Convergence and No-Regret in Multiagent Learning

Michael Bowling

January 3, 2005

TR04-11

Department of Computing Science
University of Alberta
Edmonton, Alberta
Canada T6G 2E8

Abstract

Learning in a multiagent system is a challenging problem due to two key factors. First, if other agents are simultaneously learning then the environment is no longer stationary, thus undermining convergence guarantees. Second, learning is often susceptible to deception, where the other agents may be able to exploit a learner's particular dynamics. In the worst case, this could result in poorer performance than if the agent was not learning at all. These challenges are identifiable in the two most common evaluation criteria for multiagent learning algorithms: convergence and regret. Algorithms focusing on convergence or regret in isolation are numerous. In this paper, we seek to address both criteria in a single algorithm by introducing GIGA-WoLF, a learning algorithm for normal-form games. We prove the algorithm guarantees at most zero average regret, while demonstrating the algorithm converges in many situations of self-play. We prove convergence in a limited setting and give empirical results in a wider variety of situations. These results also suggest a third new learning criterion combining convergence and regret, which we call negative non-convergence regret (NNR).

1 Introduction

Learning to select actions to achieve goals in a multiagent setting requires overcoming a number of key challenges. One of these challenges is the loss of the stationarity assumption when multiple agents are learning simultaneously. Another challenge is guaranteeing that the learner cannot be deceptively exploited by another agent. Both of these challenges distinguish the multiagent learning problem from traditional single-agent learning, and have been gaining recent attention as multiagent applications continue to proliferate.

In single-agent learning tasks, it is reasonable to assume that the same action from the same state will result in the same distribution over outcomes, both rewards and next states. In other words, the environment is stationary. In a multiagent task with other learning agents, the outcomes of an agent's action will vary with the changing policies of the other agents. Since most of the convergence results in reinforcement learning depend upon the environment being stationary, convergence is often difficult to obtain in multiagent settings.

Before continuing, we should examine the goal of convergence. Convergence alone does not seem like a desirable property as it can easily be achieved by a static, non-learning strategy. Coupled with guarantees about maximization of reward (e.g., bounds on regret or best-response guarantees) it becomes more interesting. In passing, we offer two reasons why convergence is an important property. The first is the practicality of value estimation. When a learner is placed in the context of games with state, such as stochastic games (Shapley, 1953), estimating the value of a state is critical. The variance of this estimate would be far smaller in cases with convergent strategies or rewards, and thus might make the difference between a practical and impractical algorithm. The second is far more subjective and is based on the actual play of non-converging algorithms. Although an algorithm may have high average reward, its expected reward at any moment may be arbitrarily high or low. Many algorithms have increasingly long periods of very poor play, later compensated for increasingly long periods of very successful play. This is intuitively unappealing when strategies exist with the same average payoff but without the cycles of very poor play. This is further addressed later in this paper with the introduction of negative non-convergence regret, which we believe addresses many of the arguments about the irrelevance of convergence.

Equilibrium learners (Littman, 1994; Hu & Wellman, 1998; Greenwald & Hall, 2002) are one method of handling the loss of stationarity. These algorithms learn joint-action values, which *are* stationary, and in certain circumstances guarantee these values converge to Nash (or correlated) equilibrium values. Using these values, the player's strategy corresponds to the player's component of some Nash (or correlated) equilibrium. This convergence of strategies is guaranteed nearly independently of the actions selected by the other agents, including when other agents play suboptimal responses. Equilibrium learners, therefore, can fail to learn best-response policies even against simple non-learning opponents.[1] Best-response learners (Claus & Boutilier, 1998; Singh et al., 2000; Bowling & Veloso, 2002) are an alternative approach that has sought to learn best-responses, but still considering whether the resulting algorithm converges in some form. These approaches usually examine convergence in self-play, and have included both theoretical and experimental results.

The second challenge is the avoidance of exploitation. Since learning strategies dynamically change their action selection over time, it is important to know that the change cannot be exploited by a clever opponent. A deceptive strategy may "lure" a dynamic strategy away from a safe choice in order to switch to a strategy where the learner receives much lower reward. For example, Chang and Kaelbling (2001) demonstrated that the best-response learner PHC (Bowling & Veloso, 2002) could be exploited by a particular dynamic strategy. One method of measuring whether an algorithm can be exploited is the notion of regret. Regret has been explored both in game theory (Hart & Mas-Colell, 2000) and computer science (Auer et al., 1995; Zinkevich, 2003). Regret measures how much worse an algorithm performs compared to the best static strategy, with the goal to guarantee at least zero average regret, *no-regret*, in the limit.

These two challenges result in two completely different criteria for evaluation: convergence and no-regret. In addition, they have almost exclusively been explored in isolation. For example, equilibrium learners can have arbitrarily large average regret. On the other hand, no-regret learners' strategies rarely converge in self-play (Jafari et al., 2001) in even the simplest of games.[2] In this paper, we seek to explore these two criteria in a single algorithm for learning in

[1]This work is not restricted to zero-sum games and our use of the word "opponent" refers simply to other players in the game.

[2]A notable exception is Hart and Mas-Colell's algorithm that guarantees the empirical distribution of play converges to that of a correlated equilibrium. Neither strategies nor expected values necessarily converge, though.

normal-form games. In Section 2 we present a more formal description of the problem and the two criteria. We also examine key related work in applying gradient ascent algorithms to this learning problem. In Section 3 we introduce GIGA-WoLF, an algorithm with both regret and convergence properties. The algorithm is followed by theoretical and experimental analyses in Sections 4 and 5, respectively, before concluding.

2 Online Learning in Games

A game in normal form is a tuple, $(n, \mathcal{A}_{1\ldots n}, R_{1\ldots n})$, where n is the number of players in the game, $\mathcal{A}_i$ is a set of actions available to player i ($\mathcal{A} = \mathcal{A}_1 \times \ldots \times \mathcal{A}_n$), and $R_i : \mathcal{A} \rightarrow \Re$ is a mapping from joint actions to player i's reward. The problem of learning in a normal-form game is one of repeatedly selecting an action and receiving a reward, with a goal of maximizing average reward against an unknown opponent. If there are two players then it is convenient to write a player's reward function as a $|\mathcal{A}_1| \times |\mathcal{A}_2|$ matrix. Three example normal-form games are shown in Table 1.

Table 1: Examples of games in normal-form.

$$R_1 = \begin{array}{c|cc} & \textbf{H} & \textbf{T} \\ \hline \textbf{H} & 1 & -1 \\ \textbf{T} & -1 & 1 \end{array} \qquad R_1 = \begin{array}{c|cc} & \textbf{A} & \textbf{B} \\ \hline \textbf{A} & 0 & 3 \\ \textbf{B} & 1 & 2 \end{array} \qquad R_1 = \begin{array}{c|ccc} & \textbf{R} & \textbf{P} & \textbf{S} \\ \hline \textbf{R} & 0 & -1 & 1 \\ \textbf{P} & 1 & 0 & -1 \\ \textbf{S} & -1 & 1 & 0 \end{array}$$

$$R_2 = \begin{array}{c|cc} \textbf{H} & -1 & 1 \\ \textbf{T} & 1 & -1 \end{array} \qquad R_2 = \begin{array}{c|cc} \textbf{A} & 3 & 2 \\ \textbf{B} & 0 & 1 \end{array} \qquad R_2 = \begin{array}{c|ccc} \textbf{R} & 0 & 1 & -1 \\ \textbf{P} & -1 & 0 & 1 \\ \textbf{S} & 1 & -1 & 0 \end{array}$$

(a) Matching Pennies (b) Tricky Game (c) Rock–Paper–Scissors

Unless stated otherwise we will assume the learning algorithm is player one. In the context of a particular learning algorithm and a particular opponent, let $r_t \in \Re^{|\mathcal{A}_1|}$ be the vector of actual rewards that player one would receive at time t for each of its actions. Let $x_t \in PD(\mathcal{A}_1)$ be the algorithm's strategy at time t, selected from probability distributions over actions. So, player one's expected payoff at time t is $(r_t \cdot x_t)$. Let 1_a be the probability distribution that assigns probability 1 to action $a \in \mathcal{A}_1$. Lastly, we will assume the reward for any action is bounded by $r_{\max}$ and therefore $||r_t||^2 \leq |\mathcal{A}_1| r_{\max}^2$.

2.1 Evaluation Criteria

One common evaluation criterion for learning in normal-form games is convergence. There are a number of different forms of convergence that have been examined in the literature. These include, roughly increasing in strength:

1. Average reward (i.e., $\sum(r_t \cdot x_t)/T$),
2. Empirical distribution of actions (i.e., $\sum x_t/T$),
3. Expected reward (i.e., $(r_t \cdot x_t)$), and
4. Strategies (i.e., x_t).

We focus in this paper on convergence of strategies as this implies the other three forms of convergence as well. In particular, we will say an algorithm converges against a particular opponent if and only if $\lim_{t\rightarrow\infty} x_t = x_*$.

Convergence as an important criterion has recently been called into question (e.g., (Shoham et al., 2003)). The contention is that the focus on convergence (particularly to equilibria) has diverted efforts from algorithms that simply perform well, in theory or practice. Although much of this criticism is warranted, convergence is still an important criteria when in the context of other evaluation measures, such as performance measures.

In passing, we enumerate three reasons why convergence is a desirable property. First, the variance of estimating the value of a normal-form game is much smaller if the expected reward of the game is stationary (e.g., compare to estimating the value of a cycling stochastic process with possibly ever-growing periods). Low variance is critical when the learner is placed in the context of a stochastic game, where decisions always depend on the value estimates of future states, i.e., normal-form games. Second, the use of a converging learning strategy for normal-form games can be safely extended to stochastic games with some form of temporal differencing. A non-converging strategy would mean that past rewards would not always be an accurate predictor of future reward. Third, most non-converging algorithms only make guarantees about average reward ignoring that the agent's performance may still be arbitrarily poor at any given moment. This concern can be captured by a discounted reward criterion, which becomes identical to the average reward criterion if and only if expected reward is converging. It is true that these facts do not justify convergence as an independently interesting criterion, which is why we also consider another measure of evaluation.

The second common evaluation criterion is regret. Total regret[3] is the difference between the maximum total reward of any static strategy given the past history of play and the algorithm's total reward.

$$\mathcal{R}_T \equiv \max_{a \in \mathcal{A}_1} \sum_{t=1}^{T} ((r_t \cdot 1_a) - (r_t \cdot x_t))$$

Average regret is just the asymptotic average of total regret, $\lim_{T\to\infty} \mathcal{R}_T/T$. An algorithm has no-regret if and only if the average regret is less than or equal to zero against all opponent strategies. The no-regret property makes a strong claim about the performance of the algorithm: the algorithm's expected average reward is at least as large as the expected average award any static strategy could have achieved. In other words, the algorithm is performing at least as well as any static strategy.

2.2 Gradient Ascent Learning

Gradient ascent is a simple and common technique for finding parameters that optimize a target function. In the case of learning in games, the parameters represent the player's strategy, and the target function is expected reward. We will examine three recent results evaluating gradient ascent learning algorithms in normal-form games.

Singh, Kearns, and Mansour (2000) analyzed gradient ascent in two-player, two-action games, e.g., Table 1(a) and (b). They derived the gradient of expected value with respect to a player's strategy and proposed updating a player's strategy in the direction of increasing gradient. The algorithm is often called IGA (Infinitesimal Gradient Ascent) due to their method of analysis. They examined the resulting strategy trajectories and payoffs in self-play, focusing on the continuous-time, infinitesimal step, version of the algorithm. The resulting differential equations form a two-variable affine dynamical system. These systems are well understood and are known to have at most three qualitative forms. The strategy trajectories in these three forms are shown in Figure 1. In forms (a) and (b), they showed the strategies will converge to a Nash equilibrium. In case (c), of which Table 1(a) and (b) are examples, they showed that the strategies do not converge, but instead orbit the only equilibrium in the game. They proved, instead, that the average payoffs (a weaker form of convergence) converge to the payoffs of the Nash equilibrium.

WoLF-IGA (2002) extended the work by Singh and colleagues to a stronger form of convergence, namely convergence of strategies. They introduced a variable learning rate, so players would change the size of steps taken in the direction of the gradient. Using the WoLF ("Win or Learn Fast") principle, the algorithm would choose a larger step size when the current strategy had less expected payoff than the equilibrium strategy, i.e., $r_t \cdot (x_* - x_t) > 0$. Using a similar analysis of the resulting dynamical system, they proved the variable learning rate has no effect on the first two forms where IGA's strategies converge. In the final case, though, the learning rate result in strategies following piecewise elliptical trajectories that spiral toward (rather than orbiting) the Nash equilibrium. Hence, they proved that the strategies of their modified algorithm, WoLF-IGA, would converge in self-play in all two-player, two-action games. WoLF-IGA may be a limited variant of the extragradient method (Korpelevich, 1976) for variational inequality problems. A multiagent learning view of the extragradient algorithm is that it iteratively updates a strategy vector using the gradient of the agents' rewards from one step in the future. The extragradient method is guaranteed to converge to a Nash equilibrium in self-play for all zero-sum games. Like WoLF-IGA, though, it does not have any known regret guarantees, but more importantly requires the other players' payoffs to be known.

Zinkevich (2003) looked at gradient ascent using the evaluation criterion of regret. He first extended IGA beyond two-player, two-action games. His algorithm, GIGA (Generalized Infinitesimal Gradient Ascent), updates strategies

[3]Our analysis focuses on expectations of regret (total and average), similar to (Auer et al., 1995; Zinkevich, 2003). Although note that for any self-oblivious behavior, including GIGA-WoLF, average regret of at most zero on expectation implies universal consistency, i.e., regret of at most ϵ with high probability (Zinkevich, 2003).

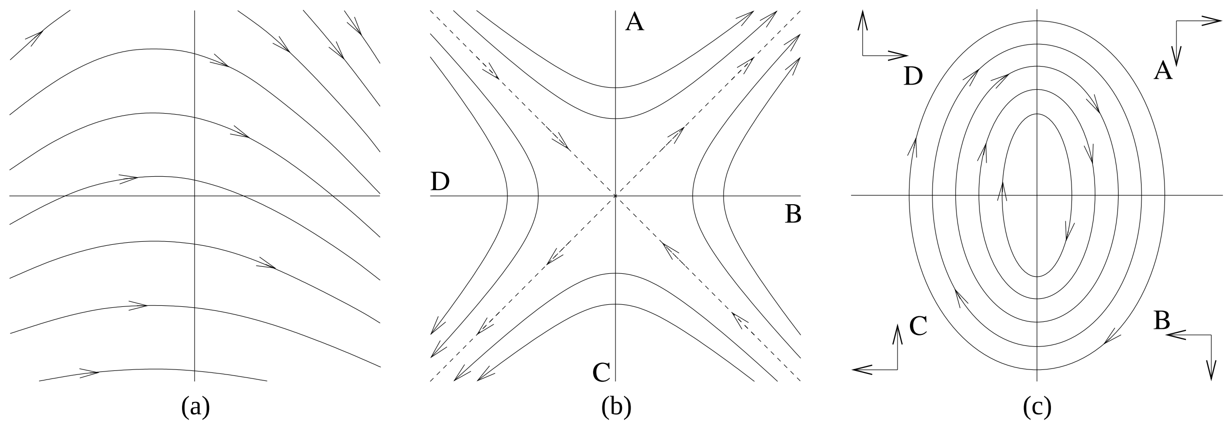


Figure 1: The three possible strategy dynamics of GIGA in self-play in two-player, two-action games. (c) corresponds to the subclass of games where GIGA's strategies do not converge.

using an unconstrained gradient, and then projects the resulting strategy vector back into the simplex of legal probability distributions,

$$x_{t+1} = P(x_t + \eta_t r_t) \tag{1}$$

where,

$$P(x) = \operatorname*{argmin}_{x' \in PD(\mathcal{A}_1)} ||x - x'||,$$

eta_t is the stepsize at time t, and $|| \cdot ||$ is the standard L2 norm.

He examined the regret of GIGA for online convex programming, a superclass containing normal-form games. He showed that for these problems GIGA has no-regret. In particular, he showed GIGA's total regret is bounded by,

$$\mathcal{R}_T \quad \leq \quad \sqrt{T} + |A| r_{\max}^2 (\sqrt{T} - 1/2). \tag{2}$$

Since GIGA is identical to IGA in two-player, two-action games, we also have that GIGA achieves the weak form of convergence in this subclass of games. It is also true, though, that GIGA's strategies do not converge in self-play even in simple games like matching pennies.

In the next section, we present an algorithm that simultaneously achieves GIGA's no-regret result and part of WoLF-IGA's convergence result. We first present the algorithm and then analyze these criteria both theoretically and experimentally.

3 GIGA-WoLF

GIGA-WoLF is a gradient based learning algorithm that internally keeps track of two different gradient-updated strategies, x_t and z_t. The algorithm chooses actions according to the distribution x_t, but updates both x_t and z_t after each iteration. The update rules consist of three steps.

$$\begin{array}{rrcl}
(1) & \hat{x}_{t+1} & = & P(x_t + \eta_t r_t) \\
(2) & z_{t+1} & = & P(z_t + \eta_t r_t / 3) \\
 & \delta_{t+1} & = & \min\left(1, \dfrac{||z_{t+1} - z_t||}{||z_{t+1} - \hat{x}_{t+1}||}\right) \\
(3) & x_{t+1} & = & \hat{x}_{t+1} + \delta_{t+1}(z_{t+1} - \hat{x}_{t+1})
\end{array}$$

Step (1) updates x_t according to GIGA's standard gradient update and stores the result as $\hat{x}_{t+1}$. Step (2) updates z_t in the same manner, but with a smaller step-size. Step (3) makes a final adjustment on x_{t+1} by moving it toward z_{t+1}. The magnitude of this adjustment is limited by the change in z_t that occurred in step (2).

A key factor in understanding this algorithm is the observance that a strategy a receives higher reward than a strategy b if and only if the gradient at a is in the direction of b (i.e., $r_t \cdot (b - a) > 0$). Therefore, the step (3) adjustment is in the direction of the gradient if and only if z_t received higher reward than x_t. Notice also that, as long as x_t is not near the boundary, the change due to step (3) is of lower magnitude than the change due to step (1). Hence, the combination of steps (1) and (3) result in a change with two key properties. First, the change is in the direction of positive gradient. Second, the magnitude of the change is larger when z_t received higher reward than x_t. So, we can interpret the update rule as a variation on the WoLF principle of "win or learn fast", i.e., the algorithm is *learning faster* if and only if its strategy x is *losing* to strategy z.

In the original presentation of WoLF-IGA, winning was determined by comparing the played strategy to an equilibrium strategy. In the practical variant called WoLF-PHC, winning was determined by comparing the played strategy to the average of past strategies. The use of a slower moving GIGA-updated strategy for determination of winning has two advantages over these past algorithms. First, it does not require the other players' payoffs to be known. Second, GIGA already has guarantees on average regret, which will turn out to be key in analyzing GIGA-WoLF's regret.

In the next section we present a theoretical examination of GIGA-WoLF's regret in n-player, n-action games, along with a limited guarantee of convergence in two-player, two-action games. In Section 5, we give experimental results of learning using GIGA-WoLF, demonstrating that convergence extends beyond the theoretical analysis presented.

4 Theoretical Analysis

We begin by examining GIGA-WoLF's regret against an unknown opponent strategy. We will prove the following bound on average regret.

Theorem 1 *If $\eta_t = 1/\sqrt{t}$, the regret of GIGA-WoLF is,*

$$\mathcal{R}_T \leq 2\sqrt{T} + |A| r_{\max}^2 (2\sqrt{T} - 1).$$

Therefore, $\lim_{T\to\infty} \mathcal{R}_T / T \leq 0$, hence GIGA-WoLF has no-regret.

Proof. We begin with a brief overview of the proof. We will find a bound on the regret of the strategy x_t with respect to the dynamic strategy z_t. Since z_t is unmodified GIGA, we already have a bound on the regret of z_t with respect to any static strategy. Hence, we can bound the regret of x_t with respect to any static strategy.

We start by examining the regret of x_t with respect to z_t using a similar analysis as used by Zinkevich (2003). Let $\rho_t^{x\to z}$ refer to the difference in expected payoff between z_t and x_t at time t, and $\mathcal{R}_T^{x\to z}$ be the sum of these differences, i.e., the total regret of x_t with respect to z_t,

$$\begin{aligned} \rho_t^{x\to z} &\equiv r_t \cdot z_t - r_t \cdot x_t = r_t \cdot (z_t - x_t) \\ \mathcal{R}_T^{x\to z} &\equiv \sum_{t=1}^{T} \rho_t^{x\to z}. \end{aligned}$$

We will use the following potential function,

$$\Phi_t \equiv \frac{(x_t - z_t)^2}{2\eta_t}.$$

We can examine how this potential changes with each step of the update. $\Delta\Phi_t^1$, $\Delta\Phi_t^2$, and $\Delta\Phi_t^3$ refers to the change in potential caused by steps (1), (2), and (3), respectively. $\Delta\Phi_t^4$ refers to the change in potential caused by the learning rate change from η_{t-1} to η_t. This gives us the following equations for the potential change.

$$\begin{aligned}
\Delta\Phi^1_{t+1} &= 1/2\eta_t((\hat{x}_{t+1} - z_t)^2 - (x_t - z_t)^2) \\
\Delta\Phi^2_{t+1} &= 1/2\eta_t((\hat{x}_{t+1} - z_{t+1})^2 - (\hat{x}_{t+1} - z_t)^2) \\
\Delta\Phi^3_{t+1} &= 1/2\eta_t((x_{t+1} - z_{t+1})^2 - (\hat{x}_{t+1} - z_{t+1})^2) \\
\Delta\Phi^4_{t+1} &= (1/2\eta_{t+1} - 1/2\eta_t)\,(x_{t+1} - z_{t+1})^2 \\
\Delta\Phi_{t+1} &= \Delta\Phi^1_{t+1} + \Delta\Phi^2_{t+1} + \Delta\Phi^3_{t+1} + \Delta\Phi^4_{t+1}
\end{aligned}$$

Notice that if $\delta_{t+1} = 1$ then $x_{t+1} = z_{t+1}$. Hence $\Phi_{t+1} = 0$, and $\Delta\Phi^2_{t+1} + \Delta\Phi^3_{t+1} \leq 0$. If $\delta_{t+1} < 1$, then $||x_{t+1} - \hat{x}_{t+1}|| = ||z_{t+1} - z_t||$. Notice also that in this case x_{t+1} is co-linear and between $\hat{x}_{t+1}$ and z_{t+1}. So,

$$\begin{aligned}
||\hat{x}_{t+1} - z_{t+1}|| &= ||\hat{x}_{t+1} - x_{t+1}|| + ||x_{t+1} - z_{t+1}|| \\
&= ||z_{t+1} - z_t|| + ||x_{t+1} - z_{t+1}||
\end{aligned}$$

We can bound the left with the triangle inequality,

$$||\hat{x}_{t+1} - z_{t+1}|| \leq ||\hat{x}_{t+1} - z_t|| + ||z_t - z_{t+1}||.$$

Combining and subtracting terms we get,

$$||x_{t+1} - z_{t+1}|| \leq ||\hat{x}_{t+1} - z_t||.$$

So regardless of δ_{t+1}, $\Delta\Phi^2_{t+1} + \Delta\Phi^3_{t+1} < 0$. Hence,

$$\Delta\Phi_{t+1} \leq \Delta\Phi^1_{t+1} + \Delta\Phi^4_{t+1}.$$

We will now use this bound on the change in the potential to bound the regret of x_t with respect to z_t. We know from Zinkevich that,

$$(\hat{x}_{t+1} - z_t)^2 - (x_t - z_t)^2 \leq 2\eta_t r_t \cdot (z_t - x_t) + \eta_t^2 r_t^2.$$

Therefore,

$$\begin{aligned}
\rho_t^{x\to z} &\leq -\frac{1}{2\eta_t}\left((\hat{x}_{t+1} - z_t)^2 - (x_t - z_t)^2 - r_t^2\right) \\
&\leq -\Delta\Phi^1_{t+1} + 1/2\eta_t r_t^2 \\
&\leq -\Delta\Phi_{t+1} + \Delta\Phi^4_{t+1} + 1/2\eta_t r_t^2.
\end{aligned}$$

Since we assume rewards are bounded by $r_{\max}$ we can bound r_t^2 by $|A|r^2_{\max}$. Summing up regret and using the fact that $\eta_t = 1/\sqrt{t}$, we get the following bound.

$$\begin{aligned}
\mathcal{R}_T^{x\to z} &\leq \sum_{t=1}^{T} -\Delta\Phi_t + \Delta\Phi_t^4 + \frac{\eta_t}{2}|A|r^2_{\max} \\
&\leq (\Phi_1 - \Phi_T) + \left(\frac{1}{\eta_T} - 1\right) + \frac{|A|r^2_{\max}}{2}\sum_{t=1}^{T}\eta_t \\
&\leq \left(2 + \frac{1}{\eta_T} - 1\right) + \frac{|A|r^2_{\max}}{2}\sum_{t=1}^{T}\eta_t \\
&\leq \sqrt{T} + |A|r^2_{\max}(\sqrt{T} - 1/2)
\end{aligned}$$

We know that GIGA's regret with respect to any strategy is bounded by the same value (see Inequality 2). Hence,

$$\mathcal{R}_T \leq 2\sqrt{T} + |A|r^2_{\max}(2\sqrt{T} - 1),$$

as claimed. □

The second criterion we want to consider is convergence. As with IGA, WoLF-IGA, and other algorithms, our theoretical analysis will be limited to two-player, two-action general-sum games. We further limit ourselves to the situation of GIGA-WoLF playing "against" GIGA. These restrictions are a limitation of the proof method, which uses a case-by-case analysis that is combinatorially impractical for the case of self-play. This is not necessarily a limitation on GIGA-WoLF's convergence. This theorem along with the empirical results we present later in Section 5 give a strong sense of GIGA-WoLF's convergence properties.

Theorem 2 *In a two-player, two-action repeated game, if one player follows the GIGA-WoLF algorithm and the other follows the GIGA algorithm, then their strategies will converge to a Nash equilibrium.*

Proof. As in the convergence proofs of IGA and WoLF-IGA, we'll focus on the continuous time dynamics, i.e., $\lim_{\eta \to 0}$. Since η is decreasing, the discrete dynamics will eventually converge to this continuous time dynamics.

In two-player, two-action games, the space of strategies is just the two-dimensional simplex and so is best represented as a single variable. The projected gradient corresponds to a change in this variable, and for GIGA the change in this variable is identical to the IGA gradient. To understand GIGA-WoLF in this context we need to consider step (3), which adjusts x_t toward z_t. This adjustment is at most a third of the gradient change in step (1). In fact, if the gradient is toward z_t then the two steps total to taking a step of size $4\eta/3$. If the gradient is away from z_t then this amounts to a step of size $2\eta/3$. Therefore, GIGA-WoLF is simply a variable learning rate added to IGA, allowing us to make use of much of the WoLF-IGA proof (Bowling & Veloso, 2002) (see Lemmas 3, 4, and 5). The only part of that proof that does not immediately hold for GIGA-WoLF is the case when the game's corresponding dynamical system has imaginary eigenvalues and the center of the dynamics is strictly inside the unit square. We now consider this subcase in detail.

GIGA's (i.e., IGA's) dynamics for this case are shown in Figure 1(c). The only Nash equilibrium is the center, and the GIGA strategies only orbit this center without converging. Let (x_*, y_*) be the center.

Notice that GIGA-WoLF's "learning rates" stay constant during parts of the trajectory that are strictly toward or away from z_t. Therefore, the two-players' strategies (x_t, y_t) will follow piecewise elliptical orbits broken when player one's gradient changes directions or when x_t crosses z_t. Our analysis will proceed by examining half orbits delineated by x_t's change in direction, which occur when the other player's strategy crosses the equilibrium. We will show that after a finite number of half-orbits, (x_t, y_t) will decrease its distance to (x_*, y_*) by a constant factor, hence (x_t, y_t) is converging to the equilibrium.

The analysis will be simpler and unaffected by shifting the equilibrium to the origin, and referring to moments in time $t = 1, 2, 3, \ldots$ as the endpoints of the strategies piecewise elliptical trajectories. Figure 2 gives an example in the case when x crosses z during the half-orbit. Odd time-points (e.g., 1, 3) refer to the start and end of half orbits. Even time-points (e.g., 2, 4) occur when x crosses z. We can now write simultaneous equations for these trajectories,

$$x_1^2 = x_2^2 + 4y_2^2 \qquad x_3^2 = x_2^2 + y_2^2 \tag{3}$$

$$z_3 = (z_2 + x_3)/2 \qquad z_4 = z_3 + (z_3 - x_3)/3 \tag{4}$$

Equations 3 come from the piecewise elliptical orbits. The factor of 4 in the first equation comes from the fact that x is moving in the direction of z while on this ellipse, and so is updated twice as quickly. Notice that z moves one half (or one quarter) the gradient of x when x is moving away (or toward) z. This fact leads to Equations 4.

We will consider four possible cases, the first three when x crosses z, and the last one when it does not. Before going into the details of the cases, note that we can solve the simultaneous equations[4] above to discover,

$$|z_2/x_1| < 1 \Rightarrow |x_3/x_1| \leq 1. \tag{5}$$

So, for the first three cases, we have that (x, y) does not increase its distance from the center along a half-orbit.

Case 1: $|z_2/x_1| < 1/3$**.** Solving the simultaneous equations with this inequality gives us, $|x_3/x_1| < \sqrt{3}/3$. Hence, in one half-orbit the distance of (x, y) decreases by a factor of $\sqrt{3}/3$.

[4]All solutions to simultaneous inequalities in this proof were found with the aid of analytic mathematical software.

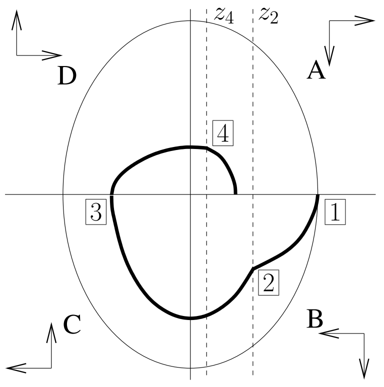


Figure 2: The dynamics of GIGA-WoLF against GIGA. The numbers delineate the time-points when the piecewise trajectories change ellipses.

Case 2: $z_2/x_1 > 1/3$**.** Solving the simultaneous equations with this inequality gives us, $|z_4/x_1| < 1/3$. Hence, the second half-orbit satisfies Case 1. In two half-orbits, by the result of Case 1 and Inequality 5, we get that the distance (x, y) decreases by a factor of $\sqrt{3}/3$.

Case 3: $z_2/x_1 < -1/3$**.** If z_2 is negative, this means x only crosses z in the second quadrant of the half-orbit. By Equations 4, if $z_2 < 0$ then $z_4 < 0$. Therefore, this satisfies either Case 1 or 2, on the second half-orbit. By the result of the cases and Inequality 5, we get that the distance of (x, y) decreases by a factor of $\sqrt{3}/3$ in at most three half-orbits.

Case 4: x **doesn't cross** z **in the half-orbit.** In this case, the ellipse will be constant during the half-orbit, so $x_3 = -x_1$ and Inequality 5 holds in this case, as well. If x crosses y on the next half-orbit then one of Cases 1, 2, or 3 will hold, and we'd get that the distance (x, y) decreases by a factor of $\sqrt{3}/3$ in at most four half-orbits. If not, then in a full orbit (x, y) will not have changed, i.e., $x_5 = x_1$. But, since z travels half the distance x travels when x is moving away from z and one quarter the distance when x is moving toward z, we get, $|z_5| = |z_1| - |x_1|/2$. After, $\frac{|z_5|-|x_1|}{|x_1|/2}$ orbits, z will be inside x's orbit, and thus one of the above cases will apply. During that time, Inequality 5 will hold, and thus after a finite number of half-orbits, the distance of (x, y) will decrease by a factor of $\sqrt{3}/3$.

In all four cases, the distance of (x, y) decreases by a factor of $\sqrt{3}/3$ in a finite number of orbits. Therefore, (x, y) is converging to the origin, which is the Nash equilibrium of the game. □

5 Experimental Analysis

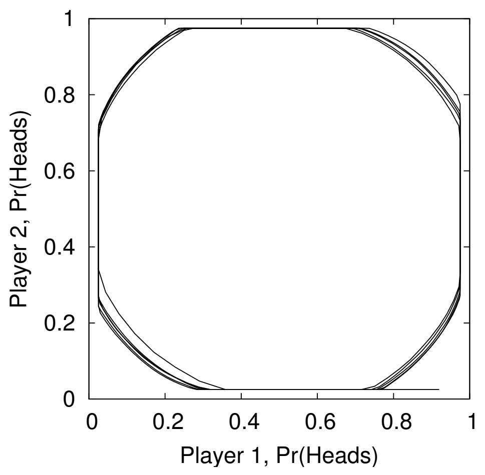
We've examined some of the theoretical properties of GIGA-WoLF, in this section we explore GIGA-WoLF experimentally on a variety of normal-form games. The purpose of these experiments is to demonstrate the theoretical results from the previous section as well as explore the extent to which the results (convergence, in particular) can be generalized. This analysis will also raise some interesting questions for further investigation, such as no-regret learners in competition, and situations when convergence fails. The games used in these experiments include all of the normal-form games for which results of WoLF-PHC, the more practical variant of WoLF-IGA, were reported (Bowling & Veloso, 2002).

One of the requirements of GIGA-WoLF (and GIGA) is knowledge of the entire reward vector (r_t), which requires knowledge of the game and observation of the opponent's action. In practical situations, one or both of these are unlikely to be available. Instead, only the reward of the selected action is likely to be observable. Therefore, these experiments involve providing GIGA-WoLF (and GIGA) with estimates of the gradient from stochastic approximation. In particular, after selecting action a and receiving reward $\hat{r}_a$, we update the current estimate of action a's component of the reward vector,

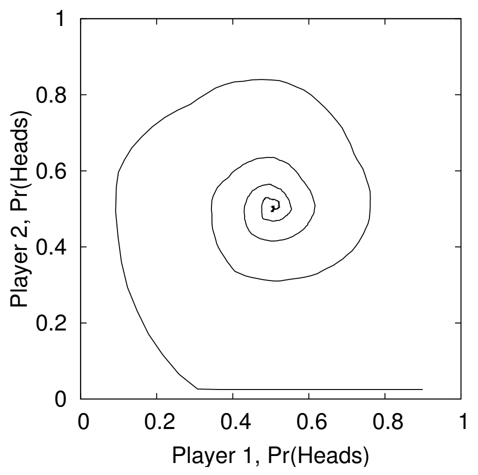
$$r_{t+1} = r_t + \alpha_t(\hat{r}_a - 1_a \cdot r_t)1_a,$$

where α_t is the learning rate. This is a standard method of estimation commonly used in reinforcement learning (e.g., Q-learning).

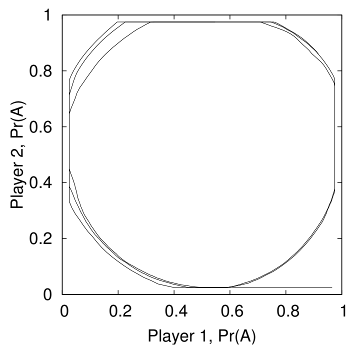
Two-Player, Two-Action Games. We begin by exploring GIGA-WoLF and GIGA in self-play in two-player, two-action games. We specifically focus on games that fall into case (c) in Figure 1 in order to explore Theorem 2 in practice. Figure 3 shows the strategy trajectories in both Matching Pennies (top), and the Tricky Game (bottom). Notice that GIGA's policies follow orbits around the equilibrium, $(1/2, 1/2)$ in both games. GIGA-WoLF's strategies, on the other hand, spiral toward the equilibrium as suggested by Theorem 2. We don't show results of the strategies when GIGA-WoLF plays against GIGA, as the trajectories are nearly identical to those shown.



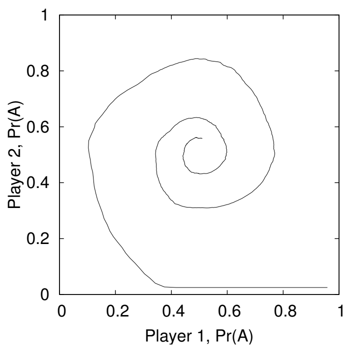
(a) GIGA



(b) GIGA-WoLF



(a) GIGA

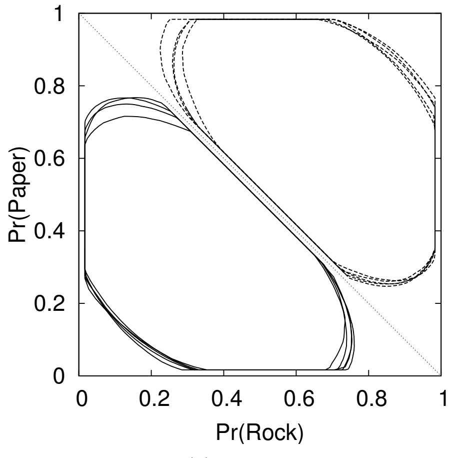


(b) GIGA-WoLF

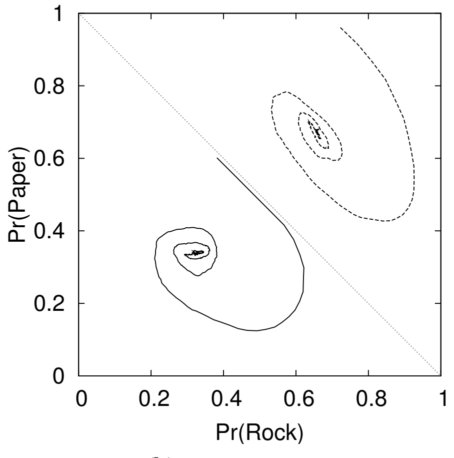
Figure 3: Trajectories of joint strategies when both players use GIGA (a) or GIGA-WoLF (b) in Matching Pennies (top) and Tricky Game (bottom).

Zero-Sum Games. Our second experiments examine the algorithms outside of two-player, two-action games. Figure 4 shows strategy trajectories for Rock-Paper-Scissors. The plots show both players' three-dimensional strategy trajectories (separated by the diagonal) projected into two-dimensions. The results are similar to the the first two games. GIGA strategies in self-play orbit the equilibrium, while GIGA-WoLF strategies spiral toward the equilibrium. In plot (c), we show strategy trajectories of GIGA playing against GIGA-WoLF. The strategies are spiraling toward the equilibrium, although the rate of convergence appears to be slower and the trajectory oddly shaped.

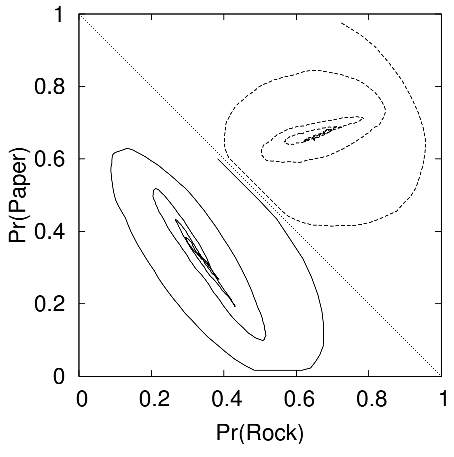
The odd trajectory in Rock-Paper-Scissors prompted an examination of the players' rewards while learning. Figure 5 shows the GIGA-WoLF player's expected reward and average reward over time. Notice that, although expected reward is oscillating around the equilibrium, the player is winning more often than losing, causing the average reward to be positive throughout. Since GIGA is a no-regret algorithm, GIGA-WoLF can't have positive average reward indefinitely, explaining its gradual decrease to the Nash equilibrium value. GIGA-WoLF's out-performance of GIGA in short-term direct competition is mostly independent of starting conditions and also occurs in Matching Pennies. Further understanding of this phenomenon requires an investigation beyond the scope of this paper.



(a) GIGA



(b) GIGA-WoLF



(c) GIGA v. GIGA-WOLF

Figure 4: Trajectories of joint strategies in Rock-Paper-Scissors when both players use GIGA (a), GIGA-WoLF (b), or when GIGA plays against GIGA-WoLF (c).

In Figure 6, we show results from the Blotto Game (Gintis, 2000), a more complicated 4×5 action zero-sum game. The plot shows player one's action probabilities over time, further demonstrating GIGA-WoLF converging to a Nash equilibrium in self-play, while GIGA strategies do not converge.

Problem Games. Convergence of GIGA-WoLF is not universal, though. Two common games where many algorithms fail to converge (often in all senses of convergence) are Shapley's Game and Three-Player Matching Pennies. The details of these games aren't critical beyond that they are non-zero-sum games. Figure 7 shows player one's action

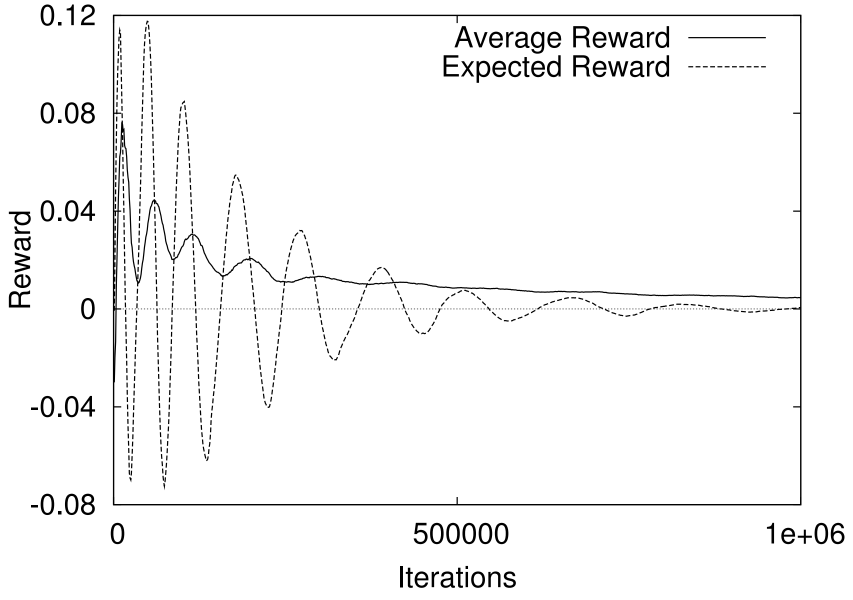


Figure 5: GIGA-WoLF's expected reward and average reward over time when playing Rock-Paper-Scissors against GIGA.

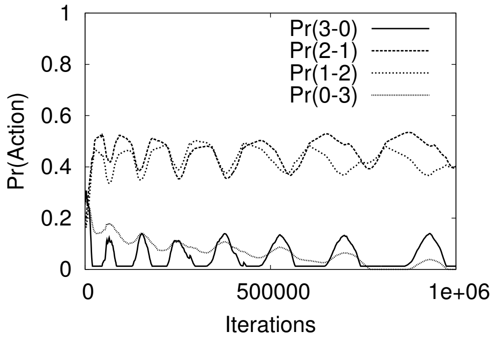
probabilities over time with both GIGA-WoLF and GIGA in self-play. The results show that these algorithms are not immune to the convergence challenges raised by these games. In light of Theorem 1's bound on GIGA-WoLF's regret, we can gain insight by exploring the players' average rewards.

Figure 8 shows both players' average reward over time with GIGA-WoLF[5] in Shapley's Game. Both players' average reward exceeds that of the game's Nash equilibrium value, which is the value that would be received if the players' strategies converged. The Nash value is also the asymptotic value of the best static strategy, meaning both players are attaining negative average regret. A similar phenomenon holds in Three-Player Matching pennies, where all three players have negative regret. Notice two things. First, a static or converging strategy cannot attain negative regret. Second, a strategy with negative regret is doing better than any static strategy. Essentially, if there's any reasonable "excuse" for strategies not converging it would be that the non-converging strategy is outperforming all static strategies, i.e., it is attaining negative regret.

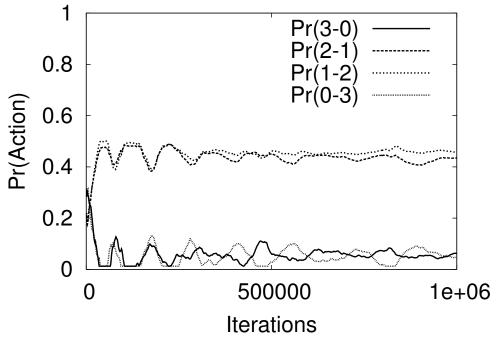
The observation is not just an excuse for failing to converge but also provides a condition for when negative regret is achievable. We can formalize this in the following property, which we propose as a new target criterion for learning. An algorithm is said to have negative non-convergence regret (NNR) if (i) it has no-regret; (ii) its expected average regret is negative or its expected reward converges. In other words, if the algorithm does not get negative regret, then it must be getting the same expected reward as the best static strategy. In addition, satisfying this property also adds a strong convergence guarantee: any two algorithms with NNR must converge to a Nash equilibrium in any zero-sum game. This result follows from the fact that both players cannot have negative regret if the game is zero-sum, therefore they both must converge and the result must be an equilibrium. It is not known if GIGA-WoLF satisfies this property, but it is solid candidate algorithm, and will be explored in future work.

Summary. These results confirm the theoretical analyses from Section 4, and give evidence of the extent that GIGA-WoLF might converge in self-play. These experiments also revealed two interesting observations that deserve further consideration. First, we noticed different short-term regret when two no-regret algorithms were paired together (see Figure 5). This suggests not all no-regret algorithms are equivalent even if they both have the same asymptotic bounds at the same asymptotic rate. We also observed that in the two common "problem" games, GIGA-WoLF strategies in self-play get greater reward than any static strategy can achieve, i.e., both players have negative regret. Therefore there

[5]The same pattern of average reward is also true of GIGA in both Shapley's Game and Three-Player Matching Pennies.



(a) GIGA



(b) GIGA-WoLF

Figure 6: Player one's strategy when both players use GIGA (a) or GIGA-WoLF (b) in Blotto Game.

is a natural limit to the desirability of convergence, since convergence would necessitate both players receiving lower payoffs. This suggested a new desirable learning property, negative non-convergence regret, that combines both the criteria of convergence and regret.

6 Conclusion

We introduced GIGA-WoLF, a new gradient-based algorithm, that we believe is the first to simultaneously address two criteria: no-regret and convergence. We proved GIGA-WoLF has no-regret. We also proved that in a small class of normal-form games, GIGA-WoLF's strategy when played against GIGA will converge to a Nash equilibrium. We presented thorough experimental results of GIGA-WoLF playing in self-play in a variety of zero-sum and general-sum games. These experiments verified our theoretical results and exposed two interesting phenomenon that deserve further study: short-term performance of no-regret learners and the new desiderata of negative non-convergence regret. We expect GIGA-WoLF and these results to be the foundation for further understanding of the connections between the regret and convergence criteria.

References

Auer, P., Cesa-Bianchi, N., Freund, Y., & Schapire, R. E. (1995). Gambling in a rigged casino: The adversarial multi-arm bandit problem. *36th Annual Symposium on Foundations of Computer Science* (pp. 322–331).

Bowling, M., & Veloso, M. (2002). Multiagent learning using a variable learning rate. *Artificial Intelligence*, *136*, 215–250.

Chang, Y.-H., & Kaelbling, L. P. (2001). Playing is believing: the role of beliefs in multi-agent learning. *Advances in Neural Information Processing Systems 14*.

Claus, C., & Boutilier, C. (1998). The dynamics of reinforcement learning in cooperative multiagent systems. *Proceedings of the Fifteenth National Conference on Artificial Intelligence* (pp. 746–752).

Gintis, H. (2000). *Game theory evolving*. Princeton University Press.

Greenwald, A., & Hall, K. (2002). Correlated Q-learning. *Proceedings of the AAAI Spring Symposium Workshop on Collaborative Learning Agents*.

Hart, S., & Mas-Colell, A. (2000). A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, *68*, 1127–1150.

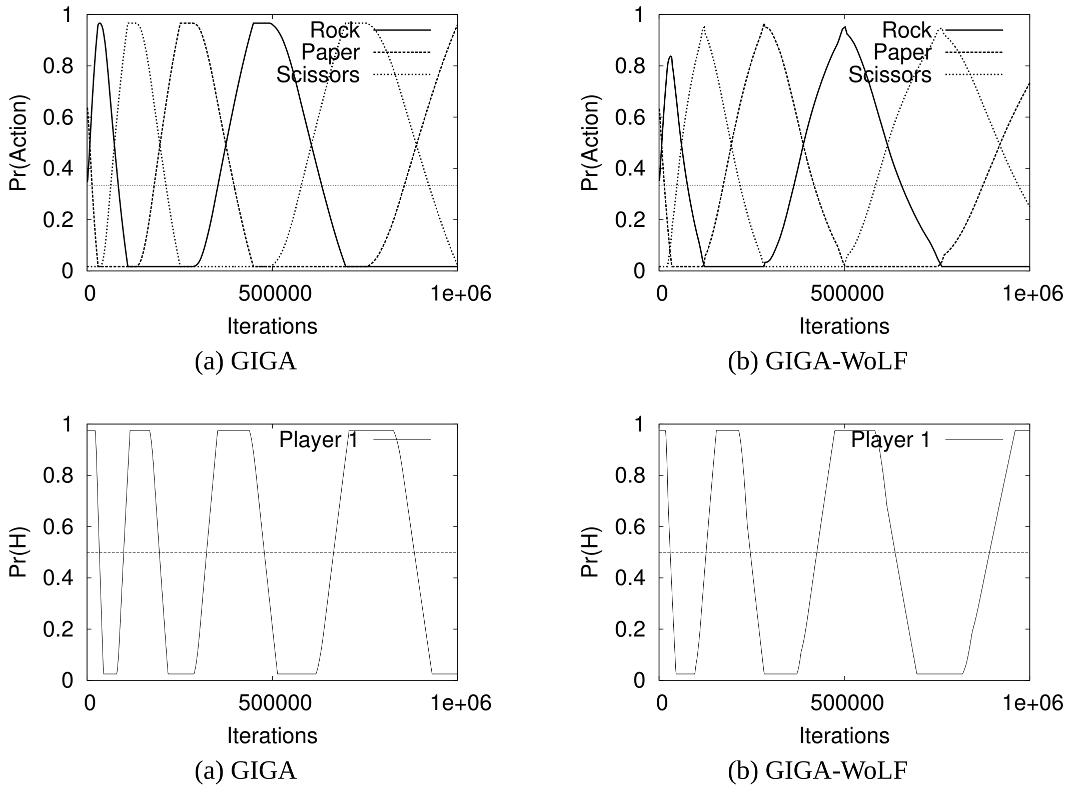


Figure 7: Player one's strategy when both players use GIGA (a) or GIGA-WoLF (b) in Shapley's Game (top) and Three-Player Matching Pennies (bottom).

Hu, J., & Wellman, M. P. (1998). Multiagent reinforcement learning: Theoretical framework and an algorithm. *Proceedings of the Fifteenth International Conference on Machine Learning* (pp. 242–250).

Jafari, A., Greenwald, A., Gondek, D., & Ercal, G. (2001). On no-regret learning, fictitious play, and nash equilibrium. *Proceedings of the Eighteenth International Conference on Machine Learning* (pp. 226–223).

Korpelevich, G. M. (1976). The extragradient method for finding saddle points and other problems. *Matecon*, *12*, 747–756.

Kuhn, H. W. (Ed.). (1997). *Classics in game theory*. Princeton University Press.

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

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

Shoham, Y., Poers, R., & Grenager, T. (2003). Multi-agent reinforcement learning: a critical survey. Unpublished manuscript.

Singh, S., Kearns, M., & Mansour, Y. (2000). Nash convergence of gradient dynamics in general-sum games. *Proceedings of the Sixteenth Conference on Uncertainty in Artificial Intelligence* (pp. 541–548).

Zinkevich, M. (2003). Online convex programming and generalized infinitesimal gradient ascent. *Proceedings of the Twentieth International Conference on Machine Learning* (pp. 928–925).

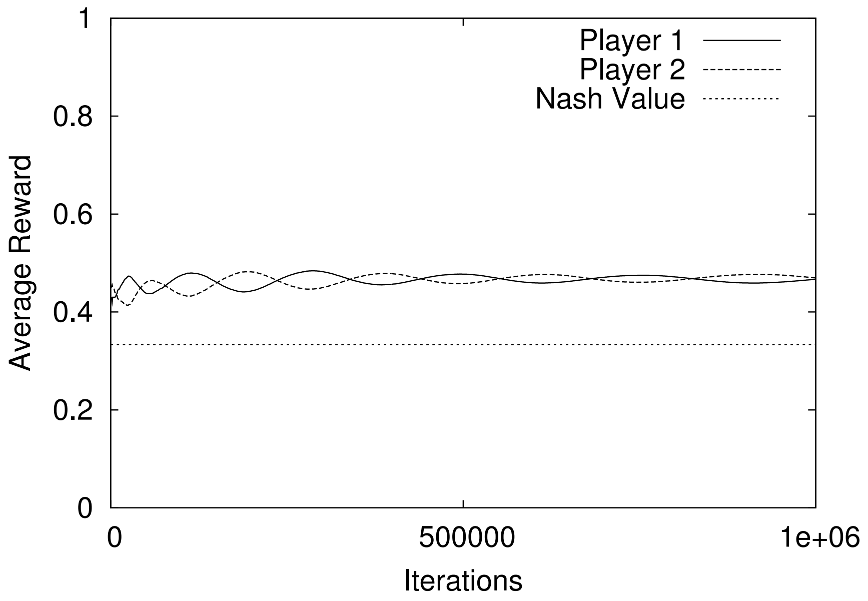


Figure 8: Both players' average reward over time when playing Shapley's game with GIGA-WoLF in self-play.