Multi-Agent Learning with Policy Prediction

Chongjie Zhang
Computer Science Department
University of Massachusetts
Amherst, MA 01003 USA
chongjie@cs.umass.edu

Victor Lesser
Computer Science Department
University of Massachusetts
Amherst, MA 01003 USA
lesser@cs.umass.edu

Abstract

Due to the non-stationary environment, learning in multi-agent systems is a challenging problem. This paper first introduces a new gradient-based learning algorithm, augmenting the basic gradient ascent approach with policy prediction. We prove that this augmentation results in a stronger notion of convergence than the basic gradient ascent, that is, strategies converge to a Nash equilibrium within a restricted class of iterated games. Motivated by this augmentation, we then propose a new practical multi-agent reinforcement learning (MARL) algorithm exploiting approximate policy prediction. Empirical results show that it converges faster and in a wider variety of situations than state-of-the-art MARL algorithms.

Introduction

Learning is a key component of multi-agent systems (MAS), which allows an agent to adapt to the dynamics of other agents and the environment and improves the agent performance or the system performance (for cooperative MAS). However, due to the non-stationary environment where multiple interacting agents are learning simultaneously, single-agent reinforcement learning techniques are not guaranteed to converge in multi-agent settings.

Several multi-agent reinforcement learning (MARL) algorithms have been proposed and studied (Singh, Kearns, and Mansour 2000; Bowling and Veloso 2002; Hu and Wellman 2003; Bowling 2005; Conitzer and Sandholm 2007; Banerjee and Peng 2007), all of which have some theoretical results of convergence in general-sum games. A common assumption of these algorithms is that an agent (or player) knows its own payoff matrix. To guarantee convergence, each algorithm has its own additional assumptions, such as requiring an agent to know a Nash Equilibrium (NE) and the strategy of the other players(Bowling and Veloso 2002; Banerjee and Peng 2007; Conitzer and Sandholm 2007), or observe what actions other agents executed and what rewards they received (Hu and Wellman 2003; Conitzer and Sandholm 2007). For practical applications, these assumptions are very constraining and unlikely to hold, and, instead, an agent can only observe the immediate reward after selecting and performing an action.

In this paper, we first propose a new gradient-based algorithm that augments a basic gradient ascent algorithm with policy prediction. The key idea behind this algorithm is that a player adjusts its strategy in response to forecasted strategies of the other players, instead of their current ones. We analyze this algorithm in two-person, two-action, general-sum iterated game and prove that if at least one player uses this algorithm (if not both, assume the other player uses the standard gradient ascent algorithm), then players' strategies will converge to a Nash equilibrium. Like other MARL algorithms, besides the common assumption, this algorithm also has additional requirements that a player knows the other player's strategy and current strategy gradient (or payoff matrix) so that it can forecast the other player's strategy.

Motivated by our theoretical convergence analysis, we then propose a new practical MARL algorithm exploiting the idea of policy prediction. Our practical algorithm only requires an agent to observe its reward when choosing a given action. We show that our practical algorithm can learn an optimal policy when other players use stationary policies. Empirical results show that it converges in more situations than that covered by our formal analysis. Compared to state-of-the-art MARL algorithms, WPL (Abdallah and Lesser 2008), WoLF-PHC (Bowling and Veloso 2002) and GIGA-WoLF (Bowling 2005), it empirically converges faster and in a wider variety of situations.

In the remainder of this paper, we first review the basic gradient ascent algorithm and then introduce our gradient-based algorithm with policy prediction followed by its theoretical analysis. We then describe a new practical MARL algorithm and evaluate it in benchmark games, distributed task allocation problem and network routing.

Notation

- Δ denotes the valid strategy space, i.e., [0, 1].
- $\Pi_\Delta : \Re \rightarrow \Delta$ denotes the projection to the valid space,
$$\Pi_\Delta[x] = \text{argmin}_{z\in\Delta}|x - z|.$$
- $P_\Delta(x, v)$ denotes the projection of a vector v on $x \in \Delta$,
$$P_\Delta(x, v) = \lim_{\eta\rightarrow 0} \frac{\Pi_\Delta(x + \eta v) - x}{\eta}$$

Gradient Ascent

We begin with a brief overview of normal-form games and then review the basic gradient ascent algorithm.

Normal-Form Games

A two-player, two-action, general-sum normal-form game is defined by a pair of matrices

$$R = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix} \text{ and } C = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix}$$

specifying the payoffs for the *row* player and the *column* player, respectively. The players simultaneously select an action from their available set, and the joint action of the players determines their payoffs according to their payoff matrices. If the row player and the column player select action $i, j \in \{1, 2\}$, respectively, then the row player receives a payoff r_{ij} and the column player receives the payoff c_{ij}.

