PAC Reinforcement Learning Algorithm for General-Sum Markov Games

Ashkan Zehfroosh, *Student Member, IEEE,* and Herbert G. Tanner, *Senior Member, IEEE*

***Abstract*— This paper presents a theoretical framework for probably approximately correct (PAC) multi-agent reinforcement learning (MARL) algorithms for Markov games. Using the idea of delayed Q-learning, the paper extends the well-known Nash Q-learning algorithm to build a new PAC MARL algorithm for general-sum Markov games. In addition to guiding the design of a provably PAC MARL algorithm, the framework enables checking whether an arbitrary MARL algorithm is PAC. Comparative numerical results demonstrate the algorithm's performance and robustness.**

***Index Terms*— Multi-agent system, Reinforcement Learning, Probability approximately correct, Markov Game, Nash Equilibrium.**

I. Introduction

DECISION-making and planning for autonomous agents in an unknown environment is often done through a reinforcement learning (RL) approach. Sometimes systems involve more than one agent, in which case the problem fall into a multi-agent reinforcement learning (MARL) domain. In MARL one is concerned with sequential decision-making for multiple autonomous agents that operate in an unknown environment, in which the behavior of all agents jointly affects the evolution of the system. Such problems are typically approached from a game-theoretic perspective, with each agent trying to maximize their own reward function. Unlike situations where agent-environment interaction dynamics are modeled as an Markov decision process (MDP), in MARL games the environment of each agent is non-stationary, adding a layer of complexity to the problem. Indeed, existing MARL algorithms that can provide a probably approximately correct (PAC) bound on the sample complexity involved in reaching a near-optimal behavior, are very rare [1]. The paper contributes to bridging this gap with a general mathematical PAC characterization of a MARL algorithm, and a new PAC MARL algorithm for general-sum Markov games.

A MARL algorithm can either be [1] fully cooperative, fully competitive, or mixed setting. *Fully cooperative* MARL handles cases where all agents collaborate to achieve some shared goal, sharing the same reward function–such a model is usually referred to as a multi-agent MDP (MMDP). As a result of shared reward function, Q-function is also identical for all agents. Hence, an early and straightforward MARL algorithm for such setting is to perform the standard Q-learning update [2]. While Littman [3] established the convergence of such algorithm to the optimal Q-values, the convergence does not necessarily imply equilibrium policy since each agent might choose a distinct equilibrium when multiple ones exist. The first MARL algorithm that provably converges to the equilibrium policy is Optimal Adaptive Learning optimal adaptive learning (OAL) [4]. Recently, researchers tried to address the scalability issue that arises when the system involves big number of agents, through value function factorization [5], [6]. Policy-based method is also developed for cooperative MARL that is provably convergent [7].

For *fully competitive* MARL, most of the literature concentrated on two-agents game with zero-sum reward functions. The reason is that, there is a huge computational barrier between solving two-player zero-sum game and multi-player one. In fact, even the simplest three-player zero-sum game is known to be PPAD-complete [8]. By defining minimax value function, MARL in two-player zero-sum games reduces into single agent case where each agent tries to maximize the worst case reward. More interestingly, optimal point of the minimax value function is the unique fixed point of a Bellman operator, which bring the strong theoretical foundation of dynamic programing in use for competitive MARL. Minimax Q-learning [9] extends the well-known Q-learning algorithm to zero-sum Markov games, and is provably convergent to minimax Q-values and constitutes the Nash equilibrium policy. For fully competitive setting, several PAC algorithms have also been developed. For instance, [10]–[12] have studied zero-sum turn-based stochastic games and with the assumption of the availability of a generative model of the game, they have proposed MARL algorithms that achieve near-optimal behavior with finite sample. Online PAC MARL algorithm has also been developed for average-reward zero-sum stochastic games using the principle of optimism in the face of uncertainty [13].

In *mixed setting*, finding a Nash equilibrium, as a general solution for mixed setting MARL, is actually PPAD-complete even for a simple two-player game [14]. Moreover, value-iteration based methods might fail in general to find stationary Nash or even correlated equilibrium for mixed setting [15]. Existing MARL algorithms for mixed

Zehfroosh and Tanner are with the Department of Mechanical Engineering, University of Delaware, Newark, DE 19716 USA. e-mail: {ashkanz,btanner}@udel.edu.

This work was supported in part by NIH under R01HD87133 and DTRA under HDTRA1-16-1-0039.

setting (e.g., Nash Q-learning [16]) are thus guaranteed to converge under quite strong assumptions. Given that finding Nash equilibria is computationally challenging [17], [18], correlated Q-learning [19] circumvents Nash equilibrium computation by computing instead (via linear programming) correlated equilibria for each stage game. Alternative methods include Bellman residue minimization for approximation of Nash equilibria [20], and modifications to Nash Q-learning where actions of agents are approximated by empirical averages [21].

The contribution of this paper is the generalization of the PAC MDP algorithm theorem [22] to PAC Markov games. This serves as the basis for the introduction of a novel PAC MARL algorithm for Markov games, based on a new extension of delayed Q-learning [23] into games, which we refer to as *Delayed Nash Q-learning*. This new PAC MARL algorithm converges based on finite samples under the same conditions that Nash Q-learning [16] imposes.

II. Technical Preliminaries

A two-player finite Markov Game M is a tuple $\{S, A^1, A^2, R^1, R^2, T, \gamma\}$ with elements

S	set of *states*
A^i	set of *actions* for player i
$R^i : S \times A^1 \times A^2 \to [0,1]$	*reward* function for player i
$T : S \times A^1 \times A^2 \times S \to [0,1]$	*transition probabilities*
$\gamma \in [0,1)$	*discount factor.*

A *stationary policy* π^i for player i is a mapping $\pi^i : S \times A^i \to [0,1]$ that selects an action a^i to be executed at state s with its corresponding probability $\pi^i(s, a^i)$. A *non-stationary policy* $\mathcal{A}^i$ can be defined as a tuple of stationary policies $\mathcal{A}^i = (\pi^i_1, \pi^i_2, \ldots)$, meaning that at step k in the game, agent i executes the policy π^i_k. In a Markov Game, each player tries to maximize its own discounted sum of rewards.

Definition 1. *In a two-player Markov Game M where players are following policies π^1 and π^2, with action a^i drawn according to policy π^i denoted $a^i_{\sim\pi^i}$, the* value of state s *for player i is defined as:*

$$v^i_M(s, \pi^1, \pi^2) := \mathbb{E}_M \left\{ \sum_{t=0}^{\infty} \gamma^t R^i\left(s_t, a^1_{\sim\pi^1}, a^2_{\sim\pi^2}\right) \,\middle|\, s_0 = s \right\}$$

Subscript M may be dropped when it clear from context.

Note that by restricting the rewards in $[0,1]$, the maximum possible value of any state is bounded by $v_{\max} = \frac{1}{1-\gamma}$. A Nash equilibrium is now a joint strategy in which the policy of each player is the best response to others.

Definition 2. *In a two-player Markov Game M, a* Nash equilibrium point *is a tuple of policies (π^1_*, π^2_*) such that for all state $s \in S$ and for all players $i = 1, 2$*

$$\begin{cases} v^1_M(s, \pi^1_*, \pi^2_*) \geq v^1_M(s, \pi^1, \pi^2_*) \\ v^2_M(s, \pi^1_*, \pi^2_*) \geq v^2_M(s, \pi^1_*, \pi^2) \end{cases} \quad \forall \pi^i \in \Pi^i \qquad (1)$$

where Π^i is the set of all available policies for player i. In addition, (π^1_*, π^2_*) is called a global optimal *Nash equilibrium point if*

$$\begin{cases} v^1_M(s, \pi^1_*, \pi^2_*) \geq v^1_M(s, \pi^1, \pi^2) \\ v^2_M(s, \pi^1_*, \pi^2_*) \geq v^2_M(s, \pi^1, \pi^2) \end{cases} \quad \forall \pi^i \in \Pi^i$$

Moreover, (π^1_*, π^2_*) is saddle point *Nash equilibrium, if in addition to* (1)*, we have*

$$\begin{cases} v^1_M(s, \pi^1_*, \pi^2_*) \leq v^1_M(s, \pi^1_*, \pi^2) \\ v^2_M(s, \pi^1_*, \pi^2_*) \leq v^2_M(s, \pi^1, \pi^2_*) \end{cases} \quad \forall \pi^i \in \Pi^i$$

Every Markov Game possesses at least one Nash equilibrium point in stationary policies [24].

