Independent Learning Approaches: Overcoming Multi-Agent Learning Pathologies In Team-Games

Thesis submitted in accordance with the requirements of
the University of Liverpool for the degree of Doctor in Philosophy by

Gregory Palmer

December 2019

Contents

Illustrations

List of Figures

List of Tables

Preface

This thesis is primarily my own work. The sources of other materials are identifed. This work has not been submitted for any other degree or professional qualification except as specified.

Abstract

Deep Neural Networks enable *Reinforcement Learning* (RL) agents to learn behaviour policies directly from high-dimensional observations. As a result, the field of *Deep Reinforcement Learning* (DRL) has seen a great number of successes. Recently the sub-field of *Multi-Agent* DRL (MADRL) has received an increased amount of attention. However, considerations are required when using RL in *Multi-Agent Systems*. *Independent Learners* (ILs) for instance lack the convergence guarantees of many single-agent RL approaches, even in domains that do not require a MADRL approach. Furthermore, ILs must often overcome a number of learning pathologies to converge upon an optimal joint-policy. Numerous IL approaches have been proposed to facilitate cooperation, including *hysteretic Q-learning* [120] and *leniency* [148]. Recently *LMRL2*, a variation of leniency, proved robust towards a number of pathologies in low-dimensional domains, including *miscoordination*, *relative overgeneralization*, *stochasticity*, *the alter-exploration problem* and *the moving target problem* [209]. In contrast, the majority of work on ILs in MADRL focuses on an amplified moving target problem, caused by neural networks being trained with potentially obsolete samples drawn from *experience replay memories*.

In this thesis we combine advances from research on ILs with DRL algorithms. However, first we evaluate the robustness of tabular approaches along each of the above pathology dimensions. Upon identifying a number of weaknesses that prevent LMRL2 from consistently converging upon optimal joint-policies we propose a new version of leniency, *Distributed-Lenient Q-learning* (DLQ). We find DLQ delivers state of the art performances in strategic-form and Markov games from *Multi-Agent Reinforcement Learning* literature. We subsequently scale leniency to MADRL, introducing *Lenient (Double) Deep Q-Network* (LDDQN). We empirically evaluate LDDQN with extensions of the Cooperative Multi-Agent Object Transportation Problem [26], finding that LDDQN outperforms hysteretic deep Q-learners in domains with multiple dropzones yielding stochastic rewards. Finally, to evaluate *deep* ILs along each pathology dimension we introduce a new MADRL environment: *the Apprentice Firemen Game* (AFG). We find lenient and hysteretic approaches fail to consistently learn near optimal joint-policies in the AFG. To address these pathologies we introduce *Negative Update Intervals-DDQN (NUI-DDQN)*, a MADRL algorithm which discards episodes yielding cumulative rewards outside the range of expanding intervals. NUI-DDQN consistently gravitates towards optimal joint-policies in deterministic and stochastic reward settings of the AFG, overcoming the outlined pathologies.

Acknowledgements

This thesis completes a journey that began just over six years ago, when I traded my job as a security manager at an investment bank in Switzerland for a chance to study the field of artificial intelligence at the University of Liverpool. I have grown a lot both personally and professionally since making this decision. More importantly, Liverpool has become a home for my family, in particular my two sons Payton and Michael, who have both developed *scouce* accents.

It goes without saying that I would not have reached this point without receiving support from a lot of people. Here is my attempt to give thanks to everyone who helped me along the way. I apologize in advance for anyone whom I fail to mention.

First I have to thank my wife Nui. To provide sufficient context, we were expecting our second child, Payton, following the completion of my BSc in Artificial Intelligence in 2016. We spent a significant amount of time considering the feasibility of me attempting a PhD while supporting and raising a young family. In the end we concluded that we would regret not attempting this path, and I am proud to say that together we have faced and overcome every single challenge along the way. Nui, thank you for your patience over the past three years. In particular during the weeks where I have been away on trips and you have had to look after the kids by yourself, and the countless weekends (and holidays) where I have been preoccupied with my research. You have sacrificed a lot, and I just want to let you know that I really appreciate everything you have done to support me.

