2011 American Control Conference
on O'Farrell Street, San Francisco, CA, USA
June 29 - July 01, 2011

Decentralized Learning in Two-Player Zero-Sum Games: A L_{R-I} Lagging Anchor Algorithm

Xiaosong Lu and Howard M. Schwartz

***Abstract*— This paper presents a L_{R-I} lagging anchor algorithm that combines a lagging anchor method to the L_{R-I} learning algorithm. We prove that this decentralized learning algorithm converges in strategies to a Nash equilibrium in two-player, zero-sum, two-action matrix games, while only needing knowledge of their own action and reward.**

I. Introduction

Multi-agent learning algorithms have received considerable attention over the past two decades [1] [2]. Among multi-agent learning algorithms, decentralized learning algorithms have become an attractive research field. Without using a centralized agent to control the learning process of all the agents, each agent will use a decentralized learning algorithm to learn its "optimal" strategy. Decentralized learning algorithms can be applied for players to learn their Nash equilibria (NE) in games with incomplete information. Incomplete information means that one player in the game may not know its reward function, the other players' strategies or their actions. The player only knows its own action and the received reward at each time step. The main challenge for designing a decentralized learning algorithm with incomplete information is to prove that the players' strategies converge to the Nash equilibrium when all the players apply the same learning algorithm.

There are a number of decentralized learning algorithms proposed in the literature which can be applied for two-player zero-sum games. A learning automata approach called linear reward-inaction approach is proposed in [3] [4] that can guarantee the convergence to the Nash equilibrium under the assumption that the matrix game only has strict Nash equilibrium in pure strategies. The linear reward-penalty approach [5] can guarantee that the expected value of the players' strategies converge to Nash equilibrium in fully mixed strategies with proper choice of certain parameters. But the convergence of the player's strategy itself has not been proved in [5]. Bowling and Veloso proposed WoLF-IGA approach that can guarantee the convergence to the Nash equilibrium for two-player two-action matrix games where the Nash equilibrium can be in fully mixed strategies or pure strategies. However, the WoLF-IGA approach is not a completely decentralized learning algorithm since the player has to know its own reward matrix and its opponent's strategy at each time step. Dahl [6] proposed the lagging anchor model approach that can guarantee the convergence to the Nash equilibrium in fully mixed strategies. But the lagging anchor algorithm is not a decentralized learning algorithm since each player has to know its reward matrix.

In this paper, we investigate these learning algorithms and propose a decentralized learning algorithm called the L_{R-I} lagging anchor algorithm which is the combination of learning automata and lagging anchor algorithms. Unlike the lagging anchor algorithm, the L_{R-I} lagging anchor algorithm is a completely decentralized algorithm where each player only needs its own action and the received reward at each time step. We also prove the convergence of the L_{R-I} lagging anchor algorithm to a Nash equilibrium in two-player two-action zero-sum matrix games while the Nash equilibrium in the game can be in pure or fully mixed strategies.

The main contributions of this paper are

- Design a decentralized learning algorithms called L_{R-I} Lagging Anchor Algorithm
- Prove the convergence of the designed L_{R-I} lagging anchor algorithm to the Nash equilibrium in two-player two-action zero-sum matrix games.

II. Decentralized Learning Algorithms for Two-Player Zero-Sum Matrix Games

A matrix game is a tuple $(n, A_1, ...A_n, R_1, ...R_n)$ where n is the number of the players, $A_i(i = 1, ..., n)$ is the action set for the player i and $R_i : A_1 \times \cdots \times A_n \rightarrow \mathbb{R}$ is the reward function for the player i. A matrix game is a game involving multiple players and a single state. Each player $i(i = 1, ..., n)$ selects an action from its action set $A_i = \{a_1^i, a_2^i, \cdots, a_{m_i}^i\}$ and receives a reward. The player i's reward function R_i is determined by all the players' joint action from joint action space $A_1 \times \cdots \times A_n$.

For a two player matrix game, we can set up a matrix with each element containing a reward for each joint action pair. Then the reward function R_i for player $i(i = 1, 2)$ becomes a matrix. If the two players are fully competitive, we will have a two-player zero-sum matrix game with $R_1 = -R_2$. In a matrix game, each player tries to maximize his own expected reward based on the player's strategy. A player's strategy is defined as a probability distribution over the player's action set. To evaluate a player's strategy, we have the following concept of Nash equilibrium. A Nash equilibrium is a collection of strategies for all the players such that no player can do better by changing its own strategy given that the other players continue playing their

X. Lu is with the Department of Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa, ON, Canada luxiaos@sce.carleton.ca

H. M. Schwartz is with the Department of Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa, ON, Canada schwartz@sce.carleton.ca

Nash equilibrium strategies. A Nash equilibrium can be in fully mixed strategies or pure strategies. If the probability of any action from the action set is greater than 0, then the player's strategy is called a fully mixed strategy. If the player selects one action with probability of 1 and other actions with probability of 0, then the player's strategy is called a pure strategy. A Nash equilibrium is called a strict Nash equilibrium in pure strategies if each player's equilibrium action is better than all its other actions, given the other players' actions [7].

We take the matching pennies game for example. Each player has two actions: Heads or Tails. If both players choose the same action, then player 1 gets a reward 1 and player 2 gets a reward -1. If the actions are different, then player 1 gets -1 and player 2 gets 1. Based on the reward matrix in Table I (a), the equilibrium strategy for each player is playing Heads and Tails with half probability each. Then we modify the matching pennies game and let player 1 get a reward of 2 when player 1 plays Heads and player 2 plays Tails. Based on the reward matrix for the modified matching pennies game in Table I (b), the equilibrium strategy for each player is playing Heads. In this modified matching pennies game, the Nash equilibrium is a strict Nash equilibrium in pure strategies.

Learning in a two-player zero-sum matrix game can be expressed as the process of the player updating its strategy according to the received reward from the environment. A learning scheme is used for each player to update its own strategy toward the Nash equilibrium based on the information from the environment. A decentralized learning algorithm requires the least information from the environment including the player's own action and the received reward from the environment. In the literature, decentralized learning algorithms for two-player zero-sum matrix games can be divided into two groups. One group is based on learning automata scheme and another group is based on gradient ascent methods. In this section, we present four existing learning algorithms based on the above two groups of learning techniques.

A. Learning Automata

learning automation is a learning unit for adaptive decision making in an unknown environment [4]. The objective of the learning automation is to learn the optimal action or strategy by updating its action probability distribution based on the environment response. The learning automata approach is a completely decentralized learning algorithm since each learner only considers its action and payoff from the environment and ignores any information from other agents such as the actions taken by other agents. The learning automation can be represented as a tuple (A, r, p, U) where $A = \{a_1, \cdots, a_m\}$ is the player's action set, $r \in [0, 1]$ is the reinforcement signal, p is the probability distribution over the actions and U is the learning algorithm to update p. There are two typical learning algorithms based on learning automata: linear reward-inaction (L_{R-I}) algorithm and linear reward-penalty (L_{R-P}) algorithm [4].

1) Linear Reward-Inaction Algorithm: The linear reward-inaction (L_{R-I}) algorithm for player $i (i = 1, ..., n)$ is defined as follows [4]

$$\begin{aligned} p_c^i(k+1) &= p_c^i(k) + \eta r^i(k)(1 - p_c^i(k)) \\ &\quad \text{(if } a_c \text{ is the current action taken at time } k) \\ p_j^i(k+1) &= p_j^i(k) - \eta r^i(k) p_j^i(k) \\ &\quad \text{(for all } a_j^i \neq a_c^i) \end{aligned} \quad (1)$$

where k is the time step, the superscripts and subscripts on p denote different player and each player's different action respectively, $0 < \eta < 1$ is the learning parameter, $r^i(k)$ is the response of the environment given the player i's action a_c^i at k and p_c^i is the probability distribution over player i's action $a_c^i (c = 1, \cdots, m)$.

In a matrix game with n players, if each player uses the L_{R-I} algorithm, then the L_{R-I} algorithm guarantees the convergence to a Nash equilibrium under the assumption that the game only has strict Nash equilibria in pure strategies [4].

2) Linear Reward-Penalty Algorithm: The linear reward-penalty (L_{R-P}) algorithm for player i is defined as follows [4]

$$\begin{aligned} p_c^i(k+1) &= p_c^i(k) + \eta_1 r^i(k)[1 - p_c^i(k)] - \eta_2[1 - \\ &\quad r^i(k)]p_c^i(k) \text{ (if } a_c^i \text{ is the current action)} \\ p_j^i(k+1) &= p_j^i(k) - \eta_1 r^i(k) p_j^i(k) + \eta_2[1 - \\ &\quad r^i(k)][\frac{1}{m-1} - p_j^i(k)] \text{ (for all } a_j^i \neq a_c^i) \end{aligned} \quad (2)$$

where $0 < \eta_1, \eta_2 < 1$ are learning parameters and m is the number of actions in the player's action set.

With the careful choice of the above two learning parameters, the linear reward-penalty algorithm can be applied to the two-player zero-sum game involving only Nash equilibrium in fully mixed strategies [4]. In a two-player zero-sum matrix game, if each player uses the L_{R-P} and chooses $\eta_2 < \eta_1$, the expected value of the fully mixed strategies for both players can be made arbitrarily close to the Nash equilibrium [4]. This means that the L_{R-P} algorithm can guarantee the convergence to the Nash equilibrium in the sense of expected value, but not the player's strategy itself.

TABLE I
TWO EXAMPLES OF TWO-PLAYER ZERO-SUM MATRIX GAMES

(a) Matching pennies game	(b) Modified matching pennies game
$R_1 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}, R_2 = -R_1$	$R_1 = \begin{bmatrix} 1 & 2 \\ -1 & 1 \end{bmatrix}, R_2 = -R_1$
NE in fully mixed strategies	NE in pure strategies

B. Gradient Ascent Learning

Gradient ascent learning can be used to update the player's strategy in a matrix game. In a gradient ascent learning algorithm, the player's strategy can be updated in the direction to the current gradient with a small step size η [8]. At each iteration, the player will adjust its strategy based

on its gradient in order to increase its expected payoff. Using a gradient ascent learning algorithm, [8] shows that the players' strategies do not converge to the Nash equilibrium for general cases. But the expected payoffs of the players converge to the Nash equilibrium. In the literature, there is a number of gradient ascent learning algorithms that can guarantee the convergence to the Nash equilibrium for some specific matrix games [9] [10]. In this section, we introduce two gradient ascent learning algorithms and analyzes them from the point of view of decentralized learning.

1) WoLF-IGA Algorithm: Win or learn fast-infinitesimal gradient ascent (WoLF-IGA) algorithm is presented by Bowling and Veloso [9] for two-player two-action matrix games. As a gradient ascent learning algorithm, the WoLF-IGA algorithm allows the player to update its strategy based on the current gradient and a variable learning rate. The value of the learning rate when the player is winning is smaller than the value of the learning rate when the player is losing. We assume p_1 is the probability of the player 1 choosing the first action. Then $1-p_1$ is the probability of the player 1 choosing the second action. Accordingly, q_1 is the probability of the player 2 choosing the first action and $1-q_1$ is the probability of the player 2 choosing the second action. The updating rules of the WoLF-IGA are listed as follows [9]

$$p_1(k+1) = p_1(k) + \eta l^1(k) \frac{\partial V^1(p_1(k), q_1(k))}{\partial p_1} \quad (3)$$

$$q_1(k+1) = q_1(k) + \eta l^2(k) \frac{\partial V^2(p_1(k), q_1(k))}{\partial q_1} \quad (4)$$

$$l^1(k) = \begin{cases} l_{min}, \text{ if } V^1(p_1(k), q_1(k)) > V^1(p_1^*, q_1(k)) \\ l_{max}, \text{ otherwise} \end{cases}$$

$$l^2(k) = \begin{cases} l_{min}, \text{ if } V^2(p_1(k), q_1(k)) > V^2(p_1(k), q_1^*) \\ l_{max}, \text{ otherwise} \end{cases}$$

where η is the step size, $l^i (i=1,2)$ is the learning rate for player $i (i=1,2)$, $V^i(p_1(k), q_1(k)) (i=1,2)$ is the expected payoff of the player i at time k given the current two players' strategy pair $(p_1(k), q_1(k))$, and (p_1^*, q_1^*) are equilibrium strategies for the players. In a two-player two-action matrix game, if each player uses the WoLF-IGA algorithm with $l_{max} > l_{min}$, the players' strategies will converge to Nash equilibrium as the step size $\eta \to 0$ [9].

This algorithm provides a gradient ascent learning algorithm that can guarantee the convergence to a Nash equilibrium in fully mixed or pure strategies for two-player two-action matrix games. However, this algorithm is not a decentralized learning algorithm. In order to compute the partial derivative of $V^i (i=1,2)$ with respect to p_1 or q_1 at time k, we need to know the player i's reward matrix and its opponent's strategy at time k. This learning algorithm also requires the knowledge of $V^1(p_1^*, q_1(k))$ and $V^2(p_1(k), q_1^*)$ in order to choose the learning parameter between l_{min} and l_{max} accordingly. With a qualified decentralized learning algorithm in matrix games, the only information a player can get is its own action and received reward. Therefore, the WoLF-IGA algorithm does not satisfy the requirement of designing a decentralized learning algorithm for matrix games. Although Bowling and Veloso provide a practical decentralized learning algorithm called WoLF policy hill-climbing based on the WoLF-IGA algorithm in [9], there is no proof of convergence to the Nash equilibrium for this practical algorithm.

2) the Lagging Anchor Algorithm: The lagging anchor algorithm for two-player zero-sum games was introduced by Dahl [10] [6]. As a gradient ascent learning method, the lagging anchor algorithm updates the players' strategies according to the gradient. We denote player 1's strategy as a vector $\mathbf{v} = [p_1, p_2, \cdots, p_{m_1}]^T$ which is the probability distribution over all the possible actions. Accordingly, the player 2's strategy is a vector $\mathbf{w} = [q_1, q_2, \cdots, q_{m_2}]^T$. The updating rules are listed as follows [6]

$$\begin{aligned} \mathbf{v}(k+1) &= \mathbf{v}(k) + \eta \mathbf{P}_{m_1} R_1 Y(k) + \eta\gamma(\bar{\mathbf{v}}(k) - \mathbf{v}(k)) \\ \bar{\mathbf{v}}(k) &= \bar{\mathbf{v}}(k) + \eta\gamma(\mathbf{v}(k) - \bar{\mathbf{v}}(k)) \\ \mathbf{w}(k+1) &= \mathbf{w}(k) + \eta \mathbf{P}_{m_2} R_2 X(k) + \eta\gamma(\bar{\mathbf{w}}(k) - \mathbf{w}(k)) \\ \bar{\mathbf{w}}(k) &= \bar{\mathbf{w}}(k) + \eta\gamma(\mathbf{w}(k) - \bar{\mathbf{w}}(k)) \end{aligned} \quad (5)$$

where η is the step size, $\gamma > 0$ is the anchor drawing factor, $\mathbf{P}_{m_i} = \mathbf{I}_{m_i} - (1/m_i)\mathbf{1}_{m_i}\mathbf{1}_{m_i}^T$ is a matrix used to maintain the summation of the elements in the vector $\mathbf{v}$ or $\mathbf{w}$ being one. $Y(k)$ is a unit vector corresponding to the actions of the player 2. If the Ith action in the player 2's action set is selected at time k, then the Ith element in $Y(k)$ is set to 1 and the other elements in $Y(k)$ are zeros. Similarly, $X(k)$ is the unit vector corresponding to the actions of player 1. R_1 and R_2 are the reward matrices for player 1 and 2 respectively. In a two-player zero-sum game with only Nash equilibrium in fully mixed strategies, if each player uses the lagging anchor algorithm, then the players' strategies converge to a Nash equilibrium as the step size $\eta \to 0$ [6].

In this algorithm, the limitation on each player's actions to be two actions is removed and the convergence to the Nash equilibrium in fully mixed strategies is guaranteed. However, the convergence to the Nash equilibrium in pure strategies in this algorithm has never been discussed. And the lagging anchor algorithm in (5) requires full information of the player's reward matrices R_1 and R_2. For a qualified decentralized learning algorithm, the reward matrix is unknown to the players during learning. Therefore, this algorithm is not a decentralized learning algorithm.

We illustrate the applicability of these algorithms based on the allowable number of actions for each player, the convergence to pure strategies or fully mixed strategies and the level of decentralization, given in Table II.

III. L_{R-I} Lagging Anchor Algorithm

In this section, we design a L_{R-I} lagging anchor algorithm that can be a completely decentralized learning algorithm and also guarantee the convergence to the Nash equilibrium in both pure and fully mixed strategies. The idea of the L_{R-I} lagging anchor algorithm is that we consider both the player's current strategy and the long-term average of the player's previous strategies at the same time. Then the

TABLE II

COMPARISON OF LEARNING ALGORITHMS IN TWO-PLAYER ZERO-SUM MATRIX GAMES

Applicability	Algorithms			
	Linear reward-inaction	Linear reward-penalty	WoLF-IGA	The lagging anchor
allowable actions for each player	no limit	2 actions	2 actions	no limit
Convergence to the pure-strategy NE or fully mixed NE	pure NE	fully mixed NE in the sense of expected value	both	fully mixed NE
Decentralization	decentralized	decentralized	not decentralized	not decentralized

player's current strategy and the long-term average will be drawn towards the equilibrium point during learning.

We take the L_{R-I} defined in (1) as the updating law of the player's strategy and add the lagging anchor term in (5) to L_{R-I}. Then the L_{R-I} lagging anchor algorithm for player i is defined as follows

$$\begin{aligned} p_c^i(k+1) &= p_c^i(k) + \eta r^i(k)(1-p_c^i(k)) + \eta(\bar{p}_c^i(k) \\ &\quad -p_c^i(k)) \text{ (if } a_c^i \text{ is the action taken at time } k) \\ \bar{p}_c^i(k+1) &= \bar{p}_c^i(k) + \eta(p_c^i(k) - \bar{p}_c^i(k)) \\ p_j^i(k+1) &= p_j^i(k) - \eta r^i(k) p_j^i(k) + \eta(\bar{p}_j^i(k) - p_j^i(k)) \\ &\quad \text{(for all } a_j^i \neq a_c^i) \\ \bar{p}_j^i(k+1) &= \bar{p}_j^i(k) + \eta(p_j^i(k) - \bar{p}_j^i(k)) \end{aligned} \tag{6}$$

where η is the step size.

To analyze the above L_{R-I} lagging anchor algorithm, we use the ODE method such that the behavior of the learning algorithm can be approximated by ordinary differential equation as the learning rate goes to zero [4]. For the L_{R-I} part in this algorithm, the equivalent ODE can be given as [4]

$$\dot{p}_c^i = \sum_{j=1}^{m_i} p_c^i p_j^i (d_c^i - d_j^i) \tag{7}$$

where d_c^i is the expected reward given that the player i is choosing action a_c^i and the other players are following their current strategies.

Based on the above ODE for L_{R-I}, we can find the equivalent ODE for our L_{R-I} lagging anchor algorithm

$$\begin{aligned} \dot{p}_c^i &= \sum_{j=1}^{m_i} p_c^i p_j^i (d_c^i - d_j^i) + (\bar{p}_c^i - p_c^i) \\ \dot{\bar{p}}_c^i &= p_c^i - \bar{p}_c^i \end{aligned} \tag{8}$$

Based on the above L_{R-I} lagging anchor algorithm, we now introduce the following theorem.

Theorem 1: Consider a two-player two-action zero-sum matrix game and each player uses the L_{R-I} lagging anchor algorithm. If we assume the game only has Nash equilibrium in mixed strategies or strict Nash equilibrium in pure strategies, then the following is true.

- All Nash equilibria are asymptotically stable.
- Any equilibrium point which is not a Nash equilibrium is unstable.

Proof: We first define the player's reward matrix

$$R_1 = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}, \; R_2 = -R_1 \tag{9}$$

We denote p_1 as the probability of player 1 taking action 1 and q_1 as the probability of player 2 taking action 1. Then the L_{R-I} lagging anchor algorithm becomes

$$\begin{aligned} \dot{p}_1 &= \sum_{j=1}^{2} p_1 p_j (d_1^1 - d_j^1) + (\bar{p}_1 - p_1) \\ \dot{\bar{p}}_1 &= p_1 - \bar{p}_1 \\ \dot{q}_1 &= \sum_{j=1}^{2} q_1 q_j (d_1^2 - d_j^2) + (\bar{q}_1 - q_1) \\ \dot{\bar{q}}_1 &= q_1 - \bar{q}_1 \end{aligned} \tag{10}$$

where $d_1^1 = r_{11}q_1 + r_{12}(1-q_1)$ and $d_2^1 = r_{21}q_1 + r_{22}(1-q_1)$. Since our game is a two-player zero-sum game, we can get $d_1^2 = -r_{11}p_1 - r_{21}(1-p_1)$ and $d_2^2 = -r_{12}p_1 - r_{22}(1-p_1)$. Then (10) becomes

$$\begin{aligned} \dot{p}_1 &= p_1(1-p_1)[uq_1 + r_{12} - r_{22}] + (\bar{p}_1 - p_1) \\ \dot{\bar{p}}_1 &= p_1 - \bar{p}_1 \\ \dot{q}_1 &= q_1(1-q_1)[-up_1 - r_{21} + r_{22}] + (\bar{q}_1 - q_1) \\ \dot{\bar{q}}_1 &= q_1 - \bar{q}_1 \end{aligned} \tag{11}$$

where $u = r_{11} - r_{12} - r_{21} + r_{22}$. If we let the right hand side of the above equation equal to zero, we can get the equilibrium points of the above equations as $(p_1^*, q_1^*) = (0,0), (0,1), (1,0), (1,1), ((r_{22}-r_{21})/u, (r_{22}-r_{12})/u)$ .To study the stability of the above learning dynamics, we use linear approximation of the above equations around the equilibrium point $(p_1^*, q_1^*, p_1^*, q_1^*)$ and the linearization matrix J is given as

$$J = \begin{bmatrix} (1-2p_1^*)(uq_1^* + r_{12} - r_{22}) - 1 & 1 & p_1^*(1-p_1^*)u & 0 \\ 1 & -1 & 0 & 0 \\ -q_1^*(1-q_1^*)u & 0 & (1-2q_1^*)(-up_1^* - r_{21} + r_{22}) - 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \tag{12}$$

The above linearization matrix can be analyzed under the following two cases.

Case 1: Nash equilibrium in pure strategies

We first consider the game only has strict Nash equilibrium in pure strategies. Without loss of generality, we assume that the Nash equilibrium in this case is $p_1^* = 1$ and $q_1^* = 1$. This means that each player is choosing action 1 from its action set as its "optimal" strategy. Then the above linearization matrix becomes

$$J = \begin{bmatrix} -c_1 - 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -c_2 - 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \quad (13)$$

where $c_1 = r_{11} - r_{21}$ and $c_2 = -r_{11} + r_{12}$. The eigenvalues of the above matrix are $0.5[-(c_1 + 2) \pm \sqrt{c_1^2 + 4})]$ and $0.5[-(c_2 + 2) \pm \sqrt{c_2^2 + 4})]$. According to the definition of a strict Nash equilibrium in [11], if the Nash equilibrium strategies are both players' first actions, we can get

$$r_{21} < r_{11} < r_{12}. \quad (14)$$

Then we can get $c_1 > 0, c_2 > 0, c_1 + 2 > \sqrt{c_1^2 + 4}$ and $c_2 + 2 > \sqrt{c_2^2 + 4}$. This results in four negative real eigenvalues for the linearization matrix. Therefore, the Nash equilibrium point $(1, 1)$ is asymptotically stable.

We now test the other equilibrium points. We first consider the equilibrium point $((r_{22} - r_{21})/u, (r_{22} - r_{12})/u)$ where $u = r_{11} - r_{12} - r_{21} + r_{22}$. To be a valid point in the probability space (unit square), the equilibrium point must satisfy

$$\begin{cases} 0 \le (r_{22} - r_{21})/u \le 1 \\ 0 \le (r_{22} - r_{12})/u \le 1 \end{cases} \Rightarrow$$
$$\begin{cases} r_{11} \ge r_{12} \\ r_{22} \ge r_{21} \\ r_{11} \ge r_{21} \\ r_{22} \ge r_{12} \end{cases} \text{if } u > 0; \begin{cases} r_{11} \le r_{12} \\ r_{22} \le r_{21} \\ r_{11} \le r_{21} \\ r_{22} \le r_{12} \end{cases} \text{if } u < 0 \quad (15)$$

The above result conflicts with the inequality in (14). Therefore, the equilibrium point $((r_{22} - r_{21})/u, (r_{22} - r_{12})/u)$ is out of the probability space. We now try the equilibrium point $(0, 1)$. We substitute $(0, 1)$ into (12) and get

$$J = \begin{bmatrix} -c_1' - 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -c_2' - 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \quad (16)$$

where $c_1' = -c_1$ and $c_2' = r_{22} - r_{21}$. Based on the similar analysis as we did for the Nash equilibrium point, in order to obtain the negative eigenvalues for the above matrix, we must have $c_1' > 0$ and $c_2' > 0$. Since we have $c_1' = -c_1$, we get $c_1 < 0 \Rightarrow r_{11} < r_{21}$ which conflicts with the inequality in (14). Therefore, the equilibrium point $(0, 1)$ is unstable. Based on the similar analysis, we can get the equilibrium points $(1, 0), (0, 0)$ are unstable as well. We ignore the details in this paper.

Thus, the strict Nash equilibrium in pure strategies is asymptotically stable while any equilibrium point which is not a Nash equilibrium is unstable. Therefore, the solution of (11) converges exponentially to the strict Nash equilibrium in pure strategies $(1, 1, 1, 1)$.

Case 2: Nash equilibrium in fully mixed strategies

We consider the game only has Nash equilibrium in fully mixed strategies. According to [11], the Nash equilibrium in fully mixed strategies for a two-player two-action zero-sum matrix game can be found as

$$(p_1^*, 1 - p_1^*) = \frac{JR_1^*}{JR_1^* J^T}, \ (q_1^*, 1 - q_1^*) = \frac{R_1^* J^T}{JR_1^* J^T}. \quad (17)$$

where R_1^* is the adjoint of R_1 and $J = (1, 1)$. Based on (17) and (9), we can get $p_1^* = (r_{22} - r_{21})/u$ and $q_1^* = (r_{22} - r_{12})/u$ $(u \neq 0)$. Then we substitute (p_1^*, q_1^*) into (12) and get

$$J = \begin{bmatrix} -1 & 1 & p_1^*(1 - p_1^*)u & 0 \\ 1 & -1 & 0 & 0 \\ -q_1^*(1 - q_1^*)u & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \quad (18)$$

The characteristic equation of the above matrix is

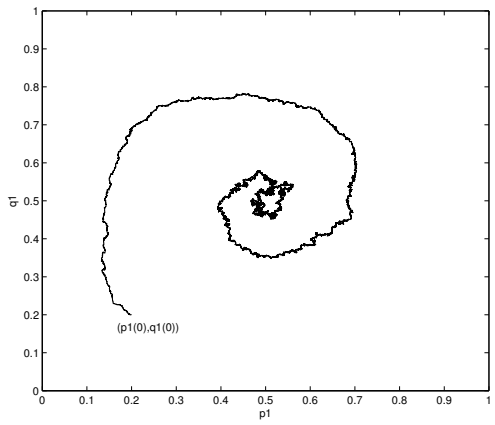
$$\lambda^4 + 4\lambda^3 + (4 + c)\lambda^2 + 2c\lambda + c = 0 \quad (19)$$

where $c = q_1^*(1 - q_1^*)p_1^*(1 - p_1^*)u^2$. Since all the coefficients of (19) are positive, we use Routh-Hurwitz stability criterion to analyze the locations of the roots in (19). We set up the Routh table as follows

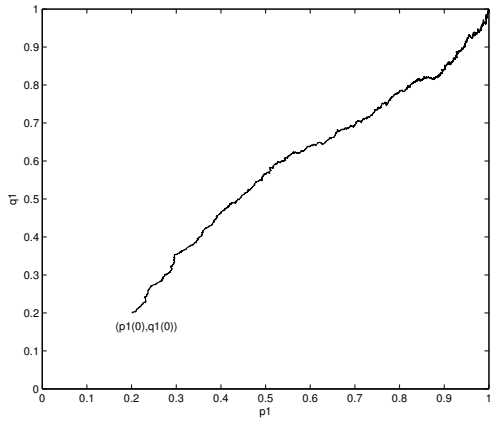
λ^4	1	4+c	c
λ^3	4	2c	
λ^2	4+0.5c	c	
λ^1	$(c^2 + 4c)/(4 + 0.5c)$		
λ^0	c		

where all the elements in the first column are positive. Therefore, all the roots in (19) have negative real part. Thus the Nash equilibrium in this case is asymptotically stable.

We now test the other equilibrium points. We try the equilibrium point $(0, 1)$. The linearization matrix around the equilibrium point $(0, 1)$ is given in (16). The condition of stability for this matrix is $c_1' > 0, c_2' > 0 \Rightarrow r_{11} < r_{21}, r_{22} > r_{21}$. Since the Nash equilibrium in this case is in fully mixed strategies, we have $p_1^* = (r_{22} - r_{21})/u$ and $q_1^* = (r_{22} - r_{12})/u$ which must satisfy the inequalities in (15). But the condition of $r_{11} < r_{21}, r_{22} > r_{21}$ conflicts with the inequalities in (15). Thus the equilibrium point $(0, 1)$ is unstable in this case. Following the same procedure, we can find that the equilibrium points $(0, 0), (1, 0), (1, 1)$ are unstable in this case. We ignore the details in this paper. Then we can conclude that the Nash equilibrium in fully



(a) Matching pennies game



(b) Modified matching pennies game

Fig. 1. Trajectories of the players' strategies during learning

mixed strategies is asymptotically stable while the other equilibrium points are unstable. Thus the solution of (11) converges exponentially to the Nash equilibrium in fully mixed strategies in case 2.

■

We now simulate the matrix games in Table I to show the performance of the proposed L_{R-I} lagging anchor algorithm. We set the step size $\eta = 0.001$ in (6) and $p_1(0) = q_1(0) = 0.2$. We run the simulation for 30000 iterations. In Fig. 1(a), the players' strategies start from (0.2,0.2) and move towards the equilibrium point (0.5,0.5) as the learning proceeds for the matching pennies game. Figure 1(b) shows that the players' strategies will converge to the "optimal" actions if the game has Nash equilibrium in pure strategies in the modified matching pennies game.

IV. Conclusion

In this paper, we investigate the existing learning algorithms for two-player zero-sum matrix games. The analysis of the learning algorithms shows that the learning automata technique including L_{R-I} and L_{R-P} methods is a good candidate for decentralized learning algorithms. The L_{R-I} learning algorithm can be applied for the game with only Nash equilibrium in pure strategies. The L_{R-P} algorithm can only guarantee the expected value of the players' strategies converge to Nash equilibrium. Inspired by the concept of lagging anchor, we propose a L_{R-I} lagging anchor algorithm as a completely decentralized learning algorithm in this paper. We prove that the L_{R-I} lagging anchor algorithm can guarantee the convergence to the Nash equilibrium in pure or fully mixed strategies in two-player two-action zero-sum matrix games.

The L_{R-I} lagging anchor algorithm can be a good candidate of designing a decentralized learning algorithm for more sophisticated games. In the future, we will prove the convergence of this algorithm to the Nash equilibrium for the player with more than 2 actions in the game. Not only for matrix games, we will also apply the L_{R-I} lagging anchor algorithm to Markov games and prove the convergence of the algorithm to the Nash equilibrium in the game.

References

[1] P. Stone and M. Veloso, "Multiagent systems: A survey from a machine learning perspective," *Autonomous Robots*, vol. 8, no. 3, pp. 345–383, 2000.

[2] L. Buşoniu, R. Babuška, and B. D. Schutter, "A comprehensive survey of multiagent reinforcement learning," *IEEE Trans. Syst., Man, Cybern. C*, vol. 38, no. 2, pp. 156–172, 2008.

[3] P. Sastry, V. Phansalkar, and M. Thathachar, "Decentralized learning of Nash equilibria in multi-person stochastic games with incomplete information," *IEEE Trans. Syst., Man, Cybern.*, vol. 24, no. 5, pp. 769–777, 1994.

[4] M. Thathachar and P. Sastry, *Networks of Learning Automata: Techniques for Online Stochastic Optimization*. Boston, Massachusetts: Kluwer Academic Publishers, 2004.

[5] S. Lakshmivarahan and K. S. Narendra, "Learning algorithms for two-person zero-sum stochastic games with incomplete information: a unified approach," *SIAM Journal on Control and Optimization*, vol. 20, no. 4, pp. 541–552, 1982.

[6] F. A. Dahl, "The lagging anchor model for game learning — a solution to the crawford puzzle," *Journal of Economic Behavior & Organization*, vol. 57, pp. 287–303, 2005.

[7] M. J. Osborne, *An Introduction to Game Theory*. Oxford University Press, USA, 2003.

[8] S. P. Singh, M. J. Kearns, and Y. Mansour, "Nash convergence of gradient dynamics in general-sum games," in *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, San Francisco, CA, USA, 2000, pp. 541–548.

[9] M. Bowling and M. Veloso, "Multiagent learning using a variable learning rate," *Artificial Intelligence*, vol. 136, no. 2, pp. 215–250, 2002.

[10] F. A. Dahl, "The lagging anchor algorithm: reinforcement learning in two-player zero-sum games with imperfect information," *Machine Learning*, vol. 49, pp. 5–37, 2002.

[11] G. Owen, *Game Theory*. San Diego, CA: Academic Press, 1995.