

**LEARNING NASH EQUILIBRIA IN ZERO-SUM STOCHASTIC GAMES VIA  
ENTROPY-REGULARIZED POLICY APPROXIMATION**

A Thesis  
Presented to  
The Academic Faculty

By

Qifan Zhang

In Partial Fulfillment  
of the Requirements for the Degree  
Master of Science in the  
College of Computing

Georgia Institute of Technology

August 2020

© Qifan Zhang 2020

**LEARNING NASH EQUILIBRIA IN ZERO-SUM STOCHASTIC GAMES VIA  
ENTROPY-REGULARIZED POLICY APPROXIMATION**

Thesis committee:

Dr. Panagiotis Tsiotras  
School of Aerospace Engineering  
*Georgia Institute of Technology*

Dr. Ayanna Howard  
College of Computing  
*Georgia Institute of Technology*

Dr. Matthew Gombolay  
College of Computing  
*Georgia Institute of Technology*

Date approved: July 24, 2020

## ACKNOWLEDGMENTS

I would like to thank Dr. Panagiotis Tsiotras for offering this thesis opportunity as well as his guidance and support. Thanks to Dr. Ayanna Howard for co-advising this work, and Dr. Matthew Gombolay for giving helpful information. I also appreciate the help from Yue Guan and the discussions with everyone in DCSL lab.

## TABLE OF CONTENTS

<b>Acknowledgments</b> . . . . .	iii
<b>List of Figures</b> . . . . .	vi
<b>List of Acronyms</b> . . . . .	vii
<b>Chapter 1: Introduction</b> . . . . .	1
1.1 Introduction . . . . .	1
1.2 Related Work . . . . .	3
<b>Chapter 2: Background</b> . . . . .	5
2.1 Reinforcement Learning for MDPs . . . . .	5
2.2 Single Agent Soft Q-Learning . . . . .	6
2.3 Two-Agent Stochastic Games . . . . .	7
<b>Chapter 3: Methodology</b> . . . . .	11
3.1 Two-Agent Soft Nash $Q^2$ -Learning . . . . .	11
3.1.1 Learning rule for $Q_{KL}$ and Soft-Optimal policies . . . . .	12
3.1.2 Nash Prior Updates . . . . .	15
3.1.3 Soft $Q^2$ -Learning Algorithm . . . . .	17
3.2 Scheduling of the Hyper-parameters . . . . .	19

3.3	Warm-starting the Soft $Q^2$ -Learning Algorithm . . . . .	20
<b>Chapter 4: Numerical Experiments and Discussion . . . . .</b>		<b>22</b>
4.1	Evaluation Criteria . . . . .	22
4.2	Game Design . . . . .	23
4.3	Implementation Detail . . . . .	25
4.4	Comparison to Existing Methods . . . . .	25
4.5	Effect of Warm-Starting . . . . .	28
4.6	Effect of Dynamic Scheduling . . . . .	29
<b>Chapter 5: Conclusion and Future Work . . . . .</b>		<b>30</b>
<b>Appendices . . . . .</b>		<b>31</b>
	Appendix A: Derivation of Single-Agent Soft-Q Learning . . . . .	32
	Appendix B: Derivation of Two-Agent Soft-Q Learning . . . . .	43
<b>References . . . . .</b>		<b>48</b>

## LIST OF FIGURES

3.1	The schematic of the SNQ2 algorithm. . . . .	19
4.1	The grids of the three pursuit-evasion games. . . . .	23
4.2	The grid of the soccer game. . . . .	24
4.3	The state transition diagram of sRPS. . . . .	24
4.4	Comparison of convergence rate of all algorithms, We cut off the computation at 600,000 episodes for $8 \times 8$ PEG, due to the large state space. . . . .	26
4.5	Comparison of computation time for all algorithms. The computation time on the right is normalized by that of Minimax-Q. We cut off the computation at 600,000 episodes for $8 \times 8$ PEG, due to the large state space. . . . .	26
4.6	Episode-wise convergence trends of different algorithms in $4 \times 4$ PEG. . . . .	27
4.7	Time-wise convergence trends of different algorithms in $4 \times 4$ PEG. . . . .	27
4.8	Episode-wise convergence trends of different algorithms in $6 \times 6$ PEG (left) and Soccer (right). . . . .	28
4.9	Cutoff time performance of the $4 \times 4$ (left) and $8 \times 8$ (right) PEGs . . . . .	28

## LIST OF ACRONYMS

**KL** Kullback-Leibler

**M3DDPG** MiniMax Multi-agent Deep Deterministic Policy Gradient

**MADDPG** Multi-agent Deep Deterministic Policy Gradient

**MARL** Multi-Agent Reinforcement Learning

**MDP** Markov Decision Process

**NE** Nash Equilibrium

**PEG** Pursuit-Evasion Game

**RL** Reinforcement Learning

**SG** Stochastic Games

**SNQ2** Soft Nash Q<sup>2</sup>-learning

**sRPS** Sequential Rock-Paper-Scissor Game

## SUMMARY

In this thesis, we explore the use of policy approximation for reducing the computational cost of learning Nash equilibria in Multi-Agent Reinforcement Learning (MARL). Existing multi-agent reinforcement learning methods are either computationally demanding or do not necessarily converge to a Nash Equilibrium (NE) without additional stringent assumptions. We propose a new algorithm for zero-sum stochastic games in which each agent simultaneously learns a Nash policy and an entropy-regularized policy. The two policies help each other towards convergence: the former guides the latter to the desired Nash equilibrium, while the latter serves as an efficient approximation of the former. We demonstrate the possibility of transferring previous training experience to a different environment, which enables the agents to adapt quickly. We also provide a dynamic hyperparameter scheduling scheme for further expedited convergence. Empirical results applied to a number of stochastic games show that the proposed algorithm converges to the Nash equilibrium while exhibiting an order of magnitude speed-up over existing algorithms.

# CHAPTER 1

## INTRODUCTION

### 1.1 Introduction

Both biological and artificial agents (e.g. robots controlled through Artificial Intelligence) tend to follow goal-directed behaviors in order to survive and achieve desired outcomes [1]. Learning to choose actions that maximize rewards is the foundation behind Reinforcement Learning (RL) [2, 3]. While the success of RL has been demonstrated in many single agent domains [4, 5], more work is needed to extend the same results to multi-agent scenarios. The difficulty of extending single-agent RL methods to multi-agent scenarios stems from the fact that the interactions between the agents make learning difficult, since changes in the policy of one agent will affect that of the other agents, and vice versa [6].

One widely adopted framework to address multi-agent systems is via Stochastic Games (SG), where one treats the interactions between all agents as a dynamic game. The resulting MARL framework assumes a group of autonomous agents that share a common environment in which the agents choose actions independently and interact with each other [7] to reach an equilibrium. When all agents are rational, and under the assumption of perfect information, the most natural solution concept is the one of a NE [3, 8, 9].

The blanket assumptions of perfect rationality [10] and perfect information behind NE are too strict for many scenarios, especially those involving humans [11]. In addition, computing NE involving more than a handful of alternative strategies is challenging. There is a need for algorithms that generate rational, yet computationally efficient policies that can be used in a wide variety of multi-agent situations. Several approaches have been proposed to find alternative ways to compute policies in multi-agent scenarios. One popular approach is to simplify the learning problem [12, 13, 14] via providing agents with extra

information during the training phase and thus allow the agents to compute “fast” but less rational policies. Such algorithms are unlikely to converge to a rational (Nash) equilibrium. Other algorithms learn directly a NE using computationally demanding operators [15, 16, 17, 18], such as Minimax-Q [15] and Nash-Q [16], which solve multiple highly-coupled optimization problems at each learning step. Under certain conditions, agents using these algorithms are more logical/rational (in the sense that they try to compute NE policies) but in doing so, these agents tend to take more time to compute the corresponding policies.

Intelligent agents also tend to efficiently utilize prior knowledge to learn new tasks and develop important competencies [19]. Entropy-regularized Q-learning [20] (also referred to as soft Q-learning), originally introduced to address the maximization-bias<sup>1</sup>, has the potential of transferring previous experiences to new environments by incorporating a prior (a reference policy or a previously learnt policy) thus restricting exploration only to policies close to the prior. The soft Q-learning idea was recently extended to two-agent zero-sum and team games [21]. This two-agent Soft-Q algorithm avoids the use of the expensive Nash operator to update the game value at each learning step; the two agents, instead, use the closed-form expressions of the soft-optimal policies. Since none of the agents computes NE policies, the resulting policies in the two-agent Soft-Q may be far from optimal, however.

In this paper, we propose a new entropy-regularized Q-learning algorithm for two-agent zero-sum games, called Soft Nash Q<sup>2</sup>-learning (SNQ2), which combines the two-agent Soft-Q and the Minimax-Q algorithms for more efficient learning. The proposed algorithm learns two different Q-values: the original Q-value and the soft Q-value. The two values are used asynchronously in a certain “feedback” learning mechanism in which soft-Q policies act as an approximation of the Nash policies to update the game Q-value, while

---

<sup>1</sup>Over-estimation (“maximization bias”) of Q-learning refers to the case where the algorithm always favors actions that have positive errors, especially when multiple actions have similar Q-values owing to the use of the max operator. These errors can be introduced from stochastic dynamics or from function approximation. With the entropy-regularization, the resulted policy no longer concentrates on the current maximizing action but instead spreads over multiple actions, which facilitate the exploration.

periodically the Nash policies are used to update the priors in the soft-Q policies of the two agents. Consequently, SNQ2 significantly reduces the frequency of using the expensive Minimax operator, and thus expedites convergence to the Nash equilibrium. Furthermore, the priors used in the soft-Q policy updates provide the opportunity to transfer knowledge from previous training experience or from human experts to a new environment, and thus “warm-start” the learning process. This is demonstrated in the numerical examples section.

Since the balance between the two Q-learning systems plays a critical role in the performance of the SNQ2 algorithm, we also introduce a dynamic scheduling scheme that actively changes the frequency of the updates of the priors with the new Nash policies. We empirically show the convergence of SNQ2 to a Nash equilibrium for several stochastic games demonstrating major speed-up over existing algorithms.

The rest of the paper is organized as follows: Section 1.2 presents and summarizes some of the related work; Chapter 2 discusses some background of RL, soft Q-learning and two-agent zero-sum stochastic games; Chapter 3 presents the proposed SNQ2 algorithm; Chapter 4 demonstrates the convergence and performance of the SNQ2 algorithm by applying it to a number of stochastic games and compares its performance to multiple existing algorithms; Chapter 5 concludes this paper and summarizes the contribution of this work.

## **1.2 Related Work**

Games, first explored in the economics community [22, 23], offer a natural framework to generalize single-agent Markov Decision Process (MDP) [3] to a multi-agent settings. The simplest approach to extend learning in multi-agent settings is to use independently learning agents. This was the approach with the use of Q-learning in [24], but this approach fails in many cases due to the non-stationarity of the environment [6]. Some previous works addressed nonstationarity by introducing an extra mechanism, including centralized critic [13, 25], conjectures on other agents’ policies [12], variable learning rates [26], and

neural fictitious play [27, 28].

Many of these works ensure convergence to a NE only in simple versions of stochastic games, such as repeated games [29]. Other algorithms are designed specifically for cooperative settings, such as, optimistic and hysteretic Q-updates [30, 31, 32], policy parameter sharing [33] and applications to Markov team games [18]. Deep learning techniques, such as deep Q-learning, have also been investigated for multi-agent scenarios; see, for example [34]. While these approaches may be computationally tractable, they are either not applicable in competitive settings or lack guarantees of convergence to a Nash equilibrium, a critical concept in competitive games [8]. Approaches that focus on solving for NE policies on the other hand [15, 16, 17], although principled are, in general, computationally demanding due to the complexity computing a Nash equilibrium [35]. Different methods have been investigated to alleviate this problem. The algorithms in [36, 37] apply a mean-field approximation to model the interactions within a population of agents to that of a single agent and the average effect from the rest of the agent population. The algorithm in [38] uses approximate dynamic programming to reduce complexity. MiniMax Multi-agent Deep Deterministic Policy Gradient (M3DDPG) [39] uses a minimax formulation to extend the popular Multi-agent Deep Deterministic Policy Gradient (MADDPG) [13] algorithm to competitive settings. These algorithms still rely on performing rather expensive minimax operations, and they could still potentially benefit from the ideas utilized in the proposed SNQ2 algorithm.

## CHAPTER 2

### BACKGROUND

#### 2.1 Reinforcement Learning for MDPs

In reinforcement learning an agent interacts with an unknown stochastic environment in order to determine an optimal policy that maximizes the total expected return. When the agent has perfect information about the effect of its interaction with the environment, these problems are best formulated in terms of MDP [3, 40]. Formally, a MDP is defined by the tuple  $\langle \mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma \rangle$ , where  $\mathcal{S}$  is the finite state space,  $\mathcal{A}$  is the finite action space, and  $\mathcal{T} : \mathcal{S} \times \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$  denotes the state transition probability. Namely,  $\mathcal{T}(s'|s, a)$  denotes the probability of being at state  $s'$  at the next time step given that the agent is located at  $s$  and chooses action  $a$  at the current time step, such that  $\sum_{s' \in \mathcal{S}} \mathcal{T}(s'|s, a) = 1$  is enforced for all  $s \in \mathcal{S}$  and  $a \in \mathcal{A}$ . For notational convenience, we will often use the notation  $\mathcal{T}_{ss'}(a)$  in lieu of  $\mathcal{T}(s'|s, a)$ . The reward function is defined as  $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ . Specifically,  $\mathcal{R}(s, a)$  represents the reward the agent collects should she choose action  $a$  at state  $s$ . The discount factor  $\gamma$  depicts the trade-off between current and future rewards. An admissible (mixed) Markov policy is defined as  $\pi : \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ , such that  $\sum_{a \in \mathcal{A}} \pi(a|s) = 1$  for all  $s$ . Since we are dealing with finite action spaces, an admissible policy  $\pi$  can be cast into vector form at each state  $s \in \mathcal{S}$  as follows:

$$\pi(s) = [\pi(a_1|s), \pi(a_2|s), \dots, \pi(a_{|\mathcal{A}|}|s)]^T \in \Delta_{|\mathcal{A}|}. \quad (2.1)$$

where  $a_i \in \mathcal{A}^{\text{pl}}$  for all  $i = 1, \dots, |\mathcal{A}^{\text{pl}}|$ . Here, we use  $\Delta_{|\mathcal{A}|}$  to denote the  $|\mathcal{A}|$ -dimensional standard simplex. Once cast into vector form, the policies  $\pi(s)$  can be interpreted as probability distributions over action space at state  $s$ .

The goal of the agent is to find a policy that maximizes the total discounted reward, i.e.,

to find  $\pi^* = \operatorname{argmax}_{\pi} V^{\pi}(s)$ , where

$$V^{\pi}(s) = \mathbb{E}_{\substack{a_t \sim \pi(a_t|s_t) \\ s_{t+1} \sim \mathcal{T}(s_{t+1}|s_t, a_t)}} \left[ \sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \mid s_0 = s \right], \quad (2.2)$$

and the expectation is taken over the state-action trajectory starting from  $s$  under the policy  $\pi$ . In the sequel, we use the shorthand notation  $V^{\pi}(s) = \mathbb{E}_s^{\pi} [\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t)]$ .

## 2.2 Single Agent Soft Q-Learning

Soft-Q algorithm has been introduced in order to reduce the overestimation problem of standard Q-learning [20] and also for constructing flexible energy-based stochastic policies in continuous domains [41]. Soft-Q approaches modify the original unconstrained optimization problem to a constrained problem as follows:

$$\begin{aligned} \max_{\pi} \quad & \mathbb{E}_s^{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \right], \\ \text{such that} \quad & \mathbb{E}_s^{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t \log \frac{\pi(a_t|s_t)}{\rho(a_t|s_t)} \right] \leq C, \end{aligned} \quad (2.3)$$

where  $C$  restricts the amount the policy  $\pi$  is allowed to deviate from the reference policy  $\rho$  measured in terms of the information<sup>1</sup>. Notice that the expectation of the information cost over the policy  $\pi$  turns out to be the Kullback-Leibler (KL) divergence between  $\pi$  and  $\rho$ . The expectation is taken over all state-action trajectories starting from  $s$  and following the policy  $\pi$ . See also notation in Equation (2.2).

To solve the constrained optimization problem in Equation (2.3), one introduces a Lagrange multiplier  $\beta \geq 0$  to rewrite an equivalent unconstrained problem as follows:

$$\mathcal{V}^*(s) = \max_{\pi} \mathbb{E}_s^{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t \left( \mathcal{R}(s_t, a_t) - \frac{1}{\beta} \frac{\pi(a_t|s_t)}{\rho(a_t|s_t)} \right) \right]. \quad (2.4)$$

---

<sup>1</sup>The KL-divergence between two distributions is the expectation of the information cost (logarithmic difference) between the two distributions.