Payton and Michael, thank you for constantly reminding your dad what is important in life. My progress pales in comparison to your recent achievements. Watching the two you learn to read, swim and master countless other activities has been a constant source of inspiration. I fear that I have often been preoccupied and impatient due to my attempts at balancing my research with demonstrating and other work related activities. For this I apologize, and I will do my best to make it up to both of you. I also want to thank my parents, Chris and Pat, and my nan, Joyce, for providing me with the necessary tools and work ethic to embark on this journey.

I have been fortunate to have received a significant amount of support and input from both my primary and secondary supervisors, Karl Tuyls and Rahul Savani. Both have become my role models with regards to my research activities and also in everyday life. Karl, thank you for introducing me to the world of multi-agent learning. For most of my adult life I have been looking for a topic to immerse myself into, and you have provided

me with the opportunity, guidance and freedom to explore my research interests ever since my final year undergraduate project. Rahul, thank you for looking out for me and always being there when I needed you. I really appreciate all the opportunities that you have provided, and thank you for taking a keen interest in my research activities throughout my PhD. I also want to take this opportunity to thank Daan Bloembergen for his guidance and support. I hope that I can continue to collaborate with the three of you in the years to come.

Learning to be researcher and a parent have not been the only challenges that I have faced throughout this experience. I am an introvert by nature, and if I am honest the prospect of running tutorial sessions used to terrify me. I therefore want to take this opportunity to thank Irina Biktasheva for essentially headhunting and forcing me to provide tutorials for the department's Biocomputation module. While I found this prospect very daunting when you first approached me, I am very grateful that you pushed me outside of my comfort zone, and allowed me to discover that I do in fact enjoy teaching.

Since February this year I have also been working as a data scientist in our university's Geographical Data Science Lab (GDSL). Balancing writing up my PhD thesis with my research activities for the GDSL has not always been an easy task, and I want to thank Alex Singleton and Mark Green for all their support during the past months. I also want to thank James Butterworth, Jacopo Castellini, Tom Spooner, Shan Luo, Frans Oliehoek, Danushka Bollegala, Shayegan Omidshafiei, Harry Flore and Adriaan Broer for valuable feedback and conversations throughout my PhD. Furthermore, I want to thank the HAL Allergy Group for partially funding my PhD and gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan X Pascal GPU that enabled this research.

The past few years have included a number of highs and lows. A particular challenge has been the recent passing of our colleague and friend Benjamin Schnieders. Benjamin, I want to take this opportunity to thank you for being a part of my life. I learnt a lot from you, and I wish I could have given you more in return. You had a positive influence on not just me, but every member of the smARTLab, and you are missed very much; not a day goes by without us thinking about you.

Chapter 1

Introduction

Intelligent autonomous agents and robots have the potential to reshape our society over the coming decades. This prediction is based on recent success stories, where solutions have been found for challenging virtual and real world problems, including: DeepMind's AlphaGo beating Go world champion Lee Sedol [168, 169], the coordination of aerial vehicles [22, 27, 51, 150], and the emergence of self-driving cars [83, 118]. A significant number of these successes can be attributed to breakthroughs in the field of *deep learning*, enabling *deep neural networks* to learn solutions to problems that humans solve using intuition [55, 56, 74, 95, 103, 119, 128].

Deep neural networks can be trained to extract compact features from complex high dimensional input samples [55, 103, 133]. Their flexibility has led to breakthroughs in numerous fields, including computer vision [55, 92, 103], language processing [34, 173, 220] and generative modeling [56, 65, 96, 158, 171]. Deep neural networks are also widely used as function approximators in the field of reinforcement learning, enabling behaviour policies to be learned using high-dimensional observations, thereby establishing the field of *deep reinforcement learning* [71, 72, 105, 127–129, 202].

