# Adaptive Policy Gradient in Multiagent Learning

Bikramjit Banerjee and Jing Peng
Department of Electrical Engineering & Computer Science
Tulane University
New Orleans, LA 70118
{banerjee,jp}@eecs.tulane.edu

## ABSTRACT

Inspired by the recent results in policy gradient learning in a general-sum game scenario, in the form of two algorithms, IGA and WoLF-IGA, we explore an alternative version of WoLF. We show that our new WoLF criterion (PDWoLF) is also accurate in $2 \times 2$ games, while being accurately computable even in more than 2-action games, unlike WoLF that relies on estimation. In particular, we show that this difference in accuracy in more than 2-action games translates to faster convergence (to Nash equilibrium policies in self-play) for PDWoLF in conjunction with the general Policy Hill Climbing algorithm. Interestingly, this expedience gets more pronounced with increasing learning rate ratio, for which we also delve into an explanation. We also show experimentally that learning faster with PDWoLF could also entail learning better policies earlier in self play. Finally we present the scalable version of PDWoLF and show that even in such domains requiring generalizations and approximations, PDWoLF could dominate WoLF in performance.

## Categories and Subject Descriptors

I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent systems, Multiagent learning*

## General Terms

Algorithms, Theory, Performance, Experimentation

## Keywords

Game Theory, Nash Equilibria, Gradient Ascent Learning

## 1. INTRODUCTION

Game theory has been a driving impetus for modeling concurrent reinforcement learning problems. With booming e-commerce (notwithstanding the recent slump), the day is not far when automated buyers and sellers will control the electronic economy. There are other potential applications like disaster relief by robots (in potentially hazardous environments), automated and robotic control of applications (viz. ranging from households to Mars exploration), etc where coordination among multiple agents will hold the key. This makes the focus on multiagent learning research extremely timely and justified. Several algorithms for multiagent learning have been proposed [2, 5, 7, 8], mostly guaranteed to converge to an equilibrium in the limit. It is noted in [3] that none of these methods simultaneously satisfies rationality and convergence, two of the desirable criteria for any multiagent learning algorithm.

A recent work [11] demonstrated that policy gradient ascent (which they called "Infinitesimal Gradient Ascent"or IGA) could achieve convergence to either Nash Equilibrium policies or Nash Equilibrium payoffs (when the policies don't converge) in self-play. This algorithm was rational [3] but not convergent to Equilibrium policies in all general-sum games. Subsequently, it was modified with a variable learning rate [3] and the resulting algorithm (WoLF-IGA) was proved to converge in all $2 \times 2$ games, even though the the WoLF criterion was impossible to compute accurately without knowing the individual equilibrium policies. Moreover, in extending it to more than 2-action games, an estimation of average policy was needed. The rationale behind *WoLF* (*W*in *o*r *L*earn *F*ast) was to allow the opponent to adapt to the learner's policy by learning slowly (i.e. changing its policy slowly) when the learner is "winning", but learn fast when it is not "winning". In an earlier work [1] we had presented an alternate criterion that could be accurately computed for such adaptive policy gradient, and theoretically proved its soundness. Henceforth, we shall refer to this modified version of WoLF as PDWoLF (Policy Dynamics based WoLF). In this paper, we address the nature of the difference between WoLF and PDWoLF, and establish the superiority of PDWoLF experimentally in some bimatrix games as well as general sum stochastic games. In particular, we show that in more than 2 action games, PDWoLF converges faster to a Nash equilibrium and this expedience gets more pronounced with a higher learning rate ratio, in section 6. In section 7 we demonstrate the validity of a common conjecture that learning faster entails learning better policies earlier in a self-play situation. In section 8 we present a scalable version of PDWoLF that we show dominates WoLF even in a domain requiring generalization and approximations. Finally we present conclusions and future directions in section 9.

## 2. DEFINITIONS

Here we provide definitions of key concepts for our work. We refer to $A_i$ as the set of possible actions available to the $i$th agent.

DEFINITION 1. *A bimatrix game is given by a pair of matrices, $(M_1, M_2)$, (each of size $|A_1| \times |A_2|$ for a two-agent game) where the payoff of the ith agent for the joint action $(a_1, a_2)$ is given by the entry $M_k(a_1, a_2)$, $\forall (a_1, a_2) \in A_1 \times A_2$, $k = 1, 2$.*

A *constant-sum game* (also called competitive games) is a special


*AAMAS'03,* July 14–18, 2003, Melbourne, Australia.

bimatrix game where

$$M_1(a_1,a_2)+M_2(a_1,a_2)=c,\ \forall(a_1,a_2)\in A_1\times A_2$$

where $c$ is a constant. If $c=0$, then it is also called a zero-sum game.

DEFINITION 2. *A mixed-strategy Nash Equilibrium for a bimatrix game $(M_1,M_2)$ is a pair of probability vectors $(\pi_1^*,\pi_2^*)$ such that*

$$\pi_1^{*T}M_1\pi_2^*\geq\pi_1^TM_1\pi_2^*\ \ \forall\pi_1\in PD(A_1).$$

$$\pi_1^{*T}M_2\pi_2^*\geq\pi_1^{*T}M_2\pi_2\ \ \forall\pi_2\in PD(A_2).$$

*where $PD(A_i)$ is the set of probability-distributions over the ith agent's action space.*

No player in this game has any incentive for unilateral deviation from the Nash equilibrium strategy, given the other's strategy. There always exists at least one such equilibrium profile for an arbitrary finite bimatrix game [9].

## 3. POLICY GRADIENT ASCENT IN BIMATRIX GAMES

The basic idea of such an algorithm is to iteratively update an agent's strategy based on the consequent improvement in the agent's expected payoff. When both the agent's fail to improve their strategies any further (which may never happen), they must have converged to some Nash Equilibrium of the game [11]. The simplified domain for studying this problem is a two-agent, two action scenario, with the payoff matrices $R=\begin{bmatrix} r_{11} & r_{12}\\ r_{21} & r_{22}\end{bmatrix}$ and $C=\begin{bmatrix} c_{11} & c_{12}\\ c_{21} & c_{22}\end{bmatrix}$.

Let $\alpha$ and $\beta$ denote the probablities of the two agents selecting the first actions from their respective sets of available actions. Then, the expected payoff of the row agent is $V_r(\alpha,\beta)=r_{11}(\alpha\beta)+r_{22}((1-\alpha)(1-\beta))+r_{12}(\alpha(1-\beta))+r_{21}((1-\alpha)\beta)$, and similarly for the column agent.

Then given a strategy pair $(\alpha,\beta)$ (constrained to lie in the unit square), and letting $u=(r_{11}+r_{22})-(r_{21}+r_{12})$ and $u'=(c_{11}+c_{22})-(c_{12}+c_{21})$, the gradients are given by

$$\frac{\partial V_r(\alpha_k,\beta_k)}{\partial\alpha}=\beta u-(r_{22}-r_{12})\tag{1}$$

$$\frac{\partial V_c(\alpha_k,\beta_k)}{\partial\beta}=\alpha u'-(c_{22}-c_{21})\tag{2}$$

and the strategy pair can be updated as

$$\alpha_{k+1}=\alpha_k+\eta\frac{\partial V_r(\alpha_k,\beta_k)}{\partial\alpha}\ \text{and}\ \beta_{k+1}=\beta_k+\eta\frac{\partial V_c(\alpha_k,\beta_k)}{\partial\beta}\tag{3}$$

The new gradients generated by the above rules are constrained to lie in the unit square by taking their projections on the boundary whenever they cross out. For $\eta\to 0$, the algorithm is called IGA. It is known from game theory [10], that the algorithm may never converge in $(\alpha,\beta)$, but their expected payoffs have been proved to always converge to that of some Nash pair [11].

## 4. WOLF-IGA

Using equations 1,2,3 and $\eta\to 0$, we get the unconstrained dynamics of the strategy pair given by the differential equations

$$\begin{bmatrix}\frac{\partial\alpha}{\partial t}\\ \frac{\partial\beta}{\partial t}\end{bmatrix}=\begin{bmatrix}0 & u\\ u' & 0\end{bmatrix}\begin{bmatrix}\alpha\\ \beta\end{bmatrix}+\begin{bmatrix}-(r_{22}-r_{12})\\ -(c_{22}-c_{21})\end{bmatrix}.\tag{4}$$

It has been proved [11] that the points of zero gradients (projected) are Nash Equilibria. However, the algorithm may not converge to such a point in case the matrix $U=\begin{bmatrix}0 & u\\ u' & 0\end{bmatrix}$ has imaginary eigenvalues and the center (point of zero gradient) lies within the unit square. Consequently, the algorithm was modified with a variable learning rate [3] to converge to a Nash pair even in this remaining subcase.

The notion of variable learning rate changes the update rules in equation 3 to

$$\alpha_{k+1}=\alpha_k+\eta l_k^r\frac{\partial V_r(\alpha_k,\beta_k)}{\partial\alpha}\ \text{and}\ \beta_{k+1}=\beta_k+\eta l_k^c\frac{\partial V_c(\alpha_k,\beta_k)}{\partial\beta}\tag{5}$$

where $l_k^{r,c}\in\{l_{min},l_{max}\}$. Since the proof of convergence of IGA for the other subcases depends only on the sign of the gradients, the above learning rules have the same convergence properties as long as $l_{min},l_{max}>0$. Moreover, for $l_{min}<l_{max}$, the algorithm can be made to converge to a Nash pair in the remaining subcase [3], by choosing

$$l^r(t)=\begin{cases} l_{min} & \text{when } V_r(\alpha_t,\beta_t)\geq V_r(\alpha^*,\beta_t)\\ l_{max} & \text{otherwise}\end{cases}$$

$$l^c(t)=\begin{cases} l_{min} & \text{when } V_c(\alpha_t,\beta_t)\geq V_c(\alpha_t,\beta^*)\\ l_{max} & \text{otherwise}\end{cases}\tag{6}$$

where $(\alpha^*,\beta^*)$ are some Nash pair. The rate of convergence is proportional [1, 3] to $(\frac{l_{max}}{l_{min}})^2$, and henceforth we shall refer to $\frac{l_{max}}{l_{min}}$ as $\lambda$. The unconstrained dynamics of the system now follows the differential equations

$$\begin{bmatrix}\frac{\partial\alpha}{\partial t}\\ \frac{\partial\beta}{\partial t}\end{bmatrix}=\begin{bmatrix}0 & l^r(t)u\\ l^c(t)u' & 0\end{bmatrix}\begin{bmatrix}\alpha\\ \beta\end{bmatrix}+\begin{bmatrix}-l^r(t)(r_{22}-r_{12})\\ -l^c(t)(c_{22}-c_{21})\end{bmatrix}.\tag{7}$$

The algorithm given by equations 5, 6 is called WoLF-IGA. When $V_r(\alpha_t,\beta_t)\geq V_r(\alpha^*,\beta_t)$, the row agent is said to be *winning*, in the sense that it would prefer its current strategy to its Nash strategy against the opponent's current strategy. However, the conditions in 6 cannot be accurately computed without knowing $\alpha^*$ (or $\beta^*$) in advance, and consequently Bowling and Veloso used estimates for those conditions [3] in actual experiments. In the next section we derive an alternative criterion, that can be easily and accurately computed and guarantees convergence.

## 5. AN ALTERNATE CRITERION FOR WINNING: PDWOLF

We are concerned with the subcase where $U$ has purely imaginary eigenvalues and the center is within the unit square. The differential equation 4 has following solution for $\alpha(t)$ for the unconstrained dynamics[11]

$$\alpha(t)=B\sqrt{u}\cos(\sqrt{uu'}t+\phi)+\alpha^*$$

where $B$ and $\phi$ are dependent on the intitial $\alpha,\beta$. This also describes the constrained motion of the row agent when they have come down to an ellipse fully contained within the unit square. We note that

$$\frac{\partial^2\alpha(t)}{\partial t^2}=-|uu'|(\alpha-\alpha^*)$$

From [3],

$$\begin{aligned} V_r(\alpha_t,\beta_t)-V_r(\alpha^*,\beta_t) &= (\alpha_t-\alpha^*)\frac{\partial V_r(\alpha_t,\beta_t)}{\partial\alpha}\\ &= -\frac{1}{|uu'|}\frac{\partial^2\alpha(t)}{\partial t^2}\frac{\partial V_r(\alpha_t,\beta_t)}{\partial\alpha}\end{aligned}$$

Hence we have $V_r(\alpha_t, \beta_t) \geq V_r(\alpha^*, \beta_t)$ when $\frac{\partial^2 \alpha(t)}{\partial t^2} \frac{\partial V_r(\alpha_t, \beta_t)}{\partial \alpha} < 0$. Thus for the iterative updates, if $\Delta_t = \alpha_t - \alpha_{t-1}$ and $\Delta_t^2 = \Delta_t - \Delta_{t-1}$, then the PDWoLF criterion for the row agent is given by

$$l^r(t) = \begin{cases} l_{min} & \text{when } \Delta_t \Delta_t^2 < 0 \\ l_{max} & \text{otherwise} \end{cases}$$

This can be computed easily without the knowledge of the equilibrium policies. Also it does not need an average policy estimation and hence accurately determines the sign of $V_r(\alpha_t, \beta_t) - V_r(\alpha^*, \beta_t)$ when $\eta \to 0$. Lastly note that it adapts the learning rate to the instantaneous position of the learners in the joint policy space, and does not require policy evaluation. There is a similar criterion of *winning* for the column agent. We note that the same criterion can be extended to the system in equation 7, since $\frac{\partial l^r(t)}{\partial t} = 0$ and $l^r(t) l^c(t) =$ constant within each quadrant. A more detailed discussion of this extensibility appears in [1].

We demonstrate the accuracy of our criterion on the bimatrix game $R = \begin{bmatrix} 0 & 3 \\ 1 & 2 \end{bmatrix}, C = \begin{bmatrix} 3 & 2 \\ 0 & 1 \end{bmatrix}$. This game is directly adopted from [3], where they used it as a counterexample for another approximate criterion, viz. $V_r(\alpha_t, \beta_t) \geq V_r(\alpha^*, \beta^*)$. The only Nash Equilibrium is $(0.5, 0.5)$, and the starting point is $(\alpha, \beta) = (0.1, 0.9)$. Figure 1 demonstrates that PDWoLF converges to the equilibrium and also checks that the approximate criterion fails as expected. The experiment was run for $10,000$ iterations, and PDWoLF converged in 1369 iterations. The choices for various parameters in both cases were thus: $l_{max} = 1.0$, $l_{min} = 0.08$, and precision[1] required was $\varepsilon = 0.01$.

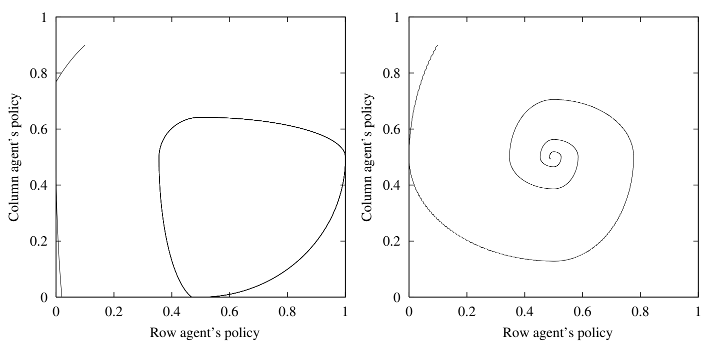


**Figure 1: Left: Approximation of WoLF fails to converge to equilibrium. Right: Succesful convergence of PDWoLF.**

# 6. WOLF VERSUS PDWOLF

We have seen from the derivations of WoLF and PDWoLF that they are accurate only in $2 \times 2$ games. However, even in such games the WoLF criterion cannot be accurately computed but only *estimated*, unlike PDWoLF. In extending these methodologies to more than 2 action games in the form of WoLF-PHC [3] and PDWoLF-PHC as shown in table 1, we note that both WoLF and PDWoLF are now only approximate criteria. As such, if there exists a "correct decision" about the learning rate at each iteration, then both are expected to make mistakes in some of those iterations. Some pertinent questions at this point are what are the comparative rates of such errors for the two methods and how significant are they? Too many errors in the earlier part of the exploration could severely affect convergence while those only in the later part would be ignorable, since the learning rates also decay. Furthermore errors at some point affect the learning process and hence the later decisions (and errors) by influencing the trajectory in the joint policy space. Also, $\lambda$ is expected to affect the convergence rates differently in case of different rates of errors for the two methods. We shall study all these questions and highlight the differences between WoLF and PDWoLF in light of the experimental results below.

We experimented with WoLF-PHC and PDWoLF-PHC on some individual bimatrix games (Figure 2 left and middle) and also on the Grid World domain (Figure 2 right), adopted from [6]. In all the following experiments a suitable decaying exploration policy was used with both the methods.

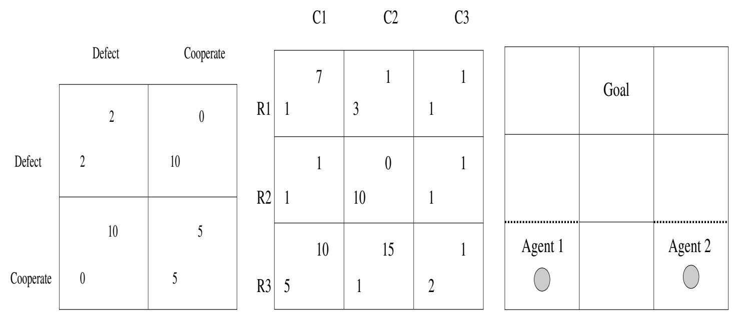


**Figure 2: The Bimatrix Games. Left: Prisoner's Dilemma. Middle: General-sum $3 \times 3$ game with mixed Nash Equilibrium. Right: Grid World general-sum stochastic game.**

## 6.1 Prisoner's Dilemma Game

The Prisoner's Dilemma game (Figure 2 left) has a unique pure Nash Equilibrium where both agents choose action "defect". We ran WoLF-PHC and PDWoLF-PHC on this bimatrix game for 1000 iterations, and noted the mean square deviation of the learned policy, at each iteration, of the first agent from its equilibrium profile. These figures were then averaged over 100 runs and the mean square deviation of the policies were plotted in Figure 3. Two learning ratios were used, $\lambda = 4, 8$. We note that PDWoLF learns faster than WoLF[2], but even faster for a higher $\lambda$. Theoretically, higher $\lambda$ should encourage faster convergence [1] but cannot explain the added advantage to PDWoLF. This is where the criticality of mistakes in selecting the learning rate gets highlighted. Figure 4 shows that both the methods make this mistake as expected (plots as proportions of maximum 100 mistakes per iteration), but PDWoLF makes less of them. Note the shifting and earlier fall of the PDWoLF curve with higher $\lambda$, which means it gets to make correct decisions when the learning rates are still significantly large (note that the mistakes in the early iterations are more crucial since the learning rates decay progressively, making later mistakes less critical), as opposed to WoLF. The latter has roughly the same error rate early on, but higher error rate later on (for higher $\lambda$) (Figure 4), thus explaining the slowdown of WoLF with higher $\lambda$ in Figures 3 left and right. Thus the plots in Figures 3 and 4 match up. However, that WoLF should at all converge with a high mistake rate early on, is itself surprising. A closer inspection reveals that for both the methods, almost all of these mistakes are of a single type, viz. selecting $l_{min}$ in place of $l_{max}$. This slows down both, more so for WoLF, but without discouraging convergence.

[1] i.e. if the stopping condition is $|\frac{\partial V_r(\alpha,\beta)}{\partial \alpha}| < \varepsilon$ *and* $|\frac{\partial V_c(\alpha,\beta)}{\partial \beta}| < \varepsilon$

[2] This is expected to some extent since WoLF-PHC decisions depend on estimating equilibrium payoffs which takes some exploring, unlike PDWoLF

1. Input $\eta \in (0,1], \delta_l > \delta_w \in (0,1]$. Initialize $Q(s,a) \leftarrow 0$, $\pi(s,a) \leftarrow \frac{1}{|A_i|}$, $\Delta(s,a) \leftarrow 0$, $\Delta^2(s,a) \leftarrow 0$.
2. Repeat
   (a) From state $s$, select action $a$ according to $\pi(s,.)$ with suitable exploration.
   (b) Observe reward $r$ and next state $s'$ and update $Q(s,a) \leftarrow (1-\eta)Q(s,a) + \eta[r + \gamma \max_b Q(s',b)]$.
   (c) Compute

$$\Delta_{sa} = \begin{cases} -\delta_{sa} & \text{if } a \neq \text{ argmax}_b Q(s,b) \\ \sum_{b \neq a} \delta_{sb} & \text{otherwise} \end{cases}$$

   where $\delta_{sa} = \min(\pi(s,a), \frac{\delta}{|A_i|-1})$, such that

$$\delta = \begin{cases} \delta_w & \text{if } \Delta(s,a) * \Delta^2(s,a) < 0 \\ \delta_l & \text{otherwise} \end{cases}$$

   (d) $\Delta^2(s,a) \leftarrow \Delta_{sa} - \Delta(s,a)$, and $\Delta(s,a) \leftarrow \Delta_{sa}$.
   (e) Finally update $\pi(s,a) \leftarrow \pi(s,a) + \Delta_{sa}$.

**Table 1: The PDWoLF-PHC algorithm**

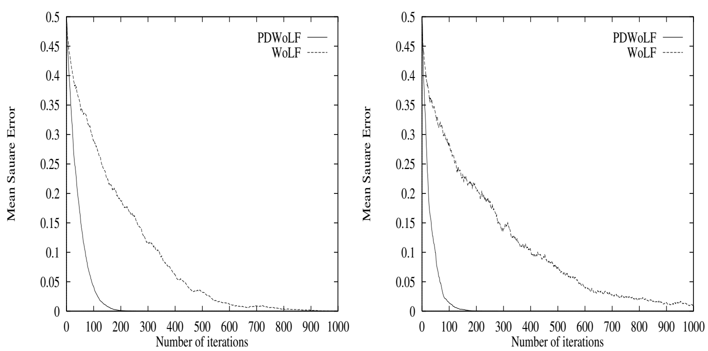


**Figure 3: Convergence curves for WoLF and PDWoLF in Prisoner's Dilemma, with $\lambda = 4$ (left) and $\lambda = 8$ (right).**

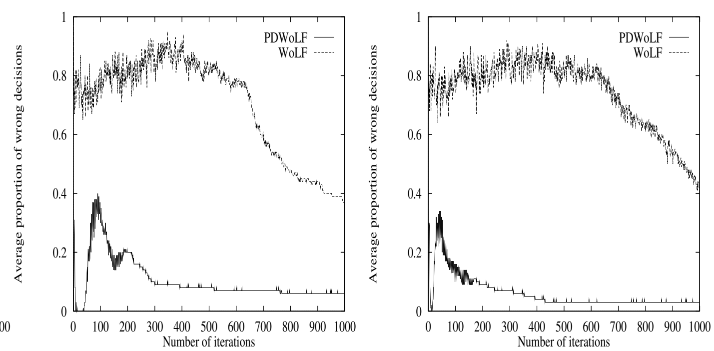


**Figure 4: Frequency plot of erroneous decisions by WoLF and PDWoLF in Prisoner's Dilemma.**

## 6.2 $3 \times 3$ Game with Mixed Nash Equilibrium

This is a sample game with 3 actions available to both agents (Figure 2 middle), and the mixed Nash Equilibrium is $\{R = (0, 0.8333, 0.1667), C = (0.6923, 0.3077, 0)\}$. As earlier, we plot the deviation of the policies (of the second agent) from the equilibrium values per iteration for 400000 iterations, and averaged over 20 runs (Figure 5 left). We used the following schedules: $\eta(t) = \frac{1}{10+\frac{t}{10000}}, l_{min} = \frac{1}{200000+t}, l_{max} = 4l_{min}$. The PDWoLF curve in Figure 5 (left) shows a damped oscillation (slight) pattern during the later iterations, since the equilibrium here is "within the unit square", unlike Prisoner's Dilemma where it was "at a corner". We also used the 3-action equivalent of the theoretical criterion of winning, viz. $\pi_1 C\pi_2 \geq \pi_1 C\pi_2^*$, and plotted the proportion of selecting wrong learning rates in Figure 5 (right). Though the higher PDWoLF decision-error rate during later iterations may be considered non-critical since it has already converged, the absence of an error-peak in Figure 5 (right) at the spot where the highest oscillatory peak occurs in Figure 5 (left) indicates that these plots have not "actually" measured decision-error rates. In other words, the criterion for winning is *not* $\pi_1 C\pi_2 \geq \pi_1 C\pi_2^*$ for more than 2 actions. This is also corroborated by the almost fixed "error rate" of 100% for WoLF (Figure 5 (right)) in spite of its convergence (Figure 5 (left)). This, in turn, questions the efficacy of the WoLF-PHC method since it directly estimates the above criterion (without modeling the opponent though). PDWoLF-PHC, on the other hand, bases its decisions directly on the dynamics of the joint trajectory in the phase space, instead of estimating $\pi_1 C\pi_2 \geq \pi_1 C\pi_2^*$ directly and then basing the decision on this estimate. This could be another reason for the efficacy of PDWoLF-PHC.

## 6.3 Grid World

The next experiment is in a general sum stochastic game (Figure 2 right), viz. Hu's gridworld. See [3, 6] for details. The game is played for 10000 iterations and the deviation of the policy of one of the agents from its (one of two possible) equilibria at the starting configuration is averaged over 20 runs, and plotted in Figures 6. The schedules for $\eta(t)$ and the learning rates were same as in [3], except that we noted the results for $\lambda = 4, 8$. We note again the pronounced difference in the deviation curves with increased $\lambda$, suggesting a similar state of affairs regarding wrong decisions with learning rates as in the $2 \times 2$ Prisoner's Dilemma game, although it cannot be similarly characterized as shown in the previous experiment. We note in passing, that when the learning rate schedule is altered to $l_{min} = \frac{1}{100+\frac{t}{10}}$ i.e. faster decaying, the deviation curves decay more slowly as one would expect.

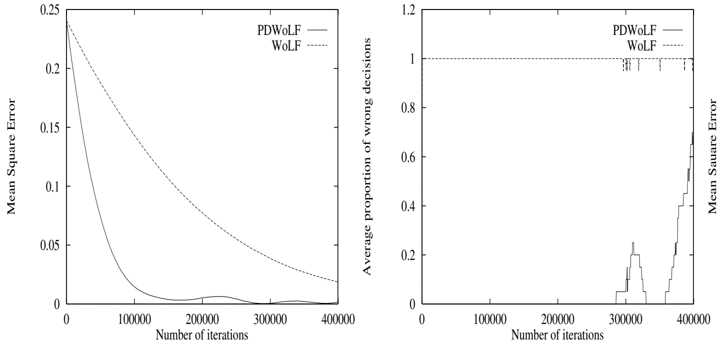


**Figure 5: Convergence (left) and error frequency (right) plots for WoLF and PDWoLF in sample 3×3 game in figure 2 (middle).**

## 7. LEARNING AND PERFORMING

It is often questioned whether learning faster is synonymous to performing better. Though it seems to be intuitively true in self play situations, it is quite clear that converging to the equilibrium faster is not necessarily better in non-self play situations, since the opponent might be learning something different that renders the equilibrium policy suboptimal. WoLF is precisely the concept that leads a learner to adapt to the opponent's policy only when not doing so hurts performance. As such it is ideal for non-self play scenarios, but unfortunately there is yet no decision mechanism for the learning rate in such scenarios. In this paper, however, we focus on self-play situations only. In the following experiment, we verify the first claim that learning faster is equivalent to performing better in a self play situation, in the block pushing domain.

The Block Pushing Domain is a $3 \times 3$ gridworld (Figure 7 left) with two agents trying to push a block to a goal cell. There is a single cell obstacle that the agents must learn to avoid while reaching the goal using minimum number of transitions. Each agent has four possible actions to choose from, viz. north, west, south and east. If they choose opposing actions they remain in the same cell, but move exactly one cell in the direction of the vector sum of their actions otherwise. Both get a reward of 100 when they reach the goal and 0 reward in all other cases. The minimum path length from the start to the goal state is 2 as shown in Figure 7 left. This amounts to one of the agents taking action north deterministically and the other selecting actions west and east with probability 0.5 each (and other actions with probability 0), at the start state. However, in this experiment we are interested in the actual path length incurred by the agents in moving from the start to the goal state using the intermediate policies. Each trial lasted 3000 iterations and at the end of each 50 iterations, the policies learned by the agents were used in simulating 100 traversals from the start to the goal state and the average path lengths from each were noted. Finally these figures were averaged over 20 runs and plotted in Figure 7 right. The various schedules were thus: $\eta(t) = \frac{1}{1+\frac{t}{400}}$, $l_{min} = \frac{1}{100+\frac{t}{100}}$ and $\lambda = 40$. Clearly PDWoLF's performance curve follows a similar pattern to the previous policy curves, showing that it can also perform better while learning faster than WoLF.

## 8. SCALING PDWOLF POLICY GRADIENT

One of the most serious drawbacks of dynamic programming based reinforcement learning is scalability. As the number of state-action tuples get larger, or the domain becomes continuous, the problem becomes intractable. A well known solution for this scenario is generalization based on function approximation [13, 12]. Recently Sutton et.al. proved that a version of policy iteration with function approximation converges to a locally optimal policy [14]. This was promptly used in conjunction with WoLF in a highly non-stationary and intractably sized stochastic game where learning was subsequently achieved with a semblance of convergence [4]. In particular, that paper noted that WoLF continues to encourage convergence even with generalization. The question we endeavor to answer at this point, is whether PDWoLF does the same and whether its superiority in tractable domains extends to these large domains.

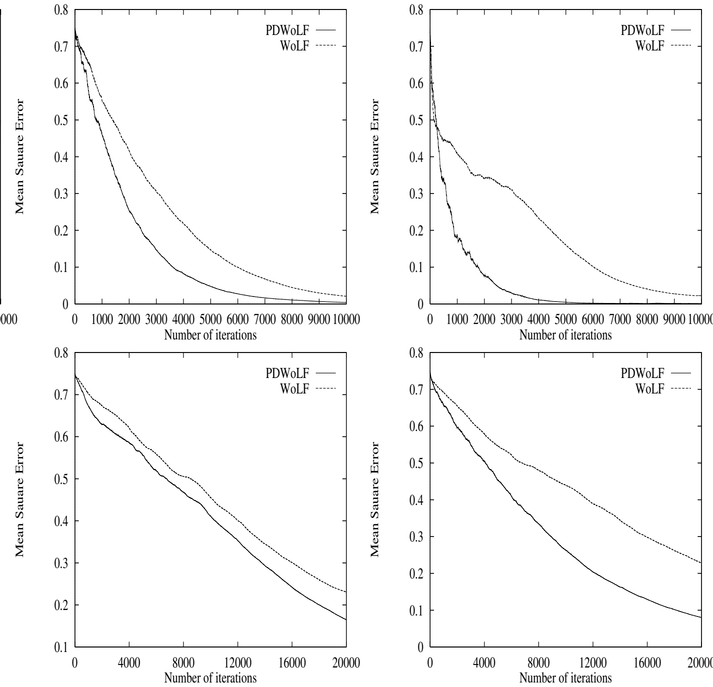


**Figure 6: Convergence curves for Grid World domain. $\lambda = 4$ (left figures) and $\lambda = 8$ (right figures). $l_{min} = 0.01$ (upper figures) and $l_{min} = 0.001$ (bottom figures).**

We experimented with the two techniques in a continuous state space block pushing domain (Figure 8). The coordinates of the agents' location can be any real number in the continuous range $[0,1]$, while they are free to choose from four available actions as earlier. We also use tile-coding [13] for approximation and parameter based generalization for policy and state action values,

$$Q_w(s,a) = \langle w, \phi_{sa} \rangle$$

$$\pi_t(s,a) = \frac{e^{\langle \theta_t, \phi_{sa} \rangle}}{\sum_b e^{\langle \theta_t, \phi_{sb} \rangle}}$$

where $\theta, w$ are parameter vectors for the policy and Q values respectively, and $\phi_{sa}$ is a tile-coding vector for the state action tuple $(s,a)$. Evidently all three vectors are of the same size. As in [4], the scalable PDWoLF gradient ascent updates the parameter vectors as

$$\theta_{t+1} \leftarrow \theta_t + \eta_t \sum_b (\pi(s,b) f_{w_t}(s,b)) \phi_{sb}$$

$$w_{t+1} \leftarrow w_t + \zeta_t (r + \gamma Q_{w_t}(s',a') - Q_{w_t}(s,a))$$

where $f_{w_t}(s,a) = \langle w_t, (\phi_{sa} - \sum_b \pi(s,b)\phi_{sb}) \rangle$, $(s',a')$ are the state and action at time $t+1$, and the PDWoLF criterion now takes the

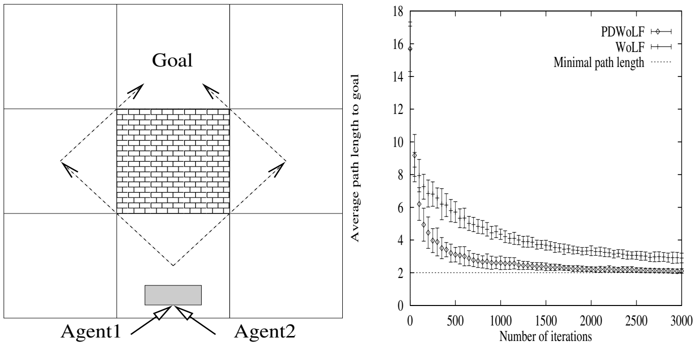


**Figure 7: Left: The Block Pushing Domain. Right: Average path lengths using the learned policies for WoLF and PDWoLF.**

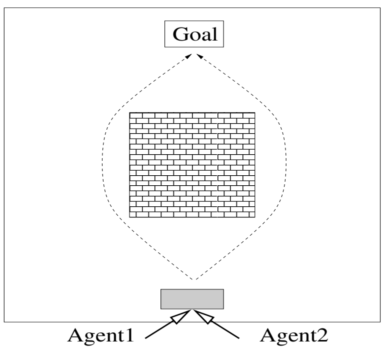


**Figure 8: The continuous state Block Pushing Domain**

form

$$\eta_t = \begin{cases} \eta_t^w & \text{if } (\pi_t(s,a) - \pi_{t-1}(s,a)) * \\ & (\pi_t(s,a) - 2\pi_{t-1}(s,a) + \pi_{t-2}(s,a)) < 0 \\ \eta_t^l > \eta_t^w & \text{otherwise} \end{cases}$$

where $\pi_{t-1}(s,a)$ and $\pi_{t-2}(s,a)$ are approximations for the last two instances of visiting $(s,a)$, since it was not the tuple actually visited in those two iterations. However, we note that with low resolution for tiling, this approximation could be good enough.

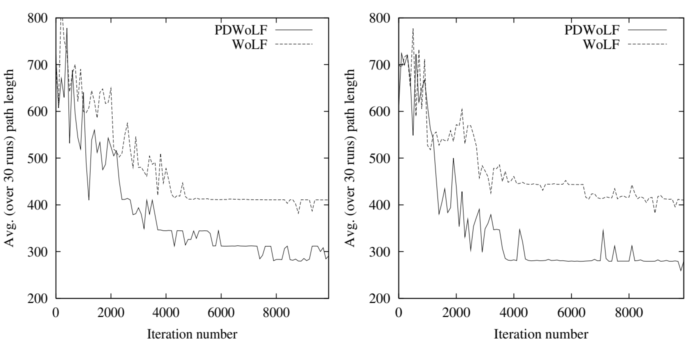


**Figure 9: Average path lengths with 16 (left) and 32 (right) tilings. The resolution was $2 \times 2$ in both cases.**

This experiment involved 10000 trials and the policies at the end of every 100 trials were used to simulate a joint trajectory over 500 trials. The average trajectory lengths were noted and these figures were later averaged over 30 runs, and plotted in Figure 9. The values used for learning rates were $l_{min} = \frac{1}{10+\frac{t}{20}}$ and $\lambda = 8$, and the stepsize for the agents' movements was 0.05. Figure 9 demonstrates that both the methods seem to produce a roughly convergent trend and PDWoLF continues to converge faster to an optimal path length, than WoLF. When the number of tilings is doubled, the sizes of the vectors $\phi_{sa}, \theta, w$ are also doubled leading to a higher capacity of the approximating model, and this is expected to improve performance in general. However, while PDWoLF seems to exploit the advantage of a more vivid model, WoLF seems to be unable to do so. The curves exhibit too much fluctuations to make any stronger conclusion.

# 9. CONCLUSIONS AND FUTURE WORK

In this paper we have studied experimentally an enhanced version of WoLF for multiagent learning. We had derived PDWoLF as an accurately computable version of WoLF for $2 \times 2$ games [1], but in this paper we demonstrated that in higher action games, this difference translates to faster convergence, with none of the methods accurate any longer. We have also shown that the criterion for winning does not have an intuitive equivalent in higher action games, thus making its discovery an open problem. This will also enable us to derive an accurate win-criterion for higher action games. Another notable observation is the appearance of a single type of error (viz. selecting $l_{min}$ in place of $l_{max}$) in all the experiments and in multiple runs, pointing to a finer truth yet to be uncovered, rather than a random phenomenon. We also experimentally verified that while PDWoLF converges faster than WoLF, it can also learn better policies earlier for better performance in self-play. Since PDWoLF may not perform as well in non self-play situations, it would be interesting to pursue a generalized WoLF strategy applicable to non self-play situations. Finally we extended our PDWoLF framework to intractably sized domains requiring function approximation and demonstrated that PDWoLF continues to dominate even in such complex domains.

# 10. REFERENCES

[1] B. Banerjee and J. Peng. Convergent gradient ascent in general sum games. In *Proceedings of 13th the European Conference on Machine Learning*, Helsinki, Finland, 2002.

[2] B. Banerjee, S. Sen, and J. Peng. Fast concurrent reinforcement learners. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, Seattle, WA, 2001.

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

[4] M. Bowling and M. Veloso. Scalable learning in stochastic games. In *AAAI Workshop Proceedings on Game Theoretic and Decision Theoretic Agents*, Edmonton, Canada, 2002.

[5] C. Claus and C. Boutilier. The dynamics of reinforcement learning in cooperative multiagent systems. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 746–752, Menlo Park, CA, 1998. AAAI Press/MIT Press.

[6] J. Hu. *Learning in dynamic noncooperative multiagent systems*. PhD thesis, Electrical Engg. and Computer Science, University of Michigan, Ann Arbor, MI, 1999.

[7] J. Hu and M. P. Wellman. Multiagent reinforcement learning: Theoretical framework and an algorithm. In *Proc. of the 15th Int. Conf. on Machine Learning (ML'98)*, pages 242–250, San Francisco, CA, 1998. Morgan Kaufmann.

[8] M. L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proc. of the 11th Int.*

*Conf. on Machine Learning*, pages 157–163, San Mateo, CA, 1994. Morgan Kaufmann.

[9] J. F. Nash. Non-cooperative games. *Annals of Mathematics*, 54:286 – 295, 1951.

[10] G. Owen. *Game Theory*. Academic Press, UK, 1995.

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

[12] R. Sutton. Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, 1996.

[13] R. Sutton and A. G. Burto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

[14] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, pages 1057 – 1063. MIT Press, 2000.