# Learning Nash Equilibria in Zero-Sum Stochastic Games via Entropy-Regularized Policy Approximation

**List of content areas: Multi-agent Learning, Reinforcement Learning.**


## Abstract

We explore the use of policy approximations to reduce the computational cost of learning Nash equilibria in zero-sum stochastic games. We propose a new Q-learning type algorithm that uses a sequence of entropy-regularized soft policies to approximate the Nash policy during the Q-function updates. We prove that under certain conditions, by updating the regularized Q-function, the algorithm converges to a Nash equilibrium. We also demonstrate the proposed algorithm's ability to transfer previous training experiences, enabling the agents to adapt quickly to new environments. We provide a dynamic hyper-parameter scheduling scheme to further expedite convergence. Empirical results applied to a number of stochastic games verify that the proposed algorithm converges to the Nash equilibrium, while exhibiting a major speed-up over existing algorithms.


## 1 Introduction

Stochastic Games (SG) [Owen, 1982] is a widely adopted framework to extend reinforcement learning [Sutton and Barto, 1998] to multiple-agent scenarios. The resulting multi-agent reinforcement learning (MARL) framework assumes a group of autonomous agents that choose actions independently and interact with each other to reach an equilibrium [Busoniu *et al.*, 2008]. When all agents are rational, the most natural solution concept is the one of a Nash Equilibrium (NE) [Nash, 1951]. The difficulty of extending single-agent RL methods to learn the NE stems from the fact that the interactions among agents result in a non-stationary environment and thus make learning computationally expensive and difficult to stabilize [Matignon *et al.*, 2012].

Several approaches have been proposed to learn policies in multi-agent scenarios. One set of algorithms learn directly a NE using computationally demanding operators, such as Minimax-Q [Littman, 1994] and Nash-Q [Hu and Wellman, 2003]. Agents adopting these algorithms follow more rational policies but in doing so, they tend to take more time to learn these policies. For example, Minimax-Q needs to solve a linear program to compute the Nash equilibrium at each Q-function update. Even with the help of neural networks, the number of linear programs solved during the learning process still grows with the cardinality of the state and action spaces, which makes Minimax-Q infeasible for large games.

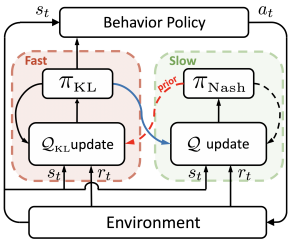



Figure 1: A schematic of the SNQ2L algorithm.

Other popular approaches that extend single-agent RL [Bertsekas and Tsitsiklis, 1996] to multi-agent scenarios include: simplifying the interactions [Foerster *et al.*, 2018b; Bowling and Veloso, 2001; Littman, 2001], or introducing extra mechanisms in the game [Lowe *et al.*, 2017; Li *et al.*, 2019; Foerster *et al.*, 2018a]. Such algorithms allow the agents to compute "fast" but less rational policies, but are unlikely to converge to a rational equilibrium, in general.

The recent works on entropy-regularized soft Q-learning [Fox *et al.*, 2016] provide an efficient way to approximate Nash policies. Specifically, the two-agent Soft-Q algorithm [Grau-Moya *et al.*, 2018] avoids the use of the expensive linear optimizations to update the Q-function. The two agents, instead, compute closed-form soft-optimal policies under an entropy regularization that explores the policy space close to the given priors. Due to fixed regularization, however, the generated policies of the two-agent Soft-Q may be far from a NE. Consequently, there is a need for algorithms that learn the NE, yet in a computationally efficient manner, that can be used in a wide variety of multi-agent situations.

To achieve convergence to NE while maintaining computational efficiency, we propose a novel adaptive entropy-regularized Q-learning algorithm for zero-sum games, referred to as Soft Nash $Q^2$-learning (SNQ2L) [1]. The proposed algorithm learns two different Q-values: a standard Q-value and an entropy-regularized soft Q-value. The two values are used asynchronously in a "feedback" learning mechanism: the soft policies act as an efficient approximation of the Nash policies to update the standard Q-value; the Nash policies

[1] SNQ2 refers to the non-learning version of the algorithm, which has access to the game dynamics.

from the standard Q-function are computed periodically to update the priors that guide the soft policies (see Figure 1). Consequently, SNQ2L reduces the frequency of using the expensive Minimax operator, and thus expedites convergence to the NE. Since the balance between the prior update frequency and computational efficiency plays a critical role in the performance of the SNQ2L algorithm, we also introduce a dynamic scheduling scheme that adaptively changes the prior update frequency. The proposed algorithm has the potential of transferring previous experiences to new environments by incorporating a prior to "warm start" the learning process.

**Contributions:** The contributions of this paper can be summarized as: (1) we propose a new algorithm for multi-agent games and prove the convergence of the proposed algorithm to a NE; (2) we demonstrate a major speed-up in convergence to a NE over existing algorithms; (3) we provide a dynamic scheduling scheme of hyper-parameters to further expedite convergence; and (4) we demonstrate the ability of the algorithm to transfer previous experience to a new environment.