The players can choose actions stochastically based on some probability distribution over their available actions. This distribution is said to be a mixed strategy. Let $\alpha \in [0, 1]$ and $\beta \in [0, 1]$ denote the probability of choosing the first action by the row and column player, respectively. With a joint strategy (α, β), the row player's expected payoff is

$$\begin{aligned} V_r(\alpha, \beta) \quad &= \quad r_{11}(\alpha\beta) + r_{12}(\alpha(1-\beta)) + r_{21}((1-\alpha)\beta) \\ &\quad\; + r_{22}((1-\alpha)(1-\beta)) \end{aligned} \tag{1}$$

and the column player's expected payoff is

$$\begin{aligned} V_c(\alpha, \beta) \quad &= \quad c_{11}(\alpha\beta) + c_{12}(\alpha(1-\beta)) + c_{21}((1-\alpha)\beta) \\ &\quad\; + c_{22}((1-\alpha)(1-\beta)). \end{aligned} \tag{2}$$

A joint strategy (α^*, β^*) is said to be a *Nash equilibrium* if (i) for any mixed strategy α of the row player, $V_r(\alpha^*, \beta^*) \geq V_r(\alpha, \beta^*)$, and (ii) for any mixed strategy β of the column player, $V_c(\alpha^*, \beta^*) \geq V_c(\alpha^*, \beta)$. In other words, no player can increase its expected payoff by changing its equilibrium strategy unilaterally. It is well-known that every game has at least one Nash equilibrium.

Learning using Gradient Ascent in Iterated Games

In an iterated normal-form game, players repeatedly play the same game. Each player seeks to maximize it own expected payoff in response to the strategy of the other player. Using the gradient ascent algorithm, a player can increase its expected payoff by moving its strategy in the direction of the current gradient with some step size. The gradient is computed as the partial derivative of the agent's expected payoff with respect to its strategy:

$$\begin{aligned} \partial_\alpha V_r(\alpha, \beta) &= \frac{\partial V_r(\alpha, \beta)}{\partial \alpha} = u_r \beta + b_r \\ \partial_\beta V_c(\alpha, \beta) &= \frac{\partial V_c(\alpha, \beta)}{\partial \beta} = u_c \alpha + b_c \end{aligned} \tag{3}$$

where $u_r = r_{11} + r_{22} - r_{12} - r_{21}, b_r = r_{12} - r_{22}, u_c = c_{11} + c_{22} - c_{12} - c_{21}$, and $b_c = c_{21} - c_{22}$.

If (α_k, β_k) are the strategies on the kth iteration and both players use gradient ascent, then the new strategies will be:

$$\begin{aligned} \alpha_{k+1} &= \Pi_\Delta[\alpha_k + \eta \partial_\alpha V_r(\alpha_k, \beta_k)] \\ \beta_{k+1} &= \Pi_\Delta[\beta_k + \eta \partial_\beta V_c(\alpha_k, \beta_k)] \end{aligned} \tag{4}$$

where η is the gradient step size. If the gradient moves the strategy out of the valid probability space, then the function Π_Δ will project it back. This will only occur on the boundaries (i.e., 0 and 1) of the probability space.

Singh, Kearns, and Mansour (2000) analyzed the gradient ascent algorithm by examining the dynamics of the strategies in the case of an infinitesimal step size ($\lim_{\eta \to 0}$). This algorithm is called *Infinitesimal Gradient Ascent (IGA)*. Its main conclusion is that, if both players use IGA, their average payoffs will converge in the limit to the expected payoffs for some Nash equilibrium.

Note that the convergence result of IGA focuses on the average payoffs of the two players. This notion of convergence is still weak, because, although the players' average payoffs converge, their strategies may not converge to a Nash equilibrium (e.g., in zero-sum games). In the next section, we will describe a new gradient ascent algorithm with policy prediction that allows players' strategies to converge to a Nash equilibrium.

Gradient Ascent With Policy Prediction

As shown in Equation 4, the gradient used by IGA to adjust the strategy is based on current strategies. Suppose that one player knows its change direction of the opponent's strategy, i.e., strategy derivative, in addition to its current strategy. Then the player can forecast the opponent's strategy and adjust its strategy in response to the forecasted strategy. Thus the strategy update rules is changed to:

$$\begin{aligned} \alpha_{k+1} &= \Pi_\Delta[\alpha_k + \eta \partial_\alpha V_r(\alpha_k, \beta_k + \gamma \partial_\beta V_c(\alpha_k, \beta_k))] \\ \beta_{k+1} &= \Pi_\Delta[\beta_k + \eta \partial_\beta V_c(\alpha_k + \gamma \partial_\alpha V_r(\alpha_k, \beta_k), \beta_k)] \end{aligned} \tag{5}$$

The new derivative terms with γ serve as a short-term prediction (i.e., with length γ) of the opponent's strategy. Each player computes its strategy gradient based on the forecasted strategy of the opponent. If the prediction length $\gamma = 0$, the algorithm is actually IGA. Because of using policy prediction (i.e., $\gamma > 0$), we call this algorithm IGA-PP (for theoretical analysis, we also consider the case of an infinitesimal step size ($\lim_{\eta \to 0}$)). As will be shown in the next section, if one player uses IGA-PP and the other uses IGA-PP or IGA, their strategies will converge to a Nash equilibrium.

The prediction length γ will affect the convergence of the IGA-PP algorithm. With a too large prediction length, a player may not predict the opponent strategy in a right way. Then the gradient based on the wrong opponent strategy deviates too much from the gradient based on the current strategy, and the player adjusts its strategy in a wrong direction. As a result, in some cases (e.g., $u_r u_c > 0$), players' strategies converge to a point that is not a Nash equilibrium. The following conditions restrict γ to be appropriate.

Condition 1: $\gamma > 0$,

Condition 2: $\gamma^2 u_r u_c \neq 1$,

Condition 3: for any $x \in \{b_r, u_r + b_r\}$ and $y \in \{b_c, u_c + b_c\}$, if $x \neq 0$, then $x(x + \gamma u_r y) > 0$, and if $y \neq 0$, then $y(y + \gamma u_c x) > 0$.