To adapt Q-learning into a multi-agent context, the first step is to recognize the need for considering joint actions rather than individual actions. For a two-agent system, for example, the Q-function is now written $Q(s, a^1, a^2)$, where the pair (a^1, a^2) is referred to as the *action profile*. With these extensions of the standard concept of Q-function, and with a Nash equilibrium as the desired solution, one can define the Nash Q-value [16]:

Definition 3. *In a two-player Markov Game M with Nash equilibrium policy (π^1_*, π^2_*), for any state-action profile (s, a^1, a^2), the* Nash Q-value $Q^i_*(s, a^1, a^2)$ *for agent i is the expected sum of discounted rewards when both players follow the Nash equilibrium policy from next period on:*

$$Q^i_*(s, a^1, a^2) = R^i(s, a^1, a^2) + \gamma \sum_{s' \in S} T(s, a^1, a^2, s') v^i_M(s, \pi^1_*, \pi^2_*) \qquad (2)$$

Using this Definition, for all players i and state s, one can equivalently write [16, Lemma 10]

$$v^i_M(s, \pi^1_*, \pi^2_*) = \sum_{(a^1, a^2)} \pi^1_*(s, a^1) \cdot \pi^2_*(s, a^2) \cdot Q^i_*(s, a^1, a^2)$$

The objective now is to design an RL algorithm that identifies the Nash equilibrium policy in a Markov Game when the actual transition probabilities and/or reward function are *not known*. The procedure for finding this policy naturally involves exploration of the Markov Game model. An RL algorithm usually maintains a table of state-action profile value estimates $Q^i(s, a^1, a^2)$ for all players, which are updated based on the exploration data. During the execution of this RL algorithm, the currently stored value for state-action profile (s, a^1, a^2) for agent i at time step t will be denoted $Q^i_t(s, a^1, a^2)$. Assume that all players are utilizing the same RL algorithm and have their own estimation of Q-values for all other players. Consequently, by considering each state of the Markov Game as a stage game (one-shot game) with $Q^i_t(s, :)$

(denoting row s of the Q matrix) as the reward of each possible action profile for each player i, define the current value of state s for player i as

$$v_t^i(s) = \text{Nash}^i\big(Q_t^1(s,:), Q_t^2(s,:)\big)$$

where the Nash^i operator calculates the Nash Q-value for agent i in the stage game with rewards $(Q_t^1(s,:), Q_t^2(s,:))$. We refer to a multi-agent RL algorithm in a game-theoretic context as *Nash-greedy* if, at any time step t and state s, it instructs players to execute policies associated with some Nash equilibrium. The policy that is in force at time step t is denoted π_t, and the greedy policy for player i would be denoted in general as

$$\pi_t^i(s,a^i) = \text{argNash}_{a^i}\big(Q_t^1(s,:), Q_t^2(s,:)\big) \tag{3}$$

III. Characterization of a PAC Multi-agent RL Algorithm

This section formally frames the characteristics of the desired multi-agent PAC RL algorithm in the form of a theorem. The necessary technical stage is set through the following definitions and lemmas.

Definition 4. *Consider a two-player Markov Game* $M = \{S, A^1, A^2, R^1, R^2, T, \gamma\}$, *which at time step* t *has Nash Q-value estimates* $Q_t^i(s,a^1,a^2)$ *for agent* i. *Let* $K_t \subseteq S \times A^1 \times A^2$ *be a set of state-action profiles which are labeled* known. *The* known state-action Markov Game

$$M_{K_t} = \Big\{ S \cup \big\{ z_{(s,a^1,a^2)} \mid (s,a^1,a^2) \notin K_t \big\}, \\ A^1, A^2, R^1_{K_t}, R^2_{K_t}, T_{K_t}, \gamma \Big\}$$

is an Markov Game derived from M *and* K_t *by defining new states* z_{s,a^1,a^2} *for each unknown state-action profile* $(s,a^1,a^2) \notin K_t$, *with self-loops for all actions, i.e.:* $T_{K_t}(z_{(s,a^1,a^2)}, \ldots, z_{(s,a^1,a^2)}) = 1$. *For all* $(s,a^1,a^2) \in K_t$ *and all players, it is* $R^i_{K_t}(s,a^1,a^2) = R^i(s,a^1,a^2)$ *and* $T_{K_t}(s,a^1,a^2,\cdot) = T(s,a^1,a^2,\cdot)$. *When an unknown state-action profile* $(s,a^1,a^2) \notin K_t$ *is experienced, the reward* $R^i_{K_t}(s,a^1,a^2) = Q_t^i(s,a^1,a^2)(1-\gamma)$ *is accumulated for player* i *and the model jumps to* $z_{(s,a^1,a^2)}$ *with* $T_{K_t}(s,a^1,a^2,z_{(s,a^1,a^2)}) = 1$; *subsequently,* $R^i_{K_t}(z_{(s,a^1,a^2)},\cdot) = Q_t^i(s,a^1,a^2)(1-\gamma)$ *for all players.*

Probably approximately correct (PAC) analysis of multi-agent RL algorithms deals with the question of how fast an RL algorithm converges to a desired fixed point. Since in the case of this paper this point is a policy that is ϵ-near or better than the Nash policy, the PAC property of a multi-agent RL algorithm is understood here in relation to the existence of a probabilistic bound on the number of exploration steps that the algorithm takes before converging to a policy that is ϵ-near or better (in terms of value) than the Nash policy.

Definition 5. *Consider a Markov Game* M *with Nash equilibrium* (π_*^1, π_*^2) *in which all players are independently executing a given* RL *algorithm* $\mathcal{A}$. *Let* s_t *be the state visited at time step* t *and* $\mathcal{A}_t^i$ *be the non-stationary policy that* $\mathcal{A}$ *computes for player* i *at* t. *For a given* $\epsilon > 0$ *and* $\delta > 0$, $\mathcal{A}$ *is a* PAC *if there is an* $N > 0$ *such that with probability at least* $1-\delta$, *and for all but* N *time steps, every player* i *satisfies*

$$v_M^i(s_t, \mathcal{A}_t^1, \mathcal{A}_t^2) \geq v_M^i(s_t, \pi_*^1, \pi_*^2) - \epsilon \tag{4}$$

Inequality (4) is henceforth referred to as the *ϵ-or-better Nash condition*, and N as the *sample complexity* of $\mathcal{A}$. With $|\cdot|$ denoting cardinality, sample complexity is a function of any combination of $|S|$, $|A^1|$, $|A^2|$, $\frac{1}{\epsilon}$, $\frac{1}{\delta}$, and $\frac{1}{1-\gamma}$.

Now let K_t be set of current known state-action profiles for an RL algorithm $\mathcal{A}$ at time step t, and allow it to be arbitrarily defined as long as it depends only on the history of exploration data up to t. Any $(s,a^1,a^2) \notin K_t$ experienced at time step t marks an *escape event*.

The proof of each of the following three lemmas is very similar to that of the original version as it appears in literature, which the reader is referred to for details.

Lemma 1. *(cf. [25, Lemma 2]) For a two-player Markov Game* M, *the* H-step value function *for policy profile* (π^1, π^2) *is defined as*

$$v_M^i(s, \pi^1, \pi^2, H) \\ := \mathbb{E}_M \left\{ \sum_{t=0}^{H} \gamma^t R^i\big(s_t, \pi^1(s_t), \pi^2(s_t)\big) \,\middle|\, s_0 = s \right\}$$

and for $H = \frac{1}{1-\gamma} \ln \frac{1}{(1-\gamma)\epsilon}$ *it holds*

$$\big| v_M^i(s, \pi^1, \pi^2) - v_M^i(s, \pi^1, \pi^2, H) \big| \leq \epsilon$$

Lemma 2. *(cf. [22, Lemma 9]) Given a two-player Markov Game* M *and a set of known state-action profiles* K_t, *the value of any state* s *under any policy profile* (π^1, π^2) *in the known state-action Markov Game* M_{K_t} *is bounded by* $\frac{2}{1-\gamma} = 2v_{\max}$.

Lemma 3. *(cf. [22, Lemma 8]) Suppose that a weighted coin that is flipped has a probability* $p > 0$ *of landing with heads up. Then, for any positive integer* k *and* $\delta \in (0,1)$, *there exists* $m = \mathcal{O}(\frac{k}{p} \ln \frac{1}{\delta})$, *such that after* m *tosses, with probability at least* $1-\delta$, *one observes* k *or more heads.*

The following theorem is one of the two main results of this paper. It offers sufficient conditions for a Nash-greedy RL algorithm in a two-player Markov game to be PAC.

Theorem 1. *Let* M *be a two-player Markov game in which players are executing a Nash-greedy* RL *algorithm* $\mathcal{A}$. *Denote* (π_*^1, π_*^2) *the Nash equilibrium of* M, *and* K_t *the set of current known state-action profiles at time step* t. *Let* M_{K_t} *be the known state-action Markov Game at time step* t *and* $\pi_t^i := \text{argNash}_{a^i}\left(Q_t^1(s,:), Q_t^2(s,:)\right)$ *for brevity. Assume that* $K_t = K_{t+1}$ *unless at time step* t, *an update to some estimated Nash Q-value, or an escape event occurs. Suppose also that*

$Q_t^i(s,a^1,a^2) \le v_{\max}$ *for all players, state-action profiles and time steps.*

If for $\epsilon > 0$ *the following conditions hold with probability at least* $1-\delta \in (0,1)$,

optimism: $v_t^i(s) \ge v_M^i(s,\pi_*^1,\pi_*^2) - \epsilon$

accuracy: $v_t^i(s) - v_{M_{K_t}}^i(s,\pi_t^1,\pi_t^2) \le \epsilon$

complexity: *the sum of the number of time steps where Nash Q-value updates occur plus number of time steps where escape events occur is upper bounded by* $\zeta(\epsilon,\delta) > 0$.

then the players will follow a policy that results at values which are at most 4ϵ *lower than a Nash policy on all but*

$$\mathcal{O}\left(\frac{\zeta(\epsilon,\delta)}{\epsilon(1-\gamma)^2}\ln\left(\tfrac{1}{\delta}\right)\ln\left(\tfrac{1}{\epsilon(1-\gamma)}\right)\right) \simeq \mathcal{O}\left(\frac{\zeta(\epsilon,\delta)}{\epsilon(1-\gamma)^2}\right) \quad (5)$$

time steps, with probability at least $1-2\delta$.

Proof. Pick $\epsilon,\delta > 0$, and suppose that algorithm $\mathcal{A}$ is executed by both players in game M, with $\mathcal{A}_t^i$ being the current non-stationary policy of player i, for $i = 1,2$. Let s_t denote the state of the game at t, and set E be one of the two possible events that can occur in the execution of algorithm $\mathcal{A}$, $H = \frac{1}{1-\gamma}\ln\frac{1}{(1-\gamma)\epsilon}$ time steps after arriving at state s_t. Event E can be:

- an update to any of the Nash Q-value estimates, or
- an experience of a state-action profile $(s,a^1,a^2) \notin K_t$ (escape event).

Let p_E be the probability that event E occurs; verify that:

$$\begin{aligned} v_M^i(s_t,\mathcal{A}_t^1,\mathcal{A}_t^2) &\ge v_M^i(s_t,\mathcal{A}_t^1,\mathcal{A}_t^2,H) \\ &\ge v_{M_{K_t}}^i(s_t,\pi_t^1,\pi_t^2,H) - 2v_{\max}\cdot p_E \end{aligned}$$

The left inequality is due to all rewards being positive, and the right follows from the fact that following $\mathcal{A}_t^1$, $\mathcal{A}_t^2$ in M results in an identical behavior of following π_t^1, π_t^2 in M_{K_t}, unless event E occurs which can at most reduce the value by $2v_{\max}$ (Lemma 2). From the above, now write:

$$\begin{aligned} v_M^i(s_t,\mathcal{A}_t^1,\mathcal{A}_t^2) &\ge v_{M_{K_t}}^i(s_t,\pi_t^1,\pi_t^2,H) - 2v_{\max}\,p_E \\ &\overset{\text{Lemma 1}}{\ge} v_{M_{K_t}}^i(s_t,\pi_t^1,\pi_t^2) - \epsilon - 2v_{\max}\,p_E \\ &\overset{accuracy}{\ge} v_t^i(s_t) - 2\epsilon - 2v_{\max}\,p_E \\ &\overset{optimism}{\ge} v_M^i(s,\pi_*^1,\pi_*^2) - 3\epsilon - 2v_{\max}\,p_E \end{aligned}$$

Now, if $p_E < \frac{\epsilon}{2v_{\max}}$ the claim is proved: we have the 4ϵ-or-better Nash condition

$$v_M^i(s_t,\mathcal{A}_t^1,\mathcal{A}_t^2) \ge v_M^i(s,\pi_*^1,\pi_*^2) - 4\epsilon$$

If, on the other hand, $p_E \ge \frac{\epsilon}{2v_{\max}}$, Lemma 3 would guarantee with probability at least $1-\delta$ that $\zeta(\epsilon,\delta)$ occurrences of event E happen within $\mathcal{O}(\frac{\zeta(\epsilon,\delta)Hv_{\max}}{\epsilon}\ln\frac{1}{\delta})$ time steps. However, the *complexity* condition guarantees with probability $1-\delta$ that $\zeta(\epsilon,\delta)$ is the maximum number of updates or escape events. With probability at least $1-2\delta$, therefore, and for all but $\mathcal{O}(\frac{\zeta(\epsilon,\delta)Hv_{\max}}{\epsilon}\ln\frac{1}{\delta}) = \mathcal{O}(\frac{\zeta(\epsilon,\delta)}{\epsilon(1-\gamma)^2}\ln\left(\frac{1}{\delta}\right)\ln\left(\frac{1}{\epsilon(1-\gamma)}\right))$ timesteps, it must be the case that $v_M^i(s_t,\mathcal{A}_t^1,\mathcal{A}_t^2) \ge v_M^i(s,\pi_*^1,\pi_*^2) - 4\epsilon$.

IV. Delayed Nash Q-learning Algorithm

This section presents the second key contribution of this paper: a new RL algorithm for two-player Markov Games, referred to as Delayed Nash Q-learning (Algorithm 1). Delayed Nash Q-learning is the first algorithm that is PAC in terms of converging to a policy which is arbitrarily near (or better than) the Nash equilibrium policy. The computational complexity of Delayed Nash Q-learning is comparable to the well-known Nash Q-learning [16].

We assume that both players are executing the (same) algorithm. Thus, players keep the Nash Q-value estimates of their opponents. This is made possible by the assumption that every player can observe the rewards of all players at each step in the Markov Game. With the same observations, players' estimates of the Nash Q-values are identical. As the "delayed" term in the name suggests, the algorithm waits until a state-action profile is experienced m times before it makes an update of its Nash Q-value. The update mechanism (of lines 24, 25) requires a successful update to change the Nash Q-value estimate, triggered by another parameter ϵ_1. Both m and ϵ_1 parameters are tunable. Similarly to Delayed Q-learning [22] or R-max [26], the algorithm reported here utilizes the principle of "optimism in the face of uncertainty" to encourage exploration by originally overestimating the Nash Q-value estimates to some $v_{\max}$.

Similar to Delayed Q-learning [22], Delayed Nash Q-learning maintains the following internal variables:

- $l^i(s,a^1,a^2)$ is the number of samples gathered for the update of $Q^i(s,a^1,a^1)$ once it is the case that $l^i(s,a^1,a^2) = m$.
- $U^i(s,a^1,a^2)$ stores the running sum of target values that will be used for the update of $Q^i(s,a^1,a^1)$ once enough samples have been gathered.
- $b^i(s,a^1,a^2)$ is the time step at which the collection of the most recent m experiences of (s,a^1,a^2) started.
- $\mathsf{learn}^i(s,a^1,a^2)$ is a Boolean flag indicating whether samples are being gathered for state-action profile (s,a^1,a^2). It is set to true initially, and is reset to true whenever some Nash Q-value of player i is updated. It changes to false when no updates to any Nash Q-values occur within a time window in which (s,a^1,a^2) is experienced m times, but subsequent attempted updates of $Q^i(s,a^1,a^2)$ fail (cf. [22]).

V. PAC Properties of Delayed Nash Q-learning

In this section, we claim that Delayed Nash Q-learning algorithm is PAC and we present its sample complexity bound. The proof requires the following assumption, which is the same requirement that the well-known Nash Q-learning needs to guarantee convergence [16].

Assumption 1. *For every state* s *and timestep* t*, Delayed Nash Q-learning encounters a stage game associated with* $\left(Q_t^1(s,:),Q_t^2(s,:)\right)$*, which possesses either a global optimum or a saddle point Nash equilibrium. The algorithm may use either of the two for its updates.*

Algorithm 1 The Delayed Nash Q-learning algorithm

1: **Inputs**: $S, A^1, A^2, \gamma, m, \epsilon_1$
2: **for all** $s, a^1, a^2, i = 1, 2$ **do**
3: $Q^i(s,a^1,a^2) \leftarrow v_{\max}$ ▷ set Nash Q-value to its maximum
4: $U^i(s,a^1,a^2) \leftarrow 0$ ▷ used for attempted updates
5: $l^i(s,a^1,a^2) \leftarrow 0$ ▷ counters
6: $b^i(s,a^1,a^2) \leftarrow 0$ ▷ starting timestep of attempted update
7: $\mathsf{learn}^i(s,a^1,a^2) \leftarrow \text{true}$ ▷ learn flags
8: **end for**
9: $t^* \leftarrow 0$ ▷ time of the most recent successful update
10: **for** $t = 1, 2, 3, \ldots$ **do**
11: let s denotes the state at time t
12: players choose (a^1, a^2) according to π_t^i as in (3)
13: observe immediate rewards r^1, r^2 & next state s'
14: **if** $b^i(s,a^1,a^2) \le t^*$ **then**
15: $\mathsf{learn}^i(s,a^1,a^2) \leftarrow \text{true}$
16: **end if**
17: **if** $\mathsf{learn}^i(s,a^1,a^2) = \text{true}$ **then**
18: **if** $l^i(s,a^1,a^2) = 0$ **then**
19: $b^i(s,a^1,a^2) \leftarrow t$
20: **end if**
21: $l^i(s,a^1,a^2) \leftarrow l^i(s,a^1,a^2) + 1$
22: $U^i(s,a^1,a^2) \leftarrow U^i(s,a^1,a^2) + r^i + \gamma v^i(s')$
23: **if** $l^i(s,a^1,a^2) = m$ **then**
24: **if** $Q^i(s,a^1,a^2) - U^i(s,a^1,a^2)/m \ge 2\epsilon_1$ **then**
25: $Q^i(s,a^1,a^2) \leftarrow U^i(s,a^1,a^2)/m + \epsilon_1$
26: $t^* \leftarrow t$
27: **else if** $b^i(s,a^1,a^2) > t^*$ **then**
28: $\mathsf{learn}^i(s,a^1,a^2) \leftarrow \text{false}$
29: **end if**
30: $U^i(s,a^1,a^2) \leftarrow 0$
31: $l^i(s,a^1,a^2) \leftarrow 0$
32: **end if**
33: **end if**
34: **end for**

Before formally stating the PAC properties of the Delayed Nash Q-learning algorithm and proving the bound on its sample complexity, some technical groundwork needs to be laid. To slightly simplify notation, let

$$\kappa \triangleq \frac{|S||A^1||A^2|}{(1-\gamma)\epsilon_1} \qquad v_*^i(s) \triangleq v_M^i(s, \pi_*^1, \pi_*^2)$$

and note that subscript t marks the value of a variable at the *beginning* of time step t (particularly line 17 of the algorithm).

Definition 6. *An* attempted update *is an event at which* $\mathsf{learn}^i(s,a^1,a^2) = \text{true}$ *and* $l^1(s,a^1,a^2) = l^2(s,a^1,a^2) = m$. *An attempted update can be successful or unsuccessful depending on the condition of line* 24 *of the algorithm.*

Definition 7. *At any time step t of the Delayed Nash Q-learning algorithm, the set of* known state-action profiles *is defined as*

$$K_t := K_t^1 \cap K_t^2 \qquad \textit{where}$$
$$K_t^i := \Big\{(s,a^1,a^2) \mid Q_t^i(s,a^1,a^2) - R^i(s,a^1,a^2) - \gamma \textstyle\sum_{s'} T(s,a^1,a^2,s') v_t^i(s') \le 3\epsilon_1\Big\} . \quad (6)$$

For the Delayed Nash Q-learning algorithm a number of facts can be shown. The proof for these claims can be found in different appendices at the end of this paper. First, the number of successful updates is bounded:

Lemma 4. *The total number of updates during the execution of Delayed Nash Q-learning algorithm is bounded by* 2κ.

Proof. In Appendix A.

Attempted updates are bounded in number:

Lemma 5. *The total number of attempted updates in Delayed Nash Q-learning algorithm is bounded by* $2|S||A^1||A^2|(1+2\kappa)$.

Proof. In Appendix B.

Players decrease their state Nash value estimates as time goes on:

Lemma 6. *Let $t_1 < t_2$ be two timesteps during the execution of Delayed Nash Q-learning algorithm. Then under the Assumption 1, for all states s and any player i, $v_{t_1}^i(s) \ge v_{t_2}^i(s)$.*

Proof. In Appendix C.

Lemma 7. *Suppose that the Delayed Nash Q-learning is executed on a Markov Game M under Assumption 1 with parameter m satisfying*

$$m \ge \frac{\ln\left(\frac{6|S||A^1||A^2|(1+2\kappa)}{\delta}\right)}{2\epsilon_1^2(1-\gamma)^2} \simeq \mathcal{O}\left(\frac{\ln\left(\frac{|S|^2|A^1|^2|A^2|^2}{\delta}\right)}{\epsilon_1^2(1-\gamma)^2}\right) \quad (7)$$

Then, $Q_t^i(s,a^1,a^2) \ge Q_*^i(s,a^1,a^2)$ and $v_t^i(s) \ge v_*^i(s)$ for any player i, state-action profile (s,a^1,a^2), and timestep t, with probability at least $1 - \frac{\delta}{3}$.

Proof. In Appendix D.

Lemma 8. *Under Assumption 1 and with the choice of m as in* (7)*, assume that the Delayed Nash Q-learning algorithm is at timestep t with $(s,a^1,a^2) \notin K_t^i$, $l^i(s,a) = 0$ and $\mathsf{learn}^i(s,a^1,a^2) = \text{true}$ for player i. Knowing that an attempted update of $Q^i(s,a^1,a^2)$ will necessarily occur within m occurrences of (s,a^1,a^2) after t, say at timestep t_m, this attempted update at t_m will be successful with probability at least* $1 - \frac{\delta}{3}$.

Proof. In Appendix E.

Lemma 9. *Let t be the timestep when an unsuccessful update of $Q^i(s,a^1,a^2)$ occurs after the conditions of Lemma 8 were satisfied. If $\mathsf{learn}^i(s,a^1,a^2) = \text{false}$ at timestep $t+1$, then $(s,a^1,a^2) \in K_{t+1}^i$.*

Proof. In Appendix F.

Lemma 10. *During the execution of the Delayed Nash Q-learning algorithm, and assuming that Lemma 8 applies, the total number of timesteps with $(s_t, a_t^1, a_t^2) \notin K_t$ (i.e. escape events) is at most $4m\kappa$.*

Proof. In Appendix G.

The PAC properties of Algorithm 1 can now be established in following form theorem, the proof of which is based on an of Theorem 1.

Theorem 2. *Consider a two-player Markov Game $M = \{S, A^1, A^2, T, R^1, R^2, \gamma\}$. Pick $\epsilon \in \left(0, \frac{1}{1-\gamma}\right)$, and $\delta \in (0,1)$. Then, with $\frac{1}{\epsilon_1} = \frac{3}{(1-\gamma)\epsilon} = \mathcal{O}\left(1/\epsilon(1-\gamma)\right)$, there exists an integer*

$$m = \mathcal{O}\left(\frac{\ln\left(|S|^2|A^1|^2|A^2|^2/\delta\right)}{\epsilon_1^2(1-\gamma)^2}\right)$$

such that if the Delayed Nash Q-learning algorithm is executed by both players under Assumption 1 and with the set K_t defined as (6)*, the players will follow a policy which with probability at least $1-2\delta$ is at most 4ϵ worse than a Nash policy, on all but*

$$\mathcal{O}\left(\frac{|S||A^1||A^2|}{\epsilon^4(1-\gamma)^8}\right) \tag{8}$$

time steps (logarithmic factors ignored).

Proof. Apply Theorem 1. It is already shown in Lemma 7 that the *optimism* condition holds throughout the execution of Delayed Nash Q-learning algorithm.

To establish the *accuracy* condition, for all player $i = 1, 2$ and all state s, write

$$\begin{aligned} v^i_{M_{K_t}}(s, \pi_t^1, \pi_t^2) = &\sum_{\substack{(a^1,a^2)\\(s,a^1,a^2)\in K_t}} \pi_t^1(s,a^1), \pi_t^2(s,a^2)\times \\ &\left(R^i(s,a^1,a^2) + \gamma \sum_{s'} T(s,a^1,a^2,s') v^i_{M_{K_t}}(s', \pi_t^1, \pi_t^2)\right) \\ &+ \sum_{\substack{(a^1,a^2)\\(s,a^1,a^2)\notin K_t}} \pi_t^1(s,a^1), \pi_t^2(s,a^2)\left(Q_t^i(s,a^1,a^2)\right) \end{aligned} \tag{9}$$

Similarly,

$$\begin{aligned} v_t^i(s) = &\sum_{(a^1,a^2)} \pi_t^1(s,a^1), \pi_t^2(s,a^2)\left(Q_t^i(s,a^1,a^2)\right) = \\ &\sum_{\substack{(a^1,a^2)\\(s,a^1,a^2)\in K_t}} \pi_t^1(s,a^1), \pi_t^2(s,a^2)\times \\ &\left(R^i(s,a^1,a^2) + \gamma \sum_{s'} T(s,a^1,a^2,s') v_t^i(s') + \beta^i(s,a^1,a^2)\right) \\ &+ \sum_{\substack{(a^1,a^2)\\(s,a^1,a^2)\notin K_t}} \pi_t^1(s,a^1), \pi_t^2(s,a^2)\left(Q_t^i(s,a^1,a^2)\right) \end{aligned} \tag{10}$$

where by the definition of K_t as (6), you know that $\beta^i(s,a^1,a^2) \le 3\epsilon_1$. To slightly simplify the notation, set $\Delta^i(s) \triangleq v_t^i(s) - v^i_{M_{K_t}}(s, \pi_t^1, \pi_t^2)$. Now, if you denote

$$\alpha := \max_s \left(\Delta^i(s)\right) = \Delta^i(s^*)$$

by (9) and (10), it follows that α affords the bound

$$\begin{aligned} \alpha = &\sum_{\substack{(a^1,a^2)\\(s,a^1,a^2)\in K_t}} \pi_t^1(s,a^1), \pi_t^2(s,a^2)\times \\ &\left(\gamma \sum_{s'} T(s^*,a^1,a^2,s')\Delta^i(s') + \beta^i(s^*,a^1,a^2)\right) \\ &\le \gamma\alpha + 3\epsilon_1 \end{aligned}$$

from which it follows that $\alpha \le \gamma\alpha + 3\epsilon_1 \implies \alpha \le \frac{3\epsilon}{1-\gamma} = \epsilon$.

Finally, to confirm the *complexity* condition, invoke Lemmas 4 and 10 to see that the learning complexity $\zeta(\epsilon, \delta)$ is bounded by $2\kappa + 4m\kappa$.

In conclusion, the conditions of Theorem 1 are satisfied and therefore the Delayed Nash Q-learning algorithm is PAC. Substituting $\zeta(\epsilon, \delta)$ into (5) yields (8) and completes the proof.

VI. Numerical Results

In this section, the performance of the Delayed Nash Q-learning algorithm is evaluated on two grid-world games used in work reported in literature [16].

The grid-world games 1 and 2 are shown in Figs. 1 and 2, respectively, where the initial and goal positions of each player is depicted. The home cells are the goal configurations for the two agents and the smiley faces mark their initial positions. In grid-world game 1 the players have different goal positions; in 2, they have the same. Cells are labeled in the order depicted in Fig. 3. Both players can move only one cell a time, exercising one of four primitive actions: down (d), left (l), up (u), and right (r). If the two players attempt to move together into any cell other than the goal, they bounce back to their previous cells.

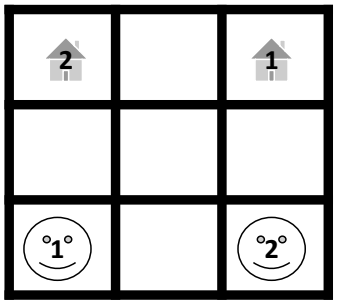

Fig. 1: The grid-world game 1 (cf. [16])

The state-space of the game is the set of pairs $S = \{(1,2), (1,3), \ldots, (9,8)\}$, in which the first number is the location of player 1 and the second number is the location of player 2. State (1,3) is the initial state, and all states in which one of the players is in its goal location are terminal states. If a player reaches its goal position, it receives the reward of 1, and for all other moves the reward is 0. Transitions in grid-world game 1 are deterministic; transitions in grid-game 2 are also deterministic except for action up in cells 1 and 3 (shown by dashed lines in Fig. 2). If player 1 (resp. 2) chooses action up in cell 1 (resp. 3), it will either move to cell 4 (resp. 6) with

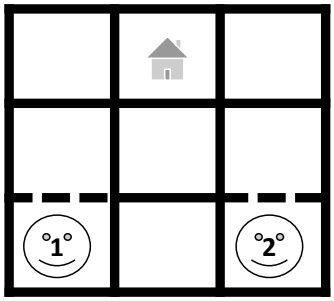

Fig. 2: The grid-world game 2 (cf. [16])

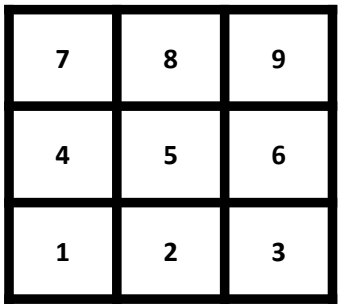

Fig. 3: Cell indexing

probability 0.5 or stay in the same cell with probability 0.5.

Multiple Nash equilibrium strategies exist for grid-world game 1. Some examples are depicted in Fig. 4. For grid-world game 2 on the other hand, there exist only two Nash equilibrium strategies: the ones shown in Fig. 5.

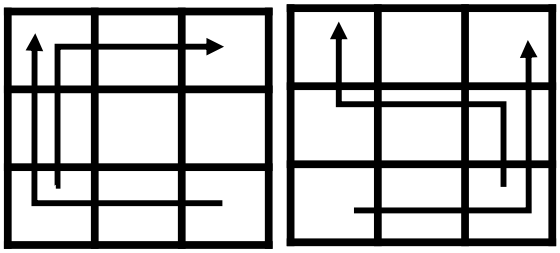

Fig. 4: Examples of Nash strategies for grid-world 1.

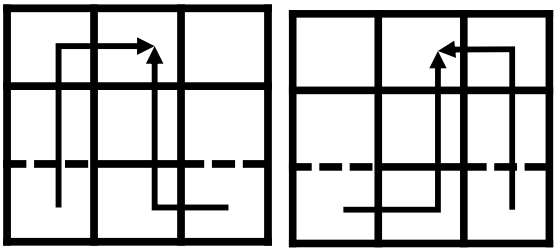

Fig. 5: The Nash equilibrium strategies for grid-world 2.

Note that all Nash equilibria in grid-world game 1 are globally optimal; however, there is no guarantee that all stage games encountered during the execution of the learning algorithm possess a globally optimal Nash equilibrium [16]. On the other hand, none of the Nash equilibrium strategies for grid-world game 2 are globally optimal or saddles.

In the earlier study of these two grid-world games, it has been reported [16] that while both games do not necessarily satisfy Assumption 1, the convergence of a Nash Q-learning algorithm is almost guaranteed for the game that has an overall globally optimal or saddle Nash equilibrium strategy (e.g., grid-world game 1). For games that do not, (e.g., grid-world game 2), a 79% probability of convergence using a Nash Q-learning algorithm was reported [16].

In this paper, Delayed Nash Q-learning algorithm was implemented on the two grid-world games with the following parameters: $\gamma = 0.8$, $m = 50$ and $\epsilon = 0.06$. The algorithm was run on each game for 50 times (matching [16]). Delayed Nash Q-learning converged to a Nash equilibrium strategy profile, in both games, *every time*. The average number of steps needed for convergence (the empirical sample complexity of the algorithm) was 445640 in the case of grid-world game 1, and 485460 in the case of grid-world game 2.

VII. Conclusion

Computational gains and analytical performance guarantees obtained recently in applications of RL to finite MDPs can be extended to Markov games yielding a unique new PAC algorithm for learning game equilibria. A new sample-efficient MARL algorithm for Markov games emerges as the outcome of careful integration of delayed Q-learning techniques with Nash Q-learning methodologies. This algorithm, referred to as delayed Nash Q-learning, is guaranteed to converge on finite samples, under the same assumptions utilized for Nash Q-learning. Numerical results not only support the theoretical PAC predictions, but also indicate that the delayed Nash Q-learning algorithm performance may degrade much more gracefully than that of Nash Q-learning when the underlying assumptions are violated.

Appendix

A. Proof of Lemma 4

Consider a fixed state-action profile (s, a^1, a^2). Its value $Q^i(s, a^1, a^2)$ for player i is initially set to $v_{\max} = \frac{1}{1-\gamma}$. When an update is successful $Q^i(s, a^1, a^2)$ is reduced by at least ϵ_1 (because of the condition of line 24 of the algorithm). Since the reward function $R^i(s, a)$ is non-negative, $Q^i(s, a^1, a^2) \geq 0$ in all timesteps, which means that there can be at most $\lfloor \frac{1}{\epsilon_1(1-\gamma)} \rfloor$ updates for $Q^i(s, a^1, a^2)$. With $|S||A^1||A^2|$ total state-action profiles and since there are 2 players, the total number updates is bounded by $2\kappa = 2\frac{|S||A^1||A^2|}{(1-\gamma)\epsilon_1}$.

B. Proof of Lemma 5

Suppose an attempted update occurs at timestep t to some $Q^i(s,a^1,a^2)$. By definition, in order for a subsequent attempted update to $Q^i(s,a^1,a^2)$ to occur at timestep $t' > t$, at least one update to any Nash Q-value estimate must occur between t and t'. Lemma 4 ensures that there can be no more than 2κ (successful) updates. In other words, the most frequent occurrence of an attempted update is interlaced between successful updates, which implies that at most $1+2\kappa$ attempted updates are possible for $Q^i(s,a^1,a^2)$. Scaling this argument to all state-action profiles and both players, we arrive at the $2|S||A^1||A^2|(1+2\kappa)$ upper bound.

C. Proof of Lemma 6

Show the lemma for player 1; the proof for the other player is identical. First note that because the Delayed Nash Q-learning algorithm only allows updates that decrease the Q estimates, for all state $s \in S$

$$Q^1_{t_1}(s,a^1,a^2) \geq Q^1_{t_2}(s,a^1,a^2) \quad \forall (a^1,a^2) \in A^1 \times A^2$$

Let $(\pi^1_{t_1},\pi^2_{t_1})$ and $(\pi^1_{t_2},\pi^2_{t_2})$ be the Nash equilibria of stage games associated with state s, at timesteps t_1 and t_2, respectively. Then, write

$$v^1_{t_1}(s) = \sum_{(a^1,a^2)} \pi^1_{t_1}(s,a^1)\cdot\pi^2_{t_1}(s,a^2)\cdot Q^1_{t_1}(s,a^1,a^2)$$
$$v^1_{t_2}(s) = \sum_{(a^1,a^2)} \pi^1_{t_2}(s,a^1)\cdot\pi^2_{t_2}(s,a^2)\cdot Q^1_{t_2}(s,a^1,a^2)$$

If the stage games possess global optimal Nash equilibrium, it will be

$$\begin{aligned} v^1_{t_1}(s) &\overset{\text{global optimal}}{\geq} \sum_{(a^1,a^2)} \pi^1_{t_2}(s,a^1),\pi^2_{t_2}(s,a^2)\big(Q^1_{t_1}(s,a^1,a^2)\big) \\ &\geq \sum_{(a^1,a^2)} \pi^1_{t_2}(s,a^1),\pi^2_{t_2}(s,a^2)\big(Q^1_{t_2}(s,a^1,a^2)\big) \\ &= v^1_{t_2}(s) \end{aligned}$$

whereas if the stage games possess saddle point Nash Equilibrium, again write

$$\begin{aligned} v^1_{t_1}(s) &\overset{\text{Nash equilibrium}}{\geq} \sum_{(a^1,a^2)} \pi^1_{t_2}(s,a^1),\pi^2_{t_2}(s,a^2)\big(Q^1_{t_1}(s,a^1,a^2)\big) \\ &\geq \sum_{(a^1,a^2)} \pi^1_{t_2}(s,a^1),\pi^2_{t_1}(s,a^2)\big(Q^1_{t_2}(s,a^1,a^2)\big) \\ &\overset{\text{saddle}}{\geq} \sum_{(a^1,a^2)} \pi^1_{t_2}(s,a^1),\pi^2_{t_2}(s,a^2)\big(Q^1_{t_2}(s,a^1,a^2)\big) \\ &= v^1_{t_2}(s) \end{aligned}$$

Thus in any case, it will be $v^1_{t_1}(s) \geq v^1_{t_2}(s)$ and the proof is completed.

D. Proof of Lemma 7

Prove the claim for player 1; the proof for the other player is identical. The proof involves strong induction for all state-action profile (s,a^1,a^2):

(i) At $t=1$, the values of all state-action profiles are set to the maximum possible Nash value of the Markov Game M. This implies that $Q^1_1(s,a^1,a^2) \geq Q^1_*(s,a^1,a^2)$ and $v^1_1(s) \geq v^1_*(s)$.

(ii) Assume that $Q^1_t(s,a^1,a^2) \geq Q^1_*(s,a^1,a^2)$ and $v^1_t(s) \geq v^1_*(s)$ for all timesteps up to and including $t = n-1$.

(iii) If no successful update happens during the timestep $t = n-1$, then

$$Q^1_n(s,a^1,a^2) = Q^1_{n-1}(s,a^1,a^2) \geq Q^1_*(s,a^1,a^2) \text{ and}$$
$$v^1_n(s) = v^1_{n-1} \geq v^1_*(s)$$

and the claim is immediately established. If not, assume that during the timestep $t = n-1$, the state-action profile for which the update occurs is (s,a^1,a^2). Suppose that the latest m experiences of (s,a^1,a^2) happened at timesteps $t_1 < t_2 < \cdots < t_m = n-1$, at which the player was rewarded $r^1[1], r^1[2], \ldots, r^1[m]$ and the game jumped to states $s[1], s[2], \ldots, s[m]$, respectively. Define the random variable $Y := r^1[i] + \gamma v^1_*(s[i])$ for $1 \leq i \leq m$ and note that $0 \leq Y \leq \frac{1}{1-\gamma}$. Then a direct application of the Hoeffding inequality for bounded random variables and with the choice of m as in (7) implies

$$\frac{1}{m}\sum_{i=1}^{m}\big(r^1[i] + \gamma v^1_*(s[i])\big) > \mathbb{E}\{Y\} - \epsilon_1 = Q^1_*(s,a^1,a^2) - \epsilon_1$$

with probability $1 - \delta/6(|S||A^1||A^2|(1+2\kappa))$. Now you have:

$$\begin{aligned} Q^1_n(s,a^1,a^2) &= \frac{1}{m}\Big(\sum_{i=1}^{m} r^1[i] + \gamma v^1_{t_i}(s[i])\Big) + \epsilon_1 \\ &\geq \frac{1}{m}\Big(\sum_{i=1}^{m} r^1[i] + \gamma v^1_*(s[i])\Big) + \epsilon_1 \\ &\geq Q^1_*(s,a^1,a^2) - \epsilon_1 + \epsilon_1 \\ &= Q^1_*(s,a^1,a^2)\ . \end{aligned}$$

To show $v^1_n(s) \geq v^1_*(s)$, first note that

$$v^1_n(s) = \sum_{(a^1,a^2)} \pi^1_n(s,a^1)\pi^2_n(s,a^2)Q^1_n(s,a^1,a^2)$$

where (π^1_n,π^2_n) denotes the policy executed at timestep $t = n$. If the stage games possess global optimum Nash equilibrium,

$$\begin{aligned} v^1_n(s) &\overset{\text{global optimum}}{\geq} \sum_{(a^1,a^2)} \pi^1_*(s,a^1)\,\pi^2_*(s,a^2)\,Q^1_n(s,a^1,a^2) \\ &\geq \sum_{(a^1,a^2)} \pi^1_*(s,a^1)\,\pi^2_*(s,a^2)\,Q^1_*(s,a^1,a^2) \\ &= v^1_*(s) \end{aligned}$$

If the stage games possess a saddle point Nash Equilibrium,

$$\begin{aligned} v_n^1(s) &\overset{\text{Nash equilibrium}}{\geq} \sum_{(a^1,a^2)} \pi_*^1(s,a^1)\,\pi_n^2(s,a^2)\,Q_n^1(s,a^1,a^2) \\ &\geq \sum_{(a^1,a^2)} \pi_*^1(s,a^1)\,\pi_n^2(s,a^2)\,Q_*^1(s,a^1,a^2) \\ &\overset{\text{saddle}}{\geq} \sum_{(a^1,a^2)} \pi_*^1(s,a^1)\,\pi_*^2(s,a^2)\,Q_*^1(s,a^1,a^2) \\ &= v_*^1(s) \end{aligned}$$

The induction argument is thus complete. Since the conclusion has to be true for all possible attempted updates, invoke Lemma 5, according to which $2|S||A^1||A^2|(1+2\kappa)$ is an upper bound for all possible attempted updates. Therefore, the statement above is true with probability at least $\left(1-\delta/6(|S||A^1||A^2|(1+2\kappa))\right)^{2|S||A^1||A^2|(1+2\kappa)}$. Another induction argument can now be employed to show that $1-\frac{\delta}{3}$ bounds the latter expression from below.

E. Proof of Lemma 8

Assume that at timestep t, $(s,a^1,a^2) \notin K_t^i$, $l^i(s,a) = 0$ and $\text{learn}^i(s,a) = \text{true}$, and suppose that m experiences of (s,a^1,a^2) happen at timesteps $t \leq t_1 < t_2 < \cdots < t_m$. Since $l^i(s,a) = 0$ and $\text{learn}^i(s,a) = \text{true}$, an attempted update will necessarily happen. Let $r^i[1], r^i[2], \ldots, r^i[m]$ and $s[1], s[2], \ldots, s[m]$ be the rewards and next states observed for the m experiences of (s,a^1,a^2) for player i. Then define the random variable $X := r^i[j] + \gamma v_t^i(s[j])$, letting j range in $\{1,\ldots,m\}$, and note that $0 \leq X \leq \frac{1}{1-\gamma}$.

A direct application of the Hoeffding inequality with the choice of m as in (7) yields

$$\frac{1}{m}\left(\sum_{j=1}^{m} r^i[j] + \gamma v_t^i(s[j])\right) - \mathbb{E}\{X\} < \epsilon_1$$

with probability $1 - \frac{\delta}{6|S||A^1||A^2|(1+2\kappa)}$. Note that there can be at most $2|S||A^1||A^2|(1+2\kappa)$ instances of such an event. Since Lemma 6 shows that all Nash Q-value estimates are decreasing during the execution of Delayed Nash Q-learning algorithm, for the condition on line 24 of Algorithm 1, write:

$$\begin{aligned} Q_t^i(s,a^1,a^2) - \frac{1}{m}\left(\sum_{j=1}^{m} r^i[j] + \gamma v_{t_j}^i(s[j])\right) \\ \geq Q_t^i(s,a^1,a^2) - \frac{1}{m}\left(\sum_{j=1}^{m} r^i[j] + \gamma v_t^i(s[j])\right) \\ > Q_t^i(s,a^1,a^2) - \mathbb{E}\{X\} - \epsilon_1 \end{aligned}$$

and because $(s,a^1,a^2) \notin K_t^i$ meaning that $Q_t^i(s,a^1,a^2) - \mathbb{E}\{X\} > 3\epsilon_1$,

$$Q_t^i(s,a^1,a^2) - \mathbb{E}\{X\} - \epsilon_1 > 2\epsilon_1 \ ,$$

guaranteeing success for the update at timestep t_m. Working similarly to the proof of Lemma 7, one concludes that the successful update will occur with probability at least $1 - \frac{\delta}{3}$.

F. Proof of Lemma 9

Suppose an unsuccessful update of $Q^i(s,a^1,a^2)$ occurs at timestep t, and right after, at timestep $t+1$ you observe $\text{learn}^i(s,a^1,a^2) = \text{false}$. Set up a contradiction argument: under those conditions, *assume that* $(s,a^1,a^2) \notin K_{t+1}^i$. Since the update at t was unsuccessful, $K_{t+1}^i = K_t^i$, which also implies that $(s,a^1,a^2) \notin K_t^i$. Now label the times of the most recent m experiences of (s,a^1,a^2) as $b^i(s,a^1,a^2) := t_1 < t_2 < \cdots < t_m = t$. The contrapositive of the statement proved in Lemma 8 states that given an unsuccessful update at t, it must be $(s,a^1,a^2) \in K_{t_1}^i$. Since $(s,a^1,a^2) \notin K_t^i$, some other update must have happened between t_1 and t. Denote the timestep of that update t^*, and note that $t^* \geq b^i(s,a^1,a^2)$. Observe now that the condition $t_1 = b^i(s,a^1,a^2) \leq t^*$ would not have allowed the learn flag to be set to false (line 27 of Algorithm 1). Therefore, there is a contradiction. The assumption originally made is invalid, which means $(s,a^1,a^2) \in K_{t+1}^i$.

G. Proof of Lemma 10

Fix a state-action profile (s,a^1,a^2) and chose a player i. Show first that if $(s,a^1,a^2) \notin K_t^i$ is experienced at timestep t, then within at most $2m$ subsequent experiences of (s,a^1,a^2), a successful update of $Q^i(s,a^1,a^2)$ must occur; to do so, follow this process:

For $(s,a^1,a^2) \notin K_t^i$, distinguish two possible cases at the beginning of timestep t: either $\text{learn}^i(s,a^1,a^2) = \text{false}$ or $\text{learn}^i(s,a^1,a^2) = \text{true}$.

(i) Consider first the case where $\text{learn}^i(s,a^1,a^2) = \text{false}$. Assume that the most recent attempted update of $Q^i(s,a^1,a^2)$ occurred at some timestep t' which was unsuccessful and set the flag $\text{learn}^i(s,a^1,a^2)$ to false. Then, according to Lemma 9, it will be $(s,a^1,a^2) \in K_{t'+1}^i$. However, now it is $(s,a^1,a^2) \notin K_t^i$, which implies that some update must have occurred at some t^* with $t'+1 < t^* < t$. Thus at the beginning of timestep t, the flag $\text{learn}^i(s,a^1,a^2)$ will set to true (line 15 of Algorithm 1). At timestep t all conditions of Lemma 8 (i.e. $\text{learn}^i(s,a^1,a^2) = \text{true}$, $(s,a^1,a^2) \notin K_t^i$ and $l^i(s,a^1,a^2) = 0$) are satisfied, and thus the attempted update of $Q^i(s,a^1,a^2)$ upon the m^{th} visit of (s,a^1,a^2) after timestep t will have to be successful.

(ii) Take now the case where $\text{learn}^i(s,a^1,a^2) = \text{true}$. It is given that an attempted update for $Q^i(s,a^1,a^2)$ will occur in at most m additional experiences of (s,a^1,a^2), which can be assumed occurring at timesteps $t_1 < \cdots < t_m$, and it is $t_1 \leq t \leq t_m$. Consider the two possibilities: $(s,a^1,a^2) \notin K_{t_1}^i$ or $(s,a^1,a^2) \in K_{t_1}^i$. In the former case, Lemma 8 indicates that the attempted update at t_m will be successful. In the latter case, given that $(s,a^1,a^2) \notin K_t^i$, some successful update at t^* must have taken place between t_1

and t (since $K^i_{t_1} \neq K^i_t$). If the attempted update at t_m is unsuccessful, $\text{learn}^i(s,a^1,a^2)$ remains true and at timestep t_m+1 it is $\text{learn}^i(s,a^1,a^2)$ = true, $l^i(s,a^1,a^2)=0$ and $(s,a)\notin K^i_{t_m+1}$; this now triggers Lemma 8, which implies that the attempted update of $Q^i(s,a^1,a^2)$ upon the m^{th} visit of (s,a^1,a^2) after timestep t_m+1 (within at most $2m$ more experiences of (s,a^1,a^1) after t), will be successful.

You have thus shown that after an event $(s,a^1,a^2)\notin K^i_t$, an update for $Q^i(s,a^1,a^2)$ must occur within at most $2m$ more experiences of (s,a^1,a^2). The proof of Lemma 4 shows why the total number of successful updates for (s,a^1,a^2) is bounded by $\frac{1}{(1-\gamma)\epsilon_1}$. Given this fact, the total number of timesteps with $(s,a^1,a^2)\notin K^i_t$ is bounded by $\frac{2m}{(1-\gamma)\epsilon_1}$.

Generalizing the above statement for all state-action profiles and all players, one concludes that the total number of escape events (timesteps t with $(s_t,a^1_t,a^2_t)\notin K_t$) is bounded by $4m\kappa$.

References

[1] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *arXiv preprint arXiv:1911.10635*, 2019.

[2] C. Szepesvári and M. L. Littman, "A unified analysis of value-function-based reinforcement-learning algorithms," *Neural Computation*, vol. 11, no. 8, pp. 2017–2060, 1999.

[3] M. L. Littman, "Value-function reinforcement learning in markov games," *Cognitive Systems Research*, vol. 2, no. 1, pp. 55–66, 2001.

[4] X. Wang and T. Sandholm, "Reinforcement learning to play an optimal nash equilibrium in team markov games," in *Advances in Neural Information Processing Systems*, pp. 1603–1610, 2003.

[5] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls, *et al.*, "Value-decomposition networks for cooperative multi-agent learning based on team reward," in *Proceedings of the 17th International Conference on Autonomous Agents and Multiagent Systems*, pp. 2085–2087, 2018.

[6] K. Son, D. Kim, W. J. Kang, D. E. Hostallero, and Y. Yi, "QTRAN: Learning to factorize with transformation for cooperative multi-agent reinforcement learning," *arXiv preprint arXiv:1905.05408*, 2019.

[7] J. Pérolat, B. Piot, and O. Pietquin, "Actor-critic fictitious play in simultaneous move multistage games," in *proceeding of the 21st International Conference on Artificial Intelligence and Statistics*, 2018.

[8] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou, "The complexity of computing a nash equilibrium," *SIAM Journal on Computing*, vol. 39, no. 1, pp. 195–259, 2009.

[9] M. L. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Machine Learning Proceedings*, pp. 157–163, 1994.

[10] Z. Jia, L. F. Yang, and M. Wang, "Feature-based Q-learning for two-player stochastic games," *arXiv preprint arXiv:1906.00423*, 2019.

[11] A. Sidford, M. Wang, X. Wu, L. Yang, and Y. Ye, "Near-optimal time and sample complexities for solving markov decision processes with a generative model," in *Advances in Neural Information Processing Systems*, pp. 5186–5196, 2018.

[12] A. Sidford, M. Wang, L. F. Yang, and Y. Ye, "Solving discounted stochastic two-player games with near-optimal time and sample complexity," *arXiv preprint arXiv:1908.11071*, 2019.

[13] C.-Y. Wei, Y.-T. Hong, and C.-J. Lu, "Online reinforcement learning in stochastic games," in *Advances in Neural Information Processing Systems*, pp. 4987–4997, 2017.

[14] X. Chen, X. Deng, and S.-H. Teng, "Settling the complexity of computing two-player nash equilibria," *Journal of the ACM*, vol. 56, no. 3, pp. 1–57, 2009.

[15] M. Zinkevich, A. Greenwald, and M. L. Littman, "Cyclic equilibria in markov games," in *Advances in Neural Information Processing Systems*, pp. 1641–1648, 2006.

[16] J. Hu and M. P. Wellman, "Nash Q-learning for general-sum stochastic games," *Journal of Machine Learning Research*, vol. 4, no. 6, pp. 1039–1069, 2003.

[17] R. Savani and B. Von Stengel, "Hard-to-solve bimatrix games," *Econometrica*, vol. 74, no. 2, pp. 397–429, 2006.

[18] P. W. Goldberg, C. H. Papadimitriou, and R. Savani, "The complexity of the homotopy method, equilibrium selection, and lemke-howson solutions," *ACM Transactions on Economics and Computation*, vol. 1, no. 2, pp. 1–25, 2013.

[19] A. Greenwald, K. Hall, and R. Serrano, "Correlated Q-learning," in *International Conference on Machine Learning*, vol. 20, p. 242, 2003.

[20] J. Pérolat, F. Strub, B. Piot, and O. Pietquin, "Learning nash equilibrium for general-sum markov games from batch data," *arXiv preprint arXiv:1606.08718*, 2016.

[21] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, "Mean field multi-agent reinforcement learning," *arXiv preprint arXiv:1802.05438*, 2018.

[22] A. L. Strehl, L. Li, and M. L. Littman, "Reinforcement learning in finite MDPs: PAC analysis," *Journal of Machine Learning Research*, vol. 10, pp. 2413–2444, 2009.

[23] A. L. Strehl, L. Li, E. Wiewiora, J. Langford, and M. L. Littman, "Pac model-free reinforcement learning," in *Proceedings of the 23rd international conference on Machine learning*, pp. 881–888, ACM, 2006.

[24] A. M. Fink, "Equilibrium in a stochastic n-person game," *Journal of Science of the Hiroshima University, series AI (Mathematics)*, vol. 28, no. 1, pp. 89–93, 1964.

[25] M. Kearns and S. Singh, "Near-optimal reinforcement learning in polynomial time," *Machine Learning*, vol. 49, no. 2-3, pp. 209–232, 2002.

[26] R. I. Brafman and M. Tennenholtz, "R-max a general polynomial time algorithm for near-optimal reinforcement learning," *Journal of Machine Learning Research*, vol. 3, pp. 213–231, 2002.

Ashkan Zehfroosh received the M.S. degree in mechanical engineering from Iran University of Science and Technology, Tehran, Iran, in 2014 and the B.S. degree in mechanical engineering from Semnan University, Semnan, Iran, in 2012. Starting from 2015, He is pursuing the Ph.D. degree in mechanical engineering at University of Delaware, DE, USA. His research interest includes machine learning, statistical learning theory, human-robot interaction, decision making and control.

Bert Tanner (SM'08) received his Ph.D. in mechanical engineering from the NTUA, Athens, Greece, in 2001. He was a postdoctoral researcher at the University of Pennsylvania from 2001 to 2003, and subsequently took a position as an assistant professor at the University of New Mexico. In 2008 he joined the Department of Mechanical Engineering at the University of Delaware, where he is currently a professor, and director of the Center for Autonomous and Robotics Systems. Tanner received NSF's Career award in 2005. He is a fellow of the ASME, and a senior member of IEEE. He has served in the editorial boards of the IEEE Transactions on Automatic Control, the IEEE Robotics and Automation Magazine and the IEEE Transactions on Automation Science and Engineering. He is currently an associate editor for Automatica, and Nonlinear Analysis Hybrid Systems. He has also been serving in several conference editorial boards of both IEEE Control Systems and IEEE Robotics and Automation Societies.