## 2 Background

**Two-agent Zero-sum Stochastic Games.** In two-agent stochastic games two agents, henceforth referred to as the Player (pl) and the Opponent (op), respectively, interact in the same stochastic environment. In this paper we are interested, in particular, in zero-sum games where one of the agent's gain is the other agent's loss [Filar and Vrieze, 2012]. A zero-sum stochastic game $\mathcal{G}$ is formalized by the tuple $\mathcal{G} = \langle \mathcal{S}, \mathcal{A}^{\text{pl}}, \mathcal{A}^{\text{op}}, \mathcal{T}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ denotes a finite state space, and $\mathcal{A}^{\text{pl}}$ and $\mathcal{A}^{\text{op}}$ are the finite action spaces. To choose actions, the Player uses a (Markovian) policy $\pi^{\text{pl}} : S \times \mathcal{A}^{\text{pl}} \to [0, 1]$ and the Opponent uses $\pi^{\text{op}} : S \times \mathcal{A}^{\text{op}} \to [0, 1]$, which together produce the next state according to the state transition function $\mathcal{T} : \mathcal{S} \times \mathcal{S} \times \mathcal{A}^{\text{pl}} \times \mathcal{A}^{\text{op}} \to [0, 1]$. As a consequence of simultaneously taking these actions, the agents receive a reward $\mathcal{R} : \mathcal{S} \times \mathcal{A}^{\text{pl}} \times \mathcal{A}^{\text{op}} \to [R_{\min}, R_{\max}]$. The constant $\gamma \in (0, 1)$ is the discount factor. For zero-sum games, the Player seeks to maximize the total expected reward, whereas the Opponent seeks to minimize it. We denote the *value* at each state induced by the policy pair $\pi = (\pi^{\text{pl}}, \pi^{\text{op}})$ as $\mathcal{V}^{\pi}(s) = \mathbb{E}^{\pi}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) | s_0 = s\right]$. We denote by $\Pi$ the set of all admissible policy pairs.

**Nash Equilibrium.** Operating at a Nash equilibrium, no agent can gain by unilaterally deviating from her policy. Formally, at a NE $(\pi^{\text{pl}\star}, \pi^{\text{op}\star})$, at each state $s \in \mathcal{S}$ satisfies

$$\mathcal{V}^{\pi^{\text{pl}}, \pi^{\text{op}\star}}(s) \leq \mathcal{V}^{\pi^{\text{pl}\star}, \pi^{\text{op}\star}}(s) \leq \mathcal{V}^{\pi^{\text{pl}\star}, \pi^{\text{op}}}(s),$$

for all admissible policies $\pi^{\text{pl}}, \pi^{\text{op}}$. Even though multiple Nash equilibria could exist in a zero-sum SG, the minimax (optimal) value at a NE is *unique* and can be computed [Von Neumann *et al.*, 2007] via $\mathcal{V}^*(s) = \max_{\pi^{\text{pl}}} \min_{\pi^{\text{op}}} \mathcal{V}^{\pi^{\text{pl}}, \pi^{\text{op}}}(s) = \min_{\pi^{\text{op}}} \max_{\pi^{\text{pl}}} \mathcal{V}^{\pi^{\text{pl}}, \pi^{\text{op}}}(s)$.

## 3 Sequential Policy Approximations

The SNQ2L algorithm uses sequential policy approximations to relieve the computational burden seen in the Minimax-Q algorithm. In this section, we introduce the baseline SNQ2 algorithm in a competitive MDP (cMDP) setting. That is, we assume the game $\mathcal{G}$ is known. Later on, we will extend the algorithm for RL problems where $\mathcal{G}$ is only partially known. The proposed algorithm uses entropy-regularized soft policies to approximate the Nash equilibrium at each iteration. We prove the convergence of such algorithm to a NE in the Supplementary material. In the process, we define four operators that are useful in presenting the proposed algorithm.

### 3.1 Shapley's Method

Shapley's method [Filar and Vrieze, 2012] is an iterative algorithm commonly used to solve the Nash equilibrium of a zero-sum stochastic game [2]. Each iteration of the algorithm consists of two operations.

**Step 1.** Compute the Nash policies at each state given the Q-function estimate at the current iteration $t$, by solving a linear program [Filar and Vrieze, 2012]. For example, the maximizing Player has the following optimization problem,

$$\begin{aligned} \max \quad & v \\ \text{subject to} \quad & v\mathbb{1}^{\mathsf{T}} - \boldsymbol{\pi}^{\text{pl}}(s)^{\mathsf{T}} \boldsymbol{\mathcal{Q}}_t(s) \leq 0 \\ & \mathbb{1}^{\mathsf{T}} \boldsymbol{\pi}^{\text{pl}}(s) = 1, \;\; \boldsymbol{\pi}^{\text{pl}}(s) \geq 0, \end{aligned} \tag{1}$$

where $\boldsymbol{\pi}^{\text{pl}}(s)$ is the policy in vector form and $\boldsymbol{\mathcal{Q}}_t(s)$ is the Q-matrix at state $s$. The Nash policy of the Opponent can be solved in a similar manner. We denote the solutions to these optimization problems as $\boldsymbol{\pi}^{\text{pl}}_{\text{Nash},t}(s)$ and $\boldsymbol{\pi}^{\text{op}}_{\text{Nash},t}(s)$, respectively. Performing the optimizations at all states $s$ one may define the overall operation via the operator $\Gamma_{\text{Nash}} : Q \to \Pi$ as

$$(\pi^{\text{pl}}_{\text{Nash},t}, \pi^{\text{op}}_{\text{Nash},t}) = \Gamma_{\text{Nash}}\, \mathcal{Q}_t. \tag{2}$$

**Step 2.** Given a policy pair $\pi$, update the Q-function via

$$\begin{aligned} \mathcal{Q}_{t+1}(s, a^{\text{pl}}, a^{\text{op}}) &= \mathcal{R}(s, a^{\text{pl}}, a^{\text{op}}) \\ &+ \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}(s'|s, a^{\text{pl}}, a^{\text{op}})\, \boldsymbol{\pi}^{\text{pl}}(s')^{\mathsf{T}} \boldsymbol{\mathcal{Q}}_t(s') \boldsymbol{\pi}^{\text{op}}(s'). \end{aligned} \tag{3}$$

Similarly, we define the standard Q-function update operator as $\Gamma_1 : Q \times \Pi \to Q$. The Shapley's Method utilizes the following operator to iterate on the Q-functions.

$$\mathcal{B}(\mathcal{Q}) = \Gamma_1(\mathcal{Q}, \Gamma_{\text{Nash}}\mathcal{Q}). \tag{4}$$

Theorem 1 [Filar and Vrieze, 2012] is the foundation for the convergence of both Shapley's method and Minimax-Q.