Condition 3 basically says the term with γ will not change the sign of the x or y, and a sufficiently small $\gamma > 0$ will always satisfy them.

Analysis of IGA-PP

In this section, we will prove the following main result.

Theorem 1. *If, in a two-person, two-action, iterated general-sum game, both players follow the IGA-PP algorithm (with sufficiently small $\gamma > 0$), then their strategies will asymptotically converge to a Nash equilibrium.*

Similar to the analysis in (Singh, Kearns, and Mansour 2000; Bowling and Veloso 2002), our proof of this theorem is accomplished by examining the possible cases of the dynamics of players' strategies following IGA-PP, as done by Lemma 3, 4, and 5. To facilitate the proof, we first prove that if players' strategies converge by following IGA-PP, then they must converge to a Nash equilibrium, i.e., Lemma 2.

For brevity, let ∂_{α_k} denote $\partial_\alpha V_r(\alpha_k, \beta_k)$, and ∂_{β_k} denote $\partial_\beta V_c(\alpha_k+, \beta_k)$. We reformulate the update rules of IGA-PP from Equation 5 using Equation 3:

$$\begin{aligned} \alpha_{k+1} &= \Pi_\Delta[\alpha_k + \eta(\partial_{\alpha_k} + \gamma u_r \partial_{\beta_k})] \\ \beta_{k+1} &= \Pi_\Delta[\beta_k + \eta(\partial_{\beta_k} + \gamma u_c \partial_{\alpha_k})] \end{aligned} \tag{6}$$

Lemma 1. *If the projected partial derivatives at a strategy pair (α^*, β^*) are zero, that is, $P_\Delta(\alpha^*, \partial_{\alpha^*}) = 0$ and $P_\Delta(\beta^*, \partial_{\beta^*}) = 0$, then (α^*, β^*) is a Nash equilibrium.*

Proof. Assume that (α^*, β^*) is not a Nash equilibrium. Then at least one player, say the column player, can increase its expected payoff by changing its strategy unilaterally. Let the improved point be (α^*, β). Because the strategy space Δ is convex and the linear dependence of $V_c(\alpha, \beta)$ on β, then, for any $\epsilon > 0$, $(\alpha^*, (1-\epsilon)\beta^* + \epsilon\beta)$ must also be an improved point, which implies the projected gradient of β at (α^*, β^*) is not zero. By contradiction, (α^*, β^*) is a Nash equilibrium. □

Lemma 2. *If, in following IGA-PP with sufficiently small $\gamma > 0$, $\lim_{k\to\infty}(\alpha_k, \beta_k) = (\alpha^*, \beta^*)$, then (α^*, β^*) is a Nash equilibrium.*

Proof. The strategy pair trajectory converges at (α^*, β^*) if and only if the projected gradients used by IGA-PP are zero, that is, $P_\Delta(\alpha^*, \partial_{\alpha^*} + \gamma u_r \partial_{\beta^*}) = 0$ and $P_\Delta(\beta^*, \partial_{\beta^*} + \gamma u_c \partial_{\alpha^*}) = 0$. Now we are showing that $P_\Delta(\alpha^*, \partial_{\alpha^*} + \gamma u_r \partial_{\beta^*}) = 0$ and $P_\Delta(\beta^*, \partial_{\beta^*} + \gamma u_c \partial_{\alpha^*}) = 0$ will imply $P_\Delta(\alpha^*, \partial_{\alpha^*}) = 0$ and $P_\Delta(\beta^*, \partial_{\beta^*}) = 0$, which, according to Lemma 1, will finish the proof and indicates (α^*, β^*) is a Nash equilibrium. Assume $\gamma > 0$ is sufficiently small that satisfies Condition 2 and 3. Consider three possible cases when the projected gradients used by IGA-PP are zero.

Case 1: both gradients are zero, that is, $\partial_{\alpha^*} + \gamma u_r \partial_{\beta^*} = 0$ and $\partial_{\beta^*} + \gamma u_c \partial_{\alpha^*} = 0$. By solving them, we get $(1 - \gamma^2 u_r u_c)\partial_{\alpha^*} = 0$ and $\partial_{\beta^*} = -\gamma u_c \partial_{\alpha^*}$, which implies $\partial_{\alpha^*} = 0$ and $\partial_{\beta^*} = 0$, due to Condition 2 (i.e., $\gamma^2 u_r u_c \neq 1$). Therefore, $P_\Delta(\alpha^*, \partial_{\alpha^*}) = 0$ and $P_\Delta(\beta^*, \partial_{\beta^*}) = 0$.

Case 2: at least one gradient is greater than zero. Without loss of generality, assume $\partial_{\alpha^*} + \gamma u_r \partial_{\beta^*} > 0$. Because its projected gradient is zero, its strategy is on the boundary of the strategy space Δ, which implies $\alpha^* = 1$. Now we consider three possible cases of the column player's partial strategy derivative $\partial_{\beta^*} = u_c \alpha^* + b_c = u_c + b_c$.

