Simultaneous Adversarial Multi-Robot Learning

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

Abstract

Multi-robot learning faces all of the challenges of robot learning with all of the challenges of multiagent learning. There has been a great deal of recent research on multiagent reinforcement learning in stochastic games, which is the intuitive extension of MDPs to multiple agents. This recent work, although general, has only been applied to small games with at most hundreds of states. On the other hand robot tasks have continuous, and often complex, state and action spaces. Robot learning tasks demand approximation and generalization techniques, which have only received extensive attention in single-agent learning. In this paper we introduce GraWoLF, a general-purpose, scalable, multiagent learning algorithm. It combines gradient-based policy learning techniques with the WoLF ("Win or Learn Fast") variable learning rate. We apply this algorithm to an adversarial multi-robot task with simultaneous learning. We show results of learning both in simulation and on the real robots. These results demonstrate that GraWoLF can learn successful policies, overcoming the many challenges in multi-robot learning.

1 Introduction

Multi-robot learning is the challenge of learning to act in an environment containing other robots. These other robots, though, have their own goals and may be learning as well. Other adapting robots make the environment no longer stationary, violating the Markov property that traditional single-agent behavior learning relies upon. Multi-robot learning combines all of these multiagent learning challenges with the problems of learning in robots, such as continuous state and action spaces and minimal training data.

A great deal of recent work on multiagent learning has looked at the problem of learning in stochastic games [Littman, 1994; Singh *et al.*, 2000; Bowling and Veloso, 2002a; Greenwald and Hall, 2002]. Stochastic games are a natural extension of Markov decision processes (MDPs) to multiple agents and have been well studied in the field of game theory. The traditional solution concept for the problem of simultaneously finding optimal policies is that of Nash equilibria. An equilibrium is simply a policy for all of 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 equilibrium.

Multiagent learning in stochastic games, thus far, has only been applied to small games with enumerable state and action spaces. Robot learning tasks though have continuous state and action spaces, and typically with more than just a couple dimensions. Discretizations of this space into an enumerable state set also do not typically perform well. In addition, data is considerably more costly to gather, and millions of training runs to learn policies is not feasible. Typical solutions to this robot learning problem is to use approximation to make the learning tractable and generalization for more efficient use of training experience.

In this paper we introduce a new algorithm, GraWoLF[1], that combines approximation and generalization techniques with the WoLF multiagent learning technique. We show empirical results of applying this algorithm to a problem of simultaneous learning in an adversarial robot task. In Section 2, we give a brief overview of key concepts from multiagent learning along with the the formal model of stochastic games. In Section 3, we describe a particular adversarial robot task and its challenges for learning. In Section 4, we present the main components of GraWoLF: policy gradient ascent and the WoLF variable learning rate. In Section 5, we present experimental results of applying this algorithm to our adversarial robot task, and then conclude.

2 Multiagent Learning

Multiagent learning has focused on the game theoretic framework of 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

[1] GraWoLF is short for "Gradient-based Win or Learn Fast", and the 'a' has the same sound as in "gradient".

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. 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 children's game rock-paper-scissors. If one player were to play any action deterministically, the other player could win every time by selecting the action that defeats it. This fact requires the consideration of mixed strategies and stochastic policies. A stochastic policy, $\pi : \mathcal{S} \rightarrow PD(\mathcal{A}_i)$, is a function that maps states to mixed strategies, which are probability distributions over the player's actions. We later show that stochastic policies are also useful for gradient-based learning techniques.

Nash Equilibria. Even with the concept of stochastic policies there are still no optimal policies that are independent of the other players' policies. We can, though, define a notion of best-response. A policy is a *best-response* to the other players' policies if it is optimal given their policies. 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 and stochastic games have such an equilibrium, possibly having multiple equilibria. Zero-sum, two-player games, like the adversarial task explored in this paper, have a single Nash equilibrium.[2]

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, e.g., [Littman, 1994; Greenwald and 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, e.g., [Claus and Boutilier, 1998; Singh *et al.*, 2000; Bowling and Veloso, 2002a]. These learners explicitly optimize their reward with respect to the other players' possibly 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 and Veloso, 2002a].