This thesis focuses on deep reinforcement learning within the context of *multi-agent systems*, a type of distributed system inhabited by autonomous agents that are capable of interacting with each other [5, 17, 137, 194, 195, 199, 200, 214]. Both static and adaptive agents may inhabit multi-agent systems [88]. However, adaptive agents are more desirable, as interactions can have unforeseen consequences, while the dynamics of an environment can potentially change over time [5, 25, 138]. As we shall see, learning in multi-agent systems is far from trivial, even within domains that do not require a deep reinforcement learning approach. However, there are numerous applications where using a centralized single-agent learning approach is infeasible, due to physically distributed components, conflicting objectives, or a scarcity in resources [19, 137, 157, 174, 181]. Markets that can benefit from adaptive agents range from virtual environments to real-world settings, including recycling robots [138, 183], sensor networks [15, 125, 138, 163, 164], decentralized network routing [137], smart cities [130], multi-robot coordination [3, 15, 32], collision avoidance [22, 31], traffic control [115, 116, 134, 137, 138], distributed

load-balancing [137], electronic auctions [137], trading [16, 76, 175], resource allocation [10, 41, 42, 166], and computer games [46, 66].

1.1 Learning in Multi-Agent Systems

Agents situated within multi-agent systems frequently receive observations and a numeric feedback signal while interacting with their environment [15, 25, 146, 197, 200]. Feedback signals can be sparse and are typically less informative than the labels required by supervised learning [181, 184]. Furthermore, learners typically receive insufficient information to infer the intentions and goals of other agents within the system [195]. As a result adaptive agents implemented with incentive based learning algorithms are frequently applied to multi-agent systems, e.g., reinforcement learning or co-evolutionary algorithms [38, 184]. Using the feedback signal, incentive based learners attempt to maximize the rewards received through interacting with their environment [146, 184]. Reinforcement learning agents for instance learn via trial and error, using a scalar reward (feedback) signal to estimate the expected utility for executing an action given an observation [184]. However, despite a large number of reinforcement learning algorithms having convergence guarantees in single-agent environments [25, 183], approaches such as Q-learning are known to fail to converge upon optimal policies, even in relatively simple multi-agent systems [17, 38, 66, 78, 90, 120, 145, 149, 157, 174, 181]. Multi-agent learning literature meanwhile provides a rich taxonomy of pathologies that adaptive agents must overcome in order to converge upon optimal policies. This has resulted in a number of incentive based learning algorithms being extended to help agents overcome the identified pathologies [17, 90, 120, 121, 145, 148, 209].

1.2 Motivation & Scope

The majority of the multi-agent reinforcement learning algorithms discussed in this thesis have been designed to facilitate coordination among learning agents. The objective is to enable two or more agents to reach a consensus regarding which joint actions to perform [90]. Our focus is on fully cooperative *team games* with one global feedback signal, where each agent receives an equal reward at each time step [121].

Typically two types of learning are studied in cooperative multi-agent systems: *joint-action learning* (also known as *team learning*) and *concurrent learning* [146, 200]. In joint-action learning one agent learns a policy for all agents within the system, whereas concurrent learning consists of each agent learning a policy independently [47, 66, 146, 157, 174, 181, 200]. Traditional single-agent learning approaches can perform well when applied to joint-action learning. However, this approach does not scale well, as increasing the number of agents results in an exponential increase of the size of the observation-action space [28, 38, 66, 78, 120, 157, 174, 181]. As a result maintaining utility values for each observation-action combination received from all agents within the system can

become infeasible for a single-agent. A further criticism of joint-action learning is that the number of real world domains where an entity can observe and control all agents within the system is limited [122]. In many multi-agent systems information from other agents is unavailable due to a physically distributed architecture or limitations with regards to the agents' ability to communicate with each other [8, 122, 174, 188]. Finally, joint-action learners fail to consistently converge upon optimal joint policies, both in traditional and high-dimensional settings [33, 146, 181]. This finding is disturbing, given that agents have far less information in real world settings, where frequently the actions taken by the other agents cannot be observed [90, 146].