1. $\partial_{\beta^*} = 0$, which implies $P_\Delta(\beta^*, \partial_{\beta^*}) = 0$. $\partial_{\alpha^*} + \gamma u_r \partial_{\beta^*} > 0$ and $\alpha^* = 1$ implies $P_\Delta(\alpha^*, \partial_{\alpha^*}) = 0$.
2. $\partial_{\beta^*} = u_c + b_c > 0$, due to Condition 3, implies $\partial_{\beta^*} + \gamma u_c \partial_{\alpha^*} > 0$. Because the projected gradient of β^* is zero, then $\beta^* = 1$, which implies $P_\Delta(\beta^*, \partial_{\beta^*}) = 0$. $\partial_{\alpha^*} + \gamma u_r \partial_{\beta^*} = u_r + b_r + \gamma u_r(u_c + b_c) > 0$ and Condition 3 implies $\partial_{\alpha^*} = u_r + b_r > 0$, which, combined with $\alpha^* = 1$, implies $P_\Delta(\alpha^*, \partial_{\alpha^*}) = 0$.
3. $\partial_{\beta^*} = u_c + b_c < 0$. The analysis of this case is similar to the case above with $\partial_{\beta^*} > 0$, except $\beta^* = 0$.

Case 3: at least one gradient is less than zero. The proof of this case is similar to Case 2. Without loss of generality, assume $\partial_{\alpha^*} + \gamma u_r \partial_{\beta^*} < 0$, which implies $\alpha^* = 0$. Then using Condition 3 and analyzing three cases of $\partial_{\beta^*} = u_c \alpha^* + b_c = b_c$ will also get $P_\Delta(\alpha^*, \partial_{\alpha^*}) = 0$ and $P_\Delta(\beta^*, \partial_{\beta^*}) = 0$.

□

To prove IGA-PP's Nash convergence, we now will examine the dynamics of the strategy pair following IGA-PP. The strategy pair (α, β) can be viewed as a point in $\Re^2$ constrained to lie in the unit square. Using Equation 3, 6, and an infinitesimal step size, it is easy to show that the *unconstrained* dynamics of the strategy pair is defined by the following differential equation

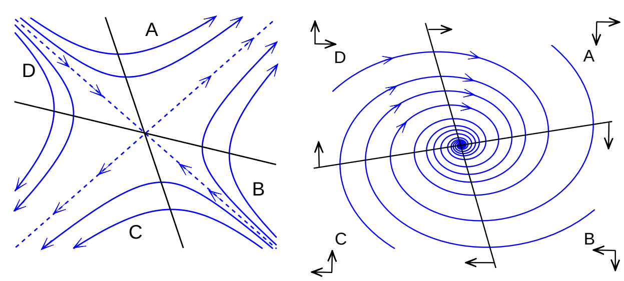
$$\begin{bmatrix} \dot\alpha \\ \dot\beta \end{bmatrix} = \begin{bmatrix} \gamma u_r u_c & u_r \\ u_c & \gamma u_c u_r \end{bmatrix} \begin{bmatrix} \alpha \\ \beta \end{bmatrix} + \begin{bmatrix} \gamma u_r b_c + b_r \\ \gamma u_c b_r + b_c \end{bmatrix} \tag{7}$$

We denote the 2×2 matrix in Equation 7 as U.

In the unconstrained dynamics, there exists at most one point of zero-gradient, which is called the center (or origin) and denoted (α^c, β^c). If the matrix U is invertible, by setting the left hand side of Equation 7 to zero, using Condition 2 (i.e., $\gamma^2 u_r u_c < 1$), and solving for the center, we get

$$(\alpha^c, \beta^c) = (\frac{-b_r}{u_r}, \frac{-b_c}{u_c}). \tag{8}$$

Note that the center is in general not at (0, 0) and may not even be in the unit square.



a) A saddle at the center   b) A stable focus at the center

Figure 1: The phase portraits of the IGA-PP dynamics: a) when U has real eigenvalues and b) when U has imaginary eigenvalues with negative real part

From dynamical systems theory (Perko 1991), if the matrix U is invertible, qualitative forms of the dynamical system specified by Equation 7 depend on eigenvalues of U, which are given by

$$\lambda_1 = \gamma u_r u_c + \sqrt{u_r u_c} \text{ and } \lambda_2 = \gamma u_r u_c - \sqrt{u_r u_c}. \quad (9)$$

If U is invertible, $u_r u_c \neq 0$. If $u_r u_c > 0$, then U has two real eigenvalues; otherwise, U has two imaginary conjugate eigenvalues with negative real part (because $\gamma > 0$). Therefore, based on linear dynamical systems theory, if U is invertible, Equation 7 has two possible phase portraits shown in Figure 1. In each diagram, there are two axes across the center. Each axis is corresponding to one player, whose strategy gradient on this axis are zero. Because $u_r, u_c \neq 0$ in Equation 7, two axes are off the horizonal or the vertical line and not orthogonal to each other. These two axes produce four quadrants.

To prove Theorem 1, we only need to show that IGA-PP always leads the strategy pair to converge a Nash equilibrium in three mutually exclusive and exhaustive cases:

- $u_r u_c = 0$, i.e., U is not invertible,
- $u_r u_c < 0$, i.e., having a *saddle* at the center,
- $u_r u_c > 0$, i.e., having a *stable focus* at the center.