[2]There can actually be multiple equilibria, but they all have equal payoffs and are interchangeable.

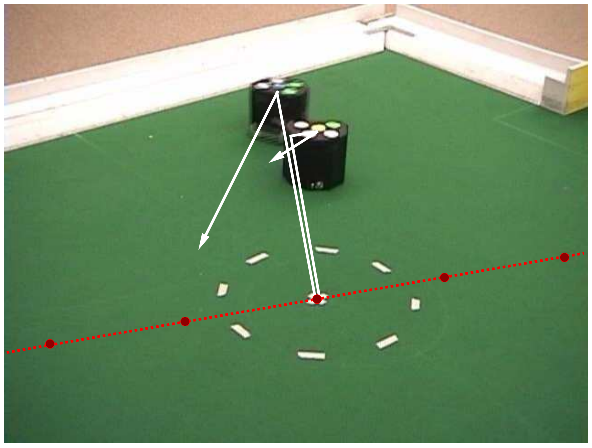

Figure 1: An adversarial robot task. The top robot is trying to get inside the circle while the bottom robot is trying to stop it. The lines show the state and action representation, which is described in Section 4.3.

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 an adversarial robot task, which can be thought of as a continuous state stochastic game. Specifically, we build on the idea of best-response learners using gradient techniques [Singh *et al.*, 2000; Bowling and Veloso, 2002a]. We first describe our robot task and then describe our algorithm and results.

3 An Adversarial Robot Task

Consider the adversarial robot task shown in Figure 1. The robot at the top of the figure, the attacker, is trying to reach the circle in the center of the field, while the robot closer to the circle, the defender, is trying to prevent this from happening. If the attacker reaches the circle, it receives a reward of one and the defender receives a reward of negative one. These are the only rewards in the task. When the attacker reaches the circle or ten seconds elapses, the trial is over and the robots reset to their initial positions, where the attacker is a meter from the circle and the defender half-way between. The robots simultaneously learn in this environment each seeking to maximize its own discounted future reward. For all of our experiments the discount factor used was 0.8 for each full second of delay.

The robots themselves are part of the CMDragons robot soccer team, which competes in the RoboCup small-size league. There are two details about the robot platform that are relevant to this learning task. First, the learning algorithm itself is situated within a large and complex architecture of existing capability. The team employs a global vision system mounted over the field. This input is processed by an elaborate tracking module that provides accurate positions and velocities of the robots. These positions comprise the input state for the learning algorithm. The team also consists of robust modules for obstacle avoidance and motion control. The actions for the learning algorithm then involve providing target points for the obstacle avoidance module. Situating the learning within the context of this larger architecture focuses the learning. Rather than having the robot learn to solve well understood problems like path planning or object tracking, the learning is directed at the heart of the problem, the multi-robot interaction.

Second, the system control loop that is partially described above has inherent, though small, latency. Specifically, after an observed change in the world 100ms will elapse before the robot's response is executed. This latency is overcome for each robot's own position and velocity by predicting through this latency using knowledge of the past, but not yet executed, actions. Since the actions of the opponent robots are not known, this prediction is not possible for other robots. Latency effectively adds an element of partial observability to the problem, since the agents do not have the complete state of the world, and in fact have separate views of this state. Notice, that this also adds a tactical element to successful policies. A robot can "fake" the opponent robot by changing its direction suddenly, knowing the other robot will not be able to respond to this change for a full latency period.

This task involves numerous challenges for existing multi-agent learning techniques. These challenges include continuous state and action spaces, elements of partial observability due to system latency, and violation of the Markov assumption since many of the system components have memory, e.g., the tracking and the obstacle avoidance modules. In fact, these limitations may even make equilibria cease to exist altogether [Bowling and Veloso, 2002b]. This is a further reason for exploring best-response learning techniques, which don't directly seek to learn an equilibrium. We now present GraWoLF, a best-response learning algorithm that can handle the challenges of multi-robot learning.

