International Journal of Innovative
Computing, Information and Control
Volume **9**, Number **1**, January **2013**

ICIC International ©2013 ISSN 1349-4198
pp. 17–32

DECENTRALIZED LEARNING IN GENERAL-SUM MATRIX GAMES: AN L_{R-I} LAGGING ANCHOR ALGORITHM

Xiaosong Lu and Howard M. Schwartz

Department of Systems and Computer Engineering
Carleton University
1125 Colonel By Drive, Ottawa, ON, Canada
{ luxiaos; schwartz }@sce.carleton.ca

Received July 2011; revised November 2011

Abstract. *This paper presents an L_{R-I} lagging anchor algorithm that combines a lagging anchor method with an L_{R-I} learning algorithm. We prove that this decentralized learning algorithm converges in strategies to Nash equilibria in two-player two-action general-sum matrix games. A practical L_{R-I} lagging anchor algorithm is introduced for players to learn their Nash equilibrium strategies in general-sum stochastic games. Simulation results show the performance of the proposed L_{R-I} lagging anchor algorithm in both matrix games and stochastic games.*
Keywords: Multiagent learning, Matrix games, Game theory

1. **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. Decentralized learning means that there is no central learning strategy for all of the agents. Instead, each agent learns its own strategy. Decentralized learning algorithms can be used for players to learn their Nash equilibria (NE) in games with incomplete information [3, 4]. When an agent has "incomplete information", it means that the agent does not know its own reward function, nor the other players' strategies nor the other players' reward functions. The agent 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 a Nash equilibrium.

There are a number of multi-agent learning algorithms proposed in the literature that can be used for two-player matrix games. Lakshmivarahan and Narendra [3] presented a linear reward-inaction approach that can guarantee the convergence to a Nash equilibrium under the assumption that the game only has strict Nash equilibria in pure strategies. The linear reward-penalty approach, introduced in [4], can guarantee that the expected value of players' strategies converge to a Nash equilibrium in fully mixed strategies with the proper choice of parameters. Bowling and Veloso proposed a WoLF-IGA approach that can guarantee the convergence to a Nash equilibrium for two-player two-action matrix games and the Nash equilibrium can be in fully mixed strategies or in pure strategies. However, the WoLF-IGA approach is not a completely decentralized learning algorithm since the player has to know its opponent's strategy at each time step. Dahl [5, 6] proposed a lagging anchor approach for two-player zero-sum matrix games that can guarantee the convergence to a Nash equilibrium in fully mixed strategies. However, the lagging anchor algorithm is not a decentralized learning algorithm because each player has to know its reward matrix. Besides the design of learning algorithms, other studies on matrix games

have been conducted more recently on finding Nash equilibria for some specific matrix games in [7, 8].

In this paper, we evaluate the learning automata algorithm L_{R-I} [3] and L_{R-P} [4], the gradient ascent algorithm WoLF-IGA [9] and the lagging anchor algorithm [5]. We then propose the new L_{R-I} lagging anchor algorithm. The L_{R-I} lagging anchor algorithm is a combination of learning automata and gradient ascent learning. It is a completely decentralized algorithm and as such, each agent only needs to know its own action and the received reward at each time step. This paper is an extension of the work in [10] from two-player two-action zero-sum matrix games to two-player two-action general-sum matrix games. We prove the convergence of the L_{R-I} lagging anchor algorithm to Nash equilibria in two-player two-action general-sum matrix games. Furthermore, the Nash equilibrium can be in games with pure or fully mixed strategies. We then simulate three matrix games to test the performance of our proposed learning algorithm.

The motivation for this research is to develop a decentralized learning algorithm for teams of mobile robots. In particular, we are interested in robots that learn to work together for security applications. We have structured these applications as stochastic games such as the guarding a territory game or the pursuit-evasion game. These games have multiple states and multiple players. In Section 3, we make theoretical advances that prove convergence of our proposed L_{R-I} lagging anchor algorithm for two-player two-action general-sum matrix games. We further extend the works to the grid game introduced by Hu and Wellman [11] and we demonstrate the practical performance of the proposed algorithm.

2. Learning in Matrix Games.

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

In a matrix game, each player tries to maximize its 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 a Nash equilibrium.

Definition 2.1. *A **Nash equilibrium** is a collection of all the players' strategies* $(\sigma_e^1, \ldots, \sigma_e^n)$ *such that, for* $i = 1, \ldots, n$,

$$\sigma_e^i \sigma_e^{-i} R^i \geq \sigma^i \sigma_e^{-i} R^i, \quad \forall \sigma^i \in PD(A^i) \tag{1}$$

where $PD(A^i)$ *is the set of all probability distributions over the action set* A^i*, and* σ^{-i} *is a joint strategy for all the players except player* i*.*

In other words, 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 Nash equilibrium strategies.

A Nash equilibrium can be in fully mixed strategies or in 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 [12].

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. Given that each player has two actions in the game, we can define a two-player two-action general-sum game as

$$R = \begin{bmatrix} r_{11} & r_{12} \\ r_{21} & r_{22} \end{bmatrix}, \quad C = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{bmatrix} \tag{2}$$

where r_{lf} and c_{lf} denote the reward to the row player (player 1) and the reward to the column player (player 2) respectively. The row player chooses action $l \in \{1, 2\}$ and the column player chooses action $f \in \{1, 2\}$. If the two players are fully competitive in the game, we will have a two-player zero-sum matrix game with $R = -C$ or $r_{lf} + c_{lf} = 0$ for $l, f \in \{1, 2\}$. Based on (2), the pure strategies l and f are called a strict Nash equilibrium in pure strategies if

$$r_{lf} > r_{-lf}, \ c_{lf} > c_{l-f} \ \text{ for } \ l, f \in \{1, 2\} \tag{3}$$

where $-l$ and $-f$ denote any row other than row l and any column other than column f respectively.

Learning in a two-player matrix game can be expressed as the process of each 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 a Nash equilibrium based on the information from the environment. In order to address the limitations of the previously published multi-agent learning algorithms for matrix games, we divide these learning algorithms into two groups. One group is based on learning automata [13] and another group is based on gradient ascent learning [14].