The soft-optimal policy can be written in closed form as [40]

$$\pi^*(a|s) = \frac{\rho(a|s)e^{\beta Q^*(s,a)}}{\sum_{a \in \mathcal{A}} \rho(a|s)e^{\beta Q^*(s,a)}}, \quad (2.5)$$

where the soft-optimal Q-value is defined as

$$Q^*(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(a) \mathcal{V}^*(s'). \quad (2.6)$$

Incorporating the soft-optimal policy into Equation (2.4), we arrive at the soft-optimal value

$$\mathcal{V}^*(s) = \frac{1}{\beta} \log \sum_{a \in \mathcal{A}} \rho(a|s) e^{\beta Q^*(s,a)}. \quad (2.7)$$

However, for the case where the transition  $\mathcal{T}$  is not available, one cannot compute the policies via exact methods like policy iteration or value iteration. One solution is to apply stochastic approximation [3]. For the Soft-Q algorithm, the soft-optimal  $Q^*(s, a)$  in Equation (2.6) can be learnt using stochastic approximation via the following recursive learning rule [20]:

$$Q_{t+1}(s_t, a_t) \leftarrow (1 - \alpha_t) Q_t(s_t, a_t) + \alpha_t \left[ \mathcal{R}(s_t, a_t) + \frac{\gamma}{\beta} \log \left( \sum_{a \in \mathcal{A}} \rho(a|s_{t+1}) e^{\beta Q_t(s_{t+1}, a)} \right) \right].$$

In Subsection 3.1.1, we discuss the extension of single-agent Soft-Q algorithm to the two-agent case. To facilitate understanding of two-agent learning algorithms, we briefly discuss two-agent stochastic games in the upcoming section.

### 2.3 Two-Agent Stochastic Games

In two-agent stochastic games, two agents, henceforth referred to as the Player and the Opponent, 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.

Formally, a zero-sum stochastic game is defined by the tuple  $\langle \mathcal{S}, \mathcal{A}^{\text{pl}}, \mathcal{A}^{\text{op}}, \mathcal{T}, \mathcal{R}, \gamma \rangle$ , where  $\mathcal{S}$  is the finite state space, and  $\mathcal{A}^{\text{pl}}$  and  $\mathcal{A}^{\text{op}}$  are the finite action spaces of the two agents. The superscripts pl and op denote the Player and the Opponent respectively. Similar to an MDP, the state transition is defined via  $\mathcal{T} : \mathcal{S} \times \mathcal{S} \times \mathcal{A}^{\text{pl}} \times \mathcal{A}^{\text{op}} \rightarrow [0, 1]$ . Namely,  $\mathcal{T}(s'|s, a^{\text{pl}}, a^{\text{op}})$  denotes the probability that the agents' joint state is at state  $s'$  at the next time step, given that the agents are currently at the joint  $s$ , and the Player chooses action  $a^{\text{pl}}$  while the Opponent chooses action  $a^{\text{op}}$ . Finally,  $\mathcal{R} : \mathcal{S} \times \mathcal{A}^{\text{pl}} \times \mathcal{A}^{\text{op}} \rightarrow \mathbb{R}$  represents the reward structure. Specifically, the quantity  $\mathcal{R}(s, a^{\text{pl}}, a^{\text{op}})$  captures the immediate reward the Player collects should she choose action  $a^{\text{pl}}$  and the Opponent chooses action  $a^{\text{op}}$  at state  $s$ . For zero-sum games, the Player seeks to maximize the cumulative reward, whereas the Opponent seeks to minimize it. The Player executes a (mixed) Markov policy  $\pi^{\text{pl}} : \mathcal{S} \times \mathcal{A}^{\text{pl}} \rightarrow [0, 1]$ , where  $\pi(a^{\text{pl}}|s)$  is the probability that the Player will choose action  $a$  when she is at state  $s$ . The policy  $\pi(a^{\text{pl}}|s)$  is admissible, if  $\sum_{a^{\text{pl}} \in \mathcal{A}^{\text{pl}}} \pi(a^{\text{pl}}|s) = 1$  for all states  $s$ . We use  $\Pi^{\text{pl}}$  to denote all the admissible policies of the Player. The policy  $\pi^{\text{op}}$  and admissible policy set  $\Pi^{\text{op}}$  of the Opponent are defined in a similar manner.

We denote the value at each state induced by the policy pair  $(\pi^{\text{pl}}, \pi^{\text{op}})$  as

$$\mathcal{V}^{\pi^{\text{pl}}, \pi^{\text{op}}}(s) = \mathbb{E}_s^{\pi^{\text{pl}}, \pi^{\text{op}}} \left[ \sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) \right],$$

where the expectation is over state and joint-action trajectories starting from  $s$  and following the admissible policy pair  $(\pi^{\text{pl}}, \pi^{\text{op}}) \in \Pi^{\text{pl}} \times \Pi^{\text{op}}$ .

The classical solution to a zero-sum stochastic game is the Nash Equilibrium [23]. According to [15], the player computes its optimal value at state  $s$  as

$$\mathcal{V}^{\text{pl}*}(s) = \max_{\pi^{\text{pl}}} \min_{\pi^{\text{op}}} \mathcal{V}^{\pi^{\text{pl}}, \pi^{\text{op}}}(s), \quad (2.8)$$

while the Opponent computes its optimal value as

$$\mathcal{V}^{\text{op}*}(s) = \min_{\pi^{\text{op}}} \max_{\pi^{\text{pl}}} \mathcal{V}^{\pi^{\text{pl}}\pi^{\text{op}}}(s). \quad (2.9)$$

Notice that the Nash Equilibrium is also a saddle point [8] of the value function for zero-sum games. Namely, if the Nash Equilibrium is achieved by the policy pair  $(\pi^{\text{pl}*}, \pi^{\text{op}*})$ , then we have

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

for all admissible policy pairs  $(\pi^{\text{pl}}, \pi^{\text{op}}) \in \Pi^{\text{pl}} \times \Pi^{\text{op}}$  and all states  $s \in \mathcal{S}$ . The Minimax theorem [42] states that for *zero-sum* stochastic games, there exists a unique value. Namely, in this case we can define

$$\mathcal{V}^*(s) = \mathcal{V}^{\text{pl}*}(s) = \mathcal{V}^{\text{op}*}(s). \quad (2.10)$$

Given the optimal value  $\mathcal{V}^*$ , and similarly to the single agent case, we can define the optimal Q-value for the game at state  $s$  as

$$\mathcal{Q}^*(s, a^{\text{pl}}, a^{\text{op}}) = \mathcal{R}(s, a^{\text{pl}}, a^{\text{op}}) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(a^{\text{pl}}, a^{\text{op}}) \mathcal{V}^*(s'). \quad (2.11)$$

Under the definition of the optimal value in Equation (2.10), a policy pair  $(\pi^{\text{pl}*}, \pi^{\text{op}*})$  is said to form a Nash equilibrium if it achieves that optimal value  $\mathcal{V}^*$ . Note that even though the optimal value  $\mathcal{V}^*$  is unique, there may exist multiple Nash equilibria that achieve the same optimal value [9]. In the zero-sum case, if  $(\tilde{\pi}^{\text{pl}}, \tilde{\pi}^{\text{op}})$  and  $(\tilde{\tilde{\pi}}^{\text{pl}}, \tilde{\tilde{\pi}}^{\text{op}})$  are both Nash equilibria, then  $(\tilde{\tilde{\pi}}^{\text{pl}}, \tilde{\pi}^{\text{op}})$  and  $(\tilde{\pi}^{\text{pl}}, \tilde{\tilde{\pi}}^{\text{op}})$  are also Nash equilibria. As a result, learning algorithms need not distinguish which Nash equilibrium they converge to in the zero-sum case.

The “unique value” property of zero-sum games enables the development of learning algorithms that solves Nash equilibrium of such games in resemblance to Q-learning [43]. The Minimax-Q algorithm [15] learns the optimal Q-value for a zero-sum game by apply-

ing the following recursive learning rule:

$$\begin{aligned} \mathcal{Q}_{t+1}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) \leftarrow & (1 - \alpha_t)\mathcal{Q}_t(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) + \alpha_t \left[ \mathcal{R}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) \right. \\ & \left. + \gamma \max_{\pi^{\text{pl}}} \min_{a^{\text{op}}} \sum_{a^{\text{pl}} \in \mathcal{A}^{\text{pl}}} \mathcal{Q}_t(s_{t+1}, a^{\text{pl}}, a^{\text{op}}) \pi^{\text{pl}}(a^{\text{pl}} | s_{t+1}) \right], \end{aligned} \quad (2.12)$$

or, equivalently,

$$\begin{aligned} \mathcal{Q}_{t+1}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) \leftarrow & (1 - \alpha_t)\mathcal{Q}_t(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) + \alpha_t \left[ \mathcal{R}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) \right. \\ & \left. + \gamma \min_{\pi^{\text{op}}} \max_{a^{\text{pl}}} \sum_{a^{\text{op}} \in \mathcal{A}^{\text{op}}} \mathcal{Q}_t(s_{t+1}, a^{\text{pl}}, a^{\text{op}}) \pi^{\text{op}}(a^{\text{op}} | s_{t+1}) \right]. \end{aligned} \quad (2.13)$$

It has been shown in [15] that under some mild conditions, the estimate  $\mathcal{Q}_t$  from Equation (2.12) or Equation (2.13) converges to the optimal Q-value in Equation (2.11). The Nash policies can then be determined from the optimal Q-value using linear programming [8]. Specifically, the Q-table at each state corresponds to a matrix game. Per Shapley's Theorem [8], the Nash policy pair at each state is the solution to the matrix game defined by the Q-table at that state, which is solvable via a linear program (see also Subsection 3.1.2).

## CHAPTER 3

### METHODOLOGY

#### 3.1 Two-Agent Soft Nash Q<sup>2</sup>-Learning

In this section, we present the two-agent Soft Nash Q<sup>2</sup>-Learning algorithm (SNQ2). As the name suggests, the algorithm relies on two different Q-values: one is the standard Q-value as in Equation (2.11) and the other one, which we denote as  $Q_{\text{KL}}$ , is the soft Q-value for the two-agent games, that is an extension of the single-agent soft Q-value in Equation (2.6). Similarly to the single-agent Soft-Q algorithm, the function  $Q_{\text{KL}}$  allows us to generate a closed-form soft-optimal policy pair that can be used as an approximation to the actual Nash policy pair. We then use the approximated policy pair to update the standard Q-value.

In the following, we first introduce the learning rule for the standard Q-value (denoted  $Q$ ) and then discuss the learning rule for the soft Q-value (denoted  $Q_{\text{KL}}$ ). To find the optimal Q-value  $Q^*(s, a^{\text{pl}}, a^{\text{op}})$  at each state we apply the following update rule recursively [15, 16]

$$Q_{t+1}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) \leftarrow (1 - \alpha_t) Q_t(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) + \alpha_t \left( \mathcal{R}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) + \gamma \mathcal{V}_t(s_{t+1}) \right), \quad (3.1)$$

where  $t$  is the learning step and  $\alpha_t$  is the learning rate. After applying the action pair  $(a_t^{\text{pl}}, a_t^{\text{op}})$ , the agents transit to state  $s_{t+1}$ , while  $\mathcal{V}_t(s_{t+1})$  is the estimated optimal value at state  $s_{t+1}$  based on the Q-values at learning step  $t$ .

When executing the update in Equation (3.1), we employ two different methods to compute the estimated value  $\mathcal{V}_t$ . The first method (used less frequently) resembles the Minimax-Q [15] and computes the Nash value based on the current Q-value estimations as in Equation (3.2).

### First method: Nash Value

$$\begin{aligned}
 \mathcal{V}_t(s) &= \max_{\pi^{\text{pl}}} \min_{a^{\text{op}}} \sum_{a^{\text{pl}}} \mathcal{Q}_t(s, a^{\text{pl}}, a^{\text{op}}) \pi^{\text{pl}}(a^{\text{pl}}|s) \\
 &= \min_{\pi^{\text{op}}} \max_{a^{\text{pl}}} \sum_{a^{\text{op}}} \mathcal{Q}_t(s_t, a^{\text{pl}}, a^{\text{op}}) \pi^{\text{op}}(a^{\text{op}}|s).
 \end{aligned} \tag{3.2}$$

The second method, soft value update, which is used most frequently, utilizes the information from the  $\mathcal{Q}_{\text{KL}}$  and the soft-optimal policies detailed in Subsection 3.1.1, Equation (3.11).

#### 3.1.1 Learning rule for $\mathcal{Q}_{\text{KL}}$ and Soft-Optimal policies

Previous work by Grau-Moya et al [21] has extended Soft-Q algorithm to the two-agent setting. Similarly to the single-agent version of the Soft-Q algorithm, this formulation augments two information costs to the reward structure to penalize each agent’s deviation from a given prior. Given the policies  $\pi^{\text{pl}}$  and  $\pi^{\text{op}}$  for the Player and the Opponent, the soft-value function is defined as

$$\mathcal{V}_{\text{KL}}^{\pi^{\text{pl}}, \pi^{\text{op}}}(s) = \mathbb{E}_s^{\pi^{\text{pl}}, \pi^{\text{op}}} \left[ \sum_{t=0}^{\infty} \gamma^t \left( \mathcal{R}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) - \frac{1}{\beta^{\text{pl}}} \log \frac{\pi^{\text{pl}}(a_t^{\text{pl}}|s_t)}{\rho^{\text{pl}}(a_t^{\text{pl}}|s_t)} - \frac{1}{\beta^{\text{op}}} \log \frac{\pi^{\text{op}}(a_t^{\text{op}}|s_t)}{\rho^{\text{op}}(a_t^{\text{op}}|s_t)} \right) \right], \tag{3.3}$$

where the expectation is taken over joint-action and state trajectories, and where  $\rho^{\text{pl}}$  and  $\rho^{\text{op}}$  are the prior policies for the Player and the Opponent, respectively. It is assumed that each agent knows each other’s prior. Since we focus on the zero-sum case, we assume that the Player is the maximizer so that  $\beta^{\text{pl}} > 0$ , while the Opponent is the minimizer so that  $\beta^{\text{op}} < 0$ .

Later in this section, we demonstrate how we can utilize the two parameters  $\beta^{\text{pl}}$  and  $\beta^{\text{op}}$  to control exploration during the learning process. At the same time, we will actively update the priors  $\rho^{\text{pl}}$  and  $\rho^{\text{op}}$ , and we treat them as some approximation of Nash policies. This gives us the advantage to “warm-start” the algorithm by initializing the priors according to some

previous experience or expert's knowledge.

The Player and the Opponent compute their optimal soft-values at state  $s$  from

$$\mathcal{V}_{\text{KL}}^{\text{op}*}(s) = \min_{\pi^{\text{op}}} \max_{\pi^{\text{pl}}} \mathcal{V}_{\text{KL}}^{\pi^{\text{pl}}, \pi^{\text{op}}}(s).$$

The uniqueness of the optimal soft-value is established in [21]. Thus, we define the unique optimal soft-value as

$$\mathcal{V}_{\text{KL}}^*(s) = \mathcal{V}_{\text{KL}}^{\text{pl},*}(s) = \mathcal{V}_{\text{KL}}^{\text{op},*}(s). \quad (3.4)$$

We then define a state-action function as

$$\mathcal{Q}_{\text{KL}}^*(s, a^{\text{pl}}, a^{\text{op}}) := \mathcal{R}(s, a^{\text{pl}}, a^{\text{op}}) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(a^{\text{pl}}, a^{\text{op}}) \mathcal{V}_{\text{KL}}^*(s'), \quad (3.5)$$

where the policy pair  $(\pi_{\text{KL}}^{\text{pl}*}, \pi_{\text{KL}}^{\text{op}*})$  achieves the optimal soft value given in Equation (3.8) below.

For action selection, neither the Player nor the Opponent directly use  $\mathcal{Q}_{\text{KL}}^*$  as it depends on the action chosen by the other agent, which is not known a priori. Instead, as shown in [21], each agent first computes the marginalized  $\mathcal{Q}_{\text{KL}}^*$  via

$$\mathcal{Q}_{\text{KL}}^{\text{pl},*}(s, a^{\text{pl}}) := \frac{1}{\beta^{\text{op}}} \log \sum_{a^{\text{op}} \in \mathcal{A}^{\text{op}}} \rho^{\text{op}}(a^{\text{op}}|s) \exp(\beta^{\text{op}} \mathcal{Q}_{\text{KL}}^*(s, a^{\text{pl}}, a^{\text{op}})), \quad (3.6a)$$

$$\mathcal{Q}_{\text{KL}}^{\text{op},*}(s, a^{\text{op}}) := \frac{1}{\beta^{\text{pl}}} \log \sum_{a^{\text{pl}} \in \mathcal{A}^{\text{pl}}} \rho^{\text{pl}}(a^{\text{pl}}|s) \exp(\beta^{\text{pl}} \mathcal{Q}_{\text{KL}}^*(s, a^{\text{pl}}, a^{\text{op}})), \quad (3.6b)$$

respectively. With these definitions, we obtain the soft-optimal policies for both the Player and the Opponent, in closed-form, as

$$\pi_{\text{KL}}^{\text{pl},*}(a^{\text{pl}}|s) = \frac{1}{Z^{\text{pl}}(s)} \rho^{\text{pl}}(a^{\text{pl}}|s) \exp(\beta^{\text{pl}} \mathcal{Q}_{\text{KL}}^{\text{pl},*}(s, a^{\text{pl}})), \quad (3.7a)$$

$$\pi_{\text{KL}}^{\text{op},*}(a^{\text{op}}|s) = \frac{1}{Z^{\text{op}}(s)} \rho^{\text{op}}(a^{\text{op}}|s) \exp(\beta^{\text{op}} \mathcal{Q}_{\text{KL}}^{\text{op},*}(s, a^{\text{op}})), \quad (3.7b)$$

where  $Z^{\text{pl}}(s)$  and  $Z^{\text{op}}(s)$  are normalizing factors. The policies in Equation (3.7) take the form of a Boltzmann distribution, where  $\beta^{\text{pl}}$  and  $\beta^{\text{op}}$  play the role of inverse temperatures. As  $\beta^{\text{pl}}$  and  $\beta^{\text{op}}$  go to zero, the soft-optimal policies get closer to the corresponding priors. On the other hand, as the magnitude of  $\beta^{\text{pl}}$  and  $\beta^{\text{op}}$  tend to infinity, the soft-optimal policies become deterministic and the Player (Opponent) chooses the action that maximizes (minimizes) the marginalized Q-value. Essentially, the values of  $\beta^{\text{pl}}$  and  $\beta^{\text{op}}$  reflect whether the algorithm trusts the priors more or the Q-values instead. If the algorithm trusts the priors, the magnitude of  $\beta^{\text{pl}}$  and  $\beta^{\text{op}}$  should be small, so that the soft-optimal policies generated are close to the reference. If the algorithm, instead, trusts the Q-values, then the magnitude of  $\beta^{\text{pl}}$  and  $\beta^{\text{op}}$  should be large, and the agents optimize based on their respective marginalized Q-values.

Based on the soft-optimal policies in Equation (3.7), the soft-optimal value  $\mathcal{V}_{\text{KL}}^*$  can be expressed in closed form as

$$\begin{aligned}\mathcal{V}_{\text{KL}}^*(s) &= \frac{1}{\beta^{\text{pl}}} \log \sum_{a^{\text{pl}}} \rho^{\text{pl}}(a^{\text{pl}}|s) \exp(\beta^{\text{pl}} \mathcal{Q}_{\text{KL}}^{\text{pl},*}(s, a^{\text{pl}})) \\ &= \frac{1}{\beta^{\text{op}}} \log \sum_{a^{\text{op}}} \rho^{\text{op}}(a^{\text{op}}|s) \exp(\beta^{\text{op}} \mathcal{Q}_{\text{KL}}^{\text{op},*}(s, a^{\text{op}})).\end{aligned}\tag{3.8}$$

We learn  $\mathcal{Q}_{\text{KL}}^*(s, a^{\text{pl}}, a^{\text{op}})$  by applying the recursive rule

$$\mathcal{Q}_{\text{KL},t+1}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) \leftarrow (1 - \eta_t) \mathcal{Q}_{\text{KL},t}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) + \eta_t \left( \mathcal{R}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) + \gamma \mathcal{V}_{\text{KL},t}(s_{t+1}) \right),\tag{3.9}$$

where  $\eta_t$  is the learning rate,  $t$  is the learning step, and  $\mathcal{V}_{\text{KL},t}(s_{t+1})$  is computed from equation Equation (3.8) using  $\mathcal{Q}_{\text{KL},t}^{\text{pl}}$  or  $\mathcal{Q}_{\text{KL},t}^{\text{op}}$  from Equation (3.6).

Using  $\mathcal{Q}_{\text{KL}}$  in Equation (3.8) and its corresponding soft-optimal policies in Equation (3.7) we can update the estimated value  $\mathcal{V}_t$  in Equation (3.1) as follows. We first cast the learnt soft-optimal policies  $\pi_{\text{KL},t}^{\text{pl}}$  and  $\pi_{\text{KL},t}^{\text{op}}$  in Equation (3.7) at each learning step  $t$  into vector

form, similar to as in Equation (2.1)

$$\boldsymbol{\pi}_{\text{KL},t}^{\text{pl}}(s) = \left[ \pi_{\text{KL},t}(a_1^{\text{pl}}|s), \pi_{\text{KL},t}(a_2^{\text{pl}}|s), \dots, \pi_{\text{KL},t}(a_{|\mathcal{A}^{\text{pl}}|}^{\text{pl}}|s) \right]^{\text{T}} \in \Delta_{|\mathcal{A}^{\text{pl}}|}, \quad (3.10\text{a})$$

$$\boldsymbol{\pi}_{\text{KL},t}^{\text{op}}(s) = \left[ \pi_{\text{KL},t}(a_1^{\text{op}}|s), \pi_{\text{KL},t}(a_2^{\text{op}}|s), \dots, \pi_{\text{KL},t}(a_{|\mathcal{A}^{\text{op}}|}^{\text{op}}|s) \right]^{\text{T}} \in \Delta_{|\mathcal{A}^{\text{op}}|}. \quad (3.10\text{b})$$

The value at each state can then be interpreted as the expected reward under the joint distribution over the joint action space governed by the policy pair  $(\boldsymbol{\pi}_{\text{KL},t}^{\text{pl}}(s), \boldsymbol{\pi}_{\text{KL},t}^{\text{op}}(s))$ .

To obtain the value of the game at a state  $s$ , one needs to further generate the matrix game at that state, which is defined through the reward matrix  $\mathcal{Q}_t(s) \in \mathbb{R}^{|\mathcal{A}^{\text{pl}}| \times |\mathcal{A}^{\text{op}}|}$ . This matrix is formed via  $[\mathcal{Q}_t]_{ij}(s) = \mathcal{Q}_t(s, a_i^{\text{pl}}, a_j^{\text{op}})$ . Specifically, the  $ij$ -entry of the matrix  $\mathcal{Q}_t(s)$  corresponds to the estimated Q-value at state  $s$  if the Player chooses action  $a_i^{\text{pl}}$  and the Opponent chooses action  $a_j^{\text{op}}$ .

At state  $s$  we use the vectorized soft-optimal policy pair  $(\boldsymbol{\pi}_{\text{KL},t}^{\text{pl}}(s), \boldsymbol{\pi}_{\text{KL},t}^{\text{op}}(s))$  and the current estimate reward matrix  $\mathcal{Q}_t(s)$  to produce the current estimate of the value  $\mathcal{V}_t(s)$  in Equation (3.1) as follows.

### Second method: Soft-Optimal Value

$$\mathcal{V}_t(s) = \boldsymbol{\pi}_{\text{KL},t}^{\text{pl}}(s)^{\text{T}} \mathcal{Q}_t(s) \boldsymbol{\pi}_{\text{KL},t}^{\text{op}}(s), \quad s \in \mathcal{S}. \quad (3.11)$$

When executing the SNQ2 algorithm at a learning step we use either of the two methods Equation (3.2) or Equation (3.11) to estimate the value. The SNQ2 algorithm follows a schedule that determines when each of the methods is selected, as shown in Algorithm 1.

#### 3.1.2 Nash Prior Updates

Aside from updating the value function using the Nash policies and soft-optimal policies to as in Equation (3.2) and Equation (3.11), the proposed SNQ2 algorithm also updates the priors required by the  $\mathcal{Q}_{\text{KL}}$  computation periodically. Given the current Q-value estimation, one can find the Nash policies at each state by solving the following two Linear

Programming problems [8]. For the player, one can solve:

$$\begin{aligned}
& \max && v, \\
\text{subject to} &&& v\mathbb{1}^\top - \boldsymbol{\pi}^{\text{pl}}(s)^\top \boldsymbol{Q}_t(s) \leq 0, \\
&&& \mathbb{1}^\top \boldsymbol{\pi}^{\text{pl}}(s) = 1, \\
&&& \boldsymbol{\pi}^{\text{pl}}(s) \geq 0,
\end{aligned} \tag{3.12}$$

and for the opponent:

$$\begin{aligned}
& \min && u, \\
\text{subject to} &&& u\mathbb{1} - \boldsymbol{Q}_t(s)\boldsymbol{\pi}^{\text{op}}(s) \geq 0, \\
&&& \mathbb{1}^\top \boldsymbol{\pi}^{\text{op}}(s) = 1, \\
&&& \boldsymbol{\pi}^{\text{op}}(s) \geq 0,
\end{aligned} \tag{3.13}$$

where  $\mathbb{1}$  is the vector of proper dimension filled with ones. Once the two linear programs in Equation (3.12) and Equation (3.13) are solved at each state  $s \in \mathcal{S}$ , we have an updated estimate for the Nash value and the Nash policies of the stochastic game at state  $s$  based on the current estimate  $\boldsymbol{Q}_t$ . The two optimal objective values in Equation (3.12) and Equation (3.13) are equal, and they correspond to the unique Nash value at the state  $s$ . We denote the solutions to Equation (3.12) and Equation (3.13) as  $\boldsymbol{\pi}_{\text{Nash},t}^{\text{pl}}(s)$  and  $\boldsymbol{\pi}_{\text{Nash},t}^{\text{op}}(s)$ , respectively, which are used to update the priors.

The frequency of performing Nash prior updates is determined by some schedule. The schedule can significantly affect the performance of the algorithm. We provide a dynamic schedule in Section 3.2 that determines that frequency based on the history of priors policies. The provided dynamic schedule also decides the decay rate for the inverse temperatures  $\beta^{\text{pl}}$  and  $\beta^{\text{op}}$ , which control the level of exploration.

### 3.1.3 Soft $Q^2$ -Learning Algorithm

We summarize the Soft Nash  $Q^2$ -Learning in Algorithm 1. From line 1 to line 6, we initialize the algorithm. If there is no specific prior knowledge available the priors can be set to uniform policies in accordance to the principle of maximum entropy [44, 45]. We use the index  $i$  to denote the episodes and we use  $t$  for the learning steps. The values of  $\beta^{\text{pl}}$  and  $\beta^{\text{op}}$  are initialized to some large values, as the agents have little trust in their initial priors and they need to explore the Q-value and the soft Q-value.

Line 8 to line 27 present the update rules for  $Q$  and  $Q_{\text{KL}}$ . As discussed in Section 3.1, we have two methods to choose from to estimate the value at a state: based on soft-optimal policies or based on the Nash policies. We use the Nash policies to compute the value  $\mathcal{V}_t$  only when the learning step is a multiple of the Nash update frequency  $T$ . At all the other learning steps we use the soft-optimal policies to estimate the value  $\mathcal{V}_t$ .

At episode  $M$ , which is determined dynamically by algorithm 2, the algorithm performs a prior update as in lines line 20 to line 26. The Nash policies are computed for all states using the current  $Q$  estimate, and the priors are replaced by the computed Nash policies. The two inverse temperatures  $\beta^{\text{pl}}$  and  $\beta^{\text{op}}$  are decreased according to a linear schedule, as the algorithm develops more trust in the priors. The algorithm repeats the whole process till convergence.

Figure 3.1 illustrates the proposed algorithm visually: the orange box represents the soft-optimal policy method to update  $Q$ . This method is employed when the learning step is not a multiple of the Nash update frequency  $T$ . At this specific learning step,  $Q_{\text{KL},t}$  generates the soft policy pair. Combined with the estimated  $Q_t$  at learning step 1, the soft policy pair  $\pi_{\text{KL},t}$  is used to update  $Q_{t+1}$ . The same soft policy pair is also used to update  $Q_{\text{KL},t+1}$ . In this case, the update of  $Q$  depends on  $Q_{\text{KL}}$ .

The green box depicts the second method that uses the Nash policies to update  $Q$ . At learning step  $T$  (or multiples of  $T$ ) the Nash policies are computed based on the current estimate  $Q_T$  and are used to update the next estimate  $Q_{T+1}$ . In this case, the updates of  $Q$

---

**Algorithm 1: Soft Nash Q<sup>2</sup>-Learning Algorithm**


---

```

1  Given  $\mathcal{A}^{\text{pl}}, \mathcal{A}^{\text{op}}, \mathcal{S}$  and learning rates  $\alpha$  and  $\eta$ ;
2  Given prior update frequency  $M$  and Nash update frequency  $T$ ;
3  Given the priors  $\rho^{\text{pl}}$  and  $\rho^{\text{op}}$ ;
4  Set  $\mathcal{Q}(s, a^{\text{pl}}, a^{\text{op}}) = \mathcal{Q}_{\text{KL}}(s, a^{\text{pl}}, a^{\text{op}}) = 0$  for all states  $s$  and actions  $a^{\text{pl}}, a^{\text{op}}$ ;
5  Set  $\beta^{\text{pl}}$  and  $\beta^{\text{op}}$  to some large values;
6  Set counters  $i = 0$  and  $t = 0$ ;
7  while  $\mathcal{Q}$  not converged do
8      while episode  $i$  not end do
9          Collect transition  $(s_t, a_t^{\text{pl}}, a_t^{\text{op}}, r_t, s_{t+1})$  where  $a_t^{\text{pl}} \sim \pi_{\text{KL}}^{\text{pl}}(s_t)$ ,
               $a_t^{\text{op}} \sim \pi_{\text{KL}}^{\text{op}}(s_t)$ ;
10         if  $t \bmod T == 0$  then
11             Compute  $\mathcal{V}(s_{t+1}) = \max_{\pi^{\text{pl}}} \min_{a^{\text{op}}} \sum_{a^{\text{pl}}} \mathcal{Q}(s_{t+1}, a^{\text{pl}}, a^{\text{op}}) \pi^{\text{pl}}(a^{\text{pl}} | s_{t+1})$ ;
12         else
13             Compute  $\mathcal{V}(s_{t+1}) = \pi_{\text{KL}}^{\text{pl}}(s_{t+1})^\top \mathcal{Q}(s_{t+1}) \pi_{\text{KL}}^{\text{op}}(s_{t+1})$ ;
14         end
15         Update
               $\mathcal{Q}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) \leftarrow (1 - \alpha) \mathcal{Q}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) + \alpha \left( \mathcal{R}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) + \gamma \mathcal{V}(s_{t+1}) \right)$ ;
16         Compute  $\mathcal{V}_{\text{KL}}(s_{t+1}) = \frac{1}{\beta^{\text{pl}}} \log \sum_{a^{\text{pl}}} \rho^{\text{pl}}(a^{\text{pl}} | s_{t+1}) \exp(\beta^{\text{pl}} \mathcal{Q}_{\text{KL}}^{\text{pl}}(s_{t+1}, a^{\text{pl}}))$ ;
17         Update  $\mathcal{Q}_{\text{KL}}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) \leftarrow$ 
               $(1 - \eta) \mathcal{Q}_{\text{KL}}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) + \eta \left( \mathcal{R}(s_t, a_t^{\text{pl}}, a_t^{\text{op}}) + \gamma \mathcal{V}_{\text{KL}}(s_{t+1}) \right)$ ;
18          $t += 1$ ;
19     end
20     if  $i == M$  then
21         Compute  $\rho_{\text{new}}^{\text{pl}}(s)$  and  $\rho_{\text{new}}^{\text{op}}(s)$  using Nash solver on Equation (3.12)
              and Equation (3.13) based on current  $\mathcal{Q}$  for all states  $s$ ;
22          $\Delta M, \beta_{\text{new}} = \text{DynamicSchedule}(\rho_{\text{new}}^{\text{pl}}, \rho_{\text{new}}^{\text{op}}, \rho^{\text{pl}}, \rho^{\text{op}}, \Delta M, \mathcal{Q})$  as in
              algorithm 2;
23          $M += \Delta M$ ;
24         Update  $\rho^{\text{pl}} \leftarrow \rho_{\text{new}}^{\text{pl}}, \rho^{\text{op}} \leftarrow \rho_{\text{new}}^{\text{op}}, \beta^{\text{pl}} \leftarrow \beta_{\text{new}}^{\text{pl}}, \beta^{\text{op}} \leftarrow \beta_{\text{new}}^{\text{op}}$ ;
25         Decrease learning rates  $\alpha$  and  $\eta$ ;
26     end
27      $i += 1$ ;
28 end
29 return  $\mathcal{Q}(s, a^{\text{pl}}, a^{\text{op}})$ .

```

---

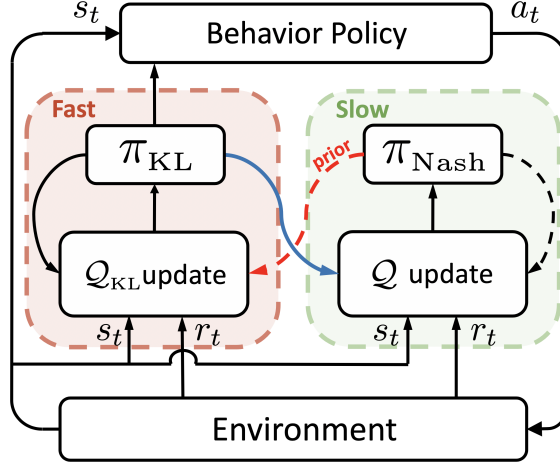


Figure 3.1: The schematic of the SNQ2 algorithm.

and  $Q_{KL}$  are decoupled.

As depicted in this schematic,  $Q_{KL}$  feeds information to  $Q$  within each episode while  $Q$  feeds information to  $Q_{KL}$  during the prior policy updates.

### 3.2 Scheduling of the Hyper-parameters

In this subsection, we provide a dynamic scheduling algorithm for the hyper-parameters  $M$  and  $\beta$  of the SNQ2 algorithm.

As presented in Algorithm 1, the trade-off between computation efficiency and approximation accuracy is embedded in the prior update frequency. 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  $\beta^{op}$  and  $\beta^{pl}$ ).

The default number of episodes between two Nash prior policy updates  $\Delta M_0$  and the default decay rate of the inverse temperature are first computed as

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

where  $\alpha_0$  is the initial learning rate and  $T_{\text{max}}$  is the maximum length of a learning episode;

$\beta_0$  and  $\beta_{\text{end}}$  are the initial and final inverse temperatures for both  $\beta^{\text{op}}$  and  $\beta^{\text{pl}}$ , and  $N_{\text{updates}}$  is the estimated number of prior updates. This value of  $M_0$  allows the algorithm to properly explore the state and action spaces 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 good set of values to use.

The dynamic scheduling scheme is summarized in Algorithm 2, where the parameter  $\sigma \in (0, 1)$  is a decrease factor and *RelativeDifference* captures the performance difference between old and new priors.

---

**Algorithm 2:** Dynamic Schedule for  $M$  and  $\beta$

---

- 1 **Inputs:** old and new priors  $\rho, \rho_{\text{new}}$ ; old prior update length  $\Delta M$ ; old inverse temperatures  $\beta$ ; current  $Q$ ;
  - 2 Compute  $\mathcal{V}_{\text{old}}(s) = [\rho^{\text{pl}}(s)]^T Q(s) \rho^{\text{pl}}(s)$  and  $\mathcal{V}_{\text{new}}(s) = [\rho_{\text{new}}^{\text{pl}}(s)]^T Q(s) \rho_{\text{new}}^{\text{pl}}(s)$ , for all  $s$ ;
  - 3 Compute *RelativeDifference*( $s$ ) =  $|\mathcal{V}_{\text{new}}(s) - \mathcal{V}_{\text{old}}(s)| / |\mathcal{V}_{\text{new}}(s)|$ , for all  $s$ ;
  - 4 Count the number of states where *RelativeDifference*( $s$ )  $< \delta$ ;
  - 5 **if**  $n/|\mathcal{S}| \geq \textit{Threshold}$  **then**
  - 6      $\Delta M = \min\{\frac{1}{\sigma} \Delta M, \Delta M_{\text{max}}\}$ ,  $\beta_{\text{new}} = \max\{\lambda \beta_{\text{old}}, \beta_{\text{min}}\}$ ;
  - 7 **else**
  - 8      $\Delta M = \max\{\sigma \Delta M, \Delta M_{\text{min}}\}$ ,  $\beta_{\text{new}} = \max\{\beta_{\text{old}}, \beta_{\text{min}}\}$ ;
  - 9 **end**
  - 10 **return**  $\Delta M, \beta_{\text{new}}$ .
- 

### 3.3 Warm-starting the Soft Q<sup>2</sup>-Learning Algorithm

As shown in Algorithm 1, one can warm-start the proposed SNQ2 algorithm by first initializing the priors  $\rho^{\text{pl}}$  and  $\rho^{\text{op}}$  based on some previously learnt policies or on expert demonstrations, and then also postponing the first prior update to exploit these priors. This gives the SNQ2 algorithm two major advantages over value-based algorithms such as Minimax-Q and Nash-Q.

First, it is more realistic and easy for human experts to directly provide expert knowledge by demonstrating policies instead of providing the optimal values for each agent. One can easily demonstrate a policy to an autonomous agent, but one can only qualitatively inform the autonomous agent how valuable a specific state is. With data from human demon-

strations, SNQ2 can easily process these data and convert them to reference policies for warm-starting. Value-based algorithms, on the other hand, need to first convert the demonstrations to value-related information, either via Monte-Carlo simulations or via the use of neural networks [46], both of which, in general, require some noticeable computation time. As a result, SNQ2 can incorporate human expert knowledge in a much streamlined and direct manner.

Second, when transferring previous training experience to a new environment, policy provides a longer and more consistent guidance than value, especially in the case where the reward structure is also changed. Value-based algorithms, such as Minimax-Q, are extremely susceptible to reward structure changes when warm-started with learnt values restored from previous learning experience. The reward change could instantaneously be reflected in the values, once the agents start to explore the new environment and use the newly received rewards to update the value function. As the value function quickly gets “corrupted”, the quality of the policies computed based on the value function deteriorates also quickly. The learnt value from previous sessions thus can only provide guidance for a limited number of episodes. On the other hand, SNQ2, if warm-started with previous training experience, can simply delay the initial prior update, which is controlled by the parameter  $M_0$ . With a large  $M_0$ , SNQ2 is able to efficiently and fully explore the state-action values under the consistent guidance of the previous experience, even with a change in the reward structure. With a more informed understanding of the new environment and the new reward structure, the subsequent updates could be performed in a much more efficient manner by SNQ2.

Next, we use numerical experiments to demonstrate the performance improvement resulting from the proposed algorithm in stochastic games over the standard Minimax-Q algorithm [15].

## CHAPTER 4

### NUMERICAL EXPERIMENTS AND DISCUSSION

To evaluate the performance of the proposed algorithm, we tested and compared SNQ2 with several existing algorithms (Minimax-Q [15], WoLF-PHC [26], Single-Q [24]) for three zero-sum game environments: a two-agent Pursuit-Evasion Game (PEG) [45], in which the Pursuer aims to capture the Evader; a Sequential Rock-Paper-Scissor Game (sRPS), in which the two agents play Rock-Paper-Scissor repeatedly till one gets four consecutive wins; and a Soccer game as in [15].

#### 4.1 Evaluation Criteria

Two metrics were used to evaluate the performance of the algorithms: the number of states achieving a Nash Equilibrium and the running time. For each game, we computed four different value functions and compare them to determine whether a state has achieved a NE:

- (1) the Nash value  $\mathcal{V}_{\text{Nash}}$  solved exactly via Shapley’s method [8], this is used as ground truth value;
- (2) the learnt value  $\mathcal{V}_{\text{Learn}}$ ;
- (3) the one-sided MDP value  $\mathcal{V}_{\text{Learn}}^{\text{pl}}$  computed by fixing the Opponent to her learnt policy;
- (4) the one-sided MDP value  $\mathcal{V}_{\text{Learn}}^{\text{op}}$  fixing the Player to her learnt policy.

We consider the learnt policies achieve a NE at state  $s$ , if

$$\frac{|\mathcal{V}_{\text{Nash}}(s) - \mathcal{V}_{\text{Learn}}(s)|}{|\mathcal{V}_{\text{Nash}}(s)|} < \epsilon \bigwedge \frac{|\mathcal{V}_{\text{Nash}}(s) - \mathcal{V}_{\text{Learn}}^{\text{pl}}(s)|}{|\mathcal{V}_{\text{Nash}}(s)|} < \epsilon \bigwedge \frac{|\mathcal{V}_{\text{Nash}}(s) - \mathcal{V}_{\text{Learn}}^{\text{op}}(s)|}{|\mathcal{V}_{\text{Nash}}(s)|} < \epsilon. \quad (4.1)$$

Here, we pick a tolerance of  $\epsilon = 0.03$ . Notice that the evaluation criterion in Equation (4.1) is stricter than simply collecting empirical win rates of different agents competing in the

environment as in [15, 47], since any deviation from the NE could be exploited by either agent.

## 4.2 Game Design

**Pursuit-Evasion Game.** The PEG is played on  $4 \times 4$ ,  $6 \times 6$  and  $8 \times 8$  grids, as depicted in Figure Figure 4.1. Both agents seek to avoid collision with the obstacles (marked in black). The Pursuer strives to capture the Evader by being in the same cell as the Evader. The goal of the Evader is to reach one of the evasion cells (marked in red) without being captured. Two agents simultaneously choose one of four actions on each turn: North, South, East, and West. Each executed action results in a stochastic transition of the agent’s position. In the default setting, an agent has a 60% chance of successfully moving to its intended cell and a 40% chance of landing in the cell to the left of the intended direction. The  $4 \times 4$  and  $8 \times 8$  PEGs use the default transitions and the  $6 \times 6$  PEG uses deterministic transitions. In such setting, a deterministic policy cannot be a Nash policy.

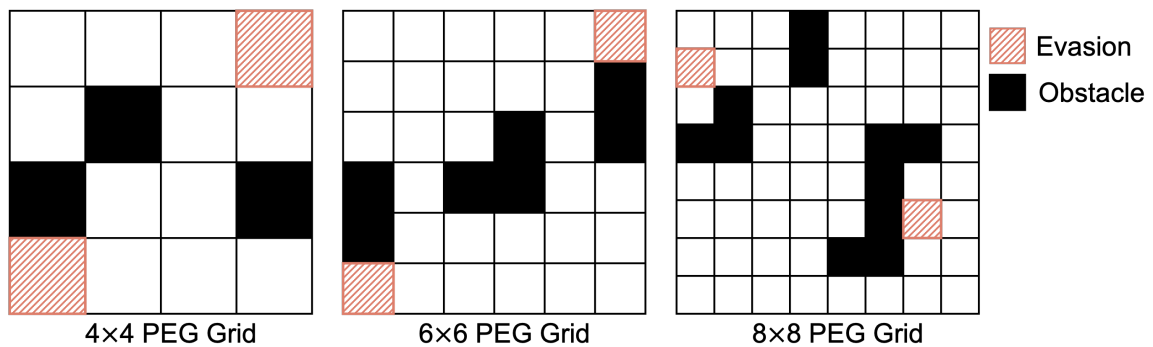


Figure 4.1: The grids of the three pursuit-evasion games.

**Soccer.** The soccer game [15] is played on a  $4 \times 5$  grid as depicted in Figure 4.2. Two agents, A and B, can choose one of five actions on each turn: North, South, East, West, and Stand, and the two actions are executed in random order which makes the game stochastic. The circle represents the “ball”. When the agent with the ball steps in to the goal (left for A and right for B), that player scores and the game restarts. When an agent executes an

action that would take it to the cell occupied by the other agent, possession of the ball goes to the stationary agent. Littman [15] argued that the Nash policies of both agents must be stochastic.

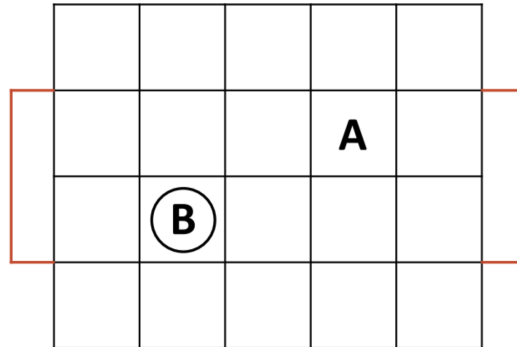


Figure 4.2: The grid of the soccer game.

**Sequential Rock-Paper-Scissor.** In a sequential Rock-Paper-Scissor game (sRPS), two agents (denoted player and opponent) play Rock-Paper-Scissor repeatedly. One episode ends when one of the two agents wins four consecutive games. The states and transitions are shown in Figure 4.3. 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 winning (terminal) states of player and opponent respectively. The Nash policies of sRPS at each state are uniform for both agents.

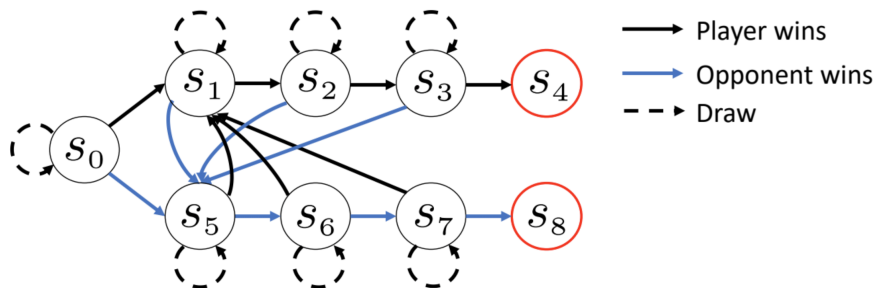


Figure 4.3: The state transition diagram of sRPS.

### 4.3 Implementation Detail

To ensure fair comparison, all the experiments are conducted single-threadedly on a PC with AMD Ryzen 1920x CPU and 32G 2666MHz RAM. The plots are averaged over 10 runs.

The game environments and algorithms are coded in Python. `Numpy` is used for vector operations and `Scipy` [48]’s linear programming package `linprog` is used to solve exact Nash values and Nash policies of the matrix games at each state.

We implemented and evaluated SNQ2 on the game environments presented in Section 4.2 using the evaluation criteria in Section 4.1. The SNQ2 algorithm was initialized with two types of priors, a uniform prior <sup>1</sup> (SNQ2-U) and a previous experience (SNQ2-PE). Previous experience for PEGs 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 (SNQ2-FS) and a dynamic schedule (SNQ2-DS). Unless otherwise specified, SNQ2 uses a dynamic schedule.

We also implemented in Python three popular alternative algorithms, namely, Minimax-Q, Single-Q, and WoLF. We fine-tuned these algorithms to get the best performance so as to demonstrate their actual capabilities.

### 4.4 Comparison to Existing Methods

We summarize the performance in terms of convergence of these algorithms on the five game environments in Figure 4.4 and Figure 4.5. In most of the experiments, SNQ2 achieves a slightly better convergence to NE than Minimax-Q, while exhibiting an significant reduction in computation time. Single-Q and WoLF are fast but fail to converge to NE in all five games hence are not shown in Figure 4.5.

The sRPS game of which the ground truth NE is an uniform policy shows that SNQ2

---

<sup>1</sup>For sRPS, the default prior is randomly generated, as uniform policies are the Nash policy for this game.

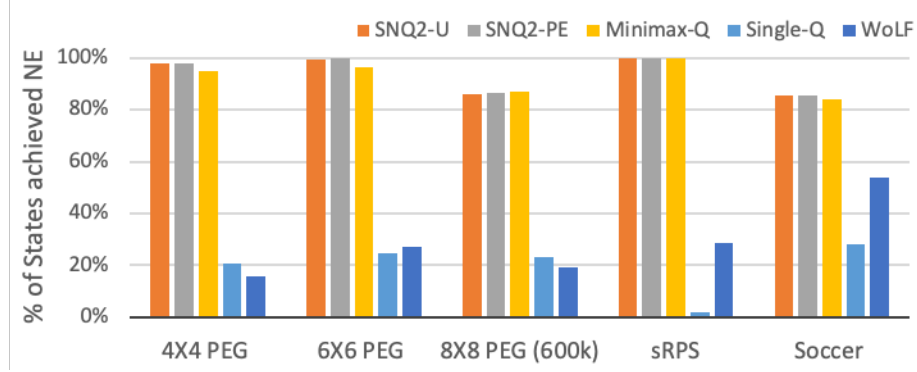


Figure 4.4: Comparison of convergence rate of all algorithms, We cut off the computation at 600,000 episodes for  $8 \times 8$  PEG, due to the large state space.

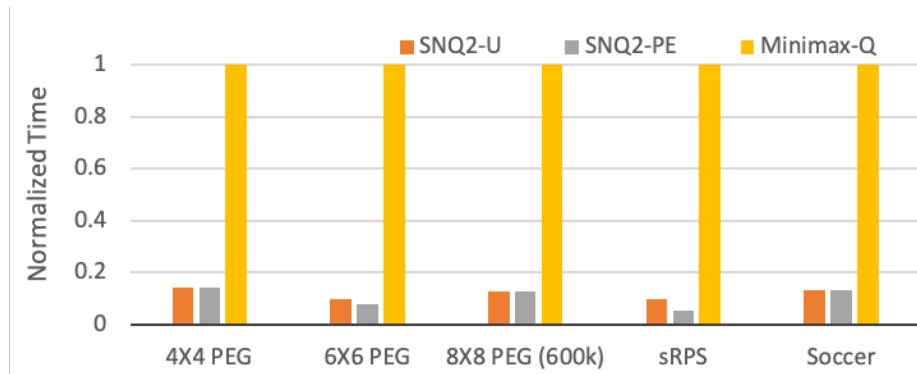


Figure 4.5: Comparison of computation time for all algorithms. The computation time on the right is normalized by that of Minimax-Q. We cut off the computation at 600,000 episodes for  $8 \times 8$  PEG, due to the large state space.

is capable of converging to a fully mixed strategy. In this game, Single-Q learning tries to learn a pure policy but does not converge; WoLF converges to NE at state  $s_3$  and  $s_8$  but with large value deviation, which propagates to other states and results in poor policies.

We then examine the  $4 \times 4$  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 300,000 episodes in Figure 4.6 and over 12,000 seconds for Minimax-Q and SNQ2 in Figure 4.7.

The time to convergence in Figure 4.7 demonstrates the expedited convergence of SNQ2 in terms of computation time. The episode-wise trend is depicted in Figure 4.6, where it is shown that SNQ2 maintains the same level of convergence performance as

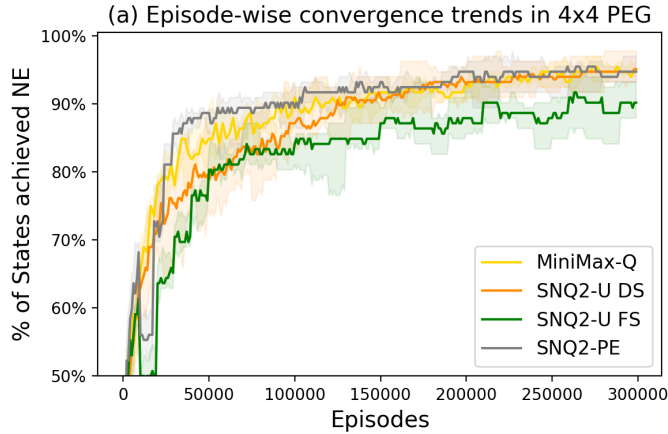


Figure 4.6: Episode-wise convergence trends of different algorithms in  $4 \times 4$  PEG.

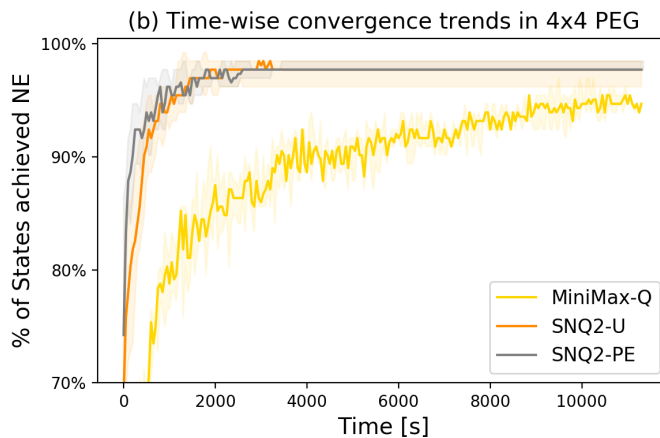


Figure 4.7: Time-wise convergence trends of different algorithms in  $4 \times 4$  PEG.

Minimax-Q, albeit with significantly reduced computation time. This shows that our entropy-regulated policy approximation approach is both accurate and computationally efficient. The performance oscillation of SNQ2 at the beginning is the result of the first few prior updates. As the Q-values are inaccurate at the beginning, the priors generated by these Q-values are also not accurate, which has long-lasting effects on the subsequent value updates as seen in Figure 4.6. Thus, updating the priors based solely on the current Q-values may be detrimental to performance. We have conducted experiments of using Polyak-averaging on the Q-values and the results showed that averaging does indeed alleviate the initial oscillations.

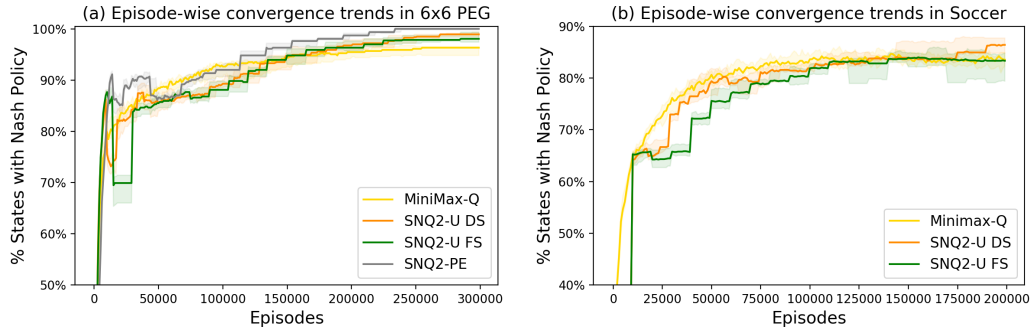


Figure 4.8: Episode-wise convergence trends of different algorithms in  $6 \times 6$  PEG (left) and Soccer (right).

In  $6 \times 6$  deterministic PEG and Soccer game, similar conclusions can be drawn. Though as depicted in Figure 4.8, episode-wisely SNQ2 has similar performance as Minimax-Q, it completely outperforms the later one in terms of computation time till convergence as illustrated in Figure 4.5, due to the large speed boost of close form solution for soft-optimal policy over exact Nash policy through linear programming.

#### 4.5 Effect of Warm-Starting

In PEGs, the agents learn their previous experience in the same grid but with a 75% success rate. To demonstrate the ability of using corrupted prior information, the prior policy fed to the agent for a new training session is the average of the previously learnt policy and a uniform policy.

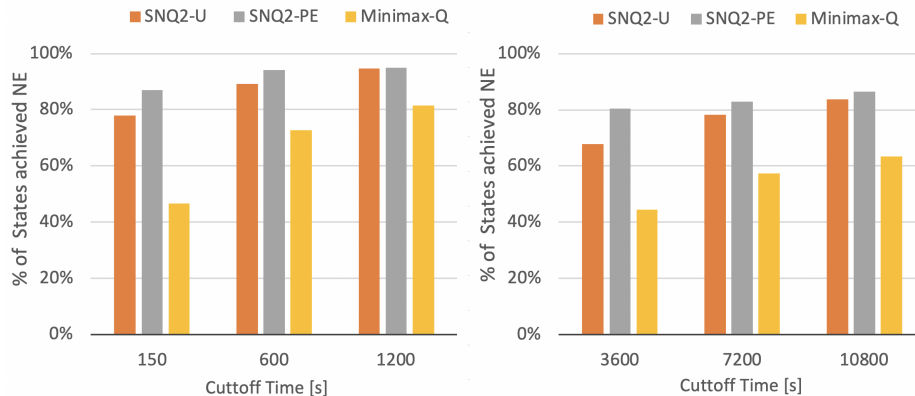


Figure 4.9: Cutoff time performance of the  $4 \times 4$ (left) and  $8 \times 8$ (right) PEGs

One can observe in Figure 4.5 that previous experience does not shorten the time till convergence, however, Figure 4.9 demonstrates that it significantly reduces the time to reach a reasonable performance. In the  $4 \times 4$  example the time to reach over 90% Nash convergence is halved from 1,200 seconds with uniform prior down to 600 seconds with previous experience. In the  $8 \times 8$  example the time to reach 80% convergence was halved from 7,200 seconds to 3,600 seconds.

One also observes from Figure 4.6 that the policies generated by SNQ2 with previous experience converge, episode-wise, slightly faster than Minimax-Q. This “warm-start” feature has appealing real-world learning applications, where the number of episodes is the main constraint instead of computation time. In this case, one can first train prior policies using a simulator and give these as priors to the agents. With SNQ2 one can train the agents to a reasonable performance with fewer episodes, while maintaining relatively low computation overhead.

#### **4.6 Effect of Dynamic Scheduling**

The effectiveness of dynamic scheduling is demonstrated in Figure 4.6 for the case of the  $4 \times 4$  PEG where it is shown that dynamic scheduling improved the number of episodes till convergence. The results for the other examples as shown in Figure 4.8 were similar. Dynamic scheduling reduces the length that SNQ2 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 demonstrated in Figure 4.6.

## CHAPTER 5

### CONCLUSION AND FUTURE WORK

We have proposed a new 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.

Empirically, the proposed algorithm converges to a Nash equilibrium as demonstrated using several stochastic games of different sizes and levels of stochasticity. Our numerical experiments showed that the algorithm significantly reduces computation time while maintaining good performance, when compared with other algorithms, such as Minimax-Q, WoLF-PHC, and independent Q-learning.

The performance of the algorithm can be improved by applying a dynamic scheduling scheme of the prior update frequency and the annealing of inverse temperature. We also demonstrated the effectiveness of warm-starting in a new environment using previous experience.

For future work, we wish to combine SNQ2 with deep learning in order to tackle more complicated games and extend this idea to continuous state and action spaces. We would also like to extend the current algorithm to generalized-sum games and games with more than two agents.

# **Appendices**

**APPENDIX A**  
**DERIVATION OF SINGLE-AGENT SOFT-Q LEARNING**

We first revisit the single-agent Soft-Q Learning formulation, which is represented as the following optimization problem.

$$\max_{\pi} \mathbb{E}^{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \right], \quad (\text{A.1})$$

$$\text{such that } \mathbb{E}^{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t \log \frac{\pi(a_t|s_t)}{\rho(a_t|s_t)} \right] \leq C, \quad (\text{A.2})$$

where  $\mathcal{R}(s, a)$  is the reward collected at state  $s$  should the agent chooses action  $a$  at that state. The agent follows a policy  $\pi$  and uses policy  $\rho$  as its reference policy. For simplicity, we assume that  $\rho$  has positive support for all actions at all states. Otherwise, one can make the action space  $\mathcal{A}$  to be state-dependent. When an action does not have positive support from  $\rho$  at a state, one can exclude it from the action space at that specific state. Then, the information cost in Equation (A.2) is well-defined.

Since we are dealing with a finite action space  $\mathcal{A}$ , the policies  $\pi$  and  $\rho$ , for each  $s \in \mathcal{S}$ , take the form

$$\begin{aligned} \boldsymbol{\pi}(s) &= [\pi(a^1|s), \pi(a^2|s), \dots, \pi(a^{|\mathcal{A}}|s)]^{\text{T}} \in \Delta_{|\mathcal{A}|}(s), \\ \boldsymbol{\rho}(s) &= [\rho(a^1|s), \rho(a^2|s), \dots, \rho(a^{|\mathcal{A}}|s)]^{\text{T}} \in \Delta_{|\mathcal{A}|}(s), \end{aligned}$$

where  $a^i \in \mathcal{A}$  for all  $i = 1, \dots, |\mathcal{A}|$ , and consequently, the policy mapping can be tabulated

via

$$\begin{aligned}\pi &= [\boldsymbol{\pi}(s^1), \boldsymbol{\pi}(s^2), \dots, \boldsymbol{\pi}(s^{|\mathcal{S}|})] \in \Delta_{|\mathcal{A}|}, \\ \rho &= [\boldsymbol{\rho}(s^1), \boldsymbol{\rho}(s^2), \dots, \boldsymbol{\rho}(s^{|\mathcal{S}|})] \in \Delta_{|\mathcal{A}|},\end{aligned}$$

where  $s^i \in \mathcal{S}$  for all  $i = 1, \dots, |\mathcal{S}|$  and where  $\Delta_{|\mathcal{A}|} = \prod_{i=1}^{|\mathcal{S}|} \Delta_{|\mathcal{A}|}(s^i)$ . We denote the set of all admissible policies as  $\Pi$ .

The expectation operator in Equation (A.1) and Equation (A.2) is over all state-action trajectories ensuing by executing the policy  $\pi$  starting at  $s_0 = s$ . Notice that the expectation of the information cost over the policy  $\pi$  turns out to be the Kullback-Leibler (KL) divergence between  $\boldsymbol{\pi}(s)$  and  $\boldsymbol{\rho}(s)$ . Specifically, for each state  $s \in \mathcal{S}$ ,

$$\mathbb{E}_{a \sim \pi} \left[ \log \frac{\pi(a|s)}{\rho(a|s)} \right] = \sum_{a \in \mathcal{A}} \pi(a|s) \log \frac{\pi(a|s)}{\rho(a|s)} = \text{KL}(\boldsymbol{\pi}(s) || \boldsymbol{\rho}(s)).$$

To solve the optimization problem Equation (A.1)-Equation (A.2) we introduce a Lagrangian multiplier  $(-1/\beta)$  to yield an unconstrained optimization problem as follows

$$\max_{\pi} \mathbb{E}^{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \right] - \frac{1}{\beta} \mathbb{E}^{\pi} \left[ \sum_{t=0}^{\infty} \gamma^t \log \frac{\pi(a_t|s_t)}{\rho(a_t|s_t)} \right].$$

As a result of this Lagrangian relaxation, a positive  $\beta$  corresponds to a maximization problem as presented in Equation (A.1). If the optimization problem in Equation (A.1) is instead a minimization problem, the multiplier  $\beta$  should take a negative value. Also, if  $\beta \rightarrow \infty$ , the information cost constraint Equation (A.2) is no longer in effect, as it can be seen from the complementary slackness condition.

Define the  $\pi$ -policy dependent soft value  $\mathcal{V}_{\text{KL}}^\pi$  as

$$\begin{aligned}
\mathcal{V}_{\text{KL}}^\pi(s) &= \mathbb{E}_{s_0=s}^\pi \left[ \sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \right] - \frac{1}{\beta} \mathbb{E}_{s_0=s}^\pi \left[ \sum_{t=0}^{\infty} \gamma^t \log \frac{\pi(a_t|s_t)}{\rho(a_t|s_t)} \right] \tag{A.3} \\
&= \mathbb{E}_{s_0=s}^\pi \left[ \sum_{t=0}^{\infty} \gamma^t \left( \mathcal{R}(s_t, a_t) - \frac{1}{\beta} \log \frac{\pi(a_t|s_t)}{\rho(a_t|s_t)} \right) \right] \\
&= \mathbb{E}_{s_0=s}^\pi [\mathcal{R}(s_0, a_0)] + \mathbb{E}_{s_0=s}^\pi \left[ \sum_{t=1}^{\infty} \gamma^t \mathcal{R}(s_t, a_t) \right] - \frac{1}{\beta} \mathbb{E}_{s_0=s}^\pi \left[ \log \frac{\pi(a_0|s_0)}{\rho(a_0|s_0)} \right] \\
&\quad - \frac{1}{\beta} \mathbb{E}_{s_0=s}^\pi \left[ \sum_{t=1}^{\infty} \gamma^t \log \frac{\pi(a_t|s_t)}{\rho(a_t|s_t)} \right] \\
&= \mathbb{E}_{s_0=s}^\pi \left[ \mathcal{R}(s_0, a_0) - \frac{1}{\beta} \log \frac{\pi(a_0|s_0)}{\rho(a_0|s_0)} \right] + \mathbb{E}_{s_0=s}^\pi \left[ \sum_{t=1}^{\infty} \gamma^t \left( \mathcal{R}(s_t, a_t) - \frac{1}{\beta} \log \frac{\pi(a_t|s_t)}{\rho(a_t|s_t)} \right) \right] \\
&= \sum_{a \in \mathcal{A}} \pi(a|s) \left[ \mathcal{R}(s, a) - \frac{1}{\beta} \log \frac{\pi(a|s)}{\rho(a|s)} \right] \\
&\quad + \sum_{s' \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{T}_{ss'}(a) \mathbb{E}_{s_1=s'}^\pi \left[ \sum_{t=1}^{\infty} \gamma^t \left( \mathcal{R}(s_t, a_t) - \frac{1}{\beta} \log \frac{\pi(a_t|s_t)}{\rho(a_t|s_t)} \right) \right] \\
&= \sum_{a \in \mathcal{A}} \pi(a|s) \left[ \mathcal{R}(s, a) - \frac{1}{\beta} \log \frac{\pi(a|s)}{\rho(a|s)} \right] \\
&\quad + \gamma \sum_{s' \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{T}_{ss'}(a) \mathbb{E}_{s_0=s'}^\pi \left[ \sum_{t=0}^{\infty} \gamma^t \left( \mathcal{R}(s_t, a_t) - \frac{1}{\beta} \log \frac{\pi(a_t|s_t)}{\rho(a_t|s_t)} \right) \right] \\
&= \sum_{a \in \mathcal{A}} \pi(a|s) \left[ \mathcal{R}(s, a) - \frac{1}{\beta} \log \frac{\pi(a|s)}{\rho(a|s)} \right] + \gamma \sum_{s' \in \mathcal{S}} \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{T}_{ss'}(a) \mathcal{V}_{\text{KL}}^\pi(s') \\
&= \sum_{a \in \mathcal{A}} \pi(a|s) \left[ \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(a) \mathcal{V}_{\text{KL}}^\pi(s') - \frac{1}{\beta} \log \frac{\pi(a|s)}{\rho(a|s)} \right]. \tag{A.4}
\end{aligned}$$

In summary, we have shown the following expression for  $\mathcal{V}_{\text{KL}}^\pi$  at each state  $s \in \mathcal{S}$

$$\mathcal{V}_{\text{KL}}^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left[ \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(a) \mathcal{V}_{\text{KL}}^\pi(s') - \frac{1}{\beta} \log \frac{\pi(a|s)}{\rho(a|s)} \right]. \tag{A.5}$$

For convenience, in the sequel we adopt the following notation

$$\begin{aligned}\mathcal{R}(s, \boldsymbol{\pi}(s)) &= \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{R}(s, a), \\ \mathcal{T}_{ss'}(\boldsymbol{\pi}(s)) &= \sum_{a \in \mathcal{A}} \pi(a|s) \mathcal{T}_{ss'}(a).\end{aligned}$$

We can then define the  $Q$ -value corresponding to the policy  $\pi$  at state  $s$  as

$$\mathcal{Q}_{\text{KL}}^\pi(s, \boldsymbol{\pi}(s)) = \mathcal{R}(s, \boldsymbol{\pi}(s)) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(\boldsymbol{\pi}(s)) \mathcal{V}_{\text{KL}}^\pi(s'). \quad (\text{A.6})$$

We view  $\boldsymbol{\pi}(s)$  as a generalized action taken at state  $s$  induced by policy  $\pi$ . Next, with a slight abuse of notation, we define the  $Q$ -value corresponding to the policy  $\pi$  at state  $s$  as

$$\mathcal{Q}_{\text{KL}}^\pi(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(a) \mathcal{V}_{\text{KL}}^\pi(s').$$

Subsequently, we can also write

$$\mathcal{V}_{\text{KL}}^\pi(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left( \mathcal{Q}_{\text{KL}}^\pi(s, a) - \frac{1}{\beta} \log \frac{\pi(a|s)}{\rho(a|s)} \right). \quad (\text{A.7})$$

From Equation (A.5), we have that

$$\begin{aligned}\mathcal{V}_{\text{KL}}^\pi(s) &= \sum_{a \in \mathcal{A}} \pi(a|s) \left( \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(a) \mathcal{V}_{\text{KL}}^\pi(s') - \frac{1}{\beta} \log \frac{\pi(a|s)}{\rho(a|s)} \right) \\ &= \mathcal{R}(s, \boldsymbol{\pi}(s)) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(\boldsymbol{\pi}(s)) \mathcal{V}_{\text{KL}}^\pi(s') - \frac{1}{\beta} \text{KL}(\boldsymbol{\pi}(s) \| \boldsymbol{\rho}(s)) \\ &= \mathcal{Q}_{\text{KL}}^\pi(s, \boldsymbol{\pi}(s)) - \frac{1}{\beta} \text{KL}(\boldsymbol{\pi}(s) \| \boldsymbol{\rho}(s)).\end{aligned}$$

We define the soft-optimal value as

$$\begin{aligned}\mathcal{V}_{\text{KL}}^*(s) &= \max_{\pi} \mathcal{V}_{\text{KL}}^{\pi}(s) \\ &= \max_{\pi(s)} \left( \mathcal{R}(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(\pi(s)) \mathcal{V}_{\text{KL}}^{\pi}(s') - \frac{1}{\beta} \text{KL}(\pi(s) \parallel \rho(s)) \right).\end{aligned}$$

Given a value vector  $\mathcal{U} \in \mathbb{R}^{|\mathcal{S}|}$ , we may define an operator  $\mathcal{B} : \mathbb{R}^{|\mathcal{S}|} \rightarrow \mathbb{R}^{|\mathcal{S}|}$  in resemblance to the ‘‘Bellman operator’’, as follows

$$(\mathcal{B}\mathcal{U})(s) = \max_{\pi(s)} \left( \mathcal{R}(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(\pi(s)) \mathcal{U}(s') - \frac{1}{\beta} \text{KL}(\pi(s) \parallel \rho(s)) \right). \quad (\text{A.8})$$

**Lemma A.1.** *The Bellman operator  $\mathcal{B}$  in Equation (A.8) defines a contraction mapping for  $\gamma < 1$ .*

*Proof.* Given a policy  $\pi \in \Pi$ , let us first define the operator for value evaluation,  $\mathcal{B}^{\pi} : \mathbb{R}^{|\mathcal{S}|} \rightarrow \mathbb{R}^{|\mathcal{S}|}$

$$(\mathcal{B}^{\pi}\mathcal{U})(s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left[ \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(a) \mathcal{U}(s') - \frac{1}{\beta} \log \frac{\pi(a|s)}{\rho(a|s)} \right]. \quad (\text{A.9})$$

We will show that  $\mathcal{B}^{\pi}$  is a contraction mapping under the supremum norm for each admissible policy  $\pi \in \Pi$ . Hence, the value  $\mathcal{V}^{\pi}(s)$  of policy  $\pi$  can be evaluated for each state  $s \in \mathcal{S}$  using the recursion

$$\mathcal{V}_{k+1}^{\pi}(s) = (\mathcal{B}^{\pi}\mathcal{V}_k^{\pi})(s), \quad k = 1, 2, \dots$$

Consider two value vectors  $\mathcal{U}$  and  $\mathcal{V}$ , and define the supremum norm between the two as

$$\|\mathcal{U} - \mathcal{V}\|_{\infty} = \max_{s \in \mathcal{S}} |\mathcal{U}(s) - \mathcal{V}(s)|.$$

Then, for each state  $s \in \mathcal{S}$ , we have

$$\begin{aligned}
\left| (\mathcal{B}^\pi \mathcal{U})(s) - (\mathcal{B}^\pi \mathcal{V})(s) \right| &= \left| \sum_{a \in \mathcal{A}} \pi(a|s) \left[ \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(a) \mathcal{U}(s') - \frac{1}{\beta} \log \frac{\pi(a|s)}{\rho(a|s)} \right] \right. \\
&\quad \left. - \sum_{a \in \mathcal{A}} \pi(a|s) \left[ \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(a) \mathcal{V}(s') - \frac{1}{\beta} \log \frac{\pi(a|s)}{\rho(a|s)} \right] \right| \\
&= \left| \gamma \sum_{a \in \mathcal{A}} \pi(a|s) \left( \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(a) [\mathcal{U}(s') - \mathcal{V}(s')] \right) \right| \\
&\leq \gamma \sum_{a \in \mathcal{A}} \pi(a|s) \left( \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(a) \left| \mathcal{U}(s') - \mathcal{V}(s') \right| \right) \\
&\leq \gamma \sum_{a \in \mathcal{A}} \pi(a|s) \left( \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(a) \|\mathcal{U} - \mathcal{V}\|_\infty \right) \\
&= \gamma \sum_{a \in \mathcal{A}} \pi(a|s) \|\mathcal{U} - \mathcal{V}\|_\infty \\
&= \gamma \|\mathcal{U} - \mathcal{V}\|_\infty.
\end{aligned}$$

By applying the max operator on the left hand side, it follows that

$$\|\mathcal{B}^\pi \mathcal{U} - \mathcal{B}^\pi \mathcal{V}\|_\infty = \max_{s \in \mathcal{S}} \left| (\mathcal{B}^\pi \mathcal{U})(s) - (\mathcal{B}^\pi \mathcal{V})(s) \right| \leq \gamma \|\mathcal{U} - \mathcal{V}\|_\infty. \quad (\text{A.10})$$

Thus,  $\mathcal{B}^\pi$  is a contraction mapping for all admissible policies  $\pi \in \Pi$  for  $\gamma < 1$ .

Next, we prove that the operator  $\mathcal{B}$  in (Equation (A.8)) is also a contraction mapping. First, note that

$$(\mathcal{B}\mathcal{U})(s) = \max_{\pi} (\mathcal{B}^\pi \mathcal{U})(s), \quad \forall s \in \mathcal{S}.$$

Now consider two value vectors  $\mathcal{U}_1$  and  $\mathcal{U}_2$ . Let  $\pi_1$  and  $\pi_2$  correspond to the maximizing policies that define the  $\mathcal{B}$  operator for the two value vectors respectively. Namely, let

$$\pi_1(s) = \operatorname{argmax}_{\pi \in \Pi} (\mathcal{B}^\pi \mathcal{U}_1)(s), \quad \pi_2(s) = \operatorname{argmax}_{\pi \in \Pi} (\mathcal{B}^\pi \mathcal{U}_2)(s).$$

Then, we have an alternative representation of the operator  $\mathcal{B}$ .

$$\mathcal{B}\mathcal{U}_1 = \mathcal{B}^{\pi_1}\mathcal{U}_1, \quad \mathcal{B}\mathcal{U}_2 = \mathcal{B}^{\pi_2}\mathcal{U}_2,$$

due to the definition of  $\pi_1$  and  $\pi_2$ , we have

$$\begin{aligned} (\mathcal{B}\mathcal{U}_1)(s) &= (\mathcal{B}^{\pi_1}\mathcal{U}_1)(s) \geq (\mathcal{B}^\pi\mathcal{U}_1)(s), \quad \forall \pi \in \Pi, \\ (\mathcal{B}\mathcal{U}_2)(s) &= (\mathcal{B}^{\pi_2}\mathcal{U}_2)(s) \geq (\mathcal{B}^\pi\mathcal{U}_2)(s), \quad \forall \pi \in \Pi, \end{aligned}$$

and the following inequality holds at each state  $s$

$$(\mathcal{B}^{\pi_2}\mathcal{U}_1)(s) - (\mathcal{B}^{\pi_2}\mathcal{U}_2)(s) \leq (\mathcal{B}^{\pi_1}\mathcal{U}_1)(s) - (\mathcal{B}^{\pi_2}\mathcal{U}_2)(s) \leq (\mathcal{B}^{\pi_1}\mathcal{U}_1)(s) - (\mathcal{B}^{\pi_1}\mathcal{U}_2)(s). \quad (\text{A.11})$$

Applying the inequality  $a \leq b \leq c \Rightarrow |b| \leq \max\{|a|, |c|\}$ , to (Equation (A.11)) yields

$$\left| (\mathcal{B}\mathcal{U}_1)(s) - (\mathcal{B}\mathcal{U}_2)(s) \right| = \left| (\mathcal{B}^{\pi_1}\mathcal{U}_1)(s) - (\mathcal{B}^{\pi_2}\mathcal{U}_2)(s) \right| \leq \max_{i=1,2} \left\{ \left| (\mathcal{B}^{\pi_i}\mathcal{U}_1)(s) - (\mathcal{B}^{\pi_i}\mathcal{U}_2)(s) \right| \right\}. \quad (\text{A.12})$$

The lemma now follows from

$$\begin{aligned} \mathcal{B}\mathcal{U}_1 - \mathcal{B}\mathcal{U}_2_\infty &= \max_{s \in \mathcal{S}} \left| (\mathcal{B}\mathcal{U}_1)(s) - (\mathcal{B}\mathcal{U}_2)(s) \right| \\ &= \max_{s \in \mathcal{S}} \left| (\mathcal{B}^{\pi_1}\mathcal{U}_1)(s) - (\mathcal{B}^{\pi_2}\mathcal{U}_2)(s) \right| \\ &\leq \max_{s \in \mathcal{S}} \max_{i=1,2} \left\{ \left| (\mathcal{B}^{\pi_i}\mathcal{U}_1)(s) - (\mathcal{B}^{\pi_i}\mathcal{U}_2)(s) \right| \right\} \\ &= \max_{i=1,2} \left\{ \max_{s \in \mathcal{S}} \left| (\mathcal{B}^{\pi_i}\mathcal{U}_1)(s) - (\mathcal{B}^{\pi_i}\mathcal{U}_2)(s) \right| \right\} \\ &= \max_{i=1,2} \left\{ \left\| \mathcal{B}^{\pi_i}\mathcal{U}_1 - \mathcal{B}^{\pi_i}\mathcal{U}_2 \right\|_\infty \right\} \leq \gamma \mathcal{U}_1 - \mathcal{U}_2_\infty, \end{aligned}$$

where last inequality follows from Equation (A.10). It follows that  $\mathcal{B}$  is a contraction for

$\gamma < 1$ . □

As a result of the Banach fixed point theorem [3], from Equation (A.8) there exists a unique soft-optimal value vector  $\mathcal{V}_{\text{KL}}^*$  that satisfies the equation

$$\mathcal{V}_{\text{KL}}^*(s) = \max_{\boldsymbol{\pi}(s)} \left[ \mathcal{R}(s, \boldsymbol{\pi}(s)) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(\boldsymbol{\pi}(s)) \mathcal{V}_{\text{KL}}^*(s') - \frac{1}{\beta} \text{KL}(\boldsymbol{\pi}(s) \parallel \boldsymbol{\rho}(s)) \right]. \quad (\text{A.13})$$

By defining the soft-optimal  $Q$ -value in terms of the generalized action  $\boldsymbol{\pi}(s)$  as

$$\mathcal{Q}_{\text{KL}}^*(s, \boldsymbol{\pi}(s)) = \mathcal{R}(s, \boldsymbol{\pi}(s)) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(\boldsymbol{\pi}(s)) \mathcal{V}_{\text{KL}}^*(s'),$$

Equation (A.13) for the soft-optimal value can be written as

$$\begin{aligned} \mathcal{V}_{\text{KL}}^*(s) &= \max_{\boldsymbol{\pi}(s)} \left[ \mathcal{Q}_{\text{KL}}^*(s, \boldsymbol{\pi}(s)) - \frac{1}{\beta} \text{KL}(\boldsymbol{\pi}(s) \parallel \boldsymbol{\rho}(s)) \right] \\ &= \max_{\boldsymbol{\pi}} \sum_{a \in \mathcal{A}} \pi(a|s) \left[ \mathcal{Q}_{\text{KL}}^*(s, a) - \frac{1}{\beta} \log \frac{\pi(a|s)}{\rho(a|s)} \right], \end{aligned} \quad (\text{A.14})$$

where

$$\mathcal{Q}_{\text{KL}}^*(s, a) = \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(a) \mathcal{V}_{\text{KL}}^*(s').$$

The following lemma can be found, for example, in [20].

**Lemma A.2.** *For each state  $s \in \mathcal{S}$ , the soft-optimal policy that solves problem (Equation (A.14)) can be written in closed form as*

$$\pi_{\text{KL}}^*(a|s) = \frac{\rho(a|s) e^{\beta \mathcal{Q}_{\text{KL}}^*(s, a)}}{\sum_{a' \in \mathcal{A}} \rho(a'|s) e^{\beta \mathcal{Q}_{\text{KL}}^*(s, a')}}. \quad (\text{A.15})$$

*Proof.* For each  $s \in \mathcal{S}$ , the design variable to be optimized is the  $|\mathcal{A}|$ -dimensional vector  $\boldsymbol{\pi}(s)(a) = \pi(a|s)$  (see Equation (3.10)). It follows that there are  $|\mathcal{S}| \times |\mathcal{A}|$  design variables. From the total probability property  $\sum_{a \in \mathcal{A}} \pi(a|s) = 1$  for all  $s \in \mathcal{S}$ . We consider the

following constrained optimization problem

$$\max_{\pi} \sum_{a \in \mathcal{A}} \pi(a|s) \left[ \mathcal{Q}_{\text{KL}}^*(s, a) - \frac{1}{\beta} \log \frac{\pi(a|s)}{\rho(a|s)} \right], \quad (\text{A.16})$$

$$\text{such that } \sum_{a \in \mathcal{A}} \pi(a|s) = 1, \quad (\text{A.17})$$

to be solved for each  $s \in \mathcal{S}$ . To this end, we consider the following Lagrangian relaxation at state  $s$

$$\mathcal{L}_s(\pi, \lambda_s) = \sum_{a \in \mathcal{A}} \pi(a|s) \left[ \mathcal{Q}_{\text{KL}}^*(s, a) - \frac{1}{\beta} \log \frac{\pi(a|s)}{\rho(a|s)} \right] + \lambda_s \left( 1 - \sum_{a \in \mathcal{A}} \pi(a|s) \right), \quad (\text{A.18})$$

where  $\lambda_s \in \mathbb{R}$  is the Lagrangian multiplier.

The soft-optimal policy can be easily obtained by taking the partial derivative of the Lagrangian in Equation (A.18) with respect to each component  $\pi(a|s)$ , and then set the partial derivatives equal to zero.

For example, if we consider a specific design variable  $\pi(\hat{a}|s)$ , we have

$$\begin{aligned} 0 &= \frac{\partial}{\partial \pi(\hat{a}|s)} \mathcal{L}_s(\pi, \lambda_s) & (\text{A.19}) \\ &= \frac{\partial}{\partial \pi(\hat{a}|s)} \left\{ \sum_{a \in \mathcal{A}} \pi(a|s) \left[ \mathcal{Q}_{\text{KL}}^*(s, a) - \frac{1}{\beta} \log \frac{\pi(a|s)}{\rho(a|s)} \right] + \lambda_s \left( 1 - \sum_{a \in \mathcal{A}} \pi(a|s) \right) \right\} \\ &= \frac{\partial}{\partial \pi(\hat{a}|s)} \left\{ \pi(\hat{a}|s) \left[ \mathcal{Q}_{\text{KL}}^*(s, \hat{a}) - \frac{1}{\beta} \log \frac{\pi(\hat{a}|s)}{\rho(\hat{a}|s)} \right] - \lambda_s \pi(\hat{a}|s) \right\} \\ &= \mathcal{Q}_{\text{KL}}^*(s, \hat{a}) - \frac{1}{\beta} \log \frac{\pi(\hat{a}|s)}{\rho(\hat{a}|s)} - \frac{1}{\beta} - \lambda_s. \end{aligned}$$

Solving for  $\pi(\hat{a}|s)$ , we have a single maximum candidate:

$$\pi(\hat{a}|s) = \rho(\hat{a}|s) e^{1 + \beta \lambda_s + \beta \mathcal{Q}_{\text{KL}}^*(s, \hat{a})}. \quad (\text{A.20})$$

After normalizing the policy to satisfy the constraint in Equation (A.17), Equation (A.20)

yields

$$\begin{aligned}
\pi(\hat{a}|s) &= \frac{\rho(\hat{a}|s)e^{1+\beta\lambda_s+\beta\mathcal{Q}_{\text{KL}}^*(s,\hat{a})}}{\sum_a \rho(a|s)e^{1+\beta\lambda_s+\beta\mathcal{Q}_{\text{KL}}^*(s,a)}} \\
&= \frac{e^{1+\beta\lambda_s}\rho(\hat{a}|s)e^{\beta\mathcal{Q}_{\text{KL}}^*(s,\hat{a})}}{e^{1+\beta\lambda_s}\sum_a \rho(a|s)e^{\beta\mathcal{Q}_{\text{KL}}^*(s,a)}} \\
&= \frac{\rho(\hat{a}|s)e^{\beta\mathcal{Q}_{\text{KL}}^*(s,\hat{a})}}{\sum_a \rho(a|s)e^{\beta\mathcal{Q}_{\text{KL}}^*(s,a)}}.
\end{aligned}$$

To verify that the solution in Equation (A.15) is indeed a maximizer, we check the second-order partial derivative Equation (A.18)

$$\frac{\partial^2}{\partial^2 \pi(\hat{a}|s)} \mathcal{L}_s(\pi, \lambda_s) = -\frac{1}{\beta \pi(\hat{a}|s)} < 0. \quad (\text{A.21})$$

We have the second order partial derivative strictly less than zero since  $\beta > 0$  and  $\pi(\hat{a}|s) > 0$ , and thus is the objective function concave.

In summary, we have the soft-optimal policy for each state  $s \in \mathcal{S}$  as a closed-form expression as

$$\pi_{\text{KL}}^*(a|s) = \frac{\rho(a|s)e^{\beta\mathcal{Q}_{\text{KL}}^*(s,a)}}{\sum_a \rho(a|s)e^{\beta\mathcal{Q}_{\text{KL}}^*(s,a)}}, \quad (\text{A.22})$$

which yields the desired result.  $\square$

The soft-optimal value can be obtained by substituting the soft-optimal policy back into Equation (A.7) to obtain

$$\mathcal{V}_{\text{KL}}^*(s) = \mathcal{V}_{\text{KL}}^{\pi_{\text{KL}}^*}(s) = \frac{1}{\beta} \log \sum_{a \in \mathcal{A}} \rho(a|s) e^{\beta\mathcal{Q}_{\text{KL}}^*(s,a)}. \quad (\text{A.23})$$

As a result, we have the following corollary.

**Corollary A.3.** *At each state  $s \in \mathcal{S}$ , the soft-optimal value is given by*

$$\mathcal{V}_{\text{KL}}^*(s) = \frac{1}{\beta} \log \sum_{a \in \mathcal{A}} \rho(a|s) e^{\beta\mathcal{Q}_{\text{KL}}^*(s,a)}. \quad (\text{A.24})$$

In particular, note that  $\mathcal{V}_{\text{KL}}^*(s)$  can be computed as the fixed point of the following implicit equation

$$\mathcal{V}_{\text{KL}}^*(s) = \frac{1}{\beta} \log \sum_{a \in \mathcal{A}} \rho(a|s) \exp \left[ \beta \mathcal{R}(s, a) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(a) \mathcal{V}_{\text{KL}}^*(s') \right]. \quad (\text{A.25})$$

**Corollary A.4.** *Consider the following minimization problem with  $\beta < 0$*

$$\mathcal{V}_{\text{KL}}^*(s) = \min_{\pi} \sum_{a \in \mathcal{A}} \pi(a|s) \left[ \mathcal{Q}_{\text{KL}}^*(s, a) - \frac{1}{\beta} \log \frac{\pi(a|s)}{\rho(a|s)} \right]. \quad (\text{A.26})$$

*For each state  $s \in \mathcal{S}$ , the soft-optimal policy that solves the minimization problem Equation (A.26) has the closed form*

$$\pi_{\text{KL}}^*(a|s) = \frac{\rho(a|s) e^{\beta \mathcal{Q}_{\text{KL}}^*(s, a)}}{\sum_{a' \in \mathcal{A}} \rho(a'|s) e^{\beta \mathcal{Q}_{\text{KL}}^*(s, a')}}.$$

*The soft-optimal value at each state  $s \in \mathcal{S}$  is then given by*

$$\mathcal{V}_{\text{KL}}^*(s) = \frac{1}{\beta} \log \sum_{a \in \mathcal{A}} \rho(a|s) e^{\beta \mathcal{Q}_{\text{KL}}^*(s, a)}. \quad (\text{A.27})$$

*Proof.* This proof is essentially the same as the proof for Lemma A.0.2. When the problem is formulated as a minimization problem as in Equation (A.26), the Lagrangian multiplier  $\beta$  becomes negative. The first-order condition in Equation (A.19) remains the same, while the second-order condition in Equation (A.21) becomes positive, which guarantees a minimum.

□

## APPENDIX B

### DERIVATION OF TWO-AGENT SOFT-Q LEARNING

Consider now the two-agent zero-sum case, where we have the policy-dependent soft value defined in a similar manner as in Equation (A.3). We expand the expectation operator and re-arrange terms to obtain

$$\begin{aligned}
\mathcal{V}_{\text{KL}}^{\pi^{\text{pl}}, \pi^{\text{op}}}(s) &= \mathbb{E}_{s_0=s}^{\pi^{\text{pl}}, \pi^{\text{op}}} \left[ \sum_{t=0}^{\infty} \gamma^t \left( \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)} \right) \right] \\
&= \mathbb{E}_{s_0=s}^{\pi^{\text{pl}}, \pi^{\text{op}}} \left[ \mathcal{R}(s, a_0^{\text{pl}}, a_0^{\text{op}}) - \frac{1}{\beta^{\text{pl}}} \log \frac{\pi^{\text{pl}}(a_0^{\text{pl}}|s)}{\rho^{\text{pl}}(a_0^{\text{pl}}|s)} - \frac{1}{\beta^{\text{op}}} \log \frac{\pi^{\text{op}}(a_0^{\text{op}}|s)}{\rho^{\text{op}}(a_0^{\text{op}}|s)} \right] \\
&\quad + \gamma \mathbb{E}_{s_0=s}^{\pi^{\text{pl}}, \pi^{\text{op}}} \left[ \mathcal{V}_{\text{KL}}^{\pi^{\text{pl}}, \pi^{\text{op}}}(s_1) \right] \\
&= \sum_{a^{\text{pl}} \in \mathcal{A}^{\text{pl}}} \sum_{a^{\text{op}} \in \mathcal{A}^{\text{op}}} \pi^{\text{pl}}(a^{\text{pl}}|s) \pi^{\text{op}}(a^{\text{op}}|s) \left( \mathcal{R}(s, a^{\text{pl}}, a^{\text{op}}) - \frac{1}{\beta^{\text{pl}}} \log \frac{\pi^{\text{pl}}(a^{\text{pl}}|s)}{\rho^{\text{pl}}(a^{\text{pl}}|s)} \right. \\
&\quad \left. - \frac{1}{\beta^{\text{op}}} \log \frac{\pi^{\text{op}}(a^{\text{op}}|s)}{\rho^{\text{op}}(a^{\text{op}}|s)} \right) + \gamma \sum_{s' \in \mathcal{S}} \sum_{a^{\text{pl}} \in \mathcal{A}^{\text{pl}}} \sum_{a^{\text{op}} \in \mathcal{A}^{\text{op}}} \pi^{\text{pl}}(a^{\text{pl}}|s) \pi^{\text{op}}(a^{\text{op}}|s) \mathcal{T}_{ss'}(a^{\text{pl}}, a^{\text{op}}) \mathcal{V}_{\text{KL}}^{\pi^{\text{pl}}, \pi^{\text{op}}}(s') \\
&= \sum_{a^{\text{pl}} \in \mathcal{A}^{\text{pl}}} \sum_{a^{\text{op}} \in \mathcal{A}^{\text{op}}} \pi^{\text{pl}}(a^{\text{pl}}|s) \pi^{\text{op}}(a^{\text{op}}|s) \left( \mathcal{R}(s, a^{\text{pl}}, a^{\text{op}}) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(a^{\text{pl}}, a^{\text{op}}) \mathcal{V}_{\text{KL}}^{\pi^{\text{pl}}, \pi^{\text{op}}}(s') \right. \\
&\quad \left. - \frac{1}{\beta^{\text{pl}}} \log \frac{\pi^{\text{pl}}(a^{\text{pl}}|s)}{\rho^{\text{pl}}(a^{\text{pl}}|s)} - \frac{1}{\beta^{\text{op}}} \log \frac{\pi^{\text{op}}(a^{\text{op}}|s)}{\rho^{\text{op}}(a^{\text{op}}|s)} \right).
\end{aligned}$$

Define the  $(\pi^{\text{pl}}, \pi^{\text{op}})$ -policy dependent Q-value as

$$\mathcal{Q}_{\text{KL}}^{\pi^{\text{pl}}, \pi^{\text{op}}}(s, a^{\text{pl}}, a^{\text{op}}) = \mathcal{R}(s, a^{\text{pl}}, a^{\text{op}}) + \gamma \sum_{s' \in \mathcal{S}} \mathcal{T}_{ss'}(a^{\text{pl}}, a^{\text{op}}) \mathcal{V}_{\text{KL}}^{\pi^{\text{pl}}, \pi^{\text{op}}}(s'). \quad (\text{B.1})$$

Then, we have

$$\mathcal{V}_{\text{KL}}^{\pi^{\text{pl}}, \pi^{\text{op}}}(s) = \sum_{a^{\text{pl}} \in \mathcal{A}^{\text{pl}}} \sum_{a^{\text{op}} \in \mathcal{A}^{\text{op}}} \pi^{\text{pl}}(a^{\text{pl}}|s) \pi^{\text{op}}(a^{\text{op}}|s) \left( \mathcal{Q}_{\text{KL}}^{\pi^{\text{pl}}, \pi^{\text{op}}}(s, a^{\text{pl}}, a^{\text{op}}) - \frac{1}{\beta^{\text{pl}}} \log \frac{\pi^{\text{pl}}(a^{\text{pl}}|s)}{\rho^{\text{pl}}(a^{\text{pl}}|s)} - \frac{1}{\beta^{\text{op}}} \log \frac{\pi^{\text{op}}(a^{\text{op}}|s)}{\rho^{\text{op}}(a^{\text{op}}|s)} \right) \quad (\text{B.2})$$

$$= \sum_{a^{\text{pl}} \in \mathcal{A}^{\text{pl}}} \left[ \pi^{\text{pl}}(a^{\text{pl}}|s) \left\{ \sum_{a^{\text{op}} \in \mathcal{A}^{\text{op}}} \left[ \pi^{\text{op}}(a^{\text{op}}|s) \left( \mathcal{Q}_{\text{KL}}^{\pi^{\text{pl}}, \pi^{\text{op}}}(s, a^{\text{pl}}, a^{\text{op}}) - \frac{1}{\beta^{\text{op}}} \log \frac{\pi^{\text{op}}(a^{\text{op}}|s)}{\rho^{\text{op}}(a^{\text{op}}|s)} \right) - \sum_{a^{\text{op}} \in \mathcal{A}^{\text{op}}} \pi^{\text{op}}(a^{\text{op}}|s) \left( \frac{1}{\beta^{\text{pl}}} \log \frac{\pi^{\text{pl}}(a^{\text{pl}}|s)}{\rho^{\text{pl}}(a^{\text{pl}}|s)} \right) \right] \right\} \right] \quad (\text{B.3})$$

$$= \sum_{a^{\text{pl}} \in \mathcal{A}^{\text{pl}}} \left[ \pi^{\text{pl}}(a^{\text{pl}}|s) \left\{ \sum_{a^{\text{op}} \in \mathcal{A}^{\text{op}}} \left[ \pi^{\text{op}}(a^{\text{op}}|s) \left( \mathcal{Q}_{\text{KL}}^{\pi^{\text{pl}}, \pi^{\text{op}}}(s, a^{\text{pl}}, a^{\text{op}}) - \frac{1}{\beta^{\text{op}}} \log \frac{\pi^{\text{op}}(a^{\text{op}}|s)}{\rho^{\text{op}}(a^{\text{op}}|s)} \right) - \frac{1}{\beta^{\text{pl}}} \log \frac{\pi^{\text{pl}}(a^{\text{pl}}|s)}{\rho^{\text{pl}}(a^{\text{pl}}|s)} \right] \right\} \right]. \quad (\text{B.4})$$

In summary, we have

$$\mathcal{V}_{\text{KL}}^{\pi^{\text{pl}}, \pi^{\text{op}}}(s) = \sum_{a^{\text{pl}} \in \mathcal{A}^{\text{pl}}} \left[ \pi^{\text{pl}}(a^{\text{pl}}|s) \left\{ \sum_{a^{\text{op}} \in \mathcal{A}^{\text{op}}} \left[ \pi^{\text{op}}(a^{\text{op}}|s) \left( \mathcal{Q}_{\text{KL}}^{\pi^{\text{pl}}, \pi^{\text{op}}}(s, a^{\text{pl}}, a^{\text{op}}) - \frac{1}{\beta^{\text{op}}} \log \frac{\pi^{\text{op}}(a^{\text{op}}|s)}{\rho^{\text{op}}(a^{\text{op}}|s)} \right) - \frac{1}{\beta^{\text{pl}}} \log \frac{\pi^{\text{pl}}(a^{\text{pl}}|s)}{\rho^{\text{pl}}(a^{\text{pl}}|s)} \right] \right\} \right]. \quad (\text{B.5})$$

We consider the case where the Player is the maximizer and the Opponent is the minimizer.

Then, the definition of the soft-optimal value for the two-agent zero-sum game is

$$\begin{aligned} V_{\text{KL}}^*(s) &= \max_{\pi^{\text{pl}}} \min_{\pi^{\text{op}}} \mathcal{V}_{\text{KL}}^{\pi^{\text{pl}}, \pi^{\text{op}}}(s) \\ &= \max_{\pi^{\text{pl}}} \sum_{a^{\text{pl}} \in \mathcal{A}^{\text{pl}}} \left[ \pi^{\text{pl}}(a^{\text{pl}}|s) \left\{ \min_{\pi^{\text{op}}} \left[ \sum_{a^{\text{op}} \in \mathcal{A}^{\text{op}}} \pi^{\text{op}}(a^{\text{op}}|s) \left( \mathcal{Q}_{\text{KL}}^*(s, a^{\text{pl}}, a^{\text{op}}) - \frac{1}{\beta^{\text{op}}} \log \frac{\pi^{\text{op}}(a^{\text{op}}|s)}{\rho^{\text{op}}(a^{\text{op}}|s)} \right) - \frac{1}{\beta^{\text{pl}}} \log \frac{\pi^{\text{pl}}(a^{\text{pl}}|s)}{\rho^{\text{pl}}(a^{\text{pl}}|s)} \right] \right\} \right], \end{aligned}$$

where the soft-optimal Q-value is defined as in Equation (B.1).

**Theorem B.1.** *The soft-optimal value in Equation (B.6) is unique.*

To prove Theorem B.1, we first present the minimax by von Neumann [49].

**Theorem B.2** (von Neumann). *Let  $X \subset \mathbb{R}^n$  and  $Y \subset \mathbb{R}^m$  be compact convex sets. Assume that  $f : X \times Y \rightarrow \mathbb{R}$  is a continuous function that is component-wise concave-convex, that is*

$$f(\cdot, y) : X \rightarrow \mathbb{R} \text{ is concave for all fixed } y \in Y,$$

$$f(x, \cdot) : Y \rightarrow \mathbb{R} \text{ is convex for all fixed } x \in X.$$

Then

$$\max_{x \in X} \min_{y \in Y} f(x, y) = \min_{y \in Y} \max_{x \in X} f(x, y). \quad (\text{B.6})$$

Now we can prove Theorem B.1.

*Proof.* First notice that the set of admissible policies for the Player,  $\Pi^{\text{Pl}}$ , can be written as

$$\Pi^{\text{Pl}} = \underbrace{\Delta_{|\mathcal{A}^{\text{Pl}}|} \times \Delta_{|\mathcal{A}^{\text{Pl}}|} \times \cdots \times \Delta_{|\mathcal{A}^{\text{Pl}}|}}_{|\mathcal{S}|}.$$

Specifically,  $\Pi^{\text{Pl}}$  is formed by the Cartesian product of  $|\mathcal{S}|$  standard  $|\mathcal{A}^{\text{Pl}}|$ -dimensional simplices. Thus,  $\Pi^{\text{Pl}}$  is compact and convex. Similarly,  $\Pi^{\text{Op}}$  is also compact and convex.

Fixing  $\pi^{\text{Op}}$ , Equation (B.2) then becomes

$$\begin{aligned} \mathcal{V}_{\text{KL}}^{\pi^{\text{Pl}}, \pi^{\text{Op}}}(s) &= \sum_{a^{\text{Pl}} \in \mathcal{A}^{\text{Pl}}} \left[ \pi^{\text{Pl}}(a^{\text{Pl}}|s) \left\{ \underbrace{\sum_{a^{\text{Op}} \in \mathcal{A}^{\text{Op}}} \left[ \pi^{\text{Op}}(a^{\text{Op}}|s) \left( \mathcal{Q}_{\text{KL}}^{\pi^{\text{Pl}}, \pi^{\text{Op}}}(s, a^{\text{Pl}}, a^{\text{Op}}) - \frac{1}{\beta^{\text{Op}}} \log \frac{\pi^{\text{Op}}(a^{\text{Op}}|s)}{\rho^{\text{Op}}(a^{\text{Op}}|s)} \right) \right]}_{\Psi(s, a^{\text{Pl}})} \right. \\ &\quad \left. - \sum_{a^{\text{Op}} \in \mathcal{A}^{\text{Op}}} \pi^{\text{Op}}(a^{\text{Op}}|s) \left( \frac{1}{\beta^{\text{Pl}}} \log \frac{\pi^{\text{Pl}}(a^{\text{Pl}}|s)}{\rho^{\text{Pl}}(a^{\text{Pl}}|s)} \right) \right\} \right] \\ &= \sum_{a^{\text{Pl}} \in \mathcal{A}^{\text{Pl}}} \left[ \pi^{\text{Pl}}(a^{\text{Pl}}|s) \Psi(s, a^{\text{Pl}}) - \frac{1}{\beta^{\text{Pl}}} \log \frac{\pi^{\text{Pl}}(a^{\text{Pl}}|s)}{\rho^{\text{Pl}}(a^{\text{Pl}}|s)} \right]. \quad (\text{B.7}) \end{aligned}$$

With  $\beta^{\text{pl}} > 0$ , it can be seen from Equation (B.7) that  $\mathcal{V}_{\text{KL}}^{\pi^{\text{pl}}, \pi^{\text{op}}}$  is concave in  $\pi^{\text{pl}}$ . To check whether  $\mathcal{V}_{\text{KL}}^{\pi^{\text{pl}}, \pi^{\text{op}}}$  is convex in  $\pi^{\text{op}}$ , we can change the order of the two summations, as the action spaces are finite. Via a similar reasoning, one can show that  $\mathcal{V}_{\text{KL}}^{\pi^{\text{pl}}, \pi^{\text{op}}}$  is convex in  $\pi^{\text{op}}$ , as  $\beta^{\text{op}} < 0$ . Thus, the soft-optimal value in Equation (B.2) is unique.  $\square$

We define the minimization conducted by the Opponent in Equation (B.6) as

$$\mathcal{Q}_{\text{KL}}^{\text{pl},*}(s, a^{\text{pl}}) = \min_{\pi^{\text{op}}} \sum_{a^{\text{op}} \in \mathcal{A}^{\text{op}}} \pi^{\text{op}}(a^{\text{op}}|s) \left[ \mathcal{Q}_{\text{KL}}^*(s, a^{\text{pl}}, a^{\text{op}}) - \frac{1}{\beta^{\text{pl}}} \log \frac{\pi^{\text{op}}(a^{\text{op}}|s)}{\rho^{\text{op}}(a^{\text{op}}|s)} \right]. \quad (\text{B.8})$$

Observe that Equation (B.8) has the same structure as the optimization problem presented in Equation (A.26). Using Corollary A.0.4, we therefore have

$$\mathcal{Q}_{\text{KL}}^{\text{pl},*}(s, a^{\text{pl}}) = \frac{1}{\beta^{\text{op}}} \log \sum_{a^{\text{op}} \in \mathcal{A}^{\text{op}}} \rho^{\text{op}}(s|a^{\text{op}}) \exp(\beta^{\text{op}} \mathcal{Q}_{\text{KL}}^*(s, a^{\text{pl}}, a^{\text{op}})).$$

Combining Equation (B.6) and Equation (B.8), we arrive at

$$V_{\text{KL}}^*(s) = \max_{\pi^{\text{pl}}} \sum_{a^{\text{pl}} \in \mathcal{A}^{\text{pl}}} \pi^{\text{pl}}(a^{\text{pl}}|s) \left[ \mathcal{Q}_{\text{KL}}^{\text{pl},*}(s, a^{\text{pl}}) - \frac{1}{\beta^{\text{pl}}} \log \frac{\pi^{\text{pl}}(a^{\text{pl}}|s)}{\rho^{\text{pl}}(a^{\text{pl}}|s)} \right], \quad (\text{B.9})$$

where the maximization is solely dependent on  $\pi^{\text{pl}}$ . Notice that Equation (B.9) has the same structure as in Equation (A.14). By Lemma A.0.2, we have the closed-form solution for the optimal policy of the Player as

$$\pi_{\text{KL}}^{\text{pl},*}(a^{\text{pl}}|s) = \frac{\rho^{\text{pl}}(a^{\text{pl}}|s) e^{\beta^{\text{pl}} \mathcal{Q}_{\text{KL}}^{\text{pl},*}(s, a^{\text{pl}})}}{\sum_{a^{\text{pl}'} \in \mathcal{A}^{\text{pl}}} \rho^{\text{pl}}(a^{\text{pl}'}|s) e^{\beta^{\text{pl}} \mathcal{Q}_{\text{KL}}^{\text{pl},*}(s, a^{\text{pl}'})}} \quad s \in \mathcal{S}.$$

Applying Corollary A.0.3 to Equation (B.9), we obtain the soft-optimal value at each state  $s \in \mathcal{S}$  as

$$\mathcal{V}_{\text{KL}}^*(s) = \frac{1}{\beta^{\text{pl}}} \log \sum_{a^{\text{pl}}} \rho^{\text{pl}}(a^{\text{pl}}|s) \exp(\beta^{\text{pl}} \mathcal{Q}_{\text{KL}}^{\text{pl},*}(s, a^{\text{pl}})). \quad (\text{B.10})$$

Since the soft-optimal value is unique, we can change the order of the max and min operators in Equation (B.6). We would then have a min-max optimization problem instead. The

soft-optimal policy of the Opponent can be then derived in a similar manner. Namely, the maximization conducted by the Player results in the following marginalization

$$\mathcal{Q}_{\text{KL}}^{\text{op},*}(s, a^{\text{op}}) = \frac{1}{\beta^{\text{pl}}} \log \sum_{a^{\text{pl}} \in \mathcal{A}^{\text{pl}}} \rho^{\text{pl}}(s|a^{\text{pl}}) \exp(\beta^{\text{pl}} \mathcal{Q}_{\text{KL}}^*(s, a^{\text{pl}}, a^{\text{op}})).$$

The soft-optimal value has the expression

$$\mathcal{V}_{\text{KL}}^*(s) = \frac{1}{\beta^{\text{op}}} \log \sum_{a^{\text{op}}} \rho^{\text{op}}(a^{\text{op}}|s) \exp(\beta^{\text{op}} \mathcal{Q}_{\text{KL}}^{\text{op},*}(s, a^{\text{op}})), \quad (\text{B.11})$$

and the soft-optimal policy for the Opponent is

$$\pi_{\text{KL}}^{\text{op},*}(a^{\text{op}}|s) = \frac{\rho^{\text{op}}(a^{\text{op}}|s) e^{\beta^{\text{op}} \mathcal{Q}_{\text{KL}}^{\text{op},*}(s, a^{\text{op}})}}{\sum_{a^{\text{op}}' \in \mathcal{A}^{\text{op}}} \rho(a^{\text{op}}'|s) e^{\beta^{\text{op}} \mathcal{Q}_{\text{KL}}^{\text{op},*}(s, a^{\text{op}}')}}, \quad s \in \mathcal{S}. \quad (\text{B.12})$$

## REFERENCES

- [1] E. O. Neftci and B. B. Averbeck, “Reinforcement learning in artificial and biological systems,” *Nature Machine Intelligence*, vol. 1, no. 3, pp. 133–143, 2019.
- [2] R. S. Sutton and A. G. Barto, *Introduction to reinforcement learning*, 1st. Cambridge, MA, USA: MIT Press, 1998, ISBN: 0262193981.
- [3] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*. Athena Scientific, 1996.
- [4] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, “Playing Atari with deep reinforcement learning,” *arXiv preprint arXiv:1312.5602*, 2013.
- [5] T. P. Lillicrap and et al., “Continuous control with deep reinforcement learning,” *Preprint arXiv:1509.02971*, 2015. arXiv: 1509.02971 [cs.LG].
- [6] L. Matignon, G. Laurent, and N. Fort-Piat, “Independent reinforcement learners in cooperative Markov games: A survey regarding coordination problems,” *The Knowledge Engineering Review*, vol. 27, pp. 1–31, Mar. 2012.
- [7] L. Busoniu, R. Babuska, and B. De Schutter, “A comprehensive survey of multiagent reinforcement learning,” *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, vol. 38, no. 2, pp. 156–172, 2008.
- [8] J. Filar and K. Vrieze, *Competitive Markov Decision Processes*. Springer Science & Business Media, 2012.
- [9] G. Owen, *Game Theory*. Academic Press, 1982, ISBN: 9780125311502.
- [10] J.-K. Chong, C. F. Camerer, and T.-H. Ho, “Cognitive hierarchy: A limited thinking theory in games,” in *Experimental Business Research*, R. Zwick and A. Rapoport, Eds., Boston, MA: Springer US, 2005, pp. 203–228, ISBN: 978-0-387-24244-6.
- [11] D. Kahneman, “Maps of bounded rationality: Psychology for behavioral economics,” *The American Economic Review*, vol. 93, no. 5, pp. 1449–1475, 2003.
- [12] J. N. Foerster, R. Y. Chen, M. Al-Shedivat, S. Whiteson, P. Abbeel, and I. Mordatch, “Learning with opponent-learning awareness,” *CoRR*, vol. abs/1709.04326, 2017. arXiv: 1709.04326.

- [13] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, “Multi-agent actor-critic for mixed cooperative-competitive environments,” *CoRR*, vol. abs/1706.02275, 2017. arXiv: 1706.02275.
- [14] L. Panait and S. Luke, “Cooperative multi-agent learning: The state of the art,” *Autonomous Agents and Multi-Agent Systems*, vol. 11, pp. 387–434, Nov. 2005.
- [15] M. L. Littman, “Markov games as a framework for multi-agent reinforcement learning,” in *ICML*, 1994, pp. 157–163, ISBN: 1558603352.
- [16] J. Hu and M. P. Wellman, “Nash Q-learning for general-sum stochastic games,” *Journal of Machine Learning Research*, vol. 4, no. Nov, pp. 1039–1069, 2003.
- [17] M. L. Littman, “Friend-or-Foe Q-learning in general-sum games,” in *ICML*, 2001, pp. 322–328, ISBN: 1558607781.
- [18] X. Wang and T. Sandholm, “Reinforcement learning to play an optimal Nash equilibrium in team markov games,” in *Proceedings of the 15th International Conference on Neural Information Processing Systems*, 2002, pp. 1603–1610.
- [19] J. D. Bransford and et al., *How People Learn*. National Academy Press, 2000, vol. 11.
- [20] R. Fox, A. Pakman, and N. Tishby, “Taming the noise in reinforcement learning via soft updates,” *arXiv preprint arXiv:1512.08562*, 2015.
- [21] J. Grau-Moya, F. Leibfried, and H. Bou-Ammar, “Balancing two-player stochastic games with soft Q-learning,” *arXiv preprint arXiv:1802.03216*, 2018.
- [22] O. Morgenstern and J. Von Neumann, *Theory of Games and Economic Behavior*. Princeton University Press, 1953.
- [23] J. Nash, “Non-cooperative games,” *Annals of Mathematics*, vol. 54, no. 2, pp. 286–295, 1951.
- [24] M. Tan, “Multi-agent reinforcement learning: Independent versus cooperative agents,” in *ICML*, 1993, pp. 330–337.
- [25] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, “Counterfactual multi-agent policy gradients,” *AAAI Conference on Artificial Intelligence*, 2018.
- [26] M. Bowling and M. Veloso, “Rational and convergent learning in stochastic games,” in *ICML*, 2001, pp. 1021–1026, ISBN: 1558608125.

- [27] K. Kawamura, N. Mizukami, and Y. Tsuruoka, “Neural fictitious self-play in imperfect information games with many players,” in *Computer Games*, Springer International Publishing, 2018, pp. 61–74.
- [28] L. Zhang, W. Wang, S. Li, and G. Pan, “Monte Carlo neural fictitious self-play: Approach to approximate Nash equilibrium of imperfect-information games,” *Preprint arXiv:1903.09569*, 2019.
- [29] M. L. Littman and P. Stone, “A polynomial-time Nash equilibrium algorithm for repeated games,” *Decision Support Systems*, vol. 39, no. 1, pp. 55–66, 2005.
- [30] M. Lauer and M. A. Riedmiller, “An algorithm for distributed reinforcement learning in cooperative multi-agent systems,” in *Proceedings of the Seventeenth International Conference on Machine Learning*, ser. ICML ’00, 2000, pp. 535–542.
- [31] L. Matignon, G. J. Laurent, and N. Le Fort-Piat, “Hysteretic Q-learning : An algorithm for decentralized reinforcement learning in cooperative multi-agent teams,” in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 64–69.
- [32] S. Omidshafiei, J. Papis, C. Amato, J. P. How, and J. Vian, “Deep decentralized multi-task multi-agent reinforcement learning under partial observability,” in *Proceedings of the 34th International Conference on Machine Learning*, 2017, pp. 2681–2690.
- [33] J. K. Gupta, M. Egorov, and M. Kochenderfer, “Cooperative multi-agent control using deep reinforcement learning,” in *Autonomous Agents and Multiagent Systems*, 2017, pp. 66–83.
- [34] A. Tampuu and et al., “Multiagent cooperation and competition with deep reinforcement learning,” *PLOS ONE*, vol. 12, pp. 1–15, 2017.
- [35] C. Daskalakis, P. W. Goldberg, and C. H. Papadimitriou, “The complexity of computing a Nash equilibrium,” in *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing*, 2006, pp. 71–78, ISBN: 1595931341.
- [36] Y. Yang, R. Luo, M. Li, M. Zhou, W. Zhang, and J. Wang, “Mean field multi-agent reinforcement learning,” *CoRR*, vol. abs/1802.05438, 2018. arXiv: 1802.05438.
- [37] X. Guo, A. Hu, R. Xu, and J. Zhang, “Learning mean-field games,” in *Advances in Neural Information Processing Systems 32*, 2019, pp. 4966–4976.
- [38] J. Perolat, B. Scherrer, B. Piot, and O. Pietquin, “Approximate dynamic programming for two-player zero-sum Markov games,” in *Proceedings of the 32nd International Conference on Machine Learning*, vol. 37, 2015, pp. 1321–1329.

- [39] S. Li, Y. Wu, X. Cui, H. Dong, F. Fang, and S. Russell, “Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 33, pp. 4213–4220, 2019.
- [40] M. L. Puterman, *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, 2014.
- [41] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, “Reinforcement learning with deep energy-based policies,” in *ICML*, 2017, pp. 1352–1361.
- [42] M. J. Osborne and A. Rubinstein, *A Course in Game Theory*. MIT press, 1994.
- [43] C. J. C. H. Watkins and P. Dayan, “Q-learning,” in *Machine Learning*, 1992, pp. 279–292.
- [44] E. T. Jaynes, “Information theory and statistical mechanics,” *Phys. Rev.*, vol. 106, pp. 620–630, 4 May 1957.
- [45] Y. Guan, D. Maity, C. M. Kroninger, and P. Tsiotras, “Bounded-rational pursuit-evasion games,” *arXiv preprint arXiv:2003.06954*, 2020. arXiv: 2003.06954.
- [46] A. Silva and M. C. Gombolay, “Pronets: Neural-encoding human experts’ domain knowledge to warm start reinforcement learning,” *CoRR*, vol. abs/1902.06007, 2019. arXiv: 1902.06007.
- [47] M. G. Lagoudakis and R. Parr, “Value function approximation in zero-sum markov games,” in *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence*, 2002, pp. 283–292.
- [48] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. Jarrod Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. Carey, Í. Polat, Y. Feng, E. W. Moore, J. Vand erPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and S. 1. 0. Contributors, “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python,” *Nature Methods*, vol. 17, pp. 261–272, 2020.
- [49] J. Von Neumann, O. Morgenstern, and H. W. Kuhn, *Theory of Games and Economic Behavior*. Princeton University Press, 2007.