4 GraWoLF

GraWoLF, or Gradient-based WoLF, combines two key ideas from reinforcement learning: policy gradient learning and the WoLF variable learning rate. Policy gradient learning is a technique to handle intractable or continuous state spaces. WoLF is a multiagent learning technique that encourages best-response learning algorithms to converge in situations of simultaneous learning. We briefly describe these techniques and how they are combined.

4.1 Policy Gradient Ascent

We use the policy gradient technique presented by Sutton and colleagues [2000]. Specifically, we define a policy as a Gibbs distribution over a linear combination of features of a candidate state and action pair. Let θ be a vector of the policy's parameters and let ϕ_{sa} be an identically sized feature vector for state s and action a, then the Gibbs distribution defines a stochastic policy according to,

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

Sutton and colleagues' main result was a convergence proof for the following policy iteration rule that updates a policy's parameters in a Gibbs distribution according to,

$$\theta_{k+1} = \theta_k + \delta_k \sum_s d^{\pi_k}(s) \sum_a \phi_{sa} \cdot \pi_{\theta_k}(s,a) f_{\mathbf{w}_k}(s,a) \quad (1)$$

$f_{\mathbf{w}_k}(s,a)$ is an independent approximation of $Q^{\pi_{\theta_k}}(s,a)$ with parameters $\mathbf{w}$, which is the expected value of taking action a from state s and then following the policy π_{θ_k}. For the Gibbs distribution, Sutton and colleagues showed that for convergence this approximation should have the following form,

$$f_{\mathbf{w}_k}(s,a) \quad = \quad \mathbf{w}_k \cdot \left[\phi_{sa} - \sum_b \pi_{\theta_k}(s,b)\phi_{sb}\right]. \quad (2)$$

As they point out, this amounts to $f_{\mathbf{w}}$ 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 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 summation over states can be removed by updating proportionately to that state's contribution to the policy's overall value. Since we are visiting states on-policy then we only need to weight later states by the discount factor to account for their smaller contribution. If t time has passed since the trial start, this turns Equation 1 into,

$$\theta_{k+1} \quad = \quad \theta_k + \gamma^t \delta_k \sum_a \phi_{sa} \cdot \pi_{\theta_k}(s,a) f_{\mathbf{w}_k}(s,a). \quad (3)$$

Since the whole algorithm is on-line, we do the policy improvement step (updating θ) simultaneously with the value estimation step (updating $\mathbf{w}$). We do value estimation using gradient-descent Sarsa(λ) [Sutton and Barto, 1998] over the same feature space as the policy. This requires maintaining an eligibility trace vector, $\mathbf{e}$. The update rule is then, if at time k the system is in state s and takes action a transitioning to state s' in time t and then taking action a', we update the trace and the weight vector using,

$$\mathbf{e}_{k+1} \quad = \quad \lambda\gamma^t \mathbf{e}_k + \phi_{sa}, \quad (4)$$

$$\mathbf{w}_{k+1} \quad = \quad \mathbf{w}_k + \mathbf{e}_{k+1}\alpha_k \begin{pmatrix} r + \gamma^t Q_{\mathbf{w}_k}(s',a') \\ -Q_{\mathbf{w}_k}(s,a) \end{pmatrix}, \quad (5)$$

where λ is the Sarsa parameter and α_k is an appropriately decayed learning rate. The addition of raising γ to the power t allows for actions to take differing amounts of time to execute, as in semi-Markov decision processes [Sutton and Barto, 1998].

The policy improvement step then uses Equation 3 where s is the state of the system at time k and the action-value estimates from Sarsa, $Q_{\mathbf{w}_k}$, are used to compute the advantage term. We then get,

$$f_{\mathbf{w}_k}(s,a) \quad = \quad Q_{\mathbf{w}_k}(s,a) - \sum_a \pi_{\theta_k}(s,a) Q_{\mathbf{w}_k}(s,a). \quad (6)$$

This forms the crux of GraWoLF. What remains is the selection of the learning rate, δ_k. This is where the WoLF variable learning rate is used.

