Artif Intell Rev (2016) 45:299–332
DOI 10.1007/s10462-015-9447-5

Exponential moving average based multiagent reinforcement learning algorithms

Mostafa D. Awheda[1] · **Howard M. Schwartz**[1]

Published online: 19 October 2015

Abstract Two multi-agent policy iteration learning algorithms are proposed in this work. The two proposed algorithms use the exponential moving average approach along with the Q-learning algorithm as a basis to update the policy for the learning agent so that the agent's policy converges to a Nash equilibrium policy. The first proposed algorithm uses a constant learning rate when updating the policy of the learning agent, while the second proposed algorithm uses two different decaying learning rates. These two decaying learning rates are updated based on either the Win-or-Learn-Fast (WoLF) mechanism or the Win-or-Learn-Slow (WoLS) mechanism. The WoLS mechanism is introduced in this article to make the algorithm learn fast when it is winning and learn slowly when it is losing. The second proposed algorithm uses the rewards received by the learning agent to decide which mechanism (WoLF mechanism or WoLS mechanism) to use for the game being learned. The proposed algorithms have been theoretically analyzed and a mathematical proof of convergence to pure Nash equilibrium is provided for each algorithm. In the case of games with mixed Nash equilibrium, our mathematical analysis shows that the second proposed algorithm converges to an equilibrium. Although our mathematical analysis does not explicitly show that the second proposed algorithm converges to a Nash equilibrium, our simulation results indicate that the second proposed algorithm does converge to Nash equilibrium. The proposed algorithms are examined on a variety of matrix and stochastic games. Simulation results show that the second proposed algorithm converges in a wider variety of situations than state-of-the-art multi-agent reinforcement learning algorithms.

Keywords Multi-agent learning systems · Reinforcement learning · Markov decision processes · Nash equilibrium

✉ Mostafa D. Awheda
mawheda@sce.carleton.ca

Howard M. Schwartz
schwartz@sce.carleton.ca

[1] Department of Systems and Computer Engineering, Carleton University, 1125 Colonel By Drive, Ottawa, ON K1S 5B6, Canada

1 Introduction

Reinforcement learning (RL) is a learning technique that maps situations to actions so that an agent learns from the experience of interacting with its environment (Sutton and Barto 1998; Kaelbling et al. 1996). Reinforcement learning has attracted attention and been widely used in intelligent robot control systems (Awheda and Schwartz 2015; Schwartz 2014; Hinojosa et al. 2011; Rodríguez et al. 2007; Dai et al. 2005; Kondo and Ito 2004; Gutnisky and Zanutto 2004; Ye et al. 2003; Smart and Kaelbling 2002). It has also been effectively used for solving nonlinear optimal control problems (Luo et al. 2014a, b, 2015a, b, c, d; Dixon 2014; Modares et al. 2014; Wu and Luo 2012). In reinforcement learning, an agent learns from the experience of interacting with its environment. After perceiving the state of its environment at each time step, the agent takes an action so that its environment transitions from a state to a new state. A scalar reward signal is used to evaluate the transition. The objective for the agent is to maximize its cumulative reward (Weiss 1999; Sutton and Barto 1998; Kaelbling et al. 1996). Reinforcement learning is also well-suited for multi-agent learning because of its simplicity and generality (Busoniu et al. 2008, 2006; Hu et al. 1998). Learning is a key element of multi-agent systems (MAS) as it allows an agent to improve its performance by adapting the dynamics of the other agents and environment (Zhang and Lesser 2010). Learning encounters some challenges when it is used in multi-agent learning. One of these challenges is that the other learning agents have to be explicitly considered by each learning agent and therefore the environment is non-stationary. The environment of a multi-agent system is no longer stationary as the Markov property is violated by the other learning agents. As a result of a non-stationary environment, single agent reinforcement learning techniques are not guaranteed to converge in multi-agent settings. In multi-agent learning, the objective of each learning agent is to adopt an equilibrium strategy that maximizes its payoffs in the long run. However, a globally optimal equilibrium may not be reached in some cases when the learning agents do not cooperate with each other (Abdallah and Lesser 2008; Claus and Boutilier 1998). The objective of each learning agent, in such cases, is to adopt a Nash equilibrium (NE), where no learning agent will do better if deviates from Nash equilibrium (Abdallah and Lesser 2008; Conitzer and Sandholm 2007; Banerjee and Peng 2007; Bowling 2005).

In this work, we consider multi-agent domains in which different agents with different independent goals, assumptions, and algorithms have to interact with each other. We are interested in multi-agent learning algorithms that make agents learn how to adapt to changes in the other agents' performance when the Markovian property is no longer valid. That is, we are interested in multi-agent learning algorithms that can make agents learn Nash equilibrium strategies in a difficult learning problem with a moving target. Several multi-agent reinforcement learning (MARL) algorithms have recently been proposed and studied (Zhang and Lesser 2010; Abdallah and Lesser 2008; Banerjee and Peng 2007; Conitzer and Sandholm 2007; Bowling 2005; Hu and Wellman 2003; Bowling and Veloso 2001a, b, 2002; Singh et al. 2000). All these algorithms assume that each learning agent knows its own immediate reward and the actions of the other learning agents. Some of these algorithms have theoretical results of convergence in general-sum games. In addition, some of these algorithms fail to converge to Nash equilibrium in some games. For example, the Infinitesimal Gradient Ascent (IGA) algorithm proposed in Singh et al. (2000) fails to converge in games with mixed Nash equilibrium. The Win-or-Learn-Fast Infinitesimal Gradient Ascent (WoLF-IGA) algorithm proposed in Bowling and Veloso (2001a) and the Win-or-Learn-Fast Generalized Infinitesimal Gradient Ascent (GIGA-WoLF) algorithm proposed in Bowling (2005) fail to converge in some challenging games such as in the Shapley's game (Abdallah and Lesser 2008). The

Win-or-Learn-Fast Policy Hill-Climbing (WoLF-PHC) algorithm proposed in Bowling and Veloso (2002) does not converge to Nash equilibrium in the Shapley's game. In addition, some of these algorithms guarantee converge to Nash equilibrium by making some strict assumptions on the knowledge that is available to each learning agent. For example, some algorithms assume that the underlying game structure (Nash equilibrium) is known to each learning agent (Banerjee and Peng 2007; Bowling and Veloso 2002). Other algorithms such as the algorithms proposed in Conitzer and Sandholm (2007), Hu and Wellman (2003) assume that each learning agent knows the actions and the immediate rewards of the other learning agents (Zhang and Lesser 2010; Abdallah and Lesser 2008). Such strict assumptions may limit the use of these algorithms because the underlying game structure (Nash equilibrium) and the rewards of the other learning agents are often unknown to the learning agent and may be learned via interacting with the other learning agents (Bowling and Veloso 2002). On the other hand, the Weighted Policy Learner (WPL) and the Policy Gradient Ascent with Approximate Policy Prediction (PGA-APP) algorithms proposed in Zhang and Lesser (2010), Abdallah and Lesser (2008) empirically converge to Nash equilibrium in a wider variety of situations without requiring the knowledge of the other agents' immediate rewards or strategies.

In this work, we are interested in proposing a multi-agent learning algorithm that can converge to Nash equilibrium in a wider variety of situations. In addition, we are interested in proposing a multi-agent learning algorithm that does not make strict assumptions that are often unknown and need to be learned via experience. In this paper, we propose two multi-agent reinforcement learning algorithms. The algorithms proposed in this work are an extended version of the work published in Schwartz (2014), Awheda and Schwartz (2013). The first proposed algorithm proposed in this work can successfully converge to Nash equilibrium policies in games that have pure Nash equilibrium. The second proposed algorithm, on the other hand, can successfully learn Nash equilibrium policies in games that have pure or mixed Nash strategies. The proposed algorithms use the exponential moving average (EMA) approach in parallel with the greedy action of the learning agent's Q-table as a basis to update the learning agent's strategy. We evaluate the proposed algorithms on a variety of matrix and stochastic games. The results show that the second proposed algorithm outperforms the state-of-the-art multi-agent learning algorithms in terms of convergence to Nash equilibrium.

This paper is organized as follows. Preliminary concepts and notations are presented in Sect. 2. The constant learning rate-based exponential moving average Q-learning (CLR-EMAQL) algorithm is proposed in Sect. 3. The exponential moving average Q-learning (EMAQL) algorithm is proposed in Sect. 4. Section 5 presents the simulation and results.

2 Preliminary concepts and notations

2.1 Markov decision processes

A Markov decision process (MDP; Howard 1960; Bellman 1957) can be described as a tuple, (S, A, R, T), where S is the discrete space of states, A is the discrete space of actions, R is the reward function and T is the transition function. The reward function defines the reward that an agent j receives when choosing an action from the available actions at the given state. The transition function describes how a probability distribution is defined over the next states as a function of the given state and the agent's action (Bowling and Veloso 2002). In a Markov decision process (MDP), the objective of the agent is to find a policy $\pi : S \rightarrow A$

Dilemma Game

$$R_{1,2} = \begin{bmatrix} 10,10 & 1,12 \\ 12,1 & 2,2 \end{bmatrix}$$

Shapley's Game

$$R_{1,2} = \begin{bmatrix} 0,0 & 1,0 & 0,1 \\ 0,1 & 0,0 & 1,0 \\ 1,0 & 0,1 & 0,0 \end{bmatrix}$$

Biased Game

$$R_{1,2} = \begin{bmatrix} 1,1.85 & 1.85,1 \\ 1.15,1 & 1,1.15 \end{bmatrix}$$

Fig. 1 Matrix games

that maps states to actions so that the discounted future reward is maximized (Bowling and Veloso 2002; Hu et al. 1998).

2.2 Matrix games

A matrix game (strategic game) can be described as a tuple $(n, A_{1,\ldots,n}, R_{1,\ldots,n})$, where n is the agents' number, A_j is the discrete space of agent j's available actions, and R_j is the payoff function that agent j receives (Bowling and Veloso 2002). In matrix games, the objective of agents is to find pure or mixed strategies that maximize their payoffs. A pure strategy is the strategy that chooses actions deterministically, whereas a mixed strategy is the strategy that chooses actions based on a probability distribution over the agent's available actions. Figure 1 shows some examples of matrix games. The dilemma game, the shapley's game, and the biased game are shown. In these games, one player chooses a row and the other chooses a column in the matrix. In these games, each cell in the payoff matrix represents the payoff received by the row and column players, respectively. The dilemma game has a pure Nash equilibrium strategy that executes the second action of each player with a probability of one. The shapley's game has one mixed Nash equilibrium strategy, $\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$. On the other hand, the biased game has a mixed Nash equilibrium strategy with probabilities not uniform across actions, (0.15,0.85) and (0.85,0.15).

2.3 Stochastic games

A stochastic game can be described as a tuple $(n, S, A_{1,\ldots,n}, R_{1,\ldots,n}, T)$, where n is the agents' number, S is the discrete space of states, A_j is the discrete space of agent j's available actions, R_j is the reward function, and T is the transition function (Bowling and Veloso 2002). Stochastic games can be described as an extension of Markov decision processes (MDP). They can also be described as an extension of matrix games as each state in a stochastic game can be dealt with as a matrix game (Bowling and Veloso 2002). Figure 2 shows two stochastic games introduced by Hu and Wellman (2003). The players in both games are located in the

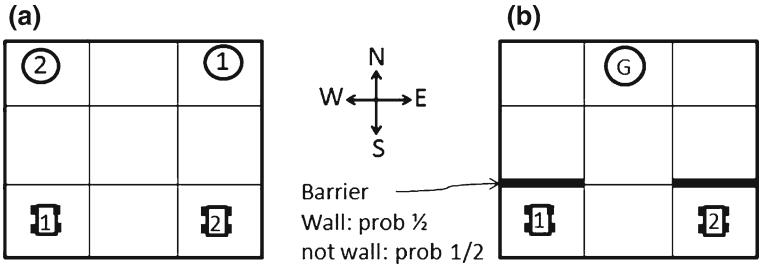

Fig. 2 Two stochastic games (Hu and Wellman 2003). **a** Grid game 1. **b** Grid game 2

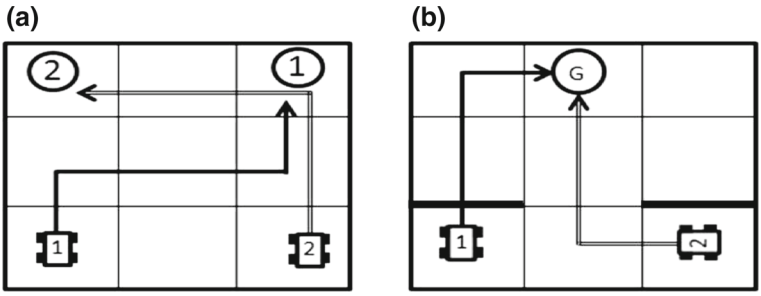

Fig. 3 **a** A Nash equilibrium of grid game 1. **b** A Nash equilibrium of grid game 2 (Hu and Wellman 2003)

lower corners and are allowed to move one cell in the four compass directions (North, East, South and West). The transition is ignored if both players move to the same cell (excluding a goal cell). The players' goals in both grid games are located as shown in Fig. 2. The transition in grid game 1 is deterministic; grid game 2, on the other hand, has deterministic and probabilistic transitions. At the lower corners in grid game 2, the probability of transition to the next cell is 0.5 when the player takes the action North. In both grid games, the player that reaches its goal is rewarded 100 points, it receives −1 points when either it hits the wall or moves into the same cell the other player moves into (excluding a goal cell), and it receives 0 points otherwise. Reaching its goal with a minimum number of steps is therefore the aim of each player in both games. As soon as a player reaches its goal, the game ends (Hu and Wellman 2003). Grid game 1 has ten different Nash equilibria; whereas grid game 2 has two different Nash equilibria (Hu and Wellman 2003). Figure 3 shows one Nash equilibrium for each grid game.

2.4 Q-learning algorithm

The Q-learning algorithm (Watkins and Dayan 1992; Watkins 1989) is one of the most well-known algorithms in reinforcement learning. The Q-learning algorithm updates the long-term payoffs of state-action pairs by interacting with the environment. The Q-learning algorithm is a single-agent learning algorithm (Watkins and Dayan 1992; Watkins 1989) that can be used in MDPs to learn optimal policies. In single-agent learning, the Q-table of a learning agent is guaranteed to converge to optimal Q-values, and hence, the learning agent learns an optimal policy by selecting the greedy actions. The Q-learning algorithm defines an optimal policy for the learning agent by learning the optimal state-action value function Q^*. Each learning agent keeps a table that saves the estimates $Q^*(s, a)$ for each state s and action a. At each state s, the learning agent selects an action a with some exploration rate so that $Q(s, a)$ is maximized. The Q-learning table of agent j is updated as follows,

$$Q^j_{t+1}(s, a_t) = (1 - \theta) Q^j_t(s, a_t) + \theta \left[r^j_t + \zeta \max_{a'} Q^j_t(s', a') \right] \tag{1}$$

where t is the number of time the state s has been visited, θ is the learning rate, r^j_t is the immediate reward of the agent j at the state s, a_t is the action chosen by the agent j at the state s, and ζ is the discount factor.