Lemma 3. *If U is not invertible, for any initial strategy pair, IGA-PP (with sufficiently small γ) leads the strategy pair trajectory to converge to a Nash equilibrium.*

Proof. If U is not invertible, $\det(U) = (\gamma^2 u_r u_c - 1)u_r u_c = 0$. A sufficiently small γ will always satisfy Condition 2, i.e., $\gamma^2 u_r u_c \neq 1$. Therefore, u_r or u_c is zero. Without loss of generality, assume u_r is zero. Then the gradient for the row player is constant (See Equation 7), i.e., b_r. As a result, if $b_r = 0$, then its strategy α keeps on its initial value; otherwise, its strategy will converge to $\alpha = 0$ (if $b_r < 0$) or $\alpha = 1$ (if $b_r > 0$). After the row player's strategy α becomes a constant, due to $u_r = 0$, the column player's strategy gradient also becomes a constant. Then its strategy β stays on a value (if the gradient is zero) or converges to one or zero, depending on the sign of the gradient. According to Lemma 2, the joint strategy converges to a Nash equilibrium. □

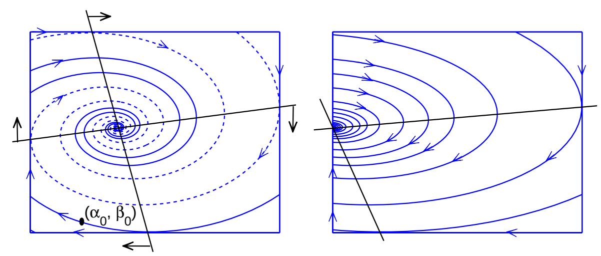
Lemma 4. *If U has real eigenvalues, for any initial strategy pair, IGA-PP leads the strategy pair trajectory to converge to a point on the boundary that is a Nash equilibrium.*

Proof. From Equation 9, real eigenvalues implies $u_r u_c > 0$. Assume $u_r > 0$ and $u_c > 0$ (the analysis for the case with $u_r < 0$ and $u_c < 0$ is analogous and omitted). In this case, the dynamics of the strategy pair has the qualitative form shown in Figure 1a.

Consider the case when the center is inside the unit square. If the initial point is at the center where the gradient is zero, it converges immediately. For an initial point in quadrant B or D, if it is on the dashed line, the trajectory will asymptotically converge to the center; otherwise, the trajectory will eventually enter either quadrant A or C. Any trajectory in quadrant A (or C) will converge to the top-right corner (or the bottom-left corner) of the unit square. Therefore, by Lemma 2, any trajectory always converges a Nash equilibrium. Cases when the center on the boundary or outside the unit square can be shown similarly to converge, and are discussed in (Singh, Kearns, and Mansour 2000). □

Lemma 5. *If U has two imaginary conjugate eigenvalues with negative real part, for any initial strategy pair, the IGA-PP algorithm leads the strategy pair trajectory to asymptotically converge to a point that is a Nash equilibrium.*

Proof. From dynamical systems theory (Perko 1991), if U has two imaginary conjugate eigenvalues with negative real part, the unconstrained dynamics of Equation 7 has a stable focus at the center, which means, starting from any point, the trajectory will asymptotically converge to the center (α^c, β^c) in a spiral way. From Equation 9, the imaginary eigenvalues implies $u_r u_c < 0$. Assume $u_r > 0$ and $u_c < 0$ (the case with $u_r < 0$ and $u_c > 0$ is analogous), whose general phase portrait is shown in Figure 1b. One observation is that the direction of the gradient of the strategy pair changes in a clockwise way through the quadrants.



a) Center within the unit square b) Center on the boundary

Figure 2: Example dynamics when U has imaginary eigenvalues with negative real part

By Lemma 2, we only need to show the strategy pair trajectory will converge a point in the constrained dynamics. We analyze three possible cases to consider depending on the location of the center (α^c, β^c).

1. **Center in the interior of the unit square.** First, we observe that all boundaries of the unit square are tangent to some spiral trajectory, and at least one boundary is tangent to a spiral trajectory, whose remaining part after the tangent point lies entirely within the unit square, e.g., two dashed trajectories in Figure 2a.
   If the initial strategy pair coincidentally is the center, it will always stay because its gradient is zero. Otherwise, the trajectory starting from the initial point either does not intersect any boundary, which will asymptotically converge to the center, or intersects with a boundary. In the latter case, when the trajectory hits a boundary, it then travels along the boundary until it reaches the point at which the boundary is tangent to some spiral, whose remaining part after the tangent point may or may not lie entirely within the unit square. If it does, then the trajectory will converge to the center along that spiral. If it does not, the trajectory will follow the tangent spiral to the next boundary in the clockwise direction. This process repeats

until the boundary is reached that is tangent to a spiral, whose remaining part after the tangent point lies entirely within the unit square. Therefore, the trajectory will eventually asymptotically converge to the center.

2. **Center on the boundary.** Consider the case where the center is on the left-side boundary of the unit square, as shown in Figure 2b. For convenience, assume the top left corner only belongs to the left boundary and the bottom left corner only belongs to the bottom boundary. If the initial strategy pair coincidentally is the center, it will always stay because of its gradient is zero. Otherwise, because of clockwise directions of the gradient, no matter where the trajectory starts, it will always finally hit the left boundary below the center, and then travels up along the left boundary and asymptotically converge to the center. A similar argument can be constructed when the center is on some other boundary of the unit square.
3. **Center outside the unit square.** In this case, the strategy trajectory will converge to some corner of the unit square depending on the location of the unit square, as discussed in (Singh, Kearns, and Mansour 2000).

□

Theorem 2. *If, in a two-person, two-action, iterated general-sum game, one player uses IGA-PP (with a sufficiently small γ) and the other player uses IGA, then their strategies will converge to a Nash equilibrium.*

The proof of this theorem is omitted, which is similar to that of Theorem 1.