**Theorem 1.** *The operator $\mathcal{B}$ is a sup-norm contraction mapping with a contraction factor of $\gamma$. The fixed point of $\mathcal{B}$ is the Q-function corresponding to a Nash equilibrium.*

### 3.2 Entropy-Regularized Policy Approximation

Entropy-regularized policy approximation [Fox *et al.*, 2016] was originally introduced to reduce the maximization bias commonly seen in learning algorithms. This idea was later extended to two-agent scenarios and is referred to as the two-agent Soft-Q algorithm [Grau-Moya *et al.*, 2018]. Two-agent Soft-Q introduces two fixed entropy-regulation terms to the reward structure and thus restricts the policy exploration to

[2] Minimax-Q is an asynchronous learning version of Shapley's method.

a neighborhood of the given priors. Given a policy pair $(\pi^{\text{pl}}, \pi^{\text{op}})$, the soft-value function is defined as

$$\mathcal{V}_{\text{KL}}^{\pi^{\text{pl}},\pi^{\text{op}}}(s) = \mathbb{E}_s^{\pi^{\text{pl}},\pi^{\text{op}}}\Bigg[\sum_{t=0}^{\infty}\gamma^t\Big(\mathcal{R}(s_t,a_t^{\text{pl}},a_t^{\text{op}}) - \frac{1}{\beta^{\text{pl}}}\log\frac{\pi^{\text{pl}}(a_t^{\text{pl}}|s_t)}{\rho^{\text{pl}}(a_t^{\text{pl}}|s_t)} - \frac{1}{\beta^{\text{op}}}\log\frac{\pi^{\text{op}}(a_t^{\text{op}}|s_t)}{\rho^{\text{op}}(a_t^{\text{op}}|s_t)}\Big)\Bigg], \tag{5}$$

where $\beta^{\text{pl}} > 0$ and $\beta^{\text{op}} < 0$ are inverse temperatures, and $\rho^{\text{pl}}$ and $\rho^{\text{op}}$ are priors for the Player and the Opponent.

The objective of the two-agent Soft-Q algorithm is to find the optimal solution to the max-min optimization problem $\max_{\pi^{\text{pl}}}\min_{\pi^{\text{op}}}\mathcal{V}_{\text{KL}}(s;\pi^{\text{pl}},\pi^{\text{op}})$. It has been shown in [Grau-Moya *et al.*, 2018] that the optimal solution can be found through an iterative algorithm [3].

**Step 1.** With the current soft Q-function $\mathcal{Q}_{\text{KL},t}$, construct the soft optimal policy for the Player through the closed-form solution

$$\pi_{\text{KL},t}^{\text{pl}}(a^{\text{pl}}|s) = \frac{1}{Z^{\text{pl}}(s)}\rho^{\text{pl}}(a^{\text{pl}}|s)\exp\left[\beta^{\text{pl}}\mathcal{Q}_{\text{KL},t}^{\text{pl}}(s,a^{\text{pl}})\right], \tag{6}$$

where $Z^{\text{pl}}(s)$ is the normalization factor and $\mathcal{Q}_{\text{KL},t}^{\text{pl}}$ is a marginalization through a log-sum-exp function

$$\mathcal{Q}_{\text{KL},t}^{\text{pl}}(s,a^{\text{pl}}) = \frac{1}{\beta^{\text{op}}}\log\sum_{a^{\text{op}}}\rho^{\text{op}}(a^{\text{op}}|s)\exp\left[\beta^{\text{op}}\mathcal{Q}_{\text{KL},t}(s,a^{\text{pl}},a^{\text{op}})\right]. \tag{7}$$

The soft-optimal policy of the Opponent can be obtained in a similar manner. One can then define the soft optimal policy generating operator $\Gamma_{\text{KL}}^{\beta}: Q\times\Pi\to\Pi$ as

$$(\pi_{\text{Nash},t}^{\text{pl}},\pi_{\text{Nash},t}^{\text{op}}) = \Gamma_{\text{KL}}^{\beta}(\mathcal{Q}_{\text{KL},t},\rho). \tag{8}$$

**Step 2.** With the marginalization in (7), compute the optimal value at state $s$ via

$$\mathcal{V}_{\text{KL},t}(s) = \frac{1}{\beta^{\text{pl}}}\log\sum_{a^{\text{pl}}}\rho^{\text{pl}}(a^{\text{pl}}|s)\exp\left[\beta^{\text{pl}}\mathcal{Q}_{\text{KL},t}^{\text{pl}}(s,a^{\text{pl}})\right] \tag{9}$$

One can then update the Q-function through

$$\mathcal{Q}_{\text{KL},t+1}(s,a^{\text{pl}},a^{\text{op}}) = R(s,a^{\text{pl}},a^{\text{op}}) + \gamma\sum_{s'\in\mathcal{S}}\mathcal{T}(s'|s,a^{\text{pl}},a^{\text{op}})\mathcal{V}_{\text{KL},t}(s'). \tag{10}$$

We denote the soft update operator $\Gamma_2^{\beta}: Q\times\Pi\to Q$ as

$$\mathcal{Q}_{\text{KL},t+1} = \Gamma_{\text{KL}}^{\beta}(\mathcal{Q}_{\text{KL},t},\rho). \tag{11}$$

Note that, due to the fixed regularization, the two-agent Soft-Q algorithm does not converge to a NE in general.

### 3.3 Sequential Policy Approximation Algorithm

In Algorithm 1, we present the baseline SNQ2 in the cMDP settings, where the game $\mathcal{G}$ is known. Different from the two-agent Soft-Q with a fixed entropy regularization, the proposed algorithm decreases $\beta$ over time. As a result, a sequence of policy approximations of different levels of regularization is applied, and thus enables the algorithm to converge to a NE.

**Algorithm 1:** Baseline SNQ2 in the cMDP Settings

**Inputs:** Game tuple $\mathcal{G} = \langle\mathcal{S},\mathcal{A}^{\text{pl}},\mathcal{A}^{\text{op}},\mathcal{T},\mathcal{R},\gamma\rangle$;
Set $\mathcal{Q}(s,a^{\text{pl}},a^{\text{op}}) = \mathcal{Q}_{\text{KL}}(s,a^{\text{pl}},a^{\text{op}}) = 0$;
Set $\beta^{\text{pl}}$ and $\beta^{\text{op}}$ to some large values;
**while** *$\mathcal{Q}$ and $\mathcal{Q}_{KL}$ not converged* **do**
  Update Nash policies: $\pi_{\text{Nash}} \leftarrow \Gamma_{\text{Nash}}(\mathcal{Q})$;
  Update soft optimal policies: $\pi_{\text{KL}} \leftarrow \Gamma_{\text{KL}}^{\beta}(\mathcal{Q}_{\text{KL}},\pi_{\text{Nash}})$;
  Update Q-function: $\mathcal{Q}_{\text{-}} \leftarrow \Gamma_1(\mathcal{Q},\pi_{\text{KL}})$;
  Update soft Q-function: $\mathcal{Q}_{\text{KL-}} \leftarrow \Gamma_2^{\beta}(\mathcal{Q}_{\text{KL}},\pi_{\text{Nash}})$;
  $\mathcal{Q} \leftarrow \mathcal{Q}_{\text{-}}$, $\mathcal{Q}_{\text{KL}} \leftarrow \mathcal{Q}_{\text{KL-}}$;
  Reduce inverse temperature $\beta$
**end**
**return** $\pi_{\text{Nash}}$ and $\mathcal{Q}(s,a^{\text{pl}},a^{\text{op}})$.

Within the while loop, the algorithm first computes the Nash policies under the current estimate of the Q-function, and it uses the Nash policies as the priors to generate the soft policies. The soft policies are then used to update the Q-functions.

To prove the convergence of the baseline SNQ2 algorithm, we define the operations within the while loop in Algorithm 1 as $\Gamma^{\beta}:\bar{\mathcal{Q}}\to\bar{\mathcal{Q}}$, such that

$$\Gamma^{\beta}\bar{\mathcal{Q}} = \Gamma^{\beta}\begin{bmatrix}\mathcal{Q}\\ \mathcal{Q}_{\text{KL}}\end{bmatrix} = \begin{bmatrix}\Gamma_1(\mathcal{Q},\Gamma_{\text{KL}}^{\beta}(\mathcal{Q}_{\text{KL}},\Gamma_{\text{Nash}}\mathcal{Q}))\\ \Gamma_2^{\beta}(\mathcal{Q}_{\text{KL}},\Gamma_{\text{Nash}}\mathcal{Q})\end{bmatrix}.$$

The baseline SNQ2 algorithm differs from the standard iterative algorithms, in the sense that the operator $\Gamma^{\beta}$ changes at every iteration, as the inverse temperature pair $\beta = (\beta^{\text{pl}},\beta^{\text{op}})$ decreases to zero. As a result, we first present our theorem regarding the convergence of sequentially applying a family of contraction mappings. The proof of Theorem 2 can be found in Appendix B.

**Theorem 2.** *Let $(\mathcal{X},\rho)$ be a complete metric space, let $f_n:\mathcal{X}\to\mathcal{X}$ be a family of $\rho$-contractions, such that, for all $n=1,2,\ldots$ there exists $d_n\in(0,1)$, such that $\rho(f_nx,f_ny)\le d_n\,\rho(x,y)$ for all $x,y\in\mathcal{X}$. Assume that $\lim_{n\to\infty}d_n = d\in(0,1)$. Let $x\in\mathcal{X}$, and let $x^{(n)} = f_n\cdots f_1x$ be the result of sequentially applying the operators $f_1,\ldots,f_n$ to $x$. If the sequence of operators $\{f_n\}_{n=1}^{\infty}$ converges pointwise to $f$, then $f$ is also a $\rho$-contraction mapping with contraction factor $d$. Furthermore, if $x^{\star}$ is the fixed point of $f$, then, for every $x\in\mathcal{X}$, $\lim_{n\to\infty}x^{(n)} = x^{\star}$.*

We now present the convergence theorem of Algorithm 1.

**Theorem 3.** *Under certain regularity assumptions, by sequentially applying $\Gamma^{\beta}$ and gradually decreasing $\beta$ to zero, Algorithm 1 converges to a Nash equilibrium.*

*Proof.* We provide a sketch of the proof here. The complete proof can be found at the supplementary material in the Appendix. We first show that under certain regularity assumptions, the operator $\Gamma^{\beta}$ is a family of contraction mapping. We then show that as $\beta$ approaches zero, the operator

[3] Derivations of the relevant formulas below can be found in the appendix.

$\Gamma^\beta$ converges pointwise to an operator, which has the concatenated Nash Q-function $[\mathcal{Q}^\star_{\text{Nash}}, \mathcal{Q}^\star_{\text{Nash}}]^\mathsf{T}$ as its unique fixed point. Then, by Theorem 2, we conclude that Algorithm 1 converges to a Nash equilibrium by sequentially applying $\Gamma^\beta$ to some initial Q-function and decreasing $\beta$ to zero. □

This baseline SNQ2 serves as a demonstration of the sequential policy approximation idea behind the SNQ2 algorithm. It does not relieve the computational burden of computing Nash policies, as we update the Nash policies at every iteration in the while loop. Due to the simplicity of the baseline algorithm, we can demonstrate its mechanism within the space limit. The standard SNQ2, on the other hand, only applies $\Gamma_{\text{Nash}}$ periodically. We present the standard SNQ2 algorithm and its convergence proof in the appendix.

# 4 Two-Agent Soft Nash Q$^2$-Learning

In this section, we present the two-agent Soft Nash Q$^2$-Learning algorithm (SNQ2L). This algorithm is an (asynchronous) learning version of the standard SNQ2 in Algorithm 1. Specifically, this algorithm does not have access to the dynamics of the game and uses stochastic approximations to estimate the expectation operator in (5). This SNQ2L algorithm only updates the Nash policies periodically according to a dynamic schedule, and thus reduces the computations required to find the Nash equilibrium.

**Learning Rule for $\mathcal{Q}_{\text{KL}}$.**

$$\mathcal{Q}_{\text{KL},t+1}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) \leftarrow (1-\eta_t)\, \mathcal{Q}_{\text{KL},t}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) + \eta_t \Big[\mathcal{R}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) + \gamma \mathcal{V}_{\text{KL},t}(s_{t+1})\Big]. \tag{12}$$

Here, $t$ is the learning step, and $\mathcal{V}_{\text{KL},t}(s_{t+1})$ is computed as in (9) using the current $\mathcal{Q}_{\text{KL},t}$ estimate. The priors used in (9) are updated periodically using the Nash policies as indicated by the red arrow in Figure 1.

**Learning Rule for $\mathcal{Q}$.**

$$\mathcal{Q}_{t+1}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) \leftarrow (1-\alpha_t)\, \mathcal{Q}_t(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) + \alpha_t \left[\mathcal{R}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) + \gamma \mathcal{V}_t(s_{t+1})\right]. \tag{13}$$

To compute the estimated optimal value $\mathcal{V}_t(s_{t+1})$, SNQ2L uses either soft policies or Nash policies. We summarize SNQ2L with dynamic schedule in Algorithm 2.

**Algorithm 2:** SNQ2-Learning Algorithm

**Inputs:** Priors $\rho$, Learning rates $\alpha$ and $\eta$; initial prior update episode $M = \Delta M_0$; Nash update frequency $T$;
Set $\mathcal{Q}(s, a^{\text{pl}}, a^{\text{op}}) = \mathcal{Q}_{\text{KL}}(s, a^{\text{pl}}, a^{\text{op}}) = 0$;
Set $\beta^{\text{pl}}$ and $\beta^{\text{op}}$ to some large values;
**while** $\mathcal{Q}$ *not converged* **do**
  **while** *episode* $i$ *not end* **do**
    Compute $\pi_{\text{KL}}(s_t) \leftarrow \left[\Gamma^\beta_{\text{KL}}(\mathcal{Q}_{\text{KL}}, \rho)\right](s_t)$;
    Collect transition $(s_t, a_t^{\text{pl}}, a_t^{\text{op}}, r_t, s_{t+1})$ where $a_t^{\text{pl}} \sim \boldsymbol{\pi}^{\text{pl}}_{\text{KL}}(s_t)$, $a_t^{\text{op}} \sim \boldsymbol{\pi}^{\text{op}}_{\text{KL}}(s_t)$;
    **if** $t$ *mod* $T$ == *0* **then**
      Compute $\mathcal{V}(s_{t+1}) = \max_{\pi^{\text{pl}}} \min_{a^{\text{op}}} \sum_{a^{\text{pl}}} \mathcal{Q}(s_{t+1}, a^{\text{pl}}, a^{\text{op}}) \pi^{\text{pl}}(a^{\text{pl}}|s_{t+1})$;
    **else**
      Compute $\mathcal{V}(s_{t+1}) = \boldsymbol{\pi}^{\text{pl}}_{\text{KL}}(s_{t+1})^\mathsf{T}\, \boldsymbol{\mathcal{Q}}(s_{t+1})\, \boldsymbol{\pi}^{\text{op}}_{\text{KL}}(s_{t+1})$ ;
    **end**
    Update $\mathcal{Q}(s_t, a_t^{\text{pl}}, a_t^{\text{op}})$ with $\mathcal{V}(s_{t+1})$ via (13);
    Update $\mathcal{Q}_{\text{KL}}(s_t, a_t^{\text{pl}}, a_t^{\text{op}})$ via (12);
  **end**
  **if** $i$ == $M$ **then**
    Compute $\pi_{\text{Nash}} \leftarrow \Gamma_{\text{Nash}} \mathcal{Q}_t$;
    Update priors $\rho_{\text{new}} \leftarrow \pi_{\text{Nash}}$;
    Update schedule as in Algorithm 3:
      $\Delta M, \beta_{\text{new}} = DS(\rho_{\text{new}}, \rho, \beta, \Delta M, \mathcal{Q})$;
    Update next prior update schedule $M$ += $\Delta M$;
    Update priors $\rho \leftarrow \rho_{\text{new}}$, $\beta \leftarrow \beta_{\text{new}}$;
    Decrease learning rates $\alpha$ and $\eta$;
  **end**
**end**
**return** $\mathcal{Q}(s, a^{\text{pl}}, a^{\text{op}})$.

## 4.1 Scheduling of the Hyper-parameters $M$ and $\beta$

As presented in Algorithm 2, the trade-off between computational efficiency and approximation accuracy is embedded in the prior update frequency $\Delta M$ and the Nash update frequency $T$. Intuitively, when the new Nash priors are close to the old ones, the algorithm is close to convergence. In this situation, the prior update frequency should be decreased (increase $\Delta M$) and the algorithm should trust and exploit the priors (decrease $\beta$).

The default number of episodes between two Nash prior policy updates $\Delta M_0$ and the default decay rate of the inverse temperature $\lambda \in (0,1)$ are given initially as

$$\Delta M_0 = \frac{N_{\text{states}} \times N_{\text{action pairs}}}{\alpha_0 \times T_{\max}}, \qquad \lambda = \left(\frac{\beta_{\text{end}}}{\beta_0}\right)^{1/N_{\text{updates}}},$$

where $\alpha_0$ is the initial learning rate and $T_{\max}$ is the maximum length of a learning episode; $\beta_0$ and $\beta_{\text{end}}$ are the initial and estimated final magnitude for both $\beta^{\text{op}}$ and $\beta^{\text{pl}}$, and $N_{\text{updates}}$ is the estimated number of prior updates. This value of $\Delta M_0$ allows the algorithm to properly explore the state-action pairs so that the first prior update is performed with an informed Q-function. In our numerical experiments we found that $\beta_0 = 20$, $\beta_{\text{end}} = 0.1$ and $N_{\text{updates}} = 10$ are a good set of values.

Algorithm 3 summarizes the dynamic scheduling scheme, where the parameter $\sigma \in (0,1)$ is a decrease factor and the *RelativeDifference* captures the performance difference between old and new priors. We demonstrate the performance boost due to dynamic scheduling in Section 5.5.

## 4.2 Warm-Starting

One can warm-start the proposed SNQ2L algorithm by first initializing the priors $\rho^{\text{pl}}$ and $\rho^{\text{op}}$ based on some previously learnt policies or expert demonstrations, and then also postponing the first prior update to exploit these priors. Using a prior policy instead of the value to warm-start the algorithm has two major advantages: first, human demonstrations can be converted into prior policies in a more streamlined manner than into value information; second, prior policies provide more consistent guidance than the values, as the priors are not updated till the first prior update. We demonstrate the effectiveness of transferring previous experience in Section 5.4.

**Algorithm 3:** Dynamic Schedule for $M$ and $\beta$

**Inputs:** old and new priors $\rho$, $\rho_{\text{new}}$; old prior update length $\Delta M$; old inverse temperatures $\beta$; current $\mathcal{Q}$;
Compute $\mathcal{V}_{\text{old}}(s) = [\boldsymbol{\rho}^{\text{pl}}(s)]^{\top} \boldsymbol{\mathcal{Q}}(s) \boldsymbol{\rho}^{\text{op}}(s)$ and $\mathcal{V}_{\text{new}}(s) = \boldsymbol{\rho}^{\text{pl}}_{\text{new}}(s)^{\top} \boldsymbol{\mathcal{Q}}(s) \boldsymbol{\rho}^{\text{op}}_{\text{new}}(s)$, for all $s$;
Compute $\textit{RelativeDifference}(s) = |\mathcal{V}_{\text{new}}(s) - \mathcal{V}_{\text{old}}(s)| / |\mathcal{V}_{\text{new}}(s)|$;
Count the number of states $n$, where $\textit{RelativeDifference}(s) < \delta$;
**if** $n/|\mathcal{S}| \geq$ *Threshold* **then**
  $\Delta M = \min\{\Delta M/\sigma, \Delta M_{\max}\}$, $\beta_{\text{new}} = \max\{\lambda\, \beta_{\text{old}}, \beta_{\min}\}$;
**else**
  $\Delta M = \max\{\sigma\, \Delta M, \Delta M_{\min}\}$, $\beta_{\text{new}} = \beta_{\text{old}}$;
**end**
**return** $\Delta M$, $\beta_{\text{new}}$.

# 5 Numerical Experiments

To evaluate the performance of the proposed algorithm, we tested and compared SNQ2L with four existing algorithms (Minimax-Q [Littman, 1994], Two-agent Soft-Q [Grau-Moya *et al.*, 2018], WoLF-PHC [Bowling and Veloso, 2001], Single-Q [Tan, 1993]) for three zero-sum game environments: a Soccer game as in [Littman, 1994], a two-agent Pursuit-Evasion Game (PEG) [Guan *et al.*, 2020] and a sequential Rock-Paper-Scissor game (sRPS).

## 5.1 Evaluation Criteria

Two metrics were used to evaluate the performance of different algorithms: the number of states achieving a NE and the running time[4]. For each game, we computed four different values and compared them to determine whether a state has achieved a NE: (a) the ground truth Nash value $\mathcal{V}_{\text{Nash}}$ solved exactly via Shapley's method; (b) the learnt value $\mathcal{V}_{\text{L}}$; (c) the one-sided MDP value $\mathcal{V}^{\text{pl}}_{\text{L}}$ computed by fixing the Opponent to her learnt policy and letting the Player maximize; (d) the one-sided MDP value $\mathcal{V}^{\text{op}}_{\text{L}}$. We assume that the learnt policies achieve a NE at state $s$, if the values at state $s$ satisfy

$$\max\left\{\frac{\left|\mathcal{V}_{\text{Nash}}(s)-\mathcal{V}_{\text{L}}(s)\right|}{\left|\mathcal{V}_{\text{Nash}}(s)\right|}, \frac{\left|\mathcal{V}_{\text{Nash}}(s)-\mathcal{V}^{\text{pl}}_{\text{L}}(s)\right|}{\left|\mathcal{V}_{\text{Nash}}(s)\right|}, \frac{\left|\mathcal{V}_{\text{Nash}}(s)-\mathcal{V}^{\text{op}}_{\text{L}}(s)\right|}{\left|\mathcal{V}_{\text{Nash}}(s)\right|}\right\} < \epsilon.$$

Notice that the evaluation criterion above is stricter than the case of only collecting empirical win rates of different agents competing in the environment as in [Littman, 1994; Lagoudakis and Parr, 2002], since any deviation from the NE could be exploited by either agent in our criteria.

## 5.2 The Game Environments

**Pursuit-Evasion.** The pursuit-evasion game (PEG) is played on three grids of different sizes, as depicted in Figure 2. Two agents simultaneously choose one of four directions on each turn. In the default setting, an agent has a 60% success rate of moving to its intended cell and a 40% chance of landing in the cell to the left of the intended direction. The 4×4 and 8×8 PEGs use the default transitions and the 6×6 PEG uses deterministic transitions. In such setting, a deterministic policy cannot be a Nash policy.

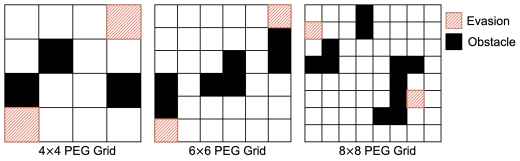



Figure 2: The grids of the three pursuit-evasion games.

**Soccer.** The soccer game [Littman, 1994] is played on a 4×5 grid as in Figure 3. Two agents, A and B, can choose one of five actions on each turn: *N, S, E, W,* and *Stand*, and the two actions selected are executed in random order. The circle represents the ball. When the agent with the ball steps in to the goal (left for A and right for B), that agent scores and the game restarts at a random state. When an agent executes an action that takes it to the cell occupied by the other agent, possession of the ball goes to the stationary agent. Littman argued that the Nash policies must be stochastic.

**Sequential Rock-Paper-Scissor.** In a sequential Rock-Paper-Scissor game (sRPS), two agents play the RPS game repeatedly. The sRPS game ends when one of the agents wins four consecutive RPS games. The states and transitions[5] are shown in Figure 3. The Nash policies of sPRS at each state are uniform for both agents.

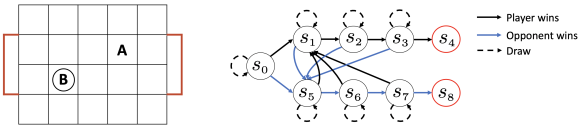



Figure 3: The grid of the soccer game (left), and the state transition diagram of sRPS (right).

## 5.3 Comparison to Existing Algorithms

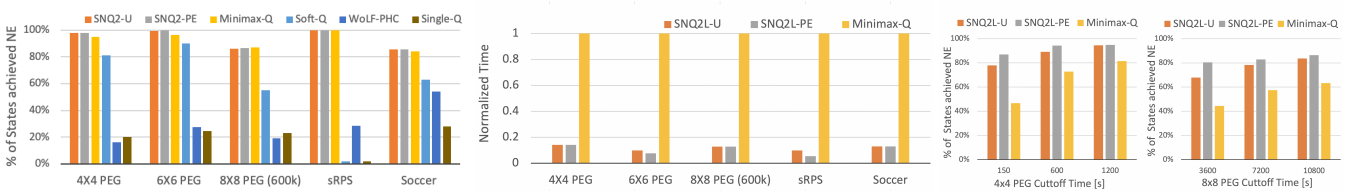
We evaluate SNQ2L on the game environments in Section 5.2 using the evaluation criteria in Section 5.1. The SNQ2L algorithm can be initialized with two types of priors, uniform[6] (SNQ2L-U) and previous experience (SNQ2L-PE). Previous experience for PEGs and Soccer is learnt in a training session of the same game but with a different dynamics. For sRPS, the previous experience is a perturbed uniform strategy. The algorithm also has the option of a fixed schedule (SNQ2L-FS) and a dynamic schedule (SNQ2L-DS). Unless otherwise specified, SNQ2L uses a dynamic schedule. SNQ2L was compared with four popular existing algorithms: Minimax-Q, two-agent Soft-Q, WoLF-PHC and Single-Q. We fine-tuned these existing algorithms to get the best performance so as to demonstrate their actual capabilities. We compare the convergence performance of the algorithms in Figure 4.

The sRPS game with a uniform Nash equilibrium shows that SNQ2L is capable of converging to a fully mixed strategy. In this game, two-agent Soft-Q learning fails to converge

[4]The single-threaded experiments were run in a Python environment with AMD Ryzen 1920x and 32G 2666MHz RAM. Scipy's optimization `linprog` (a simplex method) is used to solve the matrix games at each state.

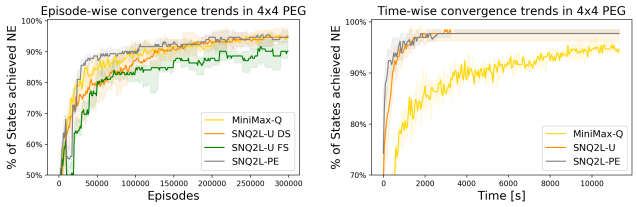
[5]State $s_0$ corresponds to the initial state where no one has won a single RPS game, and states $s_4$ and $s_8$ are the terminal states.

[6]For sRPS, the default prior is randomly generated, as uniform policies are the Nash policy for this game.




(a) Comparison of performance at convergence (b) Comparison of computation time (c) Comparison of performance at cutoff

Figure 4: All results are averaged over ten runs. The computation time in (b) is normalized by that of Minimax-Q. We cut off the computation at 600k episodes for 8×8 PEG, due to the large state space. In most of the experiments, SNQ2L achieves a slightly better convergence to NE than Minimax-Q, while exhibiting an significant reduction in computation time. Two-agent Soft-Q, Single-Q and WoLF-PHC are fast but fail to converge to NE in all five games hence are not shown in (b). Note that two-agent Soft-Q fails completely in sRPS.




(a) (b)

Figure 5: Convergence trends of different algorithms in 4×4 PEG.

to the uniform NE, as its reward structure is regularized as in (5); Single-Q learning tries to learn a pure policy but does not converge; WoLF-PHC converges to NE at state $s_3$ and $s_7$ but with large value deviation, which propagates to other states and results in poor policies.

We then examine the 4×4 PEG game. Despite the relatively small state space, the stochastic transition at all non-terminal states requires extensive exploration of the environment. We plot the convergence trends over 300k episodes and over 12k seconds for Minimax-Q and SNQ2L in Figure 5.

The time-wise convergence trends in Figure 5 demonstrate the expedited convergence of SNQ2L in terms of computation time. The episode-wise trend plot shows that SNQ2L maintains the same level of convergence to NE as Minimax-Q, albeit with significantly reduced computation time. This shows that our adaptive entropy-regularized policy approximation approach is both accurate and efficient.

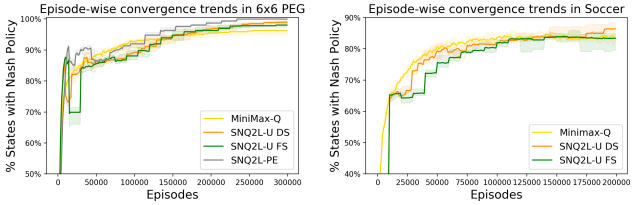
### 5.4 Effectiveness of Warm-Starting

In PEGs, the agents learn their previous experience on the same grid but with a 75% success rate. To reduce overfitting to the previous environment, the prior policy fed to the agent for a new training session is the average of the previously learnt policy and a uniform policy. As seen in Figure 5(b), previous experience does not shorten the time till convergence but instead significantly reduces the time to reach a reasonable performance. We plot the cutoff time performance of the 4×4 and 8×8 PEGs in Figure 4(c). In the 4×4 example the time to reach over 90% Nash convergence is halved from 1.2k seconds with uniform prior down to 600 seconds with previous experience. In the 8×8 example the time to 80% convergence was halved from 7.2k seconds to 3.6k seconds.

One also observes from Figure 5(a) that the policies generated by SNQ2L with previous experience converge, episode-wise, slightly faster than Minimax-Q. This "warm-start" feature has appealing real-world applications, where the number of episodes interacting with the environment is the main constraint instead of computation time. In this case, one can first train prior policies using a simulator and use these as priors to the agents. With SNQ2L one can train the agents to reach a reasonable level of performance in fewer episodes, while maintaining a relatively low computation overhead.

### 5.5 Effectiveness of Dynamic Scheduling

Figure 5(a) demonstrates that the dynamic scheduling reduces the number of episodes till convergence. For 6×6 PEG and Soccer, which have less stochasticity, dynamic scheduling improves the performance of the algorithm at convergence, as seen in Figure 6. Dynamic scheduling also reduces the number of episodes that SNQ2L exploits a potentially bad prior when the Q-values are far from convergence, especially at the beginning. As the initial learning rate is large, a Q-value based on a bad prior policy could be difficult to correct later on. The proposed dynamic scheduling also reduces the prior policy update frequency when the algorithm is close to convergence and thus reduces oscillations, as in Figure 5.




(a) (b)

Figure 6: Convergence trends in 6×6 PEG and Soccer.

## 6 Conclusions and Future Work

We have proposed a novel algorithm for solving zero-sum games where the agents use entropy-regularized policies to approximate the Nash policies and thus reduce computation time till convergence. We proved that under certain conditions, the proposed algorithm converges to a Nash equilibrium. We conducted numerical experiments on multiple games of different sizes and levels of stochasticity. The simulation results demonstrated that the algorithm significantly reduces computation time when compared to existing algorithms. We also demonstrated the effectiveness of warm-starting in a new environment using previous experience. One interesting direction of future work is to extend the current algorithm to general-sum games.

## References


[Bertsekas and Tsitsiklis, 1996] Dimitri P Bertsekas and John N Tsitsiklis. *Neuro-Dynamic Programming*. Athena Scientific, 1996.

[Bowling and Veloso, 2001] Michael Bowling and Manuela Veloso. Rational and convergent learning in stochastic games. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 1021–1026, Seattle, WA, Aug. 4–10, 2001.

[Busoniu *et al.*, 2008] L. Busoniu, R. Babuska, and B. De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics*, 38(2):156–172, 2008.

[Filar and Vrieze, 2012] Jerzy Filar and Koos Vrieze. *Competitive Markov Decision Processes*. Springer, 2012.

[Foerster *et al.*, 2018a] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. *AAAI Conference on Artificial Intelligence*, 2018.

[Foerster *et al.*, 2018b] Jakob N. Foerster, Richard Y. Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 122–130, 2018.

[Fox *et al.*, 2016] Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. In *Proceedings of the 32nd Conference on Uncertainty in Artificial Intelligence*, pages 202–211, 2016.

[Grau-Moya *et al.*, 2018] Jordi Grau-Moya, Felix Leibfried, and Haitham Bou-Ammar. Balancing two-player stochastic games with soft q-learning. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 268–274, 2018.

[Guan *et al.*, 2020] Yue Guan, Dipankar Maity, Christopher Kroninger, and Panagiotis Tsiotras. Bounded-rational pursuit-evasion games. *Preprint arXiv:2003.06954*, 2020.

[Hu and Wellman, 2003] Junling Hu and Michael P Wellman. Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research*, 4:1039–1069, 2003.

[Lagoudakis and Parr, 2002] Michail G. Lagoudakis and Ronald Parr. Value function approximation in zero-sum markov games. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 283–292, 2002.

[Li *et al.*, 2019] Shihui Li, Yi Wu, Xinyue Cui, Honghua Dong, Fei Fang, and Stuart Russell. Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:4213–4220, 2019.

[Littman, 1994] Michael L. Littman. Markov games as a framework for multi-agent reinforcement learning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 157–163, 1994.

[Littman, 2001] Michael L. Littman. Friend-or-foe Q-learning in general-sum games. In *Proceedings of the Eighteenth International Conference on Machine Learning*, pages 322–328, 2001.

[Lowe *et al.*, 2017] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6382–6393, 2017.

[Matignon *et al.*, 2012] Laetitia Matignon, Guillaume Laurent, and Nadine Fort-Piat. Independent reinforcement learners in cooperative Markov games: A survey regarding coordination problems. *The Knowledge Engineering Review*, 27:1 – 31, 2012.

[Nash, 1951] John Nash. Non-cooperative games. *Annals of Mathematics*, 54(2):286–295, 1951.

[Owen, 1982] Guillermo Owen. *Game Theory*. Academic Press, 1982.

[Sutton and Barto, 1998] Richard S. Sutton and Andrew G. Barto. *Introduction to Reinforcement Learning*. MIT Press, Cambridge, MA, USA, 1st edition, 1998.

[Tan, 1993] Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the 10th International Conference on Machine Learning*, pages 330–337, 1993.

[Von Neumann *et al.*, 2007] John Von Neumann, Oskar Morgenstern, and Harold William Kuhn. *Theory of Games and Economic Behavior*. Princeton University Press, 2007.