In this thesis we focus on *isolated concurrent learners*, which are better known as *independent learners*. Each independent learner is essentially implemented with a single-agent reinforcement learning algorithm, and therefore other agents are treated as part of the environment [15, 25, 200]. Despite being unable to observe the actions taken and rewards received by other learners [33, 101], independent learners are considered more scalable than joint-action learners [15, 28, 146, 157, 174]. However, the convergence guarantees of most machine learning methods are based on the assumption that the environment's dynamics remain stationary [15, 25, 183, 200]. Introducing independent learners to a system violates this assumption, and as a consequence each learning agent must continuously adapt to the non-stationary policies of the other agents within the system [38, 146, 200]. This particular pathology is known as the *moving target problem* (non-stationarity problem), where small changes in the learned policies can have large unpredictable consequences in the emergent behaviours of the independent learners collectively [38, 146, 195, 200]. Other pathologies that can prevent independent learners from converging upon an optimal joint-policy include [1]:

- **Miscoordination:** This pathology occurs in domains where more than one optimal policy exists for each agent, however, only some combinations of these individual optimal policies are compatible across agents [33, 90, 122, 148]. To provide an example of miscoordination we shall consider a task with two robots A and B, who must take turns passing through a narrow doorway. The robots can choose between two actions: *wait* and *move*. Assuming each robot learns a policy where an action is selected with 100% probability, we have two optimal policies: $\langle A(wait), B(move)\rangle$ and $\langle A(move), B(wait)\rangle$. However, joint-policies using the remaining action combinations will lead to miscoordination, $\langle A(wait), B(wait)\rangle$ and $\langle A(move), B(move)\rangle$, where neither robot can reach the other side of the door.

- **The stochasticity problem:** Reinforcement learners are often situated in domains that yield stochastic rewards (e.g., pulling a lever on a one-armed bandit which yields a reward chosen from a stationary probability distribution [4, 184]) and non-deterministic transitions (e.g. walking on a slippery surface [61]).

[1] We provide formal definitions for each pathology in Chapter 3, Section 3.1.

- **The alter-exploration problem:** Managing the exploration-exploitation trade-off is one of the keys to reinforcement learning's success [184]. However, exploration adds noise to the utility value estimates maintained by independent learners, who must take into account the probability of at least one agent exploring at each time-step [121].

- **Relative overgeneralization:** Independent learners can be drawn towards sub-optimal wide peaks in the reward search space due to a greater likelihood of achieving collaboration there. Relative overgeneralization occurs when pairing an independent learner's available actions with arbitrary actions by the other agents results in a sub-optimal action having the highest utility estimate [210].

- **Deception:** Independent learners can be deceived by states yielding a high local reward, that ultimately lead on to poor future rewards (also known as the delayed reward problem) [209]. As a result greedy agents can be led astray from high-value trajectories.

A significant number of independent learning algorithms have been proposed in multi-agent reinforcement learning literature to mitigate the above pathologies [15, 17, 28, 90, 120, 148, 148, 149, 157, 174, 209, 210]. Many of these algorithms are inspired by human coping mechanisms [90]. For instance, despite having a common objective, cooperative learning agents are not guaranteed to converge upon an optimal joint policy, especially when confronted with a reward space where miscoordination is associated with high penalty values [90]. Humans can be observed to adopt an optimistic disposition towards each-other when attempting to solve a problem with a high likelihood of miscoordination. As per behavioral game theory [43], football players for instance repeatedly attempt challenging passes when awarded a free-kick outside the opponent's penalty area. Despite safe passing options being available, that will allow the team to maintain possession, the free-kick taker remains optimistic that the risky pass could catch the defense off-guard and lead to a goal. Inspired by this concept numerous *optimistic* approaches have been applied to incentive based learning. Prominent examples frequently discussed in multi-agent reinforcement learning literature include *distributed Q-learning* [100], *hysteretic Q-learning* [120, 141] and *leniency* [17, 148, 149, 197].