2.1. **Learning automata.** Learning automation is a learning unit for adaptive decision making in an unknown environment [13]. 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 the received reward 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: the linear reward-inaction (L_{R-I}) algorithm and the linear reward-penalty (L_{R-P}) algorithm.

2.1.1. *Linear reward-inaction algorithm.* The linear reward-inaction (L_{R-I}) algorithm for player i ($i = 1, \ldots, n$) is defined as follows:

$$\begin{aligned} &p_c^i(k+1) = p_c^i(k) + \eta r^i(k)(1 - p_c^i(k)) \text{ if } a_c \text{ is the current action at } k \\ &p_j^i(k+1) = p_j^i(k) - \eta r^i(k) p_j^i(k) \text{ for all } a_j^i \neq a_c^i \end{aligned} \tag{4}$$

where k is the time step, the superscripts and subscripts on p denote different players and each player's different action respectively, $0 < \eta < 1$ is the learning parameter, $r^i(k)$ is the response of the environment given 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 [3].

2.1.2. *Linear reward-penalty algorithm.* The linear reward-penalty (L_{R-P}) algorithm for player i is defined as follows:

$$p_c^i(k+1) = p_c^i(k) + \eta_1 r^i(k)[1 - p_c^i(k)] - \eta_2[1 - r^i(k)]p_c^i(k) \quad (\text{if } a_c^i \text{ is the current action})$$
$$p_j^i(k+1) = p_j^i(k) - \eta_1 r^i(k) p_j^i(k) + \eta_2[1 - r^i(k)] \left[\frac{1}{m-1} - p_j^i(k)\right] \quad (\text{for all } a_j^i \neq a_c^i) \quad (5)$$

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

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

2.2. **Gradient ascent learning.** Gradient ascent learning can be used to update the player's strategy in the direction of the current gradient [14]. At each iteration, the player will adjust its strategy based on its gradient in order to increase its expected reward. Using a gradient ascent learning algorithm, Singh et al. [14] showed that the players' strategies do not converge to Nash equilibria for the general case of matrix games. However, there are a number of gradient ascent learning algorithms that can guarantee the convergence to Nash equilibria for specific matrix games such as two-player two-action matrix games. These algorithms are the WoLF-IGA algorithm [9] and the lagging anchor algorithm [5].

2.2.1. *WoLF-IGA algorithm.* Win or learn fast-infinitesimal gradient ascent (WoLF-IGA) algorithm was introduced 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 is smaller, when the player is winning and the learning rate is larger when the player is losing. The term p_1 is the probability of player 1 choosing the first action. Then $1 - p_1$ is the probability of player 1 choosing the second action. Accordingly, q_1 is the probability of player 2 choosing the first action and $1 - q_1$ is the probability of player 2 choosing the second action. The updating rules of the WoLF-IGA algorithm are as follows:

$$p_1(k+1) = p_1(k) + \eta \alpha^1(k) \frac{\partial V^1(p_1(k), q_1(k))}{\partial p_1} \quad (6)$$
$$q_1(k+1) = q_1(k) + \eta \alpha^2(k) \frac{\partial V^2(p_1(k), q_1(k))}{\partial q_1} \quad (7)$$
$$\alpha^1(k) = \begin{cases} \alpha_{\min}, & \text{if } V^1(p_1(k), q_1(k)) > V^1(p_1^*, q_1(k)) \\ \alpha_{\max}, & \text{otherwise} \end{cases}$$
$$\alpha^2(k) = \begin{cases} \alpha_{\min}, & \text{if } V^2(p_1(k), q_1(k)) > V^2(p_1(k), q_1^* *) \\ \alpha_{\max}, & \text{otherwise} \end{cases}$$

where η is the step size, α^i $(i = 1, 2)$ is the learning rate for player i $(i = 1, 2)$, $V^i(p_1(k), q_1(k))$ is the expected reward of 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 $\alpha_{\max} > \alpha_{\min}$, the players' strategies converge to a Nash equilibrium as the step size $\eta \to 0$ [9].

This algorithm is 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

general-sum matrix games. However, this algorithm is not a decentralized learning algorithm. This algorithm 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 parameters $\alpha_{\min}$ and $\alpha_{\max}$ accordingly. In order to obtain $V^1(p_1^*, q_1(k))$ and $V^2(p_1(k), q_1^*)$, we need to know each player's reward matrix and its opponent's strategy at time k. Whereas, in a decentralized learning algorithm the agents would only have their own actions and reward at time k. Although a practical decentralized learning algorithm called a WoLF policy hill-climbing method was provided in [9], there is no proof of convergence to Nash equilibrium strategies.

2.2.2. *The lagging anchor algorithm.* The lagging anchor algorithm for two-player zero-sum games was introduced by Dahl [5]. 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, player 2's strategy is denoted as a vector $\mathbf{w} = [q_1, q_2, \cdots, q_{m_2}]^T$. The updating rules are listed as follows:

$$\begin{aligned} \mathbf{v}(k+1) &= \mathbf{v}(k) + \eta \mathbf{P}_{m_1} R^1 Y(k) + \eta\gamma(\bar{\mathbf{v}}(k) - \mathbf{v}(k)) \\ \bar{\mathbf{v}}(k) &= \bar{\mathbf{v}}(k) + \eta\gamma(\mathbf{v}(k) - \bar{\mathbf{v}}(k)) \\ \mathbf{w}(k+1) &= \mathbf{w}(k) + \eta \mathbf{P}_{m_2} R^2 X(k) + \eta\gamma(\bar{\mathbf{w}}(k) - \mathbf{w}(k)) \\ \bar{\mathbf{w}}(k) &= \bar{\mathbf{w}}(k) + \eta\gamma(\mathbf{w}(k) - \bar{\mathbf{w}}(k)) \end{aligned} \tag{8}$$

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 $\mathbf{w}$ to be one. $Y(k)$ is a unit vector corresponding to the actions of player 2. If the m_ith action in player 2's action set is selected at time k, then the m_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 and R^1 and R^2 are the reward matrices for player 1 and 2 respectively. In (8), $\bar{\mathbf{v}}$ and $\bar{\mathbf{w}}$ are the anchor parameters for $\mathbf{v}$ and $\mathbf{w}$ respectively which can be represented as the weighted average of the players' strategies. In a two-player zero-sum game with only Nash equilibria 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].

This algorithm guarantees the convergence to a Nash equilibrium in fully mixed strategies. However, the convergence to a Nash equilibrium in pure strategies has never been discussed. Furthermore, the lagging anchor algorithm in (8) requires full information of the player's reward matrices R_1 and R_2. Therefore, the lagging anchor algorithm is not a decentralized learning algorithm.

Table 1 compares 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. From this table, only the WoLF-IGA algorithm can guarantee the convergence to both pure and mixed strategy Nash equilibrium (NE). However, it is not a decentralized learning algorithm. Although the L_{R-I} algorithm and the L_{R-P} algorithm are decentralized learning algorithms, neither of them can guarantee the convergence to both pure and mixed strategy NE. Therefore, the main motivation of this work is to design a decentralized learning algorithm which can guarantee the convergence to both pure and mixed strategy NE as shown in Table 1.

3. L_{R-I} Lagging Anchor Algorithm.

In this section, we design an L_{R-I} lagging anchor algorithm which is a completely decentralized learning algorithm and can guarantee the convergence to Nash equilibria in both pure and fully mixed strategies. We take the L_{R-I} algorithm defined in (4) as the updating law of the player's strategy and add the

TABLE 1. Comparison of learning algorithms in matrix games

Applicability	Existing algorithms				Our proposed algorithm
	L_{R-I}	L_{R-P}	WoLF-IGA	lagging anchor	L_{R-I} lagging anchor
Allowable actions	no limit	2 actions	2 actions	no limit	2 actions
Convergence	pure NE	fully mixed NE (expected value)	both	fully mixed NE	both
Decentralized?	Yes	Yes	No	No	Yes

lagging anchor term in (8). Then the L_{R-I} lagging anchor algorithm for player i is defined as follows:

$$\left.\begin{aligned} p_c^i(k+1) &= p_c^i(k) + \eta r^i(k)[1 - p_c^i(k)] + \eta[\bar{p}_c^i(k) - p_c^i(k)] \\ \bar{p}_c^i(k+1) &= \bar{p}_c^i(k) + \eta[p_c^i(k) - \bar{p}_c^i(k)] \end{aligned}\right\} \text{ if } a_c^i \text{ is the action taken at time } k$$

$$\left.\begin{aligned} p_j^i(k+1) &= p_j^i(k) - \eta r^i(k) p_j^i(k) + \eta[\bar{p}_j^i(k) - p_j^i(k)] \\ \bar{p}_j^i(k+1) &= \bar{p}_j^i(k) + \eta[p_j^i(k) - \bar{p}_j^i(k)] \end{aligned}\right\} \text{ for all } a_j^i \neq a_c^i \tag{9}$$

where η is the step size and $(\bar{p}_c^i, \bar{p}_j^i)$ are the lagging parameters for (p_c^i, p_j^i). The idea behind 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. We expect that the player's current strategy and the long-term average will be drawn towards the equilibrium point during learning.

To analyze the above L_{R-I} lagging anchor algorithm, we use ordinary differential equations (ODEs). The behavior of the learning algorithm can be approximated by ODEs as the step size goes to zero. Thathachar and Sastry [15] provided the equivalent ODEs of the L_{R-I} algorithm in (4) as

$$\dot{p}_c^i = \sum_{j=1}^{m_i} p_c^i p_j^i (d_c^i - d_j^i) \tag{10}$$

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

Combining the above ODEs of the L_{R-I} algorithm in (10) with the ODEs for the lagging anchor part of our algorithm, we can find the equivalent ODEs for our L_{R-I} lagging anchor algorithm given as

$$\begin{aligned} \dot{p}_c^i &= \sum_{j=1}^{m_i} p_c^i p_j^i (d_c^i - d_j^i) + (\bar{p}_c^i - p_c^i) \\ \dot{\bar{p}}_c^i &= p_c^i - \bar{p}_c^i \end{aligned} \tag{11}$$

Based on our proposed L_{R-I} lagging anchor algorithm, we now present the following theorem.

Theorem 3.1. *We consider a two-player two-action general-sum matrix game and assume the game only has a Nash equilibrium in fully mixed strategies or strict Nash equilibria in pure strategies. If both players follow the L_{R-I} lagging anchor algorithm, when the step size $\eta \to 0$, then the following is true regarding the asymptotic behavior of the algorithm.*

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

Proof: Given a two-player two-action general-sum game defined in (2), we denote p_1 as the probability of player 1 taking its first action and q_1 as the probability of player 2 taking its first action. Then the L_{R-I} lagging anchor algorithm becomes

$$\begin{aligned}
\dot{p}_1 &= \sum_{j=1}^{2} p_1 p_j (d_1^1 - d_j^1) + (\bar{p}_1 - p_1) \\
\dot{\bar{p}}_1 &= p_1 - \bar{p}_1 \\
\dot{q}_1 &= \sum_{j=1}^{2} q_1 q_j (d_1^2 - d_j^2) + (\bar{q}_1 - q_1) \\
\dot{\bar{q}}_1 &= q_1 - \bar{q}_1
\end{aligned} \tag{12}$$

where $d_1^1 = r_{11}q_1 + r_{12}(1-q_1)$, $d_2^1 = r_{21}q_1 + r_{22}(1-q_1)$, $d_1^2 = c_{11}p_1 + c_{21}(1-p_1)$ and $d_2^2 = c_{12}p_1 + c_{22}(1-p_1)$. Then (12) becomes

$$\begin{aligned}
\dot{p}_1 &= p_1(1-p_1)[u_1 q_1 + r_{12} - r_{22}] + (\bar{p}_1 - p_1) \\
\dot{\bar{p}}_1 &= p_1 - \bar{p}_1 \\
\dot{q}_1 &= q_1(1-q_1)[u_2 p_1 + c_{21} - c_{22}] + (\bar{q}_1 - q_1) \\
\dot{\bar{q}}_1 &= q_1 - \bar{q}_1
\end{aligned} \tag{13}$$

where $u_1 = r_{11} - r_{12} - r_{21} + r_{22}$ and $u_2 = c_{11} - c_{12} - c_{21} + c_{22}$. If we let the right hand side of the above equation equal to zero, we then get the equilibrium points of the above equations as $(p_1^*, q_1^*) = (0,0), (0,1), (1,0), (1,1), ((c_{22} - c_{21})/u_2, (r_{22} - r_{12})/u_1)$. To study the stability of the above learning dynamics, we use a linear approximation of the above equations around the equilibrium point $(p_1^*, q_1^*, p_1^*, q_1^*)$. Then the linearization matrix J is given as

$$J_{(p_1^*, q_1^*)} = \begin{bmatrix} (1-2p_1^*)(u_1 q_1^* + r_{12} - r_{22}) - 1 & 1 & p_1^*(1-p_1^*)u_1 & 0 \\ 1 & -1 & 0 & 0 \\ q_1^*(1-q_1^*)u_2 & 0 & (1-2q_1^*)(u_2 p_1^* + c_{21} - c_{22}) - 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}. \tag{14}$$

If we substitute each of the equilibrium points $(0,0)$, $(0,1)$, $(1,0)$, $(1,1)$ into (14), we get

$$J_{\text{pure}} = \begin{bmatrix} -e_1 - 1 & 1 & 0 & 0 \\ 1 & -1 & 0 & 0 \\ 0 & 0 & -e_2 - 1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix} \tag{15}$$

where

$$e_1 = r_{22} - r_{12},\ e_2 = c_{22} - c_{21} \quad \text{for } (0,0); \tag{16}$$
$$e_1 = r_{21} - r_{11},\ e_2 = c_{21} - c_{22} \quad \text{for } (0,1); \tag{17}$$
$$e_1 = r_{12} - r_{22},\ e_2 = c_{12} - c_{11} \quad \text{for } (1,0); \tag{18}$$
$$e_1 = r_{11} - r_{21},\ e_2 = c_{11} - c_{12} \quad \text{for } (1,1). \tag{19}$$

The eigenvalues of the above matrix J_{pure} are $\lambda_{1,2} = 0.5[-(e_1+2) \pm \sqrt{e_1^2+4}]$ and $\lambda_{3,4} = 0.5[-(e_2+2) \pm \sqrt{e_2^2+4}]$. In order to obtain a stable equilibrium point, the real parts of the eigenvalues of J_{pure} must be negative. Therefore, the equilibrium point is asymptotically stable if

$$\begin{aligned} 0.5\left[-(e_{1,2}+2) \pm \sqrt{e_{1,2}^2+4}\right] < 0 \quad &\Rightarrow \\ e_{1,2} + 2 > \sqrt{e_{1,2}^2+4} \quad &\Rightarrow \\ e_{1,2} > 0 \end{aligned} \tag{20}$$

For the equilibrium point $((c_{22}-c_{21})/u_2, (r_{22}-r_{12})/u_1)$, the linearization matrix becomes

$$J_{\text{mixed}} = \begin{bmatrix} -1 & 1 & p_1^*(1-p_1^*)u_1 & 0 \\ 1 & -1 & 0 & 0 \\ q_1^*(1-q_1^*)u_2 & 0 & -1 & 1 \\ 0 & 0 & 1 & -1 \end{bmatrix}. \tag{21}$$

The characteristic equation of the above matrix is

$$\lambda^4 + 4\lambda^3 + (4+e_3)\lambda^2 + 2e_3\lambda + e_3 = 0 \tag{22}$$

where $e_3 = -p_1^*(1-p_1^*)q_1^*(1-q_1^*)u_1u_2$. We set up the Routh table to analyze the locations of the roots in (22) as follows:

λ^4	1	$4+e_3$	e_3
λ^3	4	$2e_3$	
λ^2	$4+0.5e_3$	e_3	
λ^1	$(e_3^2+4e_3)/(4+0.5e_3)$		
λ^0	e_3		

(23)

Based on the Routh-Hurwitz stability criterion, if (22) is stable, then all the coefficients of (22) must be positive and all the elements in the first column of the Routh table in (23) are positive. In order to meet the Routh-Hurwitz stability criterion, we must have

$e_3 > 0$. Therefore, the equilibrium point $((c_{22} - c_{21})/u_2, (r_{22} - r_{12})/u_1)$ is asymptotically stable if

$$e_3 = -p_1^*(1 - p_1^*)q_1^*(1 - q_1^*)u_1u_2 > 0 \quad \Rightarrow \\ u_1u_2 < 0 \tag{24}$$

Case 1: strict Nash equilibrium in pure strategies

We first consider that the game only has strict Nash equilibrium in pure strategies. Without loss of generality, we assume that the Nash equilibrium in this case is both players' first actions. According to the definition of a strict Nash equilibrium in (3), if the Nash equilibrium strategies are both players' first actions, we can get

$$r_{11} > r_{21}, \quad c_{11} > c_{12}. \tag{25}$$

Since the Nash equilibrium in this case is the equilibrium point $(1, 1)$, we can get $e_1 = r_{11} - r_{21} > 0$ and $e_2 = c_{11} - c_{12} > 0$ based on (19) and (25). Therefore, the stability condition in (20) is satisfied and the equilibrium point $(1, 1)$ which is the Nash equilibrium in this case is asymptotically stable.

We now test the other equilibrium points. We first consider the equilibrium point $((c_{22} - c_{21})/u_2, (r_{22} - r_{12})/u_1)$. According to (24), if this equilibrium point is stable, we must have $u_1u_2 < 0$. To be a valid inner point in the probability space (unit square), the equilibrium point $((c_{22} - c_{21})/u_2, (r_{22} - r_{12})/u_1)$ must satisfy

$$\begin{cases} 0 < (c_{22} - c_{21})/u_2 < 1 \\ 0 < (r_{22} - r_{12})/u_1 < 1 \end{cases} \tag{26}$$

If $u_1u_2 < 0$, we can get

$$\begin{cases} r_{11} > r_{21},\ r_{22} > r_{12} \\ c_{11} < c_{12},\ c_{22} < c_{21} \end{cases} \quad \text{if } u_1 > 0,\ u_2 < 0 \tag{27}$$

$$\begin{cases} r_{11} < r_{21},\ r_{22} < r_{12} \\ c_{11} > c_{12},\ c_{22} > c_{21} \end{cases} \quad \text{if } u_1 < 0,\ u_2 > 0 \tag{28}$$

However, the conditions in (27) and (28) conflict with the inequalities in (25). Therefore, the inequality $u_1u_2 < 0$ will not hold and the equilibrium point $((c_{22} - c_{21})/u_2, (r_{22} - r_{12})/u_1)$ is unstable in Case 1.

For the equilibrium points $(0, 1)$ and $(1, 0)$, based on (17), (18) and (20), the stability conditions are $r_{21} > r_{11}, c_{21} > c_{22}$ for $(0, 1)$ and $r_{12} > r_{22}, c_{12} > c_{11}$ for $(1, 0)$. However, these stability conditions conflict with the inequalities $r_{11} > r_{21}, c_{11} > c_{12}$ in (25). Therefore, the equilibrium points $(0, 1)$ and $(1, 0)$ are unstable in Case 1.

For the equilibrium point $(0, 0)$, the stability condition is $r_{22} > r_{12}$, $c_{22} > c_{21}$ based on (16) and (20). From (3), we can find that this stability condition also meets the requirement for a strict Nash equilibrium (both players' second actions) in (3). Therefore, the equilibrium point $(0, 0)$ is stable only if it is also a Nash equilibrium point.

Thus, the Nash equilibrium point is asymptotically stable while any equilibrium point which is not a Nash equilibrium is unstable.

Case 2: Nash equilibrium in fully mixed strategies

We now consider that the game only has Nash equilibrium in fully mixed strategies. Singh et al. [14] showed that a Nash equilibrium in fully mixed strategies for a two-player two-action general-sum matrix game has the from of

$$(p_1^{NE}, q_1^{NE}) = \left[\frac{c_{22} - c_{21}}{u_2}, \frac{r_{22} - r_{12}}{u_1}\right] \tag{29}$$

where (p_1^{NE}, q_1^{NE}) denotes the Nash equilibrium strategies over players' first actions which happens to be the equilibrium point of (13). According to (24), the equilibrium point $((c_{22}-c_{21})/u_2, (r_{22}-r_{12})/u_1)$ is asymptotically stable if $u_1u_2 < 0$. If we assume $u_1u_2 > 0$, we can get

$$\begin{cases} 0 < (c_{22}-c_{21})/u_2 < 1 \\ 0 < (r_{22}-r_{12})/u_1 < 1 \end{cases} (u_1u_2 > 0) \Rightarrow$$

$$\begin{cases} r_{11} > r_{21},\ r_{22} > r_{12} \\ c_{11} > c_{12},\ c_{22} > c_{21} \end{cases} \text{if } u_1 > 0,\ u_2 > 0 \tag{30}$$

$$\begin{cases} r_{11} < r_{21},\ r_{22} < r_{12} \\ c_{11} < c_{12},\ c_{22} < c_{21} \end{cases} \text{if } u_1 < 0,\ u_2 < 0 \tag{31}$$

According to (3), the above equations contain multiple Nash equilibria in pure strategies: $(p_1^{NE}, q_1^{NE}) = (1,1), (0,0)$ if $u_1 > 0$, $u_2 > 0$ and $(p_1^{NE}, q_1^{NE}) = (0,1), (1,0)$ if $u_1 < 0$, $u_2 < 0$. However, under our assumption, the game in Case 2 only has a Nash equilibrium in fully mixed strategies and Nash equilibria in pure strategies do not exist. Therefore, we always have $u_1u_2 < 0$ in Case 2 and the equilibrium point $((c_{22}-c_{21})/u_2, (r_{22}-r_{12})/u_1)$, which is also the Nash equilibrium point, is asymptotically stable.

For the other equilibrium points, based on (16)-(19) and (20), the stability conditions become

$$r_{22} > r_{12},\ c_{22} > c_{21} \quad \text{for } (0,0); \tag{32}$$

$$r_{21} > r_{11},\ c_{21} > c_{22} \quad \text{for } (0,1); \tag{33}$$

$$r_{12} > r_{22},\ c_{12} > c_{11} \quad \text{for } (1,0); \tag{34}$$

$$r_{11} > r_{21},\ c_{11} > c_{12} \quad \text{for } (1,1). \tag{35}$$

As already noted, the game in Case 2 only has a Nash equilibrium in fully mixed strategies and we always have $u_1u_2 < 0$. Then the inequalities in (27) and (28) are true in Case 2. However, the stability conditions in (32)-(35) for the equilibrium points $(0,0)$, $(0,1)$, $(1,0)$, $(1,1)$ conflict with the inequalities in (27) and (28). Therefore, the equilibrium points other than $((c_{22}-c_{21})/u_2, (r_{22}-r_{12})/u_1)$ are unstable in this case.

Thus we can conclude that the Nash equilibrium point is asymptotically stable while the other equilibrium points are unstable in Case 2.

Example 3.1. *We provide an example to illustrate the stability of the Nash equilibrium points. We consider the following two-player zero-sum matrix game*

$$R^1 = \begin{bmatrix} r_{11} & 2 \\ 3 & -1 \end{bmatrix}, \quad R^2 = -R^1 \tag{36}$$

where $r_{11} \in \mathbb{R}$. We denote p_1 as the probability over player 1's first action and q_1 as the probability over player 2's first action. Based on the analysis in (3) and (29), we can find the Nash equilibrium points (p_1^{NE}, q_1^{NE}) for different values of r_{11}. Figure 1 shows the plot of Nash equilibrium points for $r_{11} \in [-10, 10]$. When $r_{11} > 2$, we have strict Nash equilibrium in pure strategies $\big(p_1^{NE} = 1, q_1^{NE} = 0\big)$. When $r_{11} < 2$, we have fully mixed strategy Nash equilibrium $\big(p_1^{NE} = 4/(6-r_{11}), q_1^{NE} = 3/(6-r_{11})\big)$. At $r_{11} = 2$, we have multiple Nash equilibria $\big(p_1^{NE} = 1, q_1^{NE} \in [0, 0.75]\big)$. According to Theorem 3.1, we consider the stability of the strict Nash equilibrium in pure strategies or the Nash equilibrium in fully mixed strategies which are the Nash equilibrium points $(1,0)$ and $\big(4/(6-r_{11}), 3/(6-r_{11})\big)$. The stability condition for the Nash equilibrium point $(1,0)$ can be found in (34). Based on the reward function in (36), we find that the stability condition for $(1,0)$ is $r_{11} > 2$. Similarly, the stability condition for the Nash equilibrium point $\big(4/(6-r_{11}), 3/(6-r_{11})\big)$

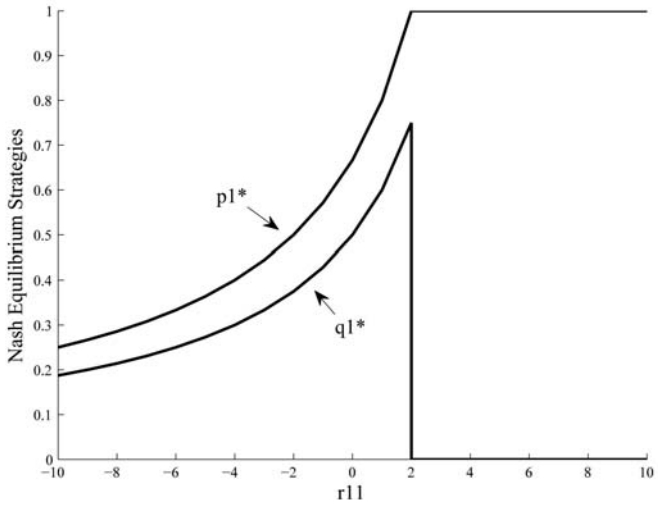


FIGURE 1. Players' Nash equilibrium strategies vs r_{11}

can be found in (28). After we substitute the rewards in (36) into (28), we find that the stability condition for $\big(4/(6-r_{11}), 3/(6-r_{11})\big)$ *is* $r_{11} < 2$. *Therefore, the Nash equilibrium points* $(1,0)$, $\big(4/(6-r_{11}), 3/(6-r_{11})\big)$ *are asymptotically stable.*

4. **Simulation.** We now simulate three matrix games to show the performance of the proposed L_{R-I} lagging anchor algorithm. We first take the matching pennies game for example. This game is a two-player zero-sum game and 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 2(a), the Nash equilibrium in this game is in fully mixed strategies such that each player plays Heads and Tails with a probability of 0.5. We set the step size $\eta = 0.001$ in (9) and $p_1(0) = q_1(0) = 0.2$. We run the simulation for 30000 iterations. In Figure 2(a), the players' probabilities of taking their first actions start from $(0.2, 0.2)$ and move close to the Nash equilibrium point $(0.5, 0.5)$ as the learning proceeds.

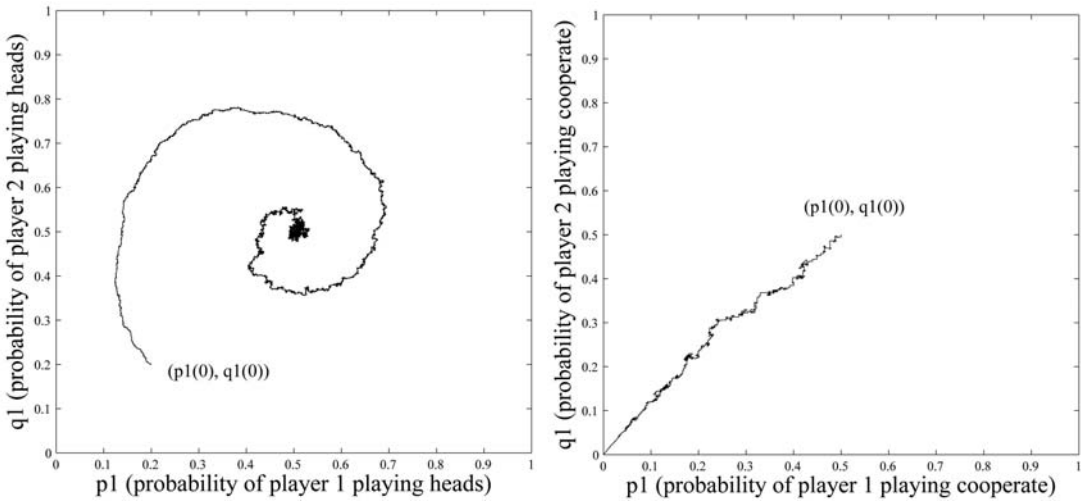
The second game we simulate is a two-player general-sum game called the prisoners' dilemma. In this game, we have two players and each player has two actions: defect or cooperate. A player receives a reward of 10 if it defects and the other player cooperates, or receives a reward of 0 if it cooperates and the other player defects. If both players

TABLE 2. Examples of two-player matrix games

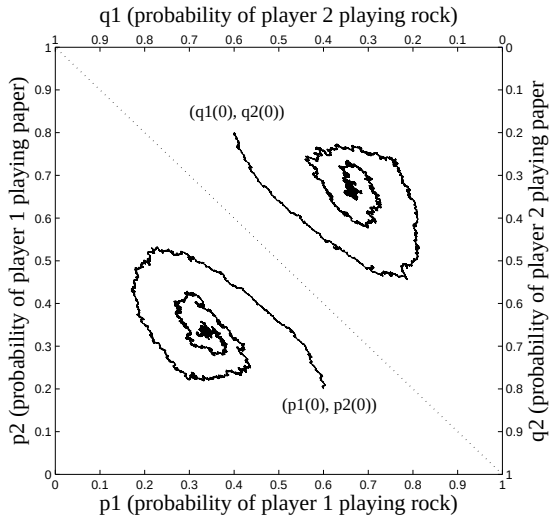
(a) Matching Pennies	(b) Prisoners' Dilemma	(c) Rock-Paper-Scissors
$R^1 = \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix}$,	$R^1 = \begin{bmatrix} 5 & 0 \\ 10 & 1 \end{bmatrix}$,	$R^1 = \begin{bmatrix} 0 & -1 & 1 \\ 1 & 0 & -1 \\ -1 & 1 & 0 \end{bmatrix}$,
$R^2 = -R^1$.	$R^2 = (R^1)^T$.	$R^2 = -R^1$.
NE in fully mixed strategies	NE in pure strategies	NE in fully mixed strategies

cooperate, each player receives a reward of 5. If they both defect, each player receives a reward of 1. The reward matrix is shown in Table 2(b) where one player's reward matrix is the transpose of the other player's reward matrix. This game has a unique Nash equilibrium in pure strategies which is both players playing defect. We set the step size $\eta = 0.001$ in (9) and $p_1(0) = q_1(0) = 0.5$. We run the simulation for 30000 iterations. Figure 2(b) shows that the players' strategies move close to the Nash equilibrium strategies (both players' second actions) as the learning proceeds.