A Practical Algorithm

Based on the idea of IGA-PP, we now present a new practical MARL algorithm, called Policy Gradient Ascent with approximate policy prediction (PGA-APP), shown in Algorithm 1. The PGA-APP algorithm only requires the observation of the reward of the selected action. To drop the assumptions of IGA-PP, PGA-APP needs to address the key question: how can an agent estimate its policy gradient with respect to the opponent's forecasted strategy without knowing the current strategy and the gradient of the opponent?

For clarity, let us consider the policy update rule of IGA-PP for the row player, shown by Equation 6. IGA-PP's policy gradient of the row player (i.e., $\partial_{\alpha_k} + \gamma u_r \partial_{\beta_k}$) contains two components: its own partial derivative (i.e., ∂_{α_k}) and the product of a constant and the column player's partial derivative (i.e., $\gamma u_r \partial_{\beta_k}$) with respect to the current joint strategies. PGA-APP estimates these two components, respectively.

To estimate the partial derivative with respect to the current strategies, PGA-APP uses Q-learning to learn the expected value of each action in each state. The value function $Q(s,a)$ returns the expected reward (or payoff) of executing action a in state s. The policy $\pi(s,a)$ returns the probability of taking action a in state s. As shown by Line 5 in Algorithm 1, Q-learning only uses the immediate reward to update the expected value. With the value function Q and the current policy π, PGA-APP then can calculate the partial derivative, as shown by Line 8. To illustrate that the calculation works properly, let us consider a

---

Algorithm 1: PGA-APP Algorithm

1 Let θ and η be the learning rates, ξ be the discount factor, γ be the derivative prediction length;
2 Initialize value function Q and policy π;
3 **repeat**
4 Select an action a in current state s according to policy $\pi(s,a)$ with suitable exploration ;
5 Observing reward r and next state s', update $Q(s,a) \leftarrow (1-\theta)Q(s,a)+\theta(r+\xi \max_{a'} Q(s',a'))$;
6 Average reward $V(s) \leftarrow \sum_{a \in A} \pi(s,a)Q(s,a)$;
7 **foreach** *action* $a \in A$ **do**
8 **if** $\pi(s,a)=1$ **then** $\hat{\delta}(s,a) \leftarrow Q(s,a) - V(s)$ **else** $\hat{\delta}(s,a) \leftarrow (Q(s,a)-V(s))/(1-\pi(s,a))$ **;**
9 $\delta(s,a) \leftarrow \hat{\delta}(s,a) - \gamma|\hat{\delta}(s,a)|\pi(s,a)$ **;**
10 $\pi(s,a) \leftarrow \pi(s,a) + \eta\delta(s,a)$ **;**
11 **end**
12 $\pi(s) \leftarrow \Pi_{\Delta}[\pi(s)]$;
13 **until** *the process is terminated* ;

---

two-person, two-action repeated game, where each agent has a single state. Let $\alpha = \pi_r(s,1)$ and $\beta = \pi_c(s,1)$ be the probability of the first action of the row player and the column player, respectively. Then $Q_r(s,1)$ is the expected value of the row player playing the first action, which will converge to $\beta * r_{11} + (1-\beta) * r_{12}$ by using Q-learning. It is easy to show that, when Q-learning converges, $(Q_r(s,1) - V(s))/(1-\pi_r(s,1)) = u_r\beta + b_r = \frac{\partial V_r(\alpha,\beta)}{\partial \alpha}$, which is the partial derivative of the row player (as shown by Equation 3).

Using Equation 3, we can expand the second component, $\gamma u_r \partial_{\beta_k} = \gamma u_r u_c \alpha + \gamma u_r b_c$. So it is actually a linear function of the row player's own strategy. PGA-APP approximates the second component by the term $-\gamma|\delta(s,a)|\pi(s,a)$, as shown in Line 9. This approximation has two advantages. First, when players' strategies converge to a Nash equilibrium, this approximated derivative will be zero and will not cause them to deviate from the equilibrium. Second, the negative sign of this approximation term is intended to simulate the partial derivative well for the case with $u_r u_c < 0$ (where IGA does not converge) and allows the algorithm to converge in all cases (properly small γ will allow convergence in other cases, i.e., $u_r u_c \geq 0$). Line 12 projects the adjusted policy to the valid space.

In some sense, PGA-APP extends Q-learning and is capable of learning mixed strategies. A player following PGA-APP with $\gamma < 1$ will learn an optimal policy if the other players are playing stationary policies. It is because, with a stationary environment, using Q-learning, the value function Q will converge to the optimal one, denoted by Q^*, with a suitable exploration strategy. With $\gamma < 1$, the approximate derivative term in Line 9 will never change the sign of the gradient, and policy π converges to a policy that is greedy with respect to Q. So when Q is converging to Q^*, π converges to a best response.

Learning parameters will affect the convergence of PGA-APP. For competitive games (with $u_r u_c < 0$), the larger the

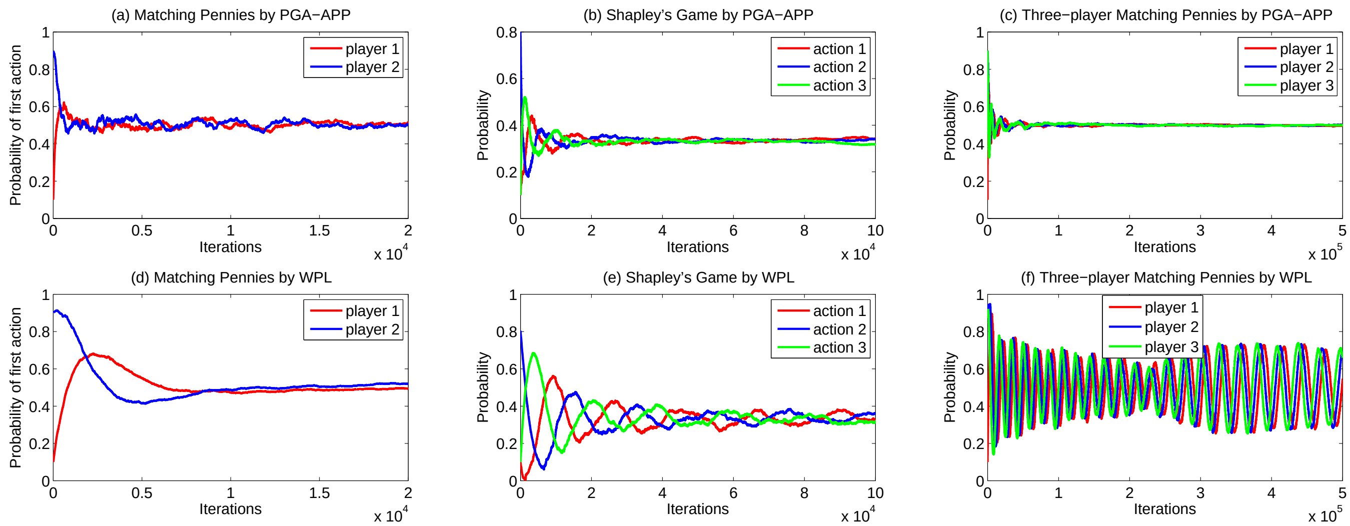


Figure 3: Convergence of PGA-APP (on the top row) and WPL (on the bottom row) in games. Plot (a), (c), (d) and (f) shows the dynamics of the probability of the first action of each player, and plot (b) and (e) shows the dynamics of the probability of each action of the first player. Parameters: $\theta = 0.8$, $\xi = 0$, $\gamma = 3$, $\eta = 5/(5000 + t)$ for PGA-APP (η is tuned and decayed slower for WPL), where t is the current number of iterations, and a fixed exploration rate = 0.05. Value function Q is initialized with zero. For two-action games, players' initial policies are (0.1, 0.9) or (0.9, 0.1), respectively, and, for three-action games, their initial policies are (0.1, 0.8, 0.1) and (0.8, 0.1, 0.1).

derivative prediction length γ, the faster the convergence. But for non-competitive games (with $u_r u_c \geq 0$), too large γ will violate Condition 3 and cause players' strategies to converge to a point that is not a Nash equilibrium. With higher learning rates θ and η, PGA-APP learns a policy faster at the early stage but the policy may oscillate at late stages. Properly decaying θ and η makes PGA-APP converge better. However, the initial value and the decay of learning rate η need to be set appropriately for the value of the learning rate θ, because we do not want to take larger policy update steps than steps with which values are updated.

Experiments: Normal-Form Games

(a) Matching Pennies	(b) Shapley's Game	(c) Three-Player Matching Pennies
$R = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$	$R = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$	Player 1 gets one dollar for matching player 2; player 2 gets one dollar for matching player 3;
$C = \begin{bmatrix} -1 & 1 \\ 1 & -1 \end{bmatrix}$	$C = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$	player 3 gets one dollar for not matching player 1; otherwise, players get zero.

Figure 4: Normal-form games.

We have evaluated PGA-APP, WoLF-PHC (Bowling and Veloso 2002), GIGA-WoLF (Bowling 2005), and WPL (Abdallah and Lesser 2008) on a variety of normal-form games. Due to space limitation, we only show results of PGA-APP and WPL in three representative benchmark games: matching pennies, Shapley's game, and three-player matching pennies, as defined in Figure 4. The results of WoLF-PHC and GIGA-WoLF have been shown and discussed in (Bowling 2005; Abdallah and Lesser 2008). As shown in Figure 3, using PGA-APP, players' strategies converge to a Nash equilibrium in all cases, including games with three players or three actions that are not covered by our formal analysis. Therefore, PGA-APP empirically has a stronger convergence property than WPL, WoLF-PHC and GIGA-WoLF, each of which does not converge in one of two games: Shapley's game and three-player matching pennies. Through experimenting with various parameter settings, we also observe that PGA-APP generally converges faster than WPL, WoLF-PHC and GIGA-WoLF. One possible reason is that, as shown in Figure 1b, the strategy trajectory following IGA-PP spirals directly into the center, while the trajectory following IGA-WoLF moves along an elliptical orbit in each quadrant and slowly approaches to the center, as discussed in (Bowling and Veloso 2002).

Experiments: Distributed Task Allocation

We used our own implementation of the distributed task allocation problem (DTAP) that was described in (Abdallah and Lesser 2008). Agents are organized in a network. Each agent may receive tasks from either the environment or its neighbors. At each time unit, an agent makes a decision for each task received during this time unit whether to execute the task locally or send it to a neighbor for processing. A task to be executed locally will be added to the local first-come-first-serve queue. The main goal of DTAP is to minimize the average total service time (ATST) of all tasks, including routing time, queuing time, and execution time.

We applied WPL, GIGA-WoLF, and PGA-APP, respectively, to learn the policy of deciding where to send a task:

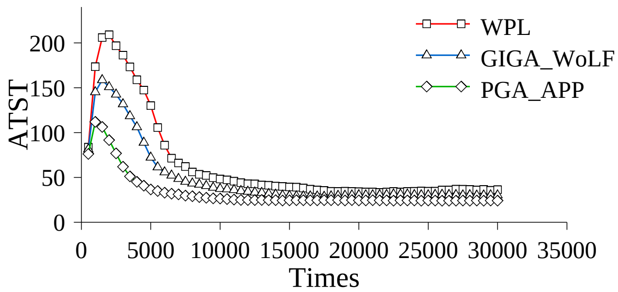


Figure 5: Performance in distributed task allocation

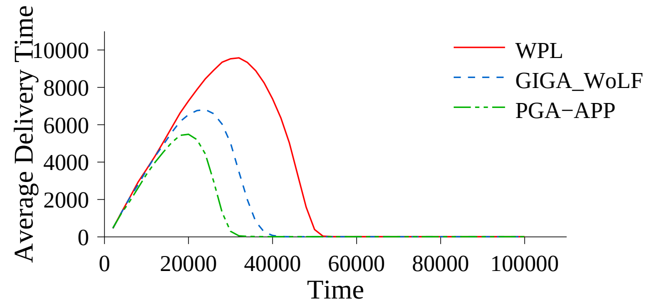


Figure 6: Performance in network routing

the local queue or one of its neighbors. The agent's state is defined by the size of the local queue, which is different from the experiments in (Abdallah and Lesser 2008) (where each agents has a single state). All algorithms use value-learning rate $\theta = 1$ and policy-learning rate $\eta = 0.0001$. PGA-APP used prediction length $\gamma = 1$.

Experiments were conducted using uniform two-dimension grid networks of agents with different sizes: 6x6, 10x10, and 18x18, and with different task arrival patterns, all of which show similar comparison results. For brevity, we only present here the results for the 10x10 grid (with 100 agents), where tasks arrive at the 4x4 sub-grid at the center at an average rate 0.5 tasks/time unit. Communication delay between two adjacent agents is one time unit. All agents can execute a task at a rate of 0.1 task/time unit.

Figure 5 shows the results of these three algorithms, all of which converge. PGA-APP converges faster and to a better ATST: WPL converges to 34.25 ± 1.46 and GIGA-WoLF to 30.30 ± 1.64, while PGA-APP converges to 24.89 ± 0.82 (results are averaged over 20 runs).

Experiments: Network Routing

We also evaluated PGA-APP in network routing. A network consists of a set of agents and links between them. Packets are periodically introduced into the network under a Poisson distribution with a random origin and destination. When a packet arrives at an agent, the agent puts it into the local queue. At each time step, an agent makes its routing decision of choosing which neighbor to forward the top packet in the queue. Once a packet reaches its destination, it is removed from the network. The main goal in this problem is to minimize the Average Delivery Time (ADT) of all packets.

We used the experimental setting that was described in (Zhang, Abdallah, and Lesser 2009). The network is a 10x10 irregular grid with some removed edges. The time cost of sending a packet down a link is a unit cost. The packet arrival rate to the network is 4. Each agent uses the learning algorithm to learn its routing policy.

Figure 6 shows the results of applying WPL, GIGA-WoLF, and PGA-APP to this problem. All three algorithms demonstrate convergence, but PGA-APP converges faster and to a better ADT: WPL converges to 11.60 ± 0.29 and GIGA-WoLF to 10.22 ± 0.24, while PGA-APP converges to 9.86 ± 0.29 (results are averaged over 20 runs).

Conclusion

We first introduced IGA-PP, a new gradient-based algorithm, augmenting the basic gradient ascent algorithm with policy prediction. We proved that, in two-player, two-action, general-sum matrix games, IGA-PP in self-play or against IGA would lead players' strategies to converge to a Nash equilibrium. Inspired by IGA-PP, we then proposed PGA-APP, a new practical MARL algorithm, only requiring the observation of an agent's local reward for selecting an specific action. Empirical results in normal-form games, distributed task allocation problem and network routing showed that PGA-APP converged faster and in a wider variety of situations than state-of-the-art MARL algorithms.

References

Abdallah, S., and Lesser, V. 2008. A multiagent reinforcement learning algorithm with non-linear dynamics. *Journal of Artificial Intelligence Research* 33:521–549.

Banerjee, B., and Peng, J. 2007. Generalized multiagent learning with performance bound. *Autonomous Agents and Multi-Agent Systems* 15(3):281–312.

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

Bowling, M. 2005. Convergence and no-regret in multiagent learning. In *NIPS'05*, 209–216.

Conitzer, V., and Sandholm, T. 2007. Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. *Machine Learning* 67(1):23–43.

Hu, J., and Wellman, M. P. 2003. Nash q-learning for general-sum stochastic games. *Journal of Machine Learning Research* 4:1039–1069.

Perko, L. 1991. *Differential equations and dynamical systems*. Springer-Verlag Inc.

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

Zhang, C.; Abdallah, S.; and Lesser, V. 2009. Integrating organizational control into multi-agent learning. In *AAMAS'09*.