However, despite these efforts none of the above approaches can completely address the outlined pathologies, even in relatively simple *stateless* games with a small discrete action space [17, 28, 33, 90, 148, 148, 149]. Furthermore, current approaches towards independent learning within high-dimensional domains with a large state-space can amplify multi-agent learning pathologies [46, 66]. For example: the architectures used in deep reinforcement learning are trained using stochastic gradient descent [128, 129]. However, gradient based methods are likely to fail when trained via strongly correlated updates that break the independent and identically distributed (i.i.d.) data assumption [160]. To break the temporal correlation between sequences of encountered states Mnih et al. [128, 129] turn to *experience replay memories* that store state-transition

tuples as the agent explores the environment [106]. However, the use of experience replay memories amplifies the moving target problem in *multi-agent deep reinforcement learning*, due to each agent's network being trained using potentially obsolete samples that may no longer reflect the current dynamics of the environment. In this thesis we empirically and algorithmically evaluate to what extent leniency [17, 145, 148] and other optimistic extensions can help *deep* reinforcement learning agents overcome multi-agent learning pathologies and converge upon an optimal joint-policy in fully-cooperative settings.

1.3 Problem Statement

In the previous section we observe that independent learning remains an open problem. Recently Wei and Luke [209] conducted an evaluation of state of the art independent learning algorithms. A variation of leniency called *Lenient Multi-agent Reinforcement Learning 2* (LRML2) emerged as the most robust algorithm. However, none of the approaches evaluated can consistently overcome a combination of relative overgeneralization and the stochasticity problem. While LMRL2 is less susceptible towards these pathologies, a significant number of LMRL2 runs failed to converge upon optimal joint-policies when confronted with excessive stochasticity. This finding is worrying, especially given that the authors' evaluation was conducted using two-player games that were either stateless, or only consisted of a small number of low-dimensional states. Questions remain whether LMRL2 struggles with the above pathologies are due to a sub-optimal choice of hyperparameters and Q-value initialization, or if further algorithmic modification are necessary to improve the robustness of lenient learners.

A further open question is the scalability of independent learning approaches that do achieve high convergence rates upon optimal joint-policies within low-dimensional settings to domains with a large high-dimensional state-space. While scalability is a challenge for most machine learning techniques, this is particularly the case for multi-agent learning [25]. Panait and Luke [146] observe that many traditional methods are likely to fail once applied to partially observable environments inhabited by a large number of agents. Furthermore, scaling frequently requires algorithmic modifications [121]. For example: a number of independent learning algorithms rely on count based methods, where learners keep track of the number of times an action has been executed within a given state [17, 90, 121, 148, 149, 190, 197]. Lenient learners for instance map a decaying temperature value to each state-action pair, which determines the amount of leniency that is applied towards a utility value update [17, 145, 148]. The temperature value mapped to the state-action pair is decayed following a utility value update, resulting in lenient learners applying less leniency towards utility value updates for frequently visited state-action pairs. In low dimensional settings temperature values can be maintained in a discrete low-dimensional data structure. However, further considerations are required

for maintaining temperature values for semantically similar state actions pairs within a high-dimensional state-space.

For independent learning algorithms that have been scaled to multi-agent deep reinforcement learning, considerations are required regarding evaluation [78]. Temporally extended high-dimensional domains are required where the learners' susceptibility to multi-agent learning pathologies can be established. However, while multi-agent reinforcement learning literature provides numerous repeated stateless and low-dimensional games to determine an algorithm's robustness towards specific pathologies, equivalent domains are lacking for multi-agent deep reinforcement learning. We therefore consider that multi-agent deep reinforcement learning can benefit from domains where independent learners' susceptibility towards each of the above pathology dimensions can be established.

1.4 Research Questions

Below are five research questions that have been distilled from the problem statement outlined in the previous section.

Q1: To what extent can existing independent learning approaches mitigate multi-agent learning pathologies within n-player repeated single-stage strategic-form games?