The state-action value function Q of Eq. (1) is guaranteed to converge to the optimal Q^* (Sutton and Barto 1998) if the following conditions are satisfied:

(i) The state-action pair is visited an infinite number of times.

(ii) The learning rate θ is decaying over time provided that

$$\sum_{t=0}^{\infty} \theta = \infty \quad \text{and} \quad \sum_{t=0}^{\infty} \theta^2 < \infty$$

Condition (i) states that each state-action pair has to be visited an infinite number of times which in turn emphasizes the importance of the exploration strategy. In this paper, the ϵ-greedy exploration policy is used. In this policy, the learning agent chooses the optimal action with a probability $1-\epsilon$ and chooses a random action with a probability ϵ. Condition (i) is guaranteed to satisfy when $\epsilon > 0$. Condition (ii), on the other hand, is a standard condition for stochastic approximation.

Although the Q-learning algorithm is a single-agent learning algorithm, it has been successfully used for multi-agent learning (Claus and Boutilier 1998; Sen et al. 1994; Tan 1993). Despite the loss of theoretical guarantees, Q-learning agents often succeed to learn Nash equilibrium policies in multi-agent environment (Fulda and Ventura 2007) because the Q-tables of the learning agents do not have to converge to optimal values in order for the agents to execute a Nash equilibrium policy, and the learning agents must adopt a Nash equilibrium if they are playing optimally.

3 The proposed constant learning rate-based exponential moving average Q-learning (CLR-EMAQL) algorithm

The exponential moving average (EMA) approach is a model-free strategy estimation approach. It is one of the statistical approaches used to analyze time series data in finance and technical analysis. Typically, EMA gives the recent observations more weight than the older ones (Burkov and Chaib-draa 2009). The EMA estimation approach is used in Tesauro (2004) by the hyper Q-learning algorithm to estimate the opponent's strategy. It is also used in Burkov and Chaib-draa (2009) by the Infinitesimal Gradient Ascent (IGA) agent to estimate its opponent's strategy. The EMA estimator used to estimate the strategy of the agent's opponent can be described by the following equation (Burkov and Chaib-draa 2009; Tesauro 2004):

$$\pi_{t+1}^{-j}(s,a) = (1-\eta)\pi_t^{-j}(s,a) + \eta u_t^{-j}(s,a^{-j}) \tag{2}$$

where $\pi^{-j}(s,a)$ is the opponent's strategy at the state s, η is a small constant step size and $0 < \eta << 1$, and $u^{-j}(s,a^{-j})$ is a unit vector representation of the action a^{-j} chosen by the opponent $(-j)$ at the state s. The unit vector $u^{-j}(s,a^{-j})$ contains the same number of elements the π^{-j} does. The elements in the unit vector $u^{-j}(s,a^{-j})$ are all equal to zero except for the element corresponding to the action a^{-j} which is equal to 1. For example, if the opponent $(-j)$ has four possible actions at each state and the opponent chooses the second action at the state s, the unit vector $u^{-j}(s,a^{-j})$ will be given in this case as follows, $u^{-j}(s,a^{-j}) = [0,1,0,0]$.

In this work, we propose the constant learning rate-based exponential moving average Q-learning (CLR-EMAQL) algorithm. The proposed CLR-EMAQL algorithm uses the exponential moving average (EMA) approach in parallel with the Q-learning algorithm as a basis to update the strategy of the learning agent itself. The CLR-EMAQL algorithm proposed in this section uses a constant learning rate η (constant step size). The Q-table of a learning agent j is updated by the Q-learning algorithm of Eq. (1). Despite the loss of theoretical guarantees, Q-learning agents often succeed to learn Nash equilibrium policies in a multi-

agent environment (Fulda and Ventura 2007). This is because the Q-tables of the learning agents do not have to converge to optimal values in order for the agents to execute a Nash equilibrium policy, and the learning agents must adopt a Nash equilibrium if they are playing optimally (Fulda and Ventura 2007). It is important here to mention that the proposed algorithm forces the Q-table of the learning agent to converge to a Nash equilibrium policy. When the Q-table of the learning agent converges to a Nash equilibrium policy, the policy π of the learning agent will also converge to a Nash equilibrium. The proposed CLR-EMAQL algorithm updates the agent j's policy by Eq. (3). Algorithm 1 lists the procedure of the CLR-EMAQL algorithm for a learning agent j when using a constant learning rate η.

$$\pi^j_{t+1}(s,a) = (1-\eta)\pi^j_t(s,a) + \eta u^j_t(s,a) \tag{3}$$

where $\eta \in (0,1)$ is a constant learning rate and $u^j_t(s_t)$ is defined as follows,

$$u^j_t(s,a) = \begin{bmatrix} u^j_1 & u^j_2 & \dots & u^j_m \end{bmatrix}^T = \begin{cases} V^j_1(s) & \text{if } a_t = \arg\max\limits_{a'} Q^j_t(s,a') \\ V^j_2(s) & \text{otherwise} \end{cases} \tag{4}$$

The elements $(u^j_1, u^j_2, \dots, u^j_m) \in [0,1]$, $u^j_1 + u^j_2 + \cdots + u^j_m = 1$, and m is the number of actions of the agent j. The vectors $V^j_1(s)$ and $V^j_2(s)$ consist of the same number of the elements π^j does. The elements in the vector $V^j_1(s)$ are all equal to zero except for the element corresponding to the action a_t which is equal to 1. On the other hand, $V^j_2(s) = \frac{1}{m-1}[\mathbf{1} - V^j_1(s)]$. When the action a_t chosen by the agent j at the state s is equal to the greedy action obtained from the agent's Q-table at the state s, the term $u^j_t(s,a)$ will equal to the vector $V^j_1(s)$. On the other hand, if the learning agent selects an action that is different from the greedy action, the term $u^j_t(s,a)$ will equal to the vector $V^j_2(s)$. To illustrate the definition of the vectors $V^j_1(s)$ and $V^j_2(s)$ more, let us assume that, for example, agent j has four possible actions at each state. Let us also assume that the action a_t chosen by agent j at the state s is the third action. The vector $u^j_t(s,a)$ will be defined in this case as $u^j_t(s,a) = V^j_1(s) = [u^j_1 \;\; u^j_2 \;\; u^j_3 \;\; u^j_4]^T = [0,0,1,0]^T$ if the greedy action obtained from the agent's Q-table at the state s is also the third action. On the other hand, the vector $u^j_t(s,a)$ will be defined as $u^j_t(s,a) = V^j_2(s) = [u^j_1 \;\; u^j_2 \;\; u^j_3 \;\; u^j_4]^T = [1/3, 1/3, 0, 1/3]^T$ if the greedy action obtained from the agent's Q-table at the state s is not the third action.

In the proposed CLR-EMAQL algorithm, the learning agent selects an action each time during learning based on his policy distribution $\pi(s,a)$. That is, the learning agent selects an action each time during learning and that action may not be the greedy action. The proposed CLR-EMAQL algorithm uses the greedy action as a criterion when it updates the policy of the learning agent. If the learning agent selects an action that is the same as the greedy action calculated from the Q-table of the Q-learning algorithm, then the proposed algorithm drags the agent's policy towards that action by giving the probability distribution corresponding to the selected action more weight than the other actions of the agent. We call this procedure by the operation of exploiting the greedy action by the CLR-EMAQL algorithm. If the selected action by the agent and the greedy action are different, the proposed algorithm then allows the learning agent to explore the other actions by decreasing the probability distribution of the selected action. We call this procedure by the operation of exploring actions by the CLR-EMAQL algorithm. The operation of exploring the other actions continue until the learning agent selects an action that is the same to the greedy action calculated from the Q-table. At that point, the operation of exploiting the greedy action takes place and the proposed algorithm

Algorithm 1 The constant learning rate-based exponential moving average Q-learning (CLR-EMAQL) algorithm for agent j:

Initialize:
learning rates $\theta \in (0,1]$ and $\eta \in (0,1)$
exploration rate ϵ
discount factor ζ
$Q^j(s,a) \leftarrow 0$ and $\pi^j(s,a) \leftarrow ICs$
Repeat
(a) From the state s, select an action a_t according to the strategy $\pi_t^j(s,a)$ with some exploration.
(b) Observe the immediate reward r_t^j and the new state s'.
(c) Update $Q_{t+1}^j(s,a_t)$ using Eq. (1).
(d) Update the strategy $\pi_{t+1}^j(s,a)$ by using Eq. (3).

keeps updating the policy of the learning agent towards that greedy action by giving more weights to the probability distribution corresponding to that selected action.

3.1 The multi-agent learning dynamics of the proposed CLR-EMAQL algorithm

To simplify the analysis, we consider two-player-two-action games. The policies of Player 1 and Player 2 updated by Eq. (3) can be written as follows,

$$\pi_{t+1}^1(s,a) = (1-\eta)\pi_t^1(s,a) + \eta u_t^1(s,a) \tag{5}$$
$$\pi_{t+1}^2(s,a) = (1-\eta)\pi_t^2(s,a) + \eta u_t^2(s,a) \tag{6}$$

where $u_t^1(s,a)$ and $u_t^2(s,a)$ are defined based on Eq. (4) as follows,

$$u_t^1(s,a) = \begin{cases} V_1^1(s) & \text{if } a_t = \arg\max_{a'} Q_t^1(s,a') \\ V_2^1(s) & \text{otherwise} \end{cases} \tag{7}$$

$$u_t^2(s,a) = \begin{cases} V_1^2(s) & \text{if } a_t = \arg\max_{a'} Q_t^2(s,a') \\ V_2^2(s) & \text{otherwise} \end{cases} \tag{8}$$

where $Q_t^1(s,a)$ and $Q_t^2(s,a)$ are the Q-tables for Player 1 and Player 2, respectively.

From Eqs. (7) and (8), it is shown that $u_t^1(s,a)$ is a function of $Q_t^1(s,a)$ and $u_t^2(s,a)$ is a function of $Q_t^2(s,a)$. That is,

$$u_t^1(s,a) = f_1\left(Q_t^1(s,a)\right) \tag{9}$$
$$u_t^2(s,a) = f_2\left(Q_t^2(s,a)\right) \tag{10}$$

From Eq. (1), it is shown that the Q-tables of Player 1 and Player 2 are functions of the player's action and the opponent's action as well. This is because the reward r_t^j of the player j depends on the player j's action and the opponent's action. Thus,

$$Q_t^1(s,a) = g_1\left(s, a^1, a^2\right) \tag{11}$$
$$Q_t^2(s,a) = g_2\left(s, a^1, a^2\right) \tag{12}$$

where a^1 and a^2 are the actions of Player 1 and Player 2, respectively.

Hence, from Eqs. (11) and (12), (9) and (10) can be rewritten as follows,

$$u_t^1(s,a) = f_1\big(Q_t^1(s,a)\big) = f_1\Big(g_1\big(s,a^1,a^2\big)\Big) \tag{13}$$

$$u_t^2(s,a) = f_2\big(Q_t^2(s,a)\big) = f_2\Big(g_2\big(s,a^1,a^2\big)\Big) \tag{14}$$

From Eqs. (13) and (14), we can say that the policy equations of Player 1 and Player 2 given in Eqs. (5) and (6) represent multi-agent learning equations. The convergence of these equations to a Nash equilibrium depends on the convergence of the Q-tables of Player 1 and Player 2. Although the Q-learning algorithm is a single-agent learning algorithm, it has been successfully used for multi-agent learning (Claus and Boutilier 1998; Sen et al. 1994; Tan 1993). Despite the loss of theoretical guarantees, Q-learning agents often succeed to learn Nash equilibrium policies in multi-agent environment (Fulda and Ventura 2007). This is because the Q-tables of the learning agents do not have to converge to optimal values in order for the agents to execute a Nash equilibrium policy. In addition, the learning agents must adopt a Nash equilibrium if they are playing optimally (Fulda and Ventura 2007). In this work, the proposed algorithm forces the Q-table of the learning agent to converge to a Nash equilibrium policy. When the Q-table of the learning agent converges to a Nash equilibrium policy, the strategy (policy) of the learning agent will also converge to Nash equilibrium.

3.2 The mathematical analysis of the proposed CLR-EMAQL algorithm

To simplify the analysis, we consider two-player-two-action games. The probability of selecting the first action of Player 1 at time t is referred to by $p_{1,t}$, whereas the probability of selecting the second action is referred to by $p_{2,t}$. Thus, the policy of Player 1 at state s and time t will be $\pi_t^1(s,a) = (p_{1,t}, p_{2,t})$, where $p_{1,t} + p_{2,t} = 1$. Similar to Player 1, the probability of selecting the first action of Player 2 at time t is referred to by $q_{1,t}$, whereas the probability of selecting the second action at time t is referred to by $q_{2,t}$. Thus, the policy of Player 2 at state s and time t will be $\pi_t^2 = (q_{1,t}, q_{2,t})$, where $q_{1,t} + q_{2,t} = 1$. To simplify notations, the term $u_t^j(s,a)$ in Eq. (3) is defined as $u_t^1(s,a) = [u_1^1 \quad u_2^1]^T$ for Player 1 and $u_t^2(s,a) = [u_1^2 \quad u_2^2]^T$ for Player 2, where the superscripts refer to the corresponding player and the subscripts refer to the corresponding action. Hence, the policies of Player 1 and Player 2 updated by Eq. (3) can be rewritten as,

$$\begin{bmatrix} p_{1,t+1} \\ p_{2,t+1} \\ q_{1,t+1} \\ q_{2,t+1} \end{bmatrix} = \begin{bmatrix} 1-\eta & 0 & 0 & 0 \\ 0 & 1-\eta & 0 & 0 \\ 0 & 0 & 1-\eta & 0 \\ 0 & 0 & 0 & 1-\eta \end{bmatrix} \begin{bmatrix} p_{1,t} \\ p_{2,t} \\ q_{1,t} \\ q_{2,t} \end{bmatrix} + \begin{bmatrix} \eta u_1^1 \\ \eta u_2^1 \\ \eta u_1^2 \\ \eta u_2^2 \end{bmatrix} \tag{15}$$

To analyze the above equation, we use the ordinary differential equation (ODE) approach. The behavior of the learning algorithm can be approximated by ODEs as the step size goes to zero (Thathachar and Sastry 2011). Thus, when the learning rate (step size) $\eta \to 0$, the ordinary differential equation of Eq. (15) can be given as follows,

$$\begin{bmatrix} \dot{p_1} \\ \dot{p_2} \\ \dot{q_1} \\ \dot{q_2} \end{bmatrix} = \begin{bmatrix} u_1^1 - p_1 \\ u_2^1 - p_2 \\ u_1^2 - q_1 \\ u_2^2 - q_2 \end{bmatrix} \tag{16}$$

When the Q-table of each learning agent converges to a Nash equilibrium policy, the above ordinary differential equation can be viewed as a linear time-invariant equation. This linear

time-invariant equation will be asymptotically stable as its eigenvalues are all real and less than zero, and they are $[-1 \ -1 \ -1 \ -1]^T$. If we let the right hand side of the above equation equal to zero, we then get the equilibrium solutions of the above equation as $p_1^* = u_1^1$, $p_2^* = u_2^1$, $q_1^* = u_1^2$, and $q_2^* = u_2^2$. For a two-player-two-action game, each parameter of the vector $u_t^1(s,a) = [u_1^1 \quad u_2^1]^T$ for Player 1 will have a value of either zero or one, as stated in Eq. (3). Similar to Player 1, each parameter of the vector $u_t^2(s,a) = [u_1^2 \quad u_2^2]^T$ for Player 2 will also have a value of either zero or one. Therefore, the possible equilibrium solutions of Eq. (16) can be specified as follows,

$$\begin{aligned}(\pi^{1*}, \pi^{2*}) &= \big((1,0),(1,0)\big)\\ (\pi^{1*}, \pi^{2*}) &= \big((1,0),(0,1)\big)\\ (\pi^{1*}, \pi^{2*}) &= \big((0,1),(1,0)\big)\\ (\pi^{1*}, \pi^{2*}) &= \big((0,1),(0,1)\big)\end{aligned} \tag{17}$$

Without loss of generality, let us assume that the last equilibrium solution, $(\pi^{1*}, \pi^{2*}) = \big((0,1),(0,1)\big)$, is the Nash equilibrium. Let us also assume that both players are adopting this Nash equilibrium. This means that selecting the second action by each player (Player 1 and Player 2) with a probability of one is the Nash equilibrium. This also means that the Q-table of each player updated by Eq. (1) converges to a Nash equilibrium policy so that the second action is the greedy action of each player's Q-table. Hence, none of the first three equilibrium solutions of Eq. (17) will be the Nash equilibrium solution as the proposed CLR-EMAQL algorithm of each player drags the player's policy away from these equilibrium solutions. Only the equilibrium solution $(\pi^{1*}, \pi^{2*}) = \big((0,1),(0,1)\big)$ will be the Nash equilibrium of the game.