4.2 Win or Learn Fast

WoLF ("Win or Learn Fast") is a method by Bowling and Veloso [2002a] for changing the learning rate to encourage convergence in a multiagent reinforcement learning scenario. Notice that the policy gradient ascent algorithm above 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 games with enumerated state spaces) that it encourages convergence in algorithms that don't otherwise converge. 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 the others' 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 $V^{\bar{\pi}}(s)$. So, if $V^{\pi}(s) > V^{\bar{\pi}}(s)$ then the algorithm is considered winning, otherwise it is losing. With the policy gradient technique above, we cannot easily compute the average policy. Instead we examine 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,

$$\sum_a \pi_{\theta_k}(s,a) Q_{\mathbf{w}_k}(s,a) > \sum_a \pi_{\bar{\theta}_k}(s,a) Q_{\mathbf{w}_k}(s,a). \quad (7)$$

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

4.3 Our Task

Returning to the robot task shown in Figure 1, in order to apply GraWoLF we need to select a policy parameterization. Specifically we need to find a mapping from the continuous space of states and actions to a useful feature vector, i.e., to define ϕ_{sa} for a given s and a. The filtered positions and velocities of the two robots form the available state information, and the available actions are points on the field for navigation. By observing that the radial angle of the attacker with respect to the circle is not relevant to the task we arrive at a seven dimensional input space. These seven dimensions are shown by the white overlaid lines in Figure 1.

We chose to use tile coding [Sutton and Barto, 1998], also known as CMACS, to construct our feature vector. Tile coding uses a number of large offset tilings to allow for both a fine discretization and large amount of generalization. We use 16 tiles whose size was 800mm in distance dimensions and 1600mm/s in velocity dimensions. We simplify the action space by requiring the attacker to select its navigation point along a perpendicular line through the circle's center. This is shown by the dark overlaid line in Figure 1. This line, whose length is three times the distance of the attacker to the circle, is then discretized into seven points evenly distributed. The defender uses the same points for its actions but then navigates to the point that bisects the line from the attacker to the selected action point. The robot's action is also included in the tiling as an eighth dimension with a tile size for that dimension of the entire line, so actions are distinguishable but there is also generalization. Agents select actions every tenth of a second, or after every three frames, unless the feature vector has not changed over that time. This keeps the robots from oscillating too much during the initial parts of learning.

5 Results

Before presenting results on applying GraWoLF to this problem we first consider some issues related to evaluation.

5.1 Evaluation

Evaluation of multi-robot learning algorithms present a number of challenges. The first is the problem of data gathering on a real robot platform. Learning often requires far more trials than is practical to execute in the physical world. We believe and demonstrate that the GraWoLF technique is practical for the time constraints of real robots. Yet, from a research standpoint, we want thorough and statistically significant evaluation, which requires far more trials than just those used for learning. We solve this problem by using both a simulator of our robot team as well as the robots themselves to show that the approach is both practical for robots while still providing an extensive analysis of the results.

The second challenge is the problem of evaluating the success of simultaneous learning. The traditional single-agent evaluation that shows performance over time converging to some optimal value is not applicable. Multiagent domains have no single optimal value independent of the other agents' behavior. The optimal value is changing over time as the other agents also learn. This is especially true of self-play experiments where both agents are using an identical learning algorithm, and any performance success by one agent is necessarily a performance failure for the other.

On the other hand we would still want learning robots, even in self-play, to improve their policy over time. In this paper, our main evaluation compares the performance of the learned policy with the the performance of the initial policy before learning. The initial policy is a random selection of the available actions, and by design of the available actions is actually a fairly capable policy for both agents. We also use the evaluation technique of challengers, which was first examined by

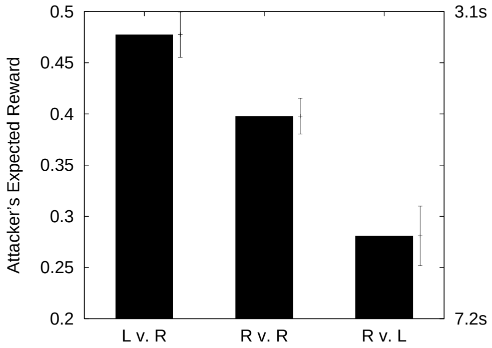

Figure 2: The value of learned policies compared to a random opponent in simulation. Lines to the right of the bars show standard deviations.

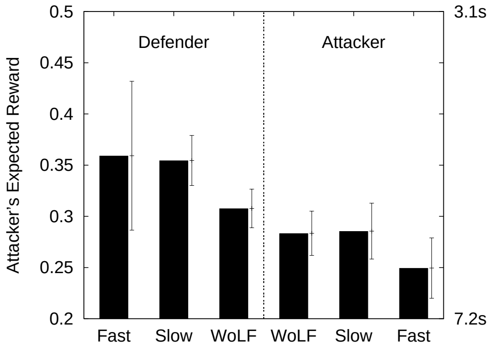

Figure 3: The value of learned policies playing their challengers in simulation. Lines to the right of the bars show standard deviations.

Littman [1994]. This technique trains a worst-case opponent, or challenger, for a particular policy to see the generality of the learned policy. In this paper we present challenger results demonstrating that the learned policies are indeed robust, and that the WoLF variable learning rate plays a critical role in keeping the learning away from easily exploited policies.

5.2 Experiments

In all of the performance graphs in this section, the y-axis shows the attacker's expected discounted reward, which roughly corresponds to the expected time it takes for the attacker to reach the circle. On the right of the graph the range is shown in seconds. All measurements of expected discounted reward are gathered empirically over the course of 1000 trials. All training occurred over 4000 trails, and takes approximately 6-7 hours of training time on the robot platform or in simulation. Unless otherwise noted, the training was always done with a simultaneously learning opponent, both using GraWoLF. The experiments in simulation were repeated nine times and the averages are shown in the graphs. We first examine the performance of the policies learned in simulation, and then examine the performance of learning on the robot.

Figure 2 shows the results of various learned policies when playing against an opponent following the random policy, which was the starting point for learning. The middle bar, "R v. R", corresponds to the expected value of both players following the random policy. "L v. R" corresponds to the value of the attacker following the policy learned in self-play against a random defender. "R v. L" corresponds to the value of a random attacker against the defender policy learned in self-play.

Notice that, as was desired, learning does improve performance over the starting policy. The learned attacker policy against random does far better than the random attacker policy against the learned defender. These experiments demonstrate that GraWoLF improves considerably on its starting policy. The next experiment explores how robust the learned policy is and the effect of the WoLF component.

Figure 3 shows results of challenger experiments. Policies are trained using simultaneous learning. The policy is then fixed and a challenger policy is trained for 4000 trails, to specifically measure the policy's worst-case performance. The better the worst-case performance, the less exploitable and more robust the policy is to unknown opponents. We trained a challenger against the policies learned after 3000, 3500, and 4000 trials, averaging together the results. We used this experiment to investigate the robustness of the learned policy and the affect of the WoLF variable learning rate on the GraWoLF algorithm. The left side of the graph shows the worst-case performance of learned defender policies, while the right side shows the worst-case performance for attacker policies. "WoLF" corresponds to the described GraWoLF algorithm, "Slow" does not use a variable learning rate but rather always uses the WoLF's winning rate, while "Fast" always uses its losing rate.

First, notice that the distance between the worst-case performance of the defender and the worst-case performance of the attacker (the third and fourth column of Table 3, respectively) is quite small. This demonstrates that the learned policies are quite close to the equilibrium policy, if one exists. This also means that the learned policies are robust and difficult to exploit.

Second, notice that the WoLF learned defender policy performs better against its challenger, i.e., keeps its challenger to a lower reward, than either of the two learning rates it switches between. For the attacker, the learned policy performs better against its challenger than the fast learning rate, but is not significantly different than the slow learning rate. There are a couple of possible reasons for this. One explanation is due to the initialization of values. Since all values were intialized to zero for both sides, this amounts to an optimistic initialization for the defender, and a pessimistic one for the attacker (as all rewards are non-negative for the attacker.) This may mean the attacker considers itself winning far more often than the defender, causing the slower learning rate to be employed most of the time. There is evidence to this effect when examining the percentage of updates that use δ_w versus δ_l. During training, the attacker used the slower, winning rate for 92.3% of its updates, while the defender used the winning rate for only 84.2%. The effect of value initialization on GraWoLF is an interesting top to be explored further. Overall, although the results are certainly not as dramatic as the unapproximated results [Bowling and Veloso, 2002a], WoLF still

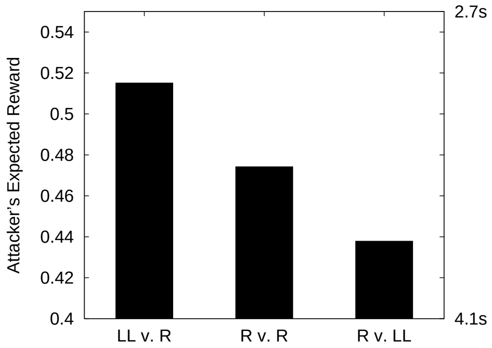

Figure 4: The value of learned policies playing the random policy on the real robots. Training was done with the first half of the trials in simulation and the last half on the robots.

seems to be providing a converging effect.

Finally, we examine results of using GraWoLF on the real robots. We took policies that were trained for 2000 trials in simulation and then did an additional 2000 trials of training on the robots. We evaluated the resulting policies against the random policy as we did with the simulator results in Figure 2. These results are shown in Figure 4

In the real robots the attacker is nearly impossible to keep from reaching the circle. The defender can at best only slow down its progress, and this is even true for a random attacker policy. This can be seen in Figure 4 by the much higher rewards as compared to the simulator results. Despite this, these results are qualitatively identical to the results produced in simulation. Simultaneous learning improves the performance for both the attacker and defender over their initial policies.

Finally, as an adversarial environment we would expect good policies to be stochastic in nature. This is true of all of the learned policies in both simulation and on the robots. For example, the most probable action in any given state has probability on average around 40% in the attacker's learned policy, and 70% in the defender's. In some states, though, the learned policies are nearly deterministic. So, the learned policies are indeed stochastic to avoid exploitation, while still learning optimal actions in states that don't depend on the other agent's behavior.

6 Conclusion

We have introduced, GraWoLF, a general-purpose multiagent learning algorithm capable of learning robot tasks in multi-robot, and even adversarial, environments. We showed results of this algorithm learning in one particular adversarial robot task, both in simulation and from actual robot experience. These results both demonstrated the effectiveness of the policy gradient learner, and the importance of the WoLF variable learning rate. It should be noted that the use of Sutton and colleagues' particular policy gradient formulation is not critical. It would be interesting to combine WoLF with other policy gradient techniques, such as [Williams, 1992; Baxter and Bartlett, 2000].

Acknowledgements

This research was sponsored by Grants F30602-00-2-0549 and DABT63-99-1-0013. The content of this publication does not necessarily reflect the position of the funding agencies and no official endorsement should be inferred. The authors also thank Brett Browning and James Bruce for the development of the CMDragons'02 robots used in this work.

References

[Baxter and Bartlett, 2000] Johnathan Baxter and Peter L. Bartlett. Reinforcement learning in POMDP's via direct gradient ascent. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 41–48, Stanford University, June 2000. Morgan Kaufman.

[Bowling and Veloso, 2002a] Michael Bowling and Manuela Veloso. Multiagent learning using a variable learning rate. *Artificial Intelligence*, 136:215–250, 2002.

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

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

[Greenwald and Hall, 2002] Amy Greenwald and Keith Hall. Correlated Q-learning. In *Proceedings of the AAAI Spring Symposium Workshop on Collaborative Learning Agents*, 2002.

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

[Nash, Jr., 1950] John F. Nash, Jr. Equilibrium points in n-person games. *PNAS*, 36:48–49, 1950.

[Singh *et al.*, 2000] Satinder Singh, Michael Kearns, and Yishay Mansour. Nash convergence of gradient dynamics in general-sum games. In *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence*, pages 541–548. Morgan Kaufman, 2000.

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

[Sutton *et al.*, 2000] Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, pages 1057–1063. MIT Press, 2000.

[Williams, 1992] R. Williams. Simple statistical gradient following algorithms for connectionisht reinforcement learning. *Machine Learning*, 8:229–256, 1992.