Chapter 4 & 5

Q2: To what extent can *synchronized lenient learners* reduce miscoordination?

Chapters 5 & 7

Q3: To what extent can we design high-dimensional domains for evaluating the susceptibility of deep reinforcement learners towards multi-agent learning pathologies?

Chapters 6 & 7

Q4: To what extent can *leniency* be scaled to multi-agent deep reinforcement learning?

Chapter 6

Q5: To what extent can independent learners overcome relative overgeneralization while making decisions using noisy utility values backed up from stochastic follow-on transitions?

Chapter 7

The chapter(s) within which each of these questions is addressed is provided in italics at the end of each question. Concrete answers are subsequently provided to each question in Chapter 8.

1.5 Relation to Published Work

The background knowledge presented in Chapters 2 and 3 is based on the work of other authors, where we cite relevant sources from literature. In Chapters 4 and 5 we present our most recent work, which is in preparation for a submission to the *Journal of Machine Learning Research*. In Chapter 4 we re-evaluate existing independent learner baselines within n-player repeated single-stage strategic-form games, while in Chapter 5 we introduce a novel *leniency* algorithm. Finally, Chapters 6 and 7 are based on two (full) conference publications, respectively:

- Gregory Palmer, Karl Tuyls, Daan Bloembergen, and Rahul Savani, Lenient Multi-Agent Deep Reinforcement Learning, In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, 2018, pp. 443–451.
- Gregory Palmer, Rahul Savani, and Karl Tuyls, Negative Update Intervals in Deep Multi-Agent Reinforcement Learning, In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, 2019, pp. 43–51.

Other Research: Not all research conducted over the past three years fits into the scope of this thesis. Other efforts include a collaboration with the HAL Allergy Group on the automated inspection of opaque liquid vaccines and work conducted with Benjamin Schnieders, Shan Luo and my supervisor Karl Tuyls on the topic of one-shot object segmentation for industrial robotics:

- Benjamin Schnieders, Shan Luo, Gregory Palmer, and Karl Tuyls, Fully Convolutional One-Shot Object Segmentation for Industrial Robotics. In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, 2019, pp. 1161-1169.

Chapter 2

Preliminaries

This chapter outlines the core concepts that serve as the foundations of the contributions presented in this thesis. First an introduction to reinforcement learning is provided, followed by a summary of the breakthroughs that enabled the field of *deep* reinforcement learning [105, 127–129, 203]. Finally, we discuss concepts from game theory that have been used to describe and evaluate multi-agent learning [19, 33, 90, 120–122, 137, 167, 178, 194, 194, 200, 209].

2.1 Reinforcement Learning

Learning through trail-and-error via interaction with an environment is a concept that underlies the majority of theories on learning and intelligence [19, 33, 122, 184, 194, 200, 200, 207]. Reinforcement learning is a branch of machine learning based on this concept [184, 194, 200, 207], where idealized learners learn through trial and error while interacting with a dynamic environment [87, 122, 133, 137, 184, 200, 200, 207]. The learners can perceive the environment's current state, perform actions and observe their impact on the environment through a numeric reward signal (as depicted in Figure 2.1) [15, 88, 184, 200]. Inspired by the pleasure and pain signals observed in biological systems, a reward signal defines the desirability of the transition that take place within the environment, and can be seen as a stochastic function that maps state-action pairs to rewards [184, 194].

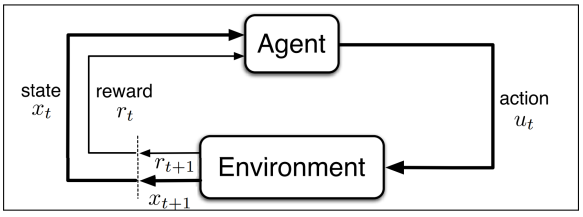


Figure 2.1: Diagram depicting the agent-environment interaction of an idealized reinforcement learning agent. Upon executing an action u_t at time t, the agent receives state x_{t+1} and reward r_{t+1} responses at time $t+1$. This illustration is adapted from Sutton and Barto [184].