The equilibrium solution $(\pi^{1*}, \pi^{2*}) = \big((1,0),(1,0)\big)$: This equilibrium solution means that Player 1 selects its first action with a probability of one ($p_1 = 1, p_2 = 0$), and Player 2 also selects its first action with a probability of one ($q_1 = 1, q_2 = 0$). Since the greedy action of Player 1's Q-table is the second action, the term $u_t^1(s,a)$ of Player 1 will be defined as stated in Eq. (3) as follows,

$$u_t^1(s,a) = \begin{bmatrix} u_1^1 & u_2^1 \end{bmatrix}^T = [0 \quad 1]^T$$

On the other hand, because the greedy action of Player 2's Q-table is the second action, the term $u_t^2(s,a)$ of Player 2 will be defined as stated in Eq. (3) as follows,

$$u_t^2(s,a) = [u_1^2 \quad u_2^2]^T = [0 \quad 1]^T$$

Hence, Eq. (16) can be defined as follows,

$$\begin{bmatrix} \dot{p_1} \\ \dot{p_2} \\ \dot{q_1} \\ \dot{q_2} \end{bmatrix} = \begin{bmatrix} -p_1 \\ 1 - p_2 \\ -q_1 \\ 1 - q_2 \end{bmatrix} \tag{18}$$

The steady state values of Eq. (18) can be calculated by setting the right half side of the equation equal to zero. Thus, the steady state values of Eq. (18) are given as follows, $p_1^* = 0$, $p_2^* = 1$, $q_1^* = 0$ and $q_2^* = 1$. This means that the equilibrium solution $(\pi^{1*}, \pi^{2*}) = \big((1,0),(1,0)\big)$ is not the Nash equilibrium. This is because the proposed CLR-EMAQL algorithm of each player drags the player's policy away from this equilibrium solution.

We can also do a similar analysis to the other equilibrium solutions given in Eq. (17). The analysis will show that only the equilibrium solution $(\pi^{1*}, \pi^{2*}) = \big((0,1),(0,1)\big)$ will be

(a) $$R_{1,2} = \begin{bmatrix} 1,1 & 0,0 \\ 0,0 & 2,2 \end{bmatrix}$$ **(b)** $$R_{1,2} = \begin{bmatrix} 1,-1 & -1,1 \\ -1,1 & 1,-1 \end{bmatrix}$$

Fig. 4 Two matrix games. **a** Coordination game. **b** Matching Pennies game

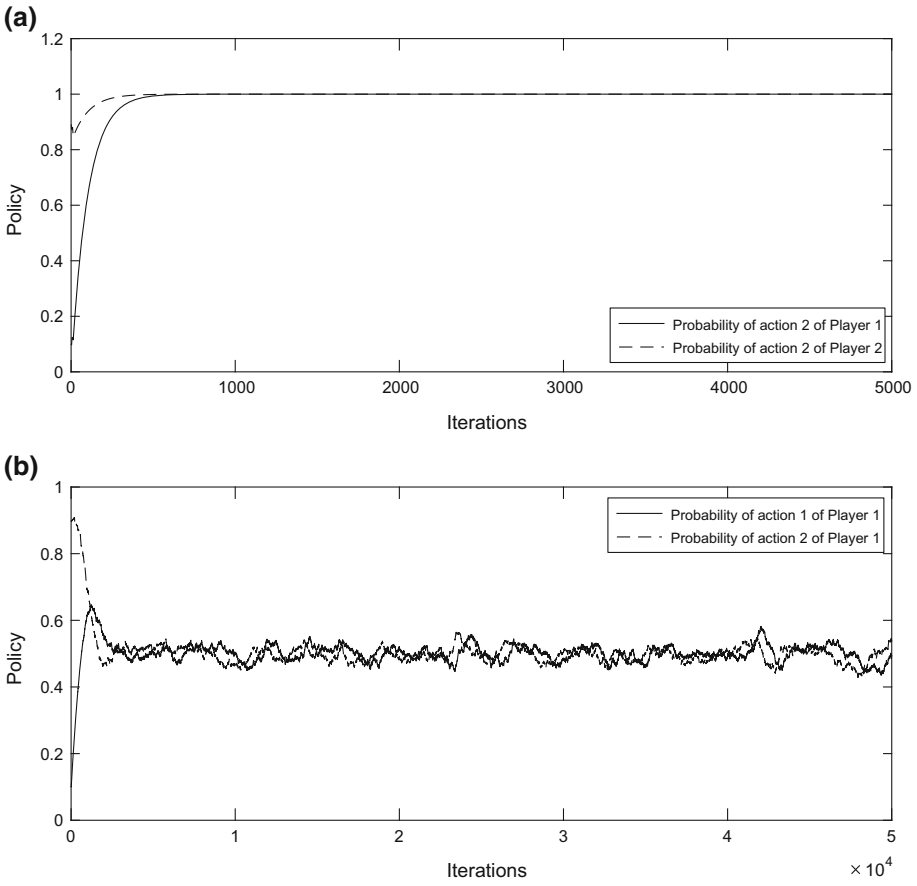

Fig. 5 Probability of selecting actions: **a** probability of choosing the second action for both players in the coordination game by CLR-EMAQL algorithm **b** probability of choosing Player 1's actions in the the matching pennies game by CLR-EMAQL algorithm

the Nash equilibrium of the game. This is because the proposed CLR-EMAQL algorithm of each player will drag the player's policy towards this equilibrium solution.

In a two-player-two-action game, we have mathematically shown that the players learning with the proposed CLR-EMAQL algorithm successfully converge to Nash equilibrium if the game has a pure Nash equilibrium. However, in a two-player-two-action game with a mixed Nash equilibrium, the players learning with the proposed CLR-EMAQL algorithm will fail to adopt a mixed Nash equilibrium. The strategies of both players will keep oscillating around the mixed Nash equilibrium. This is also true in games with more than two players and with

two actions or more. The strategies of the players learning with the proposed CLR-EMAQL algorithm will either converge to the Nash equilibrium if the game has a pure Nash strategy or oscillate around it if the game has a mixed Nash strategy.

To illustrate the performance of the proposed CLR-EMAQL algorithm, the algorithm is used to learn the games depicted in Fig. 4; the coordination game and the matching pennies game. The coordination game has a pure Nash strategy that executes the second action with a probability of 1, whereas the matching pennies game has a mixed strategy that executes each action with a probability of 0.5. Figure 5a illustrates the probability distributions of action 2 of both players (Player 1 and Player 2) when using the proposed CLR-EMAQL algorithm to learn the coordination game. As shown in this figure, both players converge to their pure Nash equilibrium. This shows that the equilibrium points of the linear time-invariant system of Eq. (16) are reached, where $p_1^* = u_1^1 = 0$, $p_2^* = u_2^1 = 1$, $q_1^* = u_1^2 = 0$, and $q_2^* = u_2^2 = 1$. On the other hand, Fig. 5b illustrates the probability distributions of action 1 and action 2 of Player 1 (policy of Player 1) when using the proposed CLR-EMAQL algorithm to learn the matching pennies game. This figure shows that the proposed CLR-EMAQL algorithm enforces the policy of Player 1 to oscillate around the Nash equilibrium strategy. As stated earlier in this section, in case of mixed Nash equilibrium, the policy of a CLR-EMAQL player will continue oscillating around the mixed Nash equilibrium. In this case, a decaying learning rate has to be used so that the oscillation around the Nash equilibrium will vanish and both players adopt their mixed Nash equilibrium.

4 The proposed exponential moving average Q-learning (EMAQL) algorithm

In this section, we introduce the proposed exponential moving average Q-learning (EMAQL) algorithm. The proposed EMAQL algorithm uses two different decaying learning rates (η_w and η_l) when updating the agent's strategy instead of only one constant learning rate η as used in the proposed CLR-EMAQL algorithm. The values of these variable learning rates are inversely proportional to the number of iterations (or episodes) and are set based on one of two different mechanisms; the Win-or-Learn-Fast (WoLF) mechanism or the Win-or-Learn-Slow (WoLS) mechanism. The Q-table of a learning agent j is updated by the Q-learning algorithm of Eq. (1) and the agent j's policy is updated by Eq. (19) as follows,

$$\pi_{t+1}^j(s,a) = (1-\eta_t)\pi_t^j(s,a) + \eta_t u_t^j(s,a) \tag{19}$$

where η_t is a decaying learning rate and $\eta_t \in (0,1)$. The term $u_t^j(s,a)$ and the learning rate η_t are defined as follows,

$$u_t^j(s,a) = \begin{bmatrix} u_1^j & u_2^j & \dots & u_m^j \end{bmatrix}^T = \begin{cases} V_1^j(s) & \text{if } a_t = \arg\max\limits_{a'} Q_t^j(s,a') \\ V_2^j(s) & \text{otherwise} \end{cases} \tag{20}$$

$$\eta_t = \begin{cases} \eta_w & \text{if } a_t = \arg\max\limits_{a'} Q_t^j(s,a') \\ \eta_l & \text{otherwise} \end{cases} \tag{21}$$

where,

$$\eta_l = \begin{cases} \eta_l^{WoLF} & \text{if WoLF mechanism is used} \\ \eta_l^{WoLS} & \text{if WoLS mechanism is used} \end{cases} \tag{22}$$

4.1 WoLF and WoLS mechanisms

The WoLF principle was used with reinforcement learning algorithms in Bowling and Veloso (2001a, b, 2002). The WoLF mechanism used by the WoLF-PHC algorithm (Bowling and Veloso 2002) uses two different learning rates, δ_w when the algorithm is winning and δ_l when it is losing. The WoLF-PHC algorithm uses the difference between the expected value of the average strategy and the expected value of the current strategy as a criterion to decide when the algorithm wins or loses. The learning rate δ_l is bigger than the learning rate δ_w. As such, when the learning agent is losing, it learns faster than when it is winning. This makes the agent adapt quickly to the changes in the strategies when it is losing and to learn cautiously when it is winning (Bowling and Veloso 2002). In this work, the proposed EMAQL algorithm uses either the WoLF mechanism or the WoLS mechanism to update the policy of the learning agent.

The WoLF mechanism used by the proposed EMAQL algorithm is similar to the WoLF mechanism used by the WoLF-PHC algorithm. It has two different learning rates (η_l^{WoLF} when losing and η_w when winning), where $\eta_l^{WoLF} > \eta_w$. However, the criterion used by the proposed EMAQL algorithm to decide when the algorithm wins or loses is different from the criterion used by the WoLF-PHC algorithm. The proposed EMAQL algorithm uses the greedy action as a criterion to decide when the algorithm wins or loses. If the learning agent selects an action that is the same as the greedy action calculated from the Q-table of the learning agent, the algorithm wins and uses η_w. If the selected action is different from the greedy action, the algorithm loses and uses η_l^{WoLF}.

On the other hand, the WoLS mechanism used by the proposed EMAQL algorithm has two different learning rates (η_l^{WoLS} when losing and η_w when winning), where $\eta_l^{WoLS} < \eta_w$. The essence of the WoLS mechanism is to learn fast when the algorithm wins and to learn slowly when the algorithm loses. Hence, when the learning agent is losing, the policy of the learning agent is changed very slowly. Learning slowly when the algorithm is losing gives the algorithm the time to explore the other actions before deciding whether the current strategy (policy) is good for the learning agent or not. In other words, learning slowly gives the learning agent the opportunity to see the response of the opponents to the learning agent's current strategy. If the opponents' response to the learning agent's current strategy benefits the learning agent, the algorithm will eventually transition from losing to winning when the greedy action obtained from the Q-table of the learning agent becomes similar to the action selected by the learning agent. On the other hand, if the opponents' response to the learning agent's current strategy does not benefit the learning agent, the algorithm will continue losing and, at the same time, exploring the other actions until a new strategy is adopted when the greedy action and the learning agent's selected action are similar to each other. At that time, the algorithm transitions from losing to winning.

4.2 When to use WoLF and WoLS mechanisms?

The proposed EMAQL algorithm uses the rewards received by the learning agent to decide which mechanism (WoLF mechanism or WoLS mechanism) to use for the game being learned. If the values of the Q-table of the learning agent for at least one action from the available actions are changing in one direction all the time [i.e. $r_t(s, a_i) > 0$ or $r_t(s, a_i) \leq 0$], then the proposed EMAQL algorithm uses the WoLS mechanism to update the policy of the learning agent. On the other hand, if the values of the of the Q-table of the learning agent for all actions are not changing in one direction, then the pro-

posed EMAQL algorithm uses the WoLF mechanism to update the policy of the learning agent. As such, the proposed EMAQL algorithm selects a learning mechanism (WoLF or WoLS) to update the policy of the learning agent based on the following reward conditions:

$$r_t(s, a_i) > 0 \tag{23}$$

$$r_t(s, a_i) \leq 0 \tag{24}$$

r_t is the reward received by the learning agent at state s when selecting the action a_i, where $i = 1, \ldots, m$ and m is the number of the available actions for the learning agent at state s.

When the agent is learning its policy, the proposed EMAQL algorithm checks whether the rewards received by the learning agent when selecting each action a_i satisfy only one of the two above reward conditions or both of them. If there is at least one action a_i from the available actions such that the rewards received by the learning agent satisfy only one of the previous two reward conditions all the time, the proposed EMAQL algorithm will use the WoLS mechanism to update the policy of the learning agent. On the other hand, if each action a_i of the available actions such that the rewards received by the learning agent satisfy both of the previous two reward conditions, the proposed EMAQL algorithm will use the WoLF mechanism to update the policy of the learning agent.

