf. on Uncertainty in Artificial Intelligence*.

Tesauro, G. 2003. Extending q-learning to general adaptive multi-agent systems. In *NIPS '03, Advances in Neural Information Processing Systems 16*.

Vidal, J., and Durfee, E. 1998. The moving target function problem in multi-agent learning. In *ICMAS '98, 3rd Int. Conf. on Multi-Agent Systems*.