Interactions with the environment produce large amounts of information regarding the consequences of actions [184, 207]. The objective of reinforcement learning agents is to utilize this information to maximize the total reward received while interacting with the environment [15, 87, 184, 194, 200]. Actions resulting in desirable transitions are to be reinforced, i.e., executed with a greater likelihood in the future [19, 184]. However, rewards may be stochastic, delayed or accumulated over a trajectory of state-transitions [19, 184, 208]. As a result action or value functions are used to estimate the long term rewards that an agent can expect to receive starting from the current state [184, 207]. Such functions account for the rewards received during follow-on states, while the reward signal only indicates the immediate reward. Therefore, while the immediate reward might appear less desirable, action and value functions can help determine long term benefits. The notion of expected utility can be used to guide the agents towards actions that lead to states with higher rewards [15, 122, 137, 184, 197].

Machine learning algorithms are typically said to belong to one of two groups: (i) *supervised learning*, where models are trained using labelled data; (ii) *unsupervised learning*, where algorithms are designed to find hidden structures in unlabelled data. However, Sutton and Barto [184] argue that reinforcement learning is in fact a third paradigm, which can be distinguished from supervised and unsupervised learning problems, as the agents: (i) are situated in a closed loop (see Figure 2.1); (ii) do not receive any labelled information regarding the desirability of actions; (iii) learn from reward signals received over multiple time steps.

A further distinction between reinforcement learning and other machine learning paradigms, is that a carefully considered trade-off is required between exploration and exploitation in order to discover preferable states and actions [15, 87, 88, 184, 194, 200]. Balancing this trade-off is non-trivial. Due to stochasticity each state-action combination must be explored sufficiently, in order to establish a reliable estimate of the expected reward [184]. As we shall see, further considerations are required within multi-agent settings, in particular within domains where exploration is likely to result in miscoordination [5, 33, 90, 120–122, 194].

Markov Decision Processes (MDPs) are well suited for modelling problems with delayed reinforcement [87]. Therefore, in the sections below we use MDPs to formalize the concepts outlined above.

2.2 Finite Markov Decision Processes

Sutton and Barto [184] consider any method capable of solving a MDP a reinforcement learning algorithm. Finite Markov Decision Processes describe a class of problems (fully observable environments) that defines the field of reinforcement learning, providing a suitable model to formally describe the interactions between reinforcement learners and their environment [79, 87, 108, 133, 153, 178, 184, 197]. Formally:

Definition 2.2.1 (Markov Decision Processes)**.** A MDP is a tuple $\langle \mathcal{X}, \mathcal{U}, \mathcal{R}, \mathcal{P}, \gamma \rangle$, where: $\mathcal{X}$ is a finite set of states; for each state $x \in \mathcal{X}$ there exists a finite set of possible actions $\mathcal{U}$; $\mathcal{R}$ is a real-valued payoff function $\mathcal{R} : \mathcal{X} \times \mathcal{U} \times \mathcal{X}' \to \mathbb{R}$, where $\mathcal{R}_u(x, x')$ is the expected payoff following a state transition from x to x' using action u; $\mathcal{P}$ is a state transition probability matrix $\mathcal{P} : \mathcal{X} \times \mathcal{U} \times \mathcal{X}' \to [0, 1]$, where $\mathcal{P}_u(x, x')$ is the probability of state x transitioning into state x' using action u; γ is a discount factor $\gamma \in [0, 1]$ weighting the value of future rewards.