To illustrate more how the proposed EMAQL algorithm decides which mechanism (WoLF mechanism or WoLS mechanism) to use for each game to update the policy of the learning agent, let us consider the two-player-two-action matrix games depicted in Figs. 1 and 4; the dilemma game, the biased game, the coordination game, and the matching pennies game. It is important to mention here that the proposed EMAQL algorithm does not know the reward matrix of the game being learned and only knows the immediate reward received by the learning agent when selecting an action a_i at the state s. In the dilemma and the biased games, the immediate rewards received by Player 1 when selecting the first action, a_1, are always greater than 0. i.e. $r_t(s, a_1) > 0$. In this case and even without looking to the second action's immediate rewards, the proposed EMAQL algorithm will decide to use the WoLS mechanism to update the policy of Player 1 in these games. This is because the immediate rewards received by Player 1 when selecting the first action, a_1, always satisfy one of the above two reward conditions, the first condition. In the coordination and the matching pennies games, the immediate rewards received by Player 1 when selecting the first action, a_1, satisfy both of the above reward conditions at the same time. i.e. the immediate rewards sometimes satisfy the first condition of the above reward conditions and sometimes satisfy the second condition of the above reward conditions. In addition, the immediate rewards received by Player 1 when selecting the second action, a_2, also satisfy both of the above reward conditions. In this case, the proposed EMAQL algorithm will decide to use the WoLF mechanism to update the policy of Player 1 in these games. This is because the immediate rewards received by Player 1 when selecting any of its actions always satisfy both of the above two reward conditions.

Algorithm 2 lists the procedure of the proposed EMAQL algorithm for a learning agent j when using a decaying learning rate η_t. As illustrated in the proposed CLR-EMAQL algorithm, when the Q-table of the learning agent converges to a Nash equilibrium policy, the policy of the learning agent will also converge to a Nash equilibrium. We will now illustrate how the proposed EMAQL algorithm works when the game has either a pure Nash equilibrium or mixed Nash equilibrium.

4.3 Pure Nash equilibrium

For simplicity and without loss of generality, let us assume that we have a two-player game with some actions for each player. Let us also assume that each player is using the proposed EMAQL algorithm to learn its policy (self-play learning). When the learning starts and a joint action is selected by both players, then each player has two possible scenarios:
Scenario 1: The action selected by Player j ($j = 1, 2$) is the same as the greedy action calculated from the Q-table of Player j. In this case, the proposed EMAQL algorithm of Player j will update the player's policy so that it is dragged towards the player's greedy action by giving the probability distribution corresponding to the player's selected action more weight than the player's other actions. As stated in Algorithm 2, the player learning with the proposed EMAQL algorithm selects its action each time based on its policy distribution π with some exploration. Player j will continue updating its policy as stated in Scenario 1 as long as its selected action is the same as the greedy action calculated from Player j's Q-table. This is because the selected action is maximizing the player's Q-table corresponding to this action. That is, maximizing the player's payoff.
Scenario 2: The action selected by Player j is different from the player's greedy action calculated from Player j's Q-table. In this case, the proposed EMAQL algorithm of Player j will update the player's policy by decreasing the probability distribution of the player's selected action so that the player may explore its other actions. The amount of this reduction will depend on the updating policy mechanism used by the proposed EMAQL algorithm; the WoLF mechanism or the WoLS mechanism. This scenario will continue as long as the selected action and the greedy action are different. Once Player j selects an action that is similar to the greedy action, then the proposed EMAQL algorithm of Player j will update the player's policy as stated in Scenario 1.

The operation of exploiting and exploring actions will continue by both players until a pure Nash equilibrium is adopted by both players. This pure Nash equilibrium will be reached through Scenario 1 when each player executes the selected action with a probability of one, and the selected action of each player is the same as the player's greedy action.

4.4 Mixed Nash equilibrium

Likewise in the pure Nash equilibrium case, the proposed EMAQL algorithm of each player continues exploiting and exploring actions. As time goes on, the players' policies oscillate around the Nash equilibrium. This is because each player keeps changing its actions trying to maximize its payoff. Thus, both players' policies will keep oscillating around the mixed Nash equilibrium as both players keep changing their actions. In this case, the importance of decaying the learning rate (step size) η_t of the proposed EMAQL algorithm arises. The more the learning rate η_t decays, the less the oscillation of the players' policies around the Nash equilibrium becomes. Therefore, when the value of the decaying learning rate η_t becomes zero, the oscillation of the players' policies around the Nash equilibrium will be zero. At this point, a mixed Nash equilibrium is adopted by both players.

4.5 Nash equilibrium $\pi^*(s, a)$

The update rule of the policy iterate $\pi_{t+1}(s, a)$ in the proposed EMAQL algorithm is given as follows,

$$\pi_{t+1}(s, a) = (1 - \eta_t)\pi_t(s, a) + \eta_t u_t(s, a) \tag{25}$$

Algorithm 2 The proposed exponential moving average Q-learning (EMAQL) algorithm for agent j:

Initialize:
learning rates $\theta \in (0,1]$ and $\eta_w \in (0,1)$
set η_l as η_l^{WoLS}
exploration rate ϵ
discount factor ζ
$Q^j(s,a) \leftarrow 0$ and $\pi^j(s,a) \leftarrow ICs$
Repeat
(a) From the state s, select an action a_t according to the strategy $\pi_t^j(s,a)$ with some exploration.
(b) Observe the immediate reward r_t^j and the new state s'.
(c) Update the learning rate η_l, η_l^{WoLF} or η_l^{WoLS}, based on Eqs. (23) and (24).
(d) Update $Q_{t+1}^j(s,a_t)$ using Eq. (1).
(e) Update the strategy $\pi_{t+1}^j(s,a)$ by using Eq. (19).

Let us for now assume that the strategy iterate $\pi_{t+1}(s,a)$ in Eq. (25) has a Nash equilibrium $\pi^*(s,a)$. Thus, when Eq. (25) converges to this Nash equilibrium $\pi^*(s,a)$, we get

$$\pi_{t+1}(s,a) = \pi_t(s,a) = \pi^*(s,a) \tag{26}$$

From Eqs. (25) and (26), we get

$$\pi^*(s,a) = (1-\eta_t)\pi^*(s,a) + \eta_t u_t(s,a)$$

Or,

$$\pi^*(s,a) = \pi^*(s,a) + \eta_t[u_t(s,a) - \pi^*(s,a)] \tag{27}$$

Thus, for the proposed EMAQL algorithm to converge to Nash equilibrium, both sides of Eq. (27) must equal to each other. Therefore, at least one of the following conditions must be achieved:

Condition 1: $\pi^*(s,a)$ is equal to $u_t(s,a)$. i.e.

$$\pi_{t+1}(s,a) \underset{t\to\infty}{\to} u_t(s,a) \tag{28}$$

Condition 2: η_t is equal to zero. i.e.

$$\eta_t \underset{t\to\infty}{\to} 0 \tag{29}$$

When the Nash equilibrium $\pi^*(s,a)$ is a pure strategy, the proposed EMAQL algorithm will enforce the strategy iterate $\pi_{t+1}(s,a)$ of the learning agent to converge to its Nash equilibrium $\pi^*(s,a)$ so that Condition 1 is achieved, and when $t \to \infty$, Condition 2 is achieved too. However, in a game with a pure Nash equilibrium, Condition 2 does not have to be satisfied in order for an EMAQL agent to converge to its pure Nash equilibrium. On the other hand, when the Nash equilibrium $\pi^*(s,a)$ is a mixed strategy, the proposed EMAQL algorithm will enforce the strategy iterate $\pi_{t+1}(s,a)$ of the learning agent to oscillate around its Nash equilibrium $\pi^*(s,a)$. When Condition 2 is achieved, the strategy iterate $\pi_{t+1}(s,a)$ converges to the Nash equilibrium $\pi^*(s,a)$.

To illustrate the effect of Condition 1 and Condition 2 on the convergence of the proposed EMAQL algorithm, the algorithm is used to learn the games depicted in Fig. 4; the coordination game and the matching pennies game. The coordination game has a pure Nash strategy that executes the second action with a probability of 1, whereas the matching pennies game has a mixed strategy that executes each action with a probability of 0.5. Figure 6a illustrates

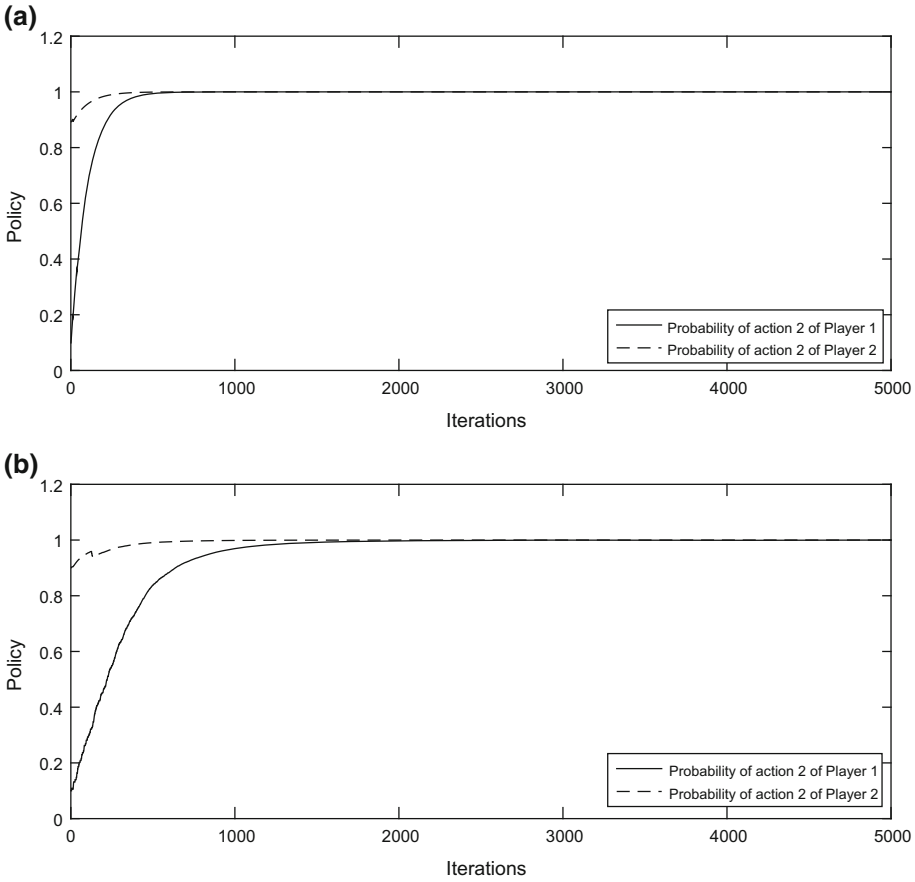

Fig. 6 Probability of choosing the second action for both players in the coordination game by EMAQL: **a** when using a constant learning rate η_t. **b** when using a decaying learning rate η_t

the probability distributions of action 2 of both players (Player 1 and Player 2) when using the proposed EMAQL algorithm to learn the coordination game (a constant learning rate η is used in this case). As shown in this figure, both players converge to their pure Nash equilibrium. This shows that Condition 1 is satisfied as each player' strategy converges to $u_t(s, a)$, where $u_t(s, a) = [0\ 1]^T$ in this case. On the other hand, Fig. 6b illustrates the probability distributions of action 2 of both players (Player 1 and Player 2) when using the proposed EMAQL algorithm to learn the coordination game (a decaying learning rate η_t is used in this case). This figure shows that both players converge to their Nash equilibrium when Condition 2 is satisfied. Figure 6 shows that in a game with a pure Nash equilibrium, an EMAQL player will converge to its Nash equilibrium when Condition 1 is satisfied even if Condition 2 is not satisfied. This figure also shows that the pure Nash equilibrium will be reached through Scenario 1 when each player executes the selected action with a probability of one, and the selected action of each player is the same as the player's greedy action. Figure 7a illustrates the probability distributions of action 1 and action 2 of Player 1 (policy of Player 1) in the matching pennies game when learning with the proposed EMAQL algorithm (a constant learning rate η is used in this case). This figure shows that the proposed EMAQL

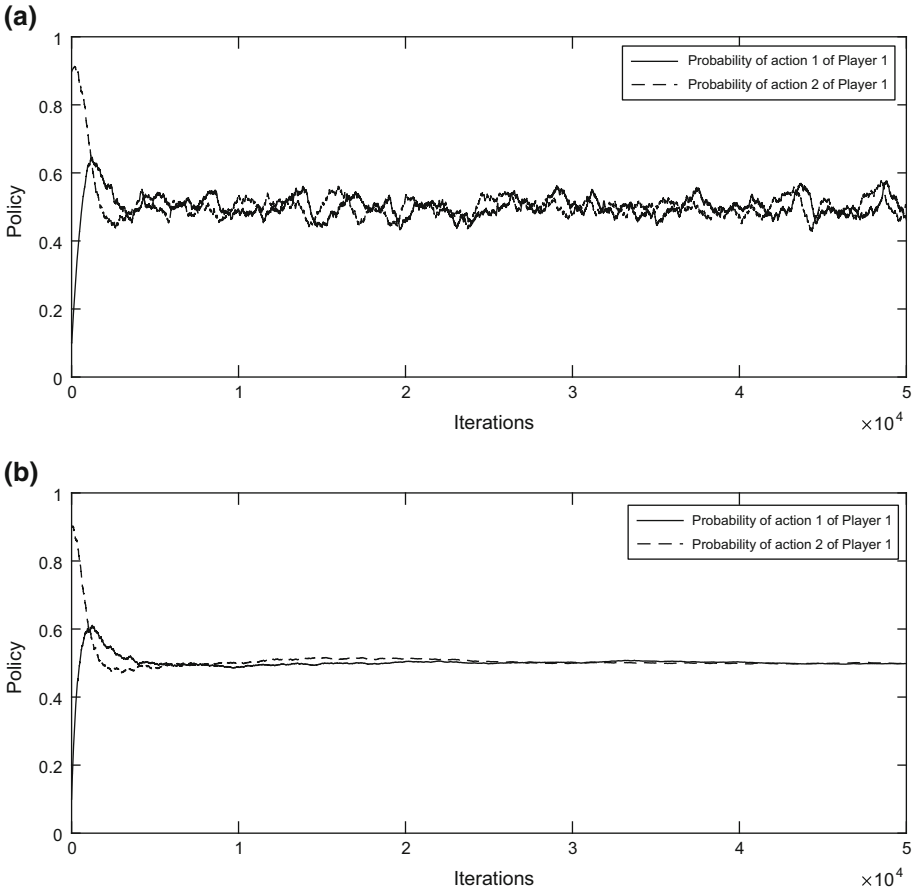

Fig. 7 Probability of choosing Player 1's actions in the the matching pennies game by EMAQL: **a** when using a constant learning rate η_t. **b** when using a decaying learning rate η_t

algorithm enforces the policy of Player 1 to oscillate around the Nash equilibrium strategy. As stated earlier, in case of mixed Nash equilibrium, the policy of the learning player will continue oscillating around the mixed Nash equilibrium when a constant learning rate η is used. In this case a decaying learning rate has to be used so that the oscillation around the Nash equilibrium vanishes and both players adopt their mixed Nash equilibrium. Figure 7b, on the other hand, shows that using the proposed EMAQL algorithm (with a decaying learning rate η_t) affects the learning performance and makes the policy of Player 1 converge to the mixed Nash equilibrium. Figure 7b shows that in a game with a mixed Nash equilibrium, an EMAQL player will converge to its Nash equilibrium when condition 2 is satisfied.

4.6 The mathematical analysis of the proposed EMAQL algorithm

To simplify analysis, we consider two-player-two-action games. However, what we are going to present here is also valid for any other multi-agent learning game. The probabilities of selecting actions at time t are referred to by $p_{1,t}$ and $p_{2,t}$ for the first and the second actions of Player 1, respectively. Similar to Player 1, the probabilities of selecting actions are referred

to by $q_{1,t}$ and $q_{2,t}$ for the first and the second actions of Player 2, respectively. Thus, the policy of Player 1 at state s and time t will be $\pi_t^1(s,a) = (p_{1,t}, p_{2,t})$, where $p_{1,t} + p_{2,t} = 1$. On the other hand, the policy of Player 2 at state s and time t will be $\pi_t^2 = (q_{1,t}, q_{2,t})$, where $q_{1,t} + q_{2,t} = 1$. To simplify notations, the term $u_t^j(s,a)$ in Eq. (19) is defined as $u_t^1(s,a) = [u_1^1 \quad u_2^1]^T$ for Player 1 and $u_t^2(s,a) = [u_1^2 \quad u_2^2]^T$ for Player 2, where the superscripts refer to the corresponding player and the subscripts refer to the corresponding action. Without loss of generality, let us rewrite the equation of updating the policy iterate $\pi_{t+1}(s,a)$ given by Eq. (19) as follows,

$$\pi_{t+1}^j(s,a) = (1 - \eta_c\eta_t)\pi_t^j(s,a) + \eta_c\eta_t u_t^j(s,a) \tag{30}$$

where η_c is a small constant step size and $\eta_c\eta_t \in (0,1)$.

Thus, the policies of Player 1 and Player 2 updated by Eq. (30) can be rewritten as follows,

$$\begin{bmatrix} p_{1,t+1} \\ p_{2,t+1} \\ q_{1,t+1} \\ q_{2,t+1} \end{bmatrix} = \begin{bmatrix} 1-\eta_c\eta_t & 0 & 0 & 0 \\ 0 & 1-\eta_c\eta_t & 0 & 0 \\ 0 & 0 & 1-\eta_c\eta_t & 0 \\ 0 & 0 & 0 & 1-\eta_c\eta_t \end{bmatrix} \begin{bmatrix} p_{1,t} \\ p_{2,t} \\ q_{1,t} \\ q_{2,t} \end{bmatrix} + \begin{bmatrix} \eta_c\eta_t u_1^1 \\ \eta_c\eta_t u_2^1 \\ \eta_c\eta_t u_1^2 \\ \eta_c\eta_t u_2^2 \end{bmatrix} \tag{31}$$

To analyze the above equation, we use the ordinary differential equation (ODE) approach. The behavior of the learning algorithm can be approximated by ODEs as the step size goes to zero (Thathachar and Sastry 2011). Thus, when the step size $\eta_c \to 0$, the ordinary differential equation of Eq. (31) can be given as follows,

$$\begin{bmatrix} \dot{p_1} \\ \dot{p_2} \\ \dot{q_1} \\ \dot{q_2} \end{bmatrix} = \begin{bmatrix} -\eta_t & 0 & 0 & 0 \\ 0 & -\eta_t & 0 & 0 \\ 0 & 0 & -\eta_t & 0 \\ 0 & 0 & 0 & -\eta_t \end{bmatrix} \begin{bmatrix} p_1 \\ p_2 \\ q_1 \\ q_2 \end{bmatrix} + \begin{bmatrix} \eta_t u_1^1 \\ \eta_t u_2^1 \\ \eta_t u_1^2 \\ \eta_t u_2^2 \end{bmatrix} \tag{32}$$

or,

$$\begin{bmatrix} \dot{p_1} \\ \dot{p_2} \\ \dot{q_1} \\ \dot{q_2} \end{bmatrix} = \begin{bmatrix} -\eta_t p_1 + \eta_t u_1^1 \\ -\eta_t p_2 + \eta_t u_2^1 \\ -\eta_t q_1 + \eta_t u_1^2 \\ -\eta_t q_2 + \eta_t u_2^2 \end{bmatrix} \tag{33}$$

To find the equilibrium solutions of the the above ordinary differential equation, let the right hand side of Eq. (33) equal to zero. Because the parameters of the vectors $u_t^1(s,a)$ and $u_t^2(s,a)$ are either 0 or 1, the possible equilibrium solutions of the above ordinary differential equation can be specified as follows,

$$\begin{aligned} (\pi^{1*}, \pi^{2*}) &= \big((1,0),(1,0)\big) \\ (\pi^{1*}, \pi^{2*}) &= \big((1,0),(0,1)\big) \\ (\pi^{1*}, \pi^{2*}) &= \big((0,1),(1,0)\big) \\ (\pi^{1*}, \pi^{2*}) &= \big((0,1),(0,1)\big) \\ (\pi^{1*}, \pi^{2*}) &= \big((p_1^*, p_2^*),(q_1^*, q_2^*)\big) \end{aligned} \tag{34}$$

Hence, the ordinary differential equation of Eq. (32) [or Eq. (33)] will have either a pure Nash equilibrium when $\left[u_1^1 - p_1,\ u_2^1 - p_2,\ u_1^2 - q_1,\ u_2^2 - q_2\right]^T = 0$ (Condition 1 in the previous subsection), or a mixed Nash equilibrium when $\eta_t \to 0$ (Condition 2 in the previous subsection).

Without loss of generality, let us assume that we have a two-player-two-action game. Thus, the possible equilibrium solutions of the game can be specified as given in Eq. (34). Without loss of generality, let us assume that the game has a pure Nash equilibrium and the equilibrium solution $(\pi^{1*}, \pi^{2*}) = \big((0, 1), (0, 1)\big)$ of Eq. (34) is the Nash equilibrium. Let us also assume that both players adopt this Nash equilibrium. That is, each player selects its second action with a probability of one. This also means that the greedy action of the Q-table of each player is the second action. Therefore, none of the first three equilibrium solutions of Eq. (34) will be the Nash equilibrium of the game. This is because of the same reasons we illustrate in the proposed CLR-EMAQL algorithm. The last equilibrium solution of Eq. (34), $(\pi^{1*}, \pi^{2*}) = \big((p_1^*, p_2^*), (q_1^*, q_2^*)\big)$, will not be the Nash equilibrium of the game either. This is because this equilibrium solution means that Player 1 selects its actions with a probability of $(p_1 = p_1^*, p_2 = p_2^*)$, and Player 2 selects its actions with a probability of $(q_1 = q_1^*, q_2 = q_2^*)$. Since the greedy action of Player 1's Q-table is the second action, the term $u_t^1(s, a)$ of Player 1 will be defined as stated in Eq. (19) as follows whatever the selected action by Player 1 is,

$$u_t^1(s, a) = \begin{bmatrix} u_1^1 & u_2^1 \end{bmatrix}^T = [0 \quad 1]^T$$

On the other hand, because the greedy action of Player 2's Q-table is the second action, the term $u_t^2(s, a)$ of Player 2 will be defined as stated in Eq. (19) as follows whatever the selected action by Player 2 is,

$$u_t^2(s, a) = [u_1^2 \quad u_2^2]^T = [0 \quad 1]^T$$

Hence, Eq. (33) can be defined as follows,

$$\begin{bmatrix} \dot{p_1} \\ \dot{p_2} \\ \dot{q_1} \\ \dot{q_2} \end{bmatrix} = \begin{bmatrix} -p_1 \\ 1 - p_2 \\ -q_1 \\ 1 - q_2 \end{bmatrix} \tag{35}$$

The steady state values of Eq. (35) are $p_1^* = 0$, $p_2^* = 1$, $q_1^* = 0$ and $q_2^* = 1$. This means that the equilibrium solution $(\pi^{1*}, \pi^{2*}) = \big((p_1^*, p_2^*), (q_1^*, q_2^*)\big)$ is not the Nash equilibrium of the game. This is because the proposed EMAQL algorithm of each player drags the player's policy away from this equilibrium solution. Hence, only the equilibrium solution $(\pi^{1*}, \pi^{2*}) = \big((0, 1), (0, 1)\big)$ will be the Nash equilibrium of the game. This is because the proposed EMAQL algorithm of each player drags the player's policy towards this equilibrium solution.

Let us now assume that the two-player-two-action game has a mixed Nash equilibrium, and the equilibrium solution $(\pi^{1*}, \pi^{2*}) = \big((p_1^*, p_2^*), (q_1^*, q_2^*)\big)$ of Eq. (34) is the mixed Nash equilibrium of the game. Let us also assume that both players adopt this Nash equilibrium. That is, Player 1 selects its actions with a probability of $(p_1 = p_1^*, p_2 = p_2^*)$, and Player 2 selects its actions with a probability of $(q_1 = q_1^*, q_2 = q_2^*)$. The greedy action of the Q-table of each player will not be a specific action all the time. This is because, in mixed Nash equilibrium games, no player will benefit from using the same action with a probability of one all the time as its opponent (the other player) will take advantage of that and maximize its payoff. This will make both players keep changing their actions. Thus, none of the first four equilibrium solutions of Eq. (34) will be the Nash equilibrium of the game as the proposed EMAQL algorithm of each player will drag the player's policy away from adopting a pure strategy. Therefore, the strategy of each player will keep oscillating around the game's Nash

equilibrium until both players adopt their mixed Nash equilibrium when $\eta_t \to 0$. Hence, the equilibrium solution $(\pi^{1*}, \pi^{2*}) = \left((p_1^*, p_2^*), (q_1^*, q_2^*)\right)$ of Eq. (34) is the only Nash equilibrium of the game. It is important to mention here that, in the case of games with mixed Nash equilibrium, our mathematical analysis shows that the proposed EMAQL algorithm converges to an equilibrium. Although our mathematical analysis does not explicitly show that the proposed EMAQL algorithm converges to a Nash equilibrium in games with mixed Nash equilibrium, our simulation results, as will be presented later, indicate that the proposed EMAQL algorithm does converge to Nash equilibrium.

4.7 The stability of the equilibrium solutions of the proposed EMAQL algorithm

In this subsection, we will study the stability of the equilibrium solutions of the proposed EMAQL algorithm when the game has a pure Nash equilibrium and when it has a mixed Nash equilibrium.

4.7.1 When the game has a pure Nash equilibrium

In pure Nash equilibrium games, when the Q-table of each learning agent converges to a Nash equilibrium policy, the ordinary differential equations of Eq. (33) can be viewed as linear time-varying equations.

Theorem 1 *The first-order linear time-varying system (D'Angelo 1970) characterized by the following equation,*

$$\begin{aligned} &\beta_0(t)\dot{x}(t) + \beta_1(t)x(t) = m(t) \\ &x(t_0) = x_0 \end{aligned} \tag{36}$$

has a transition matrix $\phi(t, t_0)$ described as follows:

$$\phi(t, t_0) = \exp\left[-\int_{t_0}^{t} \frac{\beta_1(\sigma)}{\beta_0(\sigma)} \mathrm{d}(\sigma)\right] \tag{37}$$

where t_0, x_0 and $m(t)$ are the initial time, the initial state and the system's input, respectively.

Each equation in Eq. (33) can be rewritten as follows,

$$\dot{x}(t) + \eta_t x(t) = \eta_t u(t)$$

or,

$$\dot{x}(t) + \eta_t x(t) = m(t) \tag{38}$$

From Eqs. (36) and (38), we have,

$$\frac{\beta_1(t)}{\beta_0(t)} = \eta_t$$

Without loss of generality, let $t_0 = 0$ and $\eta_t = \frac{1}{c_1 + c_2 t}$, where c_1 and c_2 are constants. Thus,

$$
\begin{aligned}
\phi(t,0) &= \exp\left[-\int_0^t \frac{\beta_1(\sigma)}{\beta_0(\sigma)}\mathrm{d}(\sigma)\right] \\
&= \exp\left[-\int_0^t \eta_\sigma \mathrm{d}(\sigma)\right] \\
&= \exp\left[-\int_0^t \frac{1}{c_1+c_2\sigma}\mathrm{d}\sigma\right] \\
&= \exp\left[-\frac{1}{c_2}\ln|c_1+c_2\sigma|_0^t\right] \\
&= \exp\left[-\frac{1}{c_2}\left(\ln|c_1+c_2t|-\ln|c_1|\right)\right] \\
&= \exp\left[-\frac{1}{c_2}\left(\ln|c_1+c_2t|+\ln\left|\frac{1}{c_1}\right|\right)\right] \\
&= \exp\left[-\frac{1}{c_2}\ln\left|\frac{c_1+c_2t}{c_1}\right|\right] \\
&= \exp\left[\ln\left|\frac{c_1+c_2t}{c_1}\right|^{-\frac{1}{c_2}}\right] \\
&= \exp\left[\ln\left|\frac{c_1}{c_1+c_2t}\right|^{\frac{1}{c_2}}\right]
\end{aligned}
$$

Thus,

$$\phi(t,0) = \left(\frac{c_1}{c_1+c_2t}\right)^{\frac{1}{c_2}} \tag{39}$$

Therefore, from Eq. (39), the transition matrix of the diagonal matrix of the linear time-varying system of Eq. (32) is given as follows,

$$\phi(t,0) = \begin{bmatrix} \phi_1(t,0) & 0 & 0 & 0 \\ 0 & \phi_1(t,0) & 0 & 0 \\ 0 & 0 & \phi_2(t,0) & 0 \\ 0 & 0 & 0 & \phi_2(t,0) \end{bmatrix} \tag{40}$$

where $\phi_1(t,0) = \left(\frac{a_1}{a_1+a_2t}\right)^{\frac{1}{a_2}}$ and $\phi_2(t,0) = \left(\frac{b_1}{b_1+b_2t}\right)^{\frac{1}{b_2}}$.

Theorem 2 *The equilibrium solution of the linear time-varying system (DeCarlo 1989; D'Angelo 1970) given by the following equation*

$$\dot{x}(t) = A(t)x(t) + B(t)m(t) \tag{41}$$

is (globally) uniformly asymptotically stable if and only if

$$\sup_{t_0\geq 0}\ \sup_{t\geq t_0}\|\phi(t,t_0)\|_i = m_0 < \infty \tag{42}$$

$$\|\phi(t+t_0,t_0)\|_i \to 0 \quad as \quad t\to\infty \tag{43}$$

where the notation $\|.\|_i$ refers to some weighted maximum norm.

Hence, the pure Nash equilibrium of the linear time-varying system of Eq. (33) is (globally) uniformly asymptotically stable as $\sup_{t\geq 0}\|\phi(t,0)\|_i < \infty$ and $\lim_{t\to\infty}\|\phi(t,0)\|_i \to 0$.

4.7.2 When the game has a mixed Nash equilibrium

In mixed Nash equilibrium games, we view the term $\eta_t u$ as an input, $m(t)$, to the differential equations of Eq. (33). Because the learning rate η_t decays as time goes on, the input, $m(t)$, to the differential equations of Eq. (33) becomes very small and continuous as well. At that point, the differential equations of Eq. (33) become linear time-varying differential equations. Hence, the stability analysis performed in case of pure Nash equilibrium games can be applied to these differential equations too.

5 Simulation and results

We have evaluated the proposed EMAQL, the WoLF-PHC (Bowling and Veloso 2002), the GIGA-WoLF (Bowling 2005), the WPL (Abdallah and Lesser 2008), and the PGA-APP (Zhang and Lesser 2010) algorithms on a variety of matrix and stochastic games. Given that both the WoLF-PHC and the GIGA-WoLF algorithms give nearly identical results over the same games (Bowling 2005), we choose to only show the proposed EMAQL, the PGA-APP, the WPL and the WoLF-PHC algorithms. The results of applying the WoLF-PHC, the WPL, the PGA-APP and the proposed EMAQL algorithms to different matrix and stochastic games are presented in this section. A comparison among the four algorithms in terms of the convergence to Nash equilibrium is provided. The learning rates of each algorithm are carefully chosen based on trial and error basis for a number of combinations of these learning rates so that each algorithm achieves the best performance.

5.1 Matrix games

The proposed EMAQL, the PGA-APP, the WPL and the WoLF-PHC algorithms are applied to the matrix games depicted in Fig. 1. Figure 8 shows the probabilities of selecting the second actions by both players in the dilemma game. The proposed EMAQL, the PGA-APP, the WPL, and the WoLF-PHC algorithms are shown. In this game, the parameters of the proposed EMAQL algorithm are set as follows, $\eta_w = \frac{1}{100+i/25}$, $\eta_l^{WoLF} = 2\eta_w$, $\eta_l^{WoLS} = 0.001\eta_w$, $\zeta = 0.25$, and $\theta = \frac{0.1}{1+0.0001i}$ with an exploration rate $\epsilon = 0.05$. The parameters of the WoLF-PHC algorithm are selected as those of the proposed EMAQL algorithm with $\delta_w = \frac{1}{100+i}$ and $\delta_l = 2\delta_w$. In the PGA-APP algorithm, the parameter ζ is set as $\zeta = 0$, and the parameter γ is set as $\gamma = 0.5$. In the WPL algorithm, the parameters are set as follows, $\eta = 0.05$, $\zeta = 0$, and $\theta = 0.01$. As stated in Sect. 2, the dilemma game has a pure Nash equilibrium strategy that executes the second action of each player with a probability of one. As can be seen in Fig. 8, when learning with all different algorithms, both players adopt their Nash equilibrium strategy as the probabilities of selecting the second actions by both players converge to one after some time. The probabilities of selecting actions by both players in Fig. 8a go through the scenarios we stated in Sect. 4.3 until a Nash equilibrium strategy is finally adopted by both players. Figure 8a confirms our claim in Sect. 4.3 that the pure Nash equilibrium is reached through Scenario 1 when the selected action of each player has a probability distribution of one, and at the same time this selected action is the same as the greedy action calculated from the Q-table of that player.

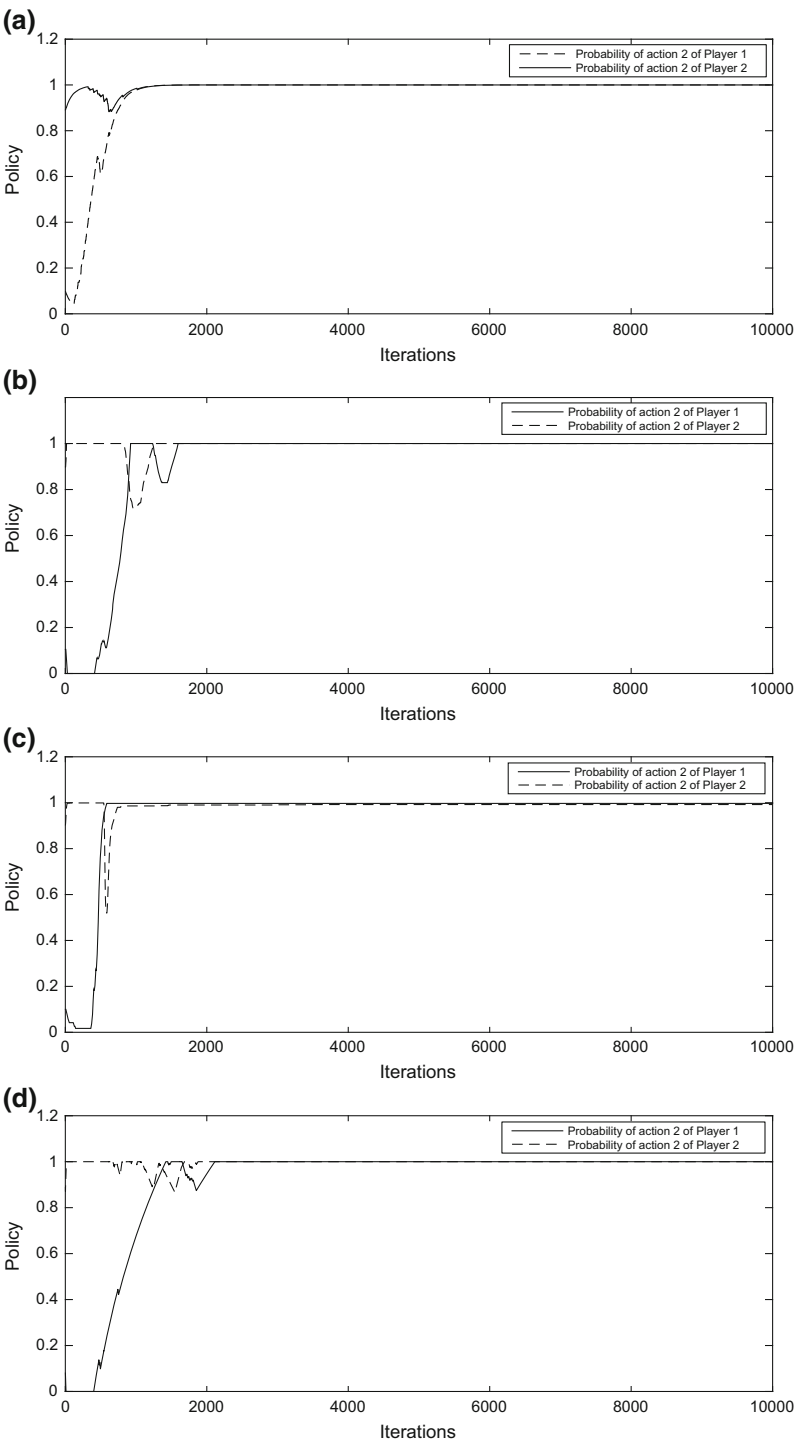

Fig. 8 Probability of choosing the second action for both players in the dilemma game. The proposed EMAQL (**a**), the PGA-APP (**b**), the WPL (**c**), and the WoLF-PHC (**d**) algorithms are shown

Figure 9 shows the probability of choosing Player 1's actions in the shapley's game when learning with the proposed EMAQL, the PGA-APP, the WPL, and the WoLF-PHC algorithms. In this game, the parameters of the proposed EMAQL algorithm are set as follows, $\eta_w = \frac{1}{100+i}$, $\eta_l^{WoLF} = 2\eta_w$, $\eta_l^{WoLS} = 0.001\eta_w$, $\zeta = 0.95$, and $\theta = \frac{2}{10+0.0001i}$ with an exploration rate $\epsilon = 0.05$. The learning rates of the WoLF-PHC algorithm are set as follows, $\delta_w = \frac{1}{100+i/10}$ and $\delta_l = 2\delta w$. The parameters of the PGA-APP algorithm are set as follows, $\eta = \frac{1}{100+i/50}$ and $\gamma = 3$. On the other hand, the parameters of the WPL algorithm are set as follows, $\eta = \frac{1}{100+i/500}$, $\zeta = 0$, and $\theta = 0.95$. As stated in Sect. 2, the shapley's game has one mixed Nash equilibrium strategy that executes each action of each player with a probability of $\frac{1}{3}$. As can be seen in Fig. 9a, when learning with the proposed EMAQL algorithm, both players adopt their Nash equilibrium strategy as the probabilities of selecting each action of Player 1 converge to $\frac{1}{3}$ after some time. Figure 9a shows that the probabilities of selecting actions by Player 1 converge to their mixed Nash equilibrium when Condition 2 we stated in Sect. 4.5 is satisfied. Figure 9b, c show that the PGA-APP and the WPL algorithms succeed to learn the Nash equilibrium of the shapley's game. Figure 9d, on the other hand, shows that the WoLF-PHC algorithm fails to learn the Nash equilibrium of the shapley's game.

Figure 10 shows the probability of choosing the first action for both players in the biased game while learning with the proposed EMAQL, the PGA-APP, the WPL, and the WoLF-PHC algorithms. As stated in Sect. 2, the biased game has a mixed Nash equilibrium strategy with probabilities not uniform across actions, (0.15, 0.85) and (0.85, 0.15). In this game, the parameters of the proposed EMAQL algorithm are set as follows: $\eta_w = \frac{1}{10+i/5}$, $\eta_l^{WoLF} = 2\eta_w$, $\eta_l^{WoLS} = 0.001\eta_w$, $\zeta = 0.95$, and $\theta = \frac{1}{10+0.0001i}$ with an exploration rate $\epsilon = 0.05$. The WoLF-PHC algorithm's learning rates are set as $\delta_w = \frac{1}{10+i}$ and $\delta_l = 2\delta w$. In the PGA-APP algorithm, the values of ζ and γ are set as follows: $\zeta = 0$ and $\gamma = 3$. In the WPL algorithm, the parameters are set as follows, $\zeta = 0$, $\theta = 0.01$ and $\eta = 0.05$. Figure 10a shows that the proposed EMAQL algorithm succeeds to converge to a Nash equilibrium in the biased game. The players' strategies in this figure go through the scenarios stated in Sects. 4.3 and 4.4 until a Nash equilibrium is adopted. Figure 10b shows that the PGA-APP algorithm fails to learn the Nash equilibrium of the biased game. Figure 10c, d, on the other hand, show that the WPL and the WoLF-PHC algorithms converge to values that are close to the Nash equilibrium of the biased game. Figures 8, 9 and 10 show that the proposed EMAQL algorithm outperforms the state-of-the-art (PGA-APP, WPL, and WoLF-PHC) algorithms in terms of the convergence to Nash equilibrium in the matrix games depicted in Fig. 1.

5.2 Stochastic games

The proposed EMAQL, the WoLF-PHC, the PGA-APP and the WPL algorithms are used to learn the grid games depicted in Fig. 2; grid game 1 and grid game 2. Each game of these grid games has more than one Nash equilibrium; grid game 1 has ten different Nash equilibria, whereas grid game 2 has two different Nash equilibria (Hu and Wellman 2003). We run grid game 1 until the game converges to the Nash equilibrium shown in Fig. 3a. We also run grid game 2 until the game converges to the Nash equilibrium shown in Fig. 3b. In both games, we only show the probabilities of the players' actions when they first move from their initial state. This is to investigate whether or not the players' probabilities of the selected actions will converge to Nash equilibria shown in Fig. 3 corresponding to this stage game (the stage game at the initial state). Therefore, the figures that will be presented in this subsection will only represent the probabilities of players' actions at the initial state.

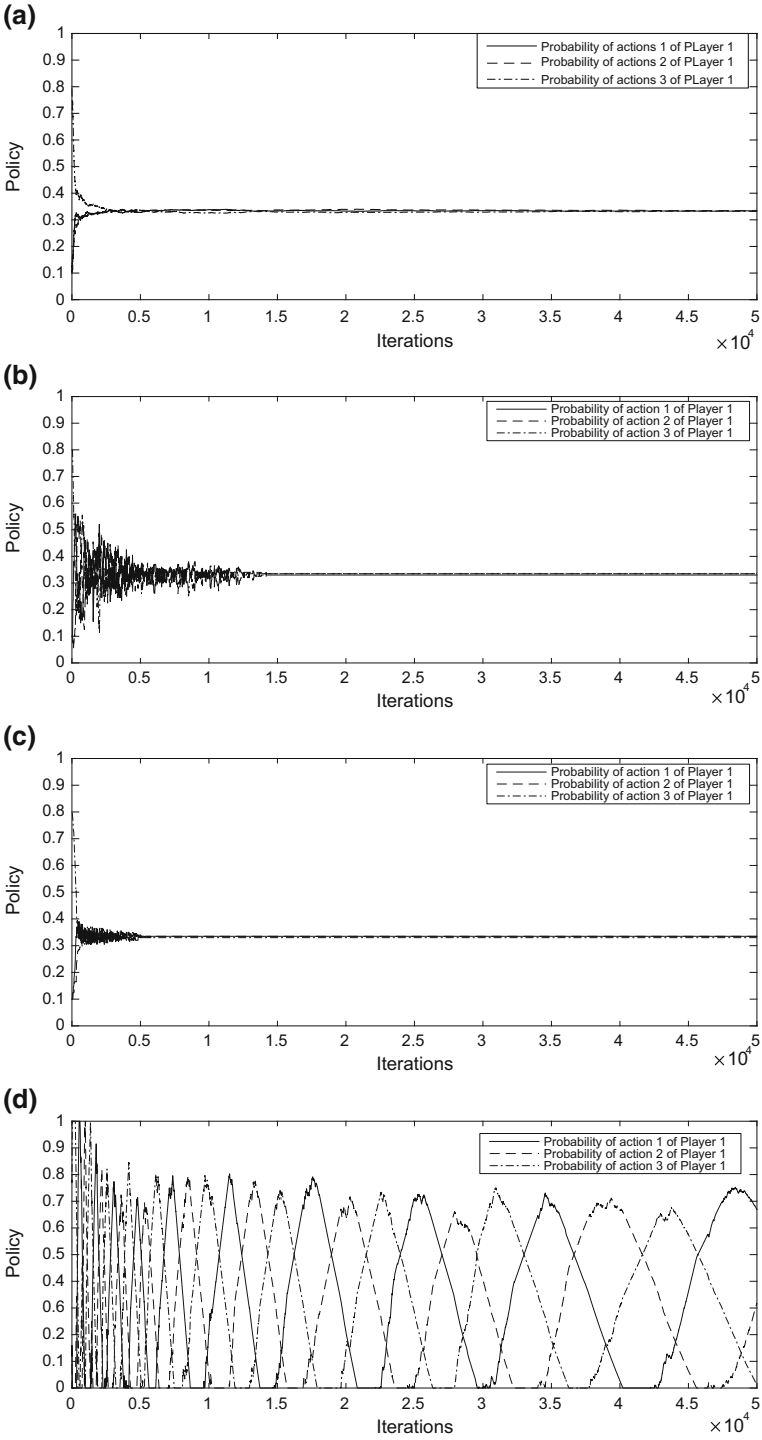

Fig. 9 Probability of choosing Player 1's actions in the shapley's game. The proposed EMAQL (**a**), the PGA-APP (**b**), the WPL (**c**), and the WoLF-PHC (**d**) algorithms are shown

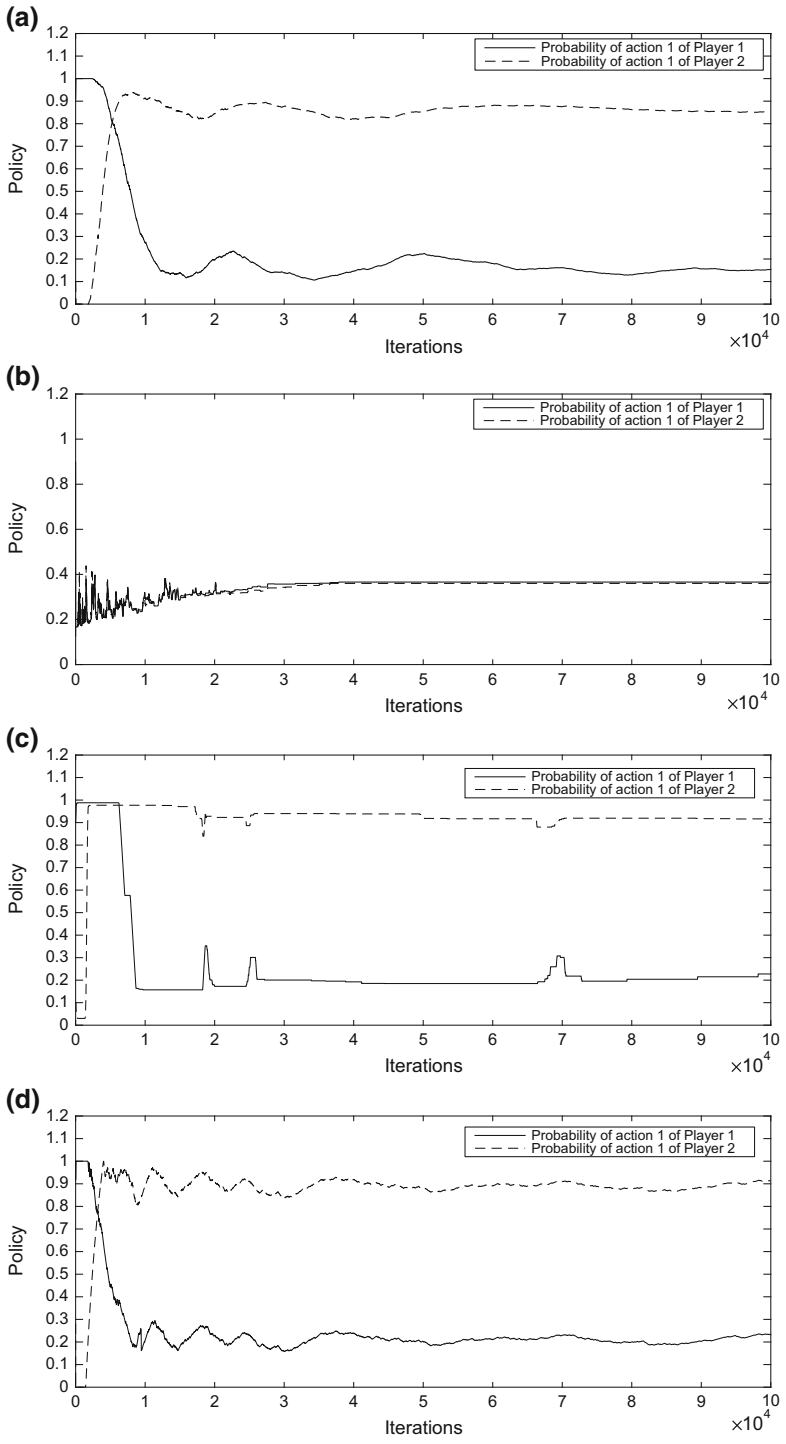

Fig. 10 Probability of choosing the first action for both players in the biased game. The proposed EMAQL (**a**), the PGA-APP (**b**), the WPL (**c**), and the WoLF-PHC (**d**) algorithms are shown

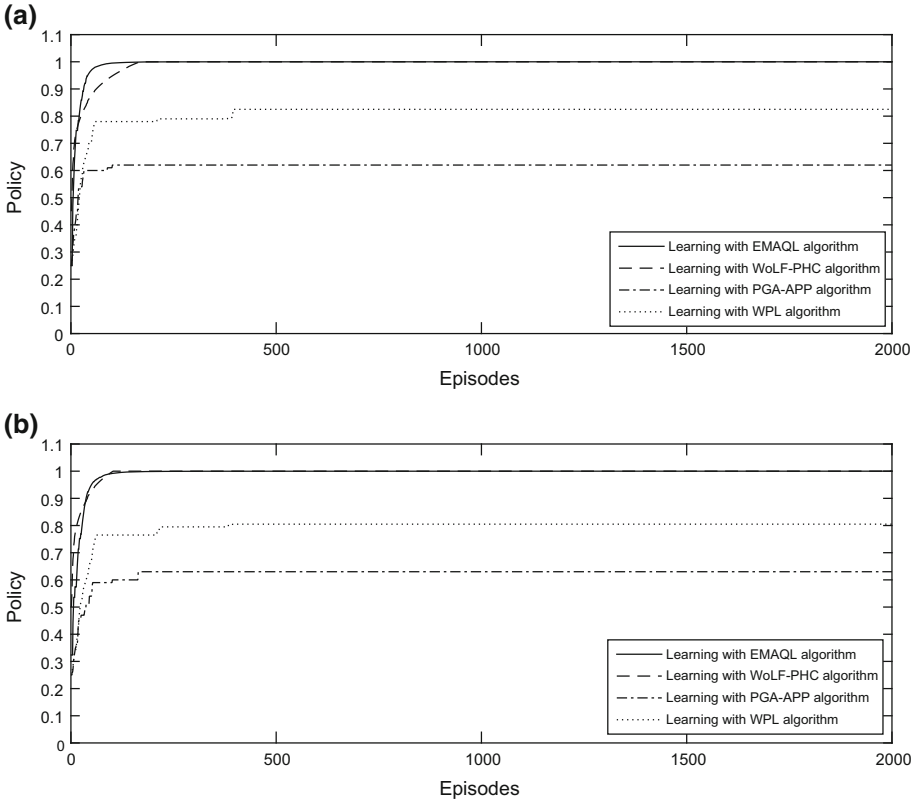

Fig. 11 Grid game 1: **a** the probability of selecting action North by Player 1 when learning with the proposed EMAQL, the WoLF-PHC, the PGA-APP, and the WPL algorithms. **b** The probability of selecting action North by Player 2 when learning with the proposed EMAQL, the WoLF-PHC, the PGA-APP, and the WPL algorithms

5.2.1 Grid game 1

The proposed EMAQL, the WoLF-PHC, the PGA-APP and the WPL algorithms are used to learn grid game 1 depicted in Fig. 2a. Grid game 1 has ten different Nash equilibria (Hu and Wellman 2003). One of these Nash equilibria (optimal paths) is shown in Fig. 3a. Figure 3a shows that the action North is the Nash equilibrium action for both players when they are at their initial state. The learning parameters of the proposed EMAQL algorithm are set as follows, $\eta_w = \frac{1}{10+i/5}$, $\eta_l^{WoLF} = 2\eta_w$, $\eta_l^{WoLS} = 0.001\eta_w$, $\zeta = 0$, and $\theta = \frac{1}{10+0.001i}$ with an exploration rate $\epsilon = 0.05$, where i is the current number of episodes. The parameters of the WoLF-PHC algorithm are set the same as those of the proposed EMAQL algorithm and with $\delta_w = \frac{1}{10+5i}$ and $\delta_l = 2\delta_w$. The values of the parameters of the PGA-APP algorithm are the same as those of the proposed EMAQL algorithm except that $\gamma = 3$, $\theta = \frac{8}{10+0.005i}$, and $\eta = 0.1$. On the other hand, the learning parameters of the WPL algorithm are set as follows, $\theta = \frac{1}{10+0.001i}$ and $\eta = 0.1$.

Figure 11a, b show the probabilities of selecting action North by both players at their initial state when learning with the proposed EMAQL, the WoLF-PHC, the PGA-APP, and the WPL algorithms. Figure 11a, b show that the probabilities of taking action North by

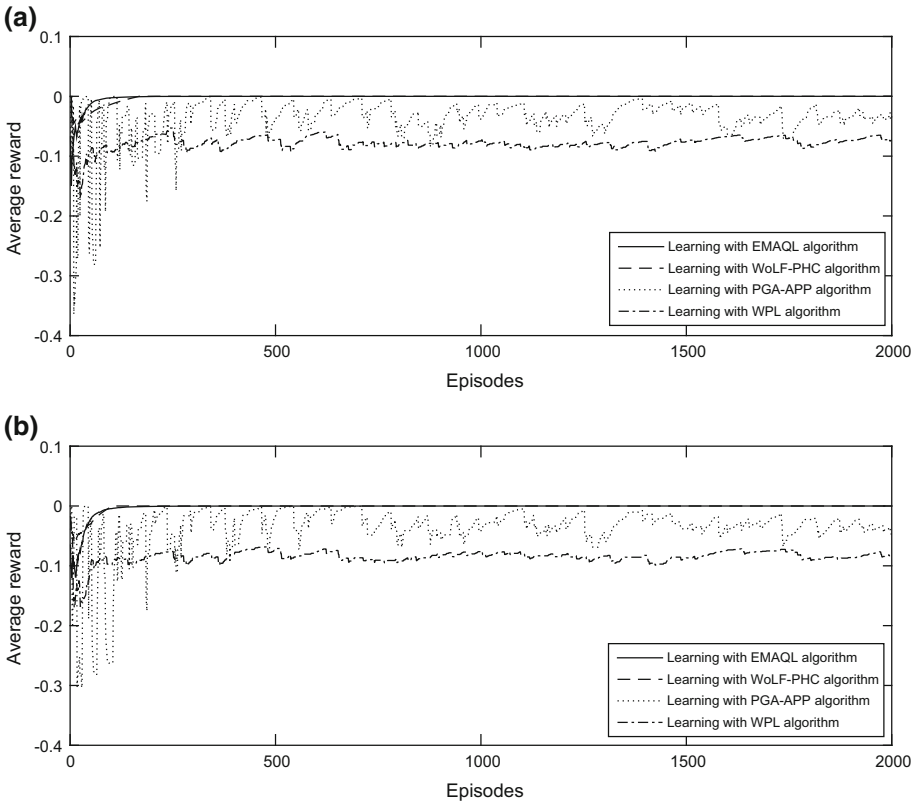

Fig. 12 Grid game 1: **a** the average reward of Player 1 when learning with the proposed EMAQL, the WoLF-PHC, the PGA-APP, and the WPL algorithms. **b** The average reward of Player 2 when learning with the proposed EMAQL, the WoLF-PHC, the PGA-APP, and the WPL algorithms

both players at the initial state converge to the Nash equilibrium (converge to one) when learning with the proposed EMAQL and WoLF-PHC algorithms. However, the PGA-APP and the WPL algorithms fail to make the players' strategies converge to the Nash equilibria. Figure 11 shows that the proposed EMAQL algorithm outperforms the PGA-APP and the WPL algorithms in terms of the convergence to Nash equilibrium in grid game 1.

The maximum reward that each player can get at the initial state if both players adopt the Nash equilibrium shown in Fig. 3a is zero. Thus, the Nash equilibrium reward for both players at the initial state is zero. Figure 12a, b show the average rewards of Player 1 and Player 2, respectively, at the initial state when learning with the proposed EMAQL, the WoLF-PHC, the PGA-APP, and the WPL algorithms. Figure 12a, b show that the average rewards of both players at the initial state converge to the corresponding Nash equilibrium reward when only learning with the proposed EMAQL and the WoLF-PHC algorithms. On the other hand, when learning with the PGA-APP aand the WPL algorithms, the average rewards of both players converge to other values close to the Nash equilibrium reward of both players.

5.2.2 Grid game 2

The proposed EMAQL, the WoLF-PHC, the PGA-APP and the WPL algorithms are also used to learn grid game 2 depicted in Fig. 2b. Grid game 2 has two Nash equilibria (Hu

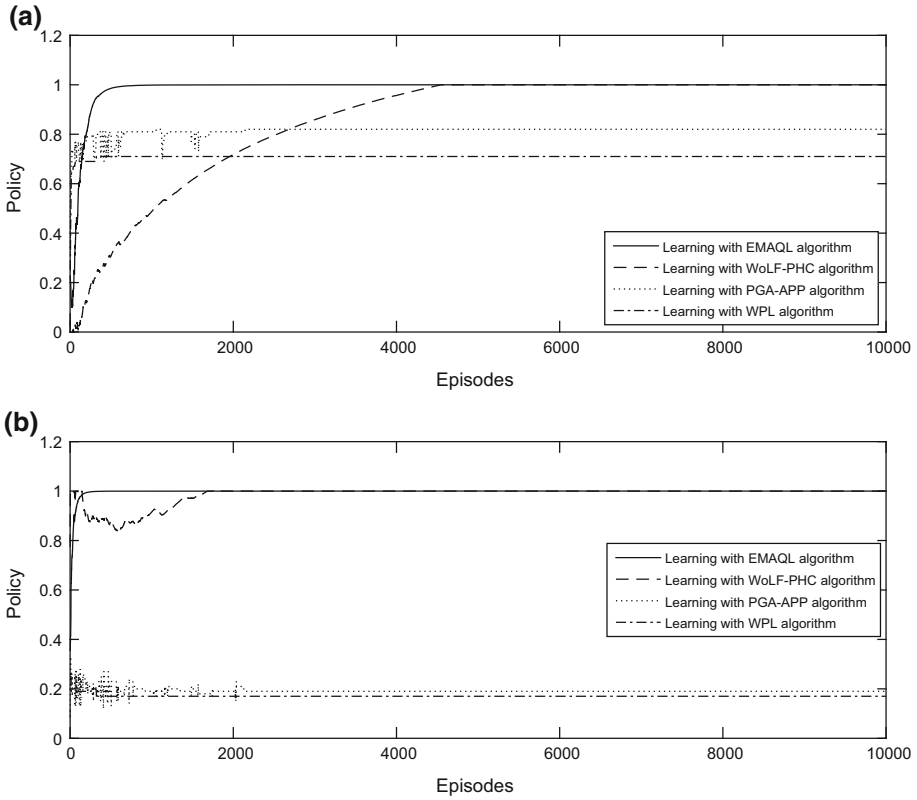

Fig. 13 Grid game 2: **a** the probability of selecting action East by Player 1 when learning with the proposed EMAQL, the WoLF-PHC, the PGA-APP, and the WPL algorithms. **b** The probability of selecting action North by Player 2 when learning with the proposed EMAQL, the WoLF-PHC, the PGA-APP, and the WPL algorithms

and Wellman 2003). Figure 3b shows one of these Nash equilibria. As can be seen from this particular Nash equilibrium, the action East is the Nash equilibrium action for Player 1 at the initial state; whereas the action North is the Nash equilibrium action for Player 2. Thus, for the algorithms to converge to this particular Nash equilibrium at the initial state, the probability of selecting the action East by Player 1 should converge to one. The probability of selecting the action North by Player 2 at the initial state, on the other hand, should also converge to one. The learning parameters of the proposed EMAQL algorithm are set as follows, $\eta_w = \frac{1}{20+i/5}$, $\eta_l^{WoLF} = 2\eta_w$, $\eta_l^{WoLS} = 0.001\eta_w$, $\zeta = 0.25$, and $\theta = 0.8$ with an exploration rate $\epsilon = \frac{2}{10+0.01i}$, where i is the current number of episodes. The parameters of the WoLF-PHC algorithm are set the same as those of the proposed EMAQL algorithm and with $\delta_w = \frac{1}{20+i}$ and $\delta_l = 2\delta_w$. The values of the parameters of the PGA-APP algorithm are the same as those of the proposed EMAQL algorithm except that $\gamma = 3$, $\theta = \frac{8}{10+0.001i}$, $\zeta = 0.1$, and $\eta = 0.1$. On the other hand, the learning parameters of the WPL algorithm are set as follows, $\theta = \frac{1}{10+0.001i}$, $\zeta = 0.1$ and $\eta = 0.1$.

Figure 13a shows the probability of selecting action East by Player 1 at the initial state when learning with the proposed EMAQL, the WoLF-PHC, the PGA-APP, and the WPL

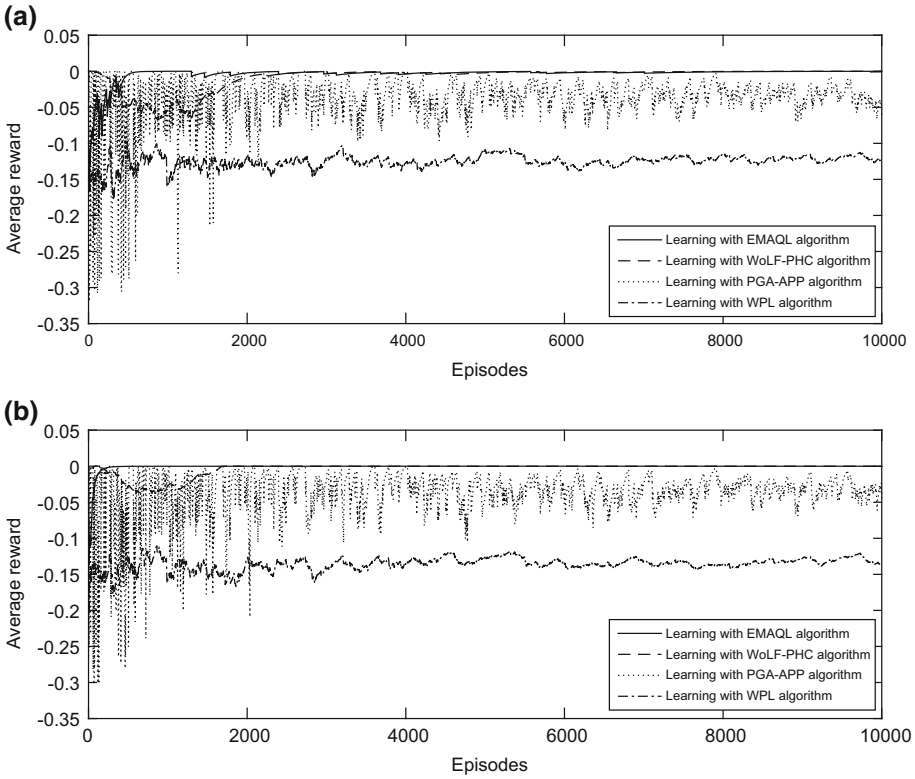

Fig. 14 Grid game 2: **a** the average reward of Player 1 when learning with the proposed EMAQL, the WoLF-PHC, the PGA-APP, and the WPL algorithms. **b** The average reward of Player 2 when learning with the proposed EMAQL, the WoLF-PHC, the PGA-APP, and the WPL algorithms

algorithms. Figure 13a illustrates that the probability of selecting the action East by Player 1 at the initial state successfully converges to one (Nash equilibrium) when learning with the proposed EMAQL and WoLF-PHC algorithms. However, the PGA-APP and the WPL algorithms fail to make Player 1 choose the action East with a probability of one at the initial state. Figure 13b shows the probability of selecting action North by Player 2 at the initial state when learning with the proposed EMAQL, the WoLF-PHC, the PGA-APP, and the WPL algorithms. As can be seen from Fig. 13b, the probability of selecting action North by Player 2 at the initial state successfully converges to one (Nash equilibrium) when Player 2 learns its strategy with the proposed EMAQL and the WoLF-PHC algorithms. The PGA-APP and the WPL algorithms, on the other hand, fail to make Player 2 choose action North with a probability of one at the initial state. Figure 13 shows that the proposed EMAQL algorithm outperforms the PGA-APP and the WPL algorithms in terms of the convergence to Nash equilibrium in grid game 2.

In grid game 2, the Nash equilibrium reward for both players at the initial state is zero. Figure 14a, b show the average rewards of Player 1 and Player 2 at the initial state, respectively, when learning with the proposed EMAQL, the WoLF-PHC, the PGA-APP, and the WPL algorithms. Figure 14a, b show that the average rewards of both players at the initial state converge to the corresponding Nash equilibrium reward when only learning with the proposed

EMAQL and the WoLF-PHC algorithms. On the other hand, when learning with the PGA-APP aand the WPL algorithms, the average rewards of both players converge to other values close to the Nash equilibrium reward of both players at the initial state.

6 Conclusions

New multi-agent policy iteration learning algorithms are proposed in this work; the CLR-EMAQL algorithm and the EMAQL algorithm. The proposed algorithms use the exponential moving average (EMA) estimation technique along with the Q-learning algorithm to update the learning agent's policy. The proposed CLR-EMAQL algorithm uses one constant learning rate (η_c) when updating the learning agent's strategy. The proposed CLR-EMAQL algorithm succeeds to converge to Nash equilibrium only when the game has a pure Nash equilibrium. A theoretical analysis that shows the convergence of the proposed CLR-EMAQL algorithm to Nash equilibrium in games with pure Nash equilibrium is provided in this article. On the other hand, the proposed EMAQL algorithm uses two different decaying learning rates (η_w and η_l) when updating the agent's strategy. The values of these variable learning rates are inversely proportional to the number of iterations (or episodes) and are set based on one of two different mechanisms; the Win-or-Learn-Fast (WoLF) mechanism or the Win-or-Learn-Slow (WoLS) mechanism. The WoLS mechanism is introduced in this article to make the algorithm learn fast when it is winning and learn slowly when it is losing. The proposed EMAQL algorithm uses the rewards received by the learning agent to decide which mechanism (WoLF mechanism or WoLS mechanism) to use for the game being learned. The proposed EMAQL algorithm succeeds to converge to Nash equilibrium in games with pure or mixed Nash equilibrium. A theoretical analysis that shows the convergence of the proposed EMAQL algorithm to Nash equilibrium in games with pure Nash equilibria is provided. In the case of games with mixed Nash equilibrium, our mathematical analysis shows that the proposed EMAQL algorithm converges to an equilibrium. Although our mathematical analysis does not explicitly show that the proposed EMAQL algorithm converges to a Nash equilibrium, our simulation results indicate that the proposed EMAQL algorithm does converge to Nash equilibrium. To verify that our theoretical analysis and our simulation results are consistent, the proposed EMAQL algorithm is applied to a variety of matrix and stochastic games. The results show that the proposed EMAQL algorithm outperforms the PGA-APP, the WPL, and the WoLF-PHC algorithms in terms of the convergence to Nash equilibrium.

References

Abdallah S, Lesser V (2008) A multiagent reinforcement learning algorithm with non-linear dynamics. J Artif Intell Res 33:521–549

Awheda MD, Schwartz HM (2013) Exponential moving average Q-learning algorithm. In: Adaptive dynamic programming and reinforcement learning (ADPRL), 2013 IEEE symposium on, IEEE, pp 31–38. IEEE

Awheda MD, Schwartz HM (2015) The residual gradient FACL algorithm for differential games. In Electrical and computer engineering (CCECE). 2015 IEEE 28th Canadian conference on, IEEE, pp 1006–1011. IEEE

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

Bellman R (1957) Dynamic programming. Princeton University Press, Princeton

Bowling M (2005) Convergence and no-regret in multiagent learning. Adv Neural Inf Process Syst 17:209–216

Bowling M, Veloso M (2001a) Convergence of gradient dynamics with a variable learning rate. In: ICML, pp 27–34

Bowling M, Veloso M (2001b) Rational and convergent learning in stochastic games. In: International joint conference on artificial intelligence, vol. 17. Lawrence Erlbaum Associates Ltd, pp 1021–1026
Bowling M, Veloso M (2002) Multiagent learning using a variable learning rate. Artif Intell 136(2):215–250
Burkov A, Chaib-draa B (2009) Effective learning in the presence of adaptive counterparts. J Algorithms 64(4):127–138
Busoniu L, Babuska R, De Schutter B (2006) Multi-agent reinforcement learning: A survey. In: Control, automation, robotics and vision, 2006. ICARCV'06. 9th international conference on, IEEE, pp 1–6. IEEE
Busoniu L, Babuska R, De Schutter B (2008) A comprehensive survey of multiagent reinforcement learning. Syst Man Cybern Part C: Appl Rev, IEEE Trans 38(2):156–172
Claus C, Boutilier C (1998) The dynamics of reinforcement learning in cooperative multiagent systems. In: AAAI/IAAI, pp 746–752
Conitzer V, Sandholm T (2007) Awesome: A general multiagent learning algorithm that converges in self-play and learns a best response against stationary opponents. Mach Learn 67(1–2):23–43
Dai X, Li C-K, Rad AB (2005) An approach to tune fuzzy controllers based on reinforcement learning for autonomous vehicle control. Intell Transp Syst, IEEE Trans 6(3):285–293
D'Angelo H (1970) Linear time-varying systems: analysis and synthesis. Allyn & Bacon, Newton
DeCarlo RA (1989) Linear systems: a state variable approach with numerical implementation. Prentice-Hall Inc, Upper Saddle River
Dixon W (2014) Optimal adaptive control and differential games by reinforcement learning principles. J Guid Control Dyn 37(3):1048–1049
Fulda N, Ventura D (2007) Predicting and preventing coordination problems in cooperative Q-learning systems. In: IJCAI, vol. 2007, pp 780–785
Gutnisky DA, Zanutto BS (2004) Learning obstacle avoidance with an operant behavior model. Artif Life 10(1):65–81
Hinojosa W, Nefti S, Kaymak U (2011) Systems control with generalized probabilistic fuzzy-reinforcement learning. Fuzzy Syst, IEEE Trans 19(1):51–64
Howard RA (1960) Dynamic programming and markov processes. MIT Press, Cambridge
Hu J, Wellman MP (2003) Nash Q-learning for general-sum stochastic games. J Mach Learn Res 4:1039–1069
Hu J, Wellman MP, et al (1998) Multiagent reinforcement learning: theoretical framework and an algorithm. In: ICML, vol. 98, Citeseer, pp 242–250
Kaelbling LP, Littman ML, Moore AW (1996) Reinforcement learning: a survey. J Artif Intell Res 4:237–285
Kondo T, Ito K (2004) A reinforcement learning with evolutionary state recruitment strategy for autonomous mobile robots control. Robot Auton Syst 46(2):111–124
Luo B, Wu H-N, Li H-X (2014a) Data-based suboptimal neuro-control design with reinforcement learning for dissipative spatially distributed processes. Ind Eng Chem Res 53(19):8106–8119
Luo B, Wu H-N, Huang T, Liu D (2014b) Data-based approximate policy iteration for nonlinear continuous-time optimal control design. Automatica 50(12):3281–3290
Luo B, Wu H-N, Huang T (2015a) Off-policy reinforcement learning for H_∞ control design. Cybern, IEEE Trans 45(1):65–76
Luo B, Wu H-N, Li H-X (2015b) Adaptive optimal control of highly dissipative nonlinear spatially distributed processes with neuro-dynamic programming. Neural Netw Learn Syst, IEEE Trans 26(4):684–696
Luo B, Huang T, Wu H-N, Yang X (2015c) Data-driven H_∞ control for nonlinear distributed parameter systems. Neural Netw Learn Syst, IEEE Trans 26(11):2949–2961
Luo B, Wu H-N, Huang T, Liu D (2015d) Reinforcement learning solution for HJB equation arising in constrained optimal control problem. Neural Netw 71:150–158
Modares H, Lewis FL, Naghibi-Sistani M-B (2014) Integral reinforcement learning and experience replay for adaptive optimal control of partially-unknown constrained-input continuous-time systems. Automatica 50(1):193–202
Rodríguez M, Iglesias R, Regueiro CV, Correa J, Barro S (2007) Autonomous and fast robot learning through motivation. Robot Auton Syst 55(9):735–740
Schwartz HM (2014) Multi-Agent Machine Learning: A Reinforcement Approach. Wiley, New York
Sen S, Sekaran M, Hale J (1994) Learning to coordinate without sharing information. In: AAAI, pp 426–431
Singh S, Kearns M, Mansour Y (2000) Nash convergence of gradient dynamics in general-sum games. In: Proceedings of the sixteenth conference on Uncertainty in artificial intelligence, Morgan Kaufmann Publishers Inc., pp 541–548
Smart WD, Kaelbling LP (2002) Effective reinforcement learning for mobile robots. In: Robotics and automation. Proceedings. ICRA'02. IEEE international conference on, vol. 4, IEEE, 2002, pp. 3404–3410. IEEE
Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. The MIT Press, Cambridge

Tan M (1993) Multi-agent reinforcement learning: Independent vs. cooperative agents. In: Proceedings of the tenth international conference on machine learning, pp 330–337
Tesauro G (2004) Extending Q-learning to general adaptive multi-agent systems. In: Advances in neural information processing systems, vol. 16. MIT press, pp 871–878
Thathachar MA, Sastry PS (2011) Networks of learning automata: techniques for online stochastic optimization. Springer, Boston
Watkins CJ, Dayan P (1992) Q-learning. Mach Learn 8(3–4):279–292
Watkins CJCH (1989) Learning from delayed rewards, Ph.D. thesis, University of Cambridge England
Weiss G (1999) Multiagent systems: a modern approach to distributed artificial intelligence. MIT Press
Wu H-N, Luo B (2012) Neural network based online simultaneous policy update algorithm for solving the HJI equation in nonlinear control. Neural Netw Learn Syst, IEEE Trans 23(12):1884–1895
Ye C, Yung NH, Wang D (2003) A fuzzy controller with supervised learning assisted reinforcement learning algorithm for obstacle avoidance. Syst Man Cybern Part B: Cybern, IEEE Trans 33(1):17–27
Zhang C, Lesser VR (2010) Multi-agent learning with policy prediction. In: AAAI