The third game we simulate in this paper is the rock-paper-scissors game. This game has two players and each player has three actions: rock, paper and scissors. A winner in the game is determined by the following rules: paper defeats rock, scissors defeats paper, and rock defeats scissors. The winner receives a reward of 1 and the loser receives -1. If both players choose the same action, each player gets 0. The reward matrix is shown in Table 2(c). This game has a Nash equilibrium in fully mixed strategies which



(a) Matching pennies (b) Prisoners' dilemma



(c) Rock-paper-scissors

FIGURE 2. Trajectories of players' strategies during learning in matrix games

is each player choosing any action with the same probability of 1/3. We set the step size $\eta = 0.001$, $p_1(0) = q_1(0) = 0.6$ and $p_2(0) = q_2(0) = 0.2$. We run the simulation for 50000 iterations. Although we only prove the convergence for two-player two-action games, the result in Figure 2(c) shows that the proposed L_{R-I} lagging anchor algorithm may be applicable to a two-player matrix game with more than two actions.

5. **Extension of Matrix Games to Stochastic Games.** A stochastic game is a tuple $(n, S, A^1, \ldots, A^n, T, \gamma, R^1, \ldots, R^n)$ where n is the number of the players, S is a finite state space, $T : S \times A^1 \times \cdots \times A^n \times S \to [0, 1]$ is the transition function, A^i $(i = 1, \ldots, n)$ is the action set for the player i, $\gamma \in [0, 1]$ is the discount factor and $R^i : S \times A^1 \times \cdots \times A^n \times S \to \mathbb{R}$ is the reward function for player i. The transition function in a stochastic game is a probability distribution over next states given the current state and joint action of the players. The reward function $R^i(s, a^1, \ldots, a^n, s')$ denotes the reward received by player i in state s' after taking joint action $(a^1, \ldots, a^n)$ in state s. A matrix game can be considered as a special case of stochastic games with $S = \emptyset$.

The proposed L_{R-I} lagging anchor algorithm is designed based on matrix games. In this section, we extend the algorithm to the more general stochastic games. Inspired by the WoLF-PHC algorithm in [9], we design a practical decentralized learning algorithm for stochastic games based on the L_{R-I} lagging anchor approach in (9). The practical algorithm is shown in Algorithm 1.

Algorithm 1 A practical L_{R-I} lagging anchor algorithm for player i

1: Initialize $Q(s, a) \leftarrow 0$ and $\pi(s, a) \leftarrow \dfrac{1}{|A^i|}$. Choose the learning rate α, η and the discount factor γ.
2: **for** Each iteration **do**
3: Select action a_c at current state s based on mixed exploration-exploitation strategy
4: Take action a_c and observe the reward r and the subsequent state s'
5: Update $Q(s, a_c)$

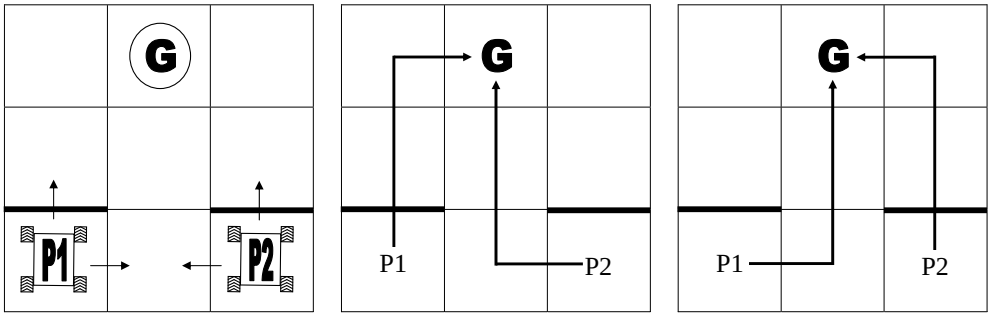
$$Q(s, a_c) = Q(s, a_c) + \alpha\big[r + \gamma \max_{a'} Q(s', a') - Q(s, a_c)\big]$$

6: Update the player's policy $\pi(s, \cdot)$

$$\begin{aligned}
&\pi(s, a_c) = \pi(s, a_c) + \eta Q(s, a_c)\big[1 - \pi(s, a_c)\big] + \eta\big[\bar{\pi}(s, a_c) - \pi(s, a_c)\big] \\
&\bar{\pi}(s, a_c) = \bar{\pi}(s, a_c) + \eta\big[\pi(s, a_c) - \bar{\pi}(s, a_c)\big] \\
&\pi(s, a_j) = \pi(s, a_j) - \eta Q(s, a_c)\pi(s, a_j) + \eta\big[\bar{\pi}(s, a_j) - \pi(s, a_j)\big] \\
&\bar{\pi}(s, a_j) = \bar{\pi}(s, a_j) + \eta\big[\pi(s, a_j) - \bar{\pi}(s, a_j)\big]
\end{aligned} \quad \text{for all } a_j \neq a_c$$

7: **end for**
$\big(Q(s, a)$ is the action-value function, $\pi(s, a)$ is the probability of player i taking action a at state s and a_c is the current action taken by player i at state $s\big)$

We now apply Algorithm 1 to a stochastic game to test the performance. The stochastic game we simulate is a general-sum grid game introduced by Hu and Wellman [11]. The game runs under a 3×3 grid field as shown in Figure 3(a). We have two players whose initial positions are located at the bottom left corner for player 1 and the bottom right corner for player 2. Both players try to reach the goal denoted as "G" in Figure 3(a). Each player has four possible moves which are moving up, down, left or right unless the player is on the sides of the grid. In Hu and Wellman's game, the movement that will take the player to a wall is ignored. Since we use the exact same game as Hu and Wellman,



(a) Grid game (b) Nash equilibrium path 1 (c) Nash equilibrium path 2

FIGURE 3. An example of a general-sum grid game

the possible actions of hitting a wall have been removed from the players' action sets. For example, if the player is at the bottom left corner, its available moves are moving up or right. If both players move to the same cell at the same time, they will bounce back to their original positions. The two thick lines in Figure 3(a) represent two barriers such that the player can pass through the barrier with a probability of 0.5. For example, if player 1 tries to move up from the bottom left corner, it will stay still or move to the upper cell with a probability of 0.5. The game ends when either one of the players reaches the goal. To reach the goal in minimum steps, the player needs to avoid the barrier and first move to the bottom center cell. Since both players cannot move to the bottom center cell simultaneously, the players need to cooperate such that one of the players has to take the risk and move up. The reward function for player i ($i = 1, 2$) in this game is defined as

$$r^i = \begin{cases} 100 & \text{player } i \text{ reaches the goal} \\ -1 & \text{both players move to the same cell (except the goal)} \\ 0 & \text{otherwise} \end{cases} \tag{37}$$

According to [11], this grid game has two Nash equilibrium paths as shown in Figure 3(b) and Figure 3(c). Starting from the initial state, the Nash equilibrium strategies of the players are player 1 moving up and player 2 moving left or player 1 moving right and player 2 moving up.

We set the step size as $\eta = 0.001$, the learning rate as $\alpha = 0.001$ and the discount factor as $\gamma = 0.9$. The mixed exploration-exploitation strategy is chosen such that the player chooses a random action with probability 0.05 and the greedy action with probability 0.95. We run the simulation for 10000 episodes. An episode is when the game starts with the players' initial positions and ends when either one of the players reaches the goal. Figure 4 shows the result of two players' learning trajectories. We define p_1 as player 1's probability of moving up and q_1 as player 2's probability of moving up from their initial positions. The result in Figure 4 shows that the two players' strategies at the initial state converge to one of the two Nash equilibrium strategies (player 1 moving right and player 2 moving up). Therefore, the proposed practical L_{R-I} lagging anchor algorithm may be applicable to general-sum stochastic games.

6. **Conclusions.** In this paper, we investigate the existing learning algorithms for matrix games. The analysis of the existing learning algorithms shows that the learning automata technique including the L_{R-I} algorithm and the L_{R-P} algorithm is a good candidate for decentralized learning. However, none of them can guarantee the convergence to Nash equilibria in both pure and fully mixed strategies. The lagging anchor algorithm considers

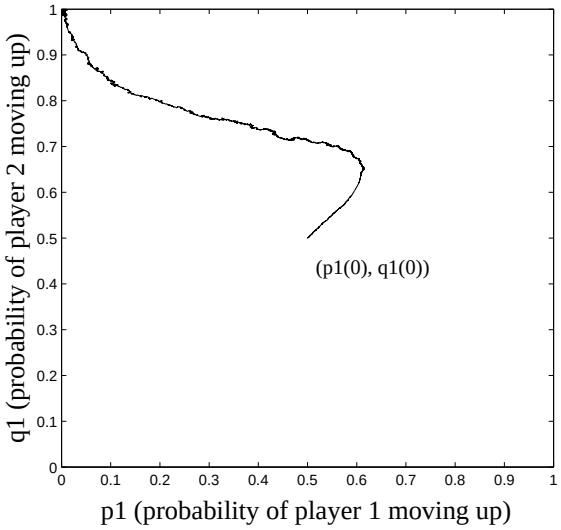


FIGURE 4. Learning trajectories of players' strategies at the initial state in the grid game

the player's current strategy and the long term average strategy during learning. Although the lagging anchor algorithm can guarantee the convergence to a Nash equilibrium in fully mixed strategies, it is not a decentralized learning algorithm. Inspired by the concept of lagging anchor, we propose an L_{R-I} lagging anchor algorithm as a completely decentralized learning algorithm. We prove that the L_{R-I} lagging anchor algorithm can guarantee the convergence to a Nash equilibrium in pure or fully mixed strategies in two-player two-action general-sum matrix games. Through simulations, we show the performance of the proposed L_{R-I} lagging anchor algorithm in three matrix games and the practical L_{R-I} lagging anchor algorithm in a general-sum stochastic grid game. Simulation results show the possibility of applying the proposed L_{R-I} lagging anchor algorithm to a two-player matrix game with more than two actions and the possibility of applying the proposed practical L_{R-I} lagging anchor algorithm to general-sum stochastic games.

Future research will examine the proof of convergence to Nash equilibria for general-sum matrix games with more than two players. We will also extend this work to the multi-player grid game of guarding a territory.

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] S. Lakshmivarahan and K. S. Narendra, Learning algorithms for two-person zero-sum stochastic games with incomplete information, *Mathematics of Operations Research*, vol.6, no.3, pp.379-386, 1981.

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

[5] 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.

[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] D. Jiang and S. Zhang, Realizablity of expected Nash equilibria of n-person condition games under strong knowledge system, *International Journal of Innovative Computing, Information and Control*, vol.2, no.4, pp.761-770, 2006.

[8] D. Jiang, Static, completely static, and rational games of complete information and their different Nash equilibria, *International Journal of Innovative Computing, Information and Control*, vol.4, no.3, pp.651-659, 2008.

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

[10] X. Lu and H. M. Schwartz, Decentralized learning in two-player zero-sum games: A L_{R-I} lagging anchor algorithm, *2011 American Control Conference*, San Francisco, CA, USA, 2011.

[11] J. Hu and M. P. Wellman, Nash Q-learning for general-sum stochastic games, *Journal of Machine Learning Research*, vol.4, pp.1039-1069, 2003.

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

[13] K. S. Narendra and M. A. L. Thathachar, *Learning Automata: An Introduction*, Prentice Hall, Englewood Cliffs, 1989.

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

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