In this thesis we shall assume that the environments proceed along evenly spaced discrete time-steps t [197]. Furthermore, a MDP is an environment within which every state has the Markov property: every state retains all relevant information independent of the history of transitions leading up to the state, therefore the state transition probability fully depends on the current state [178, 184, 194, 200, 207]. A policy within a MDP can be formulated as a function that takes a state $\mathcal{X}$ and returns an action $\mathcal{U}$ [88]:

$$\pi : \mathcal{X} \to \mathcal{U}. \tag{2.1}$$

Providing an agent is situated within a domain with the Markov property, and is given sufficient time to explore, then reinforcement learning guarantees convergence upon an optimal policy [85, 170, 184, 185, 194, 201]. As outlined above, the aim of a reinforcement learning agent situated within a MDP is to maximize the discounted sum of rewards [88]:

$$\sum_{t}^{\infty} \gamma^t \mathcal{R}_{\pi(x_t)}(x_t, x_{t+1}). \tag{2.2}$$

To obtain the value of being in a state x conditioned on the current policy π we can use the value function $\mathcal{V}^{\pi}(x)$ [88]:

$$\mathcal{V}^{\pi}(x) = \sum_{x'} \mathcal{P}_{\pi(x)}(x, x')[\mathcal{R}_{\pi(x)}(x, x') + \gamma \mathcal{V}^{\pi}(x')]. \tag{2.3}$$

The value is calculated recursively using all follow-on states x'. As is custom we define the optimal value function indicating the value of a state given an optimal policy $\pi^*(x)$ as $\mathcal{V}^*(x) = max_{\pi} \mathcal{V}^{\pi}(x)$, where $\pi^*(x)$ selects the action u with the maximum expected utility for state x [88]:

$$\pi^*(x) = \underset{u}{\operatorname{argmax}} \Big\{ \sum_{x'} \mathcal{P}_u(x, x')[\mathcal{R}_u(x, x') + \gamma \mathcal{V}^*(x')] \Big\} \tag{2.4}$$

If the transitions and reward functions for a MDP are known, then using the Bellman equation [13], *value iteration* can be applied to obtain estimates for the value function [88]:

$$\mathcal{V}(x) \leftarrow \max_{u} \Big\{ \sum_{x'} \mathcal{P}_u(x, x')[\mathcal{R}_u(x, x') + \gamma \mathcal{V}(x')] \Big\} \tag{2.5}$$

However, the focus of this thesis is on independent learners, which update their policies concurrently within a shared domain. As such, the reward and transition functions depend on the joint-actions taken by all agents within the system. Therefore the Markov property no longer holds, and we also no longer have any guarantees of convergence [17, 66, 78, 90, 120, 125, 137, 145, 149, 157, 174, 181, 194]. We therefore turn to *model free* reinforcement learning approaches.

2.3 Q-learning

The Q-learning algorithm introduced by Watkins [207, 208] is considered one of the most important breakthroughs in reinforcement learning [183]. Using a dynamic programming approach, the algorithm learns action-value estimates (Q-values) independent of the agent's current policy. Q-values are estimates of the discounted sum of future rewards (the return) that can be obtained at time t through selecting an action $u \in \mathcal{U}$ in a state x_t, providing the optimal policy is selected in each state that follows. Q-learning is therefore an *off-policy* temporal-difference learning algorithm.

In domains with a low dimensional state space Q-values can be maintained using a Q-table. Q-values are updated as follows: upon choosing an action u_t in state x_t according to a policy π, the Q-table is updated by bootstrapping the immediate reward r_{t+1} received in state x_{t+1} plus the discounted expected future reward from the next state, using a discount factor $\gamma \in (0, 1]$ and scalar α to control the learning rate:

$$Q_{t+1}(x_t, u_t) \leftarrow Q_t(x_t, u_t) + \alpha\big(r_{t+1} + \gamma \max_{u \in U} Q_t\left(x_{t+1}, u\right) - Q_t\left(x_t, u_t\right)\big) \tag{2.6}$$

However, sequential decision problems can have a high-dimensional state space. In such instances Q-values can be approximated using a function approximator, for instance using tile coding [11, 59, 70, 175] or a neural network [7, 97, 105, 127–129, 160, 185, 203]. The parameters θ of the function approximator can also be learned via experiences gathered by the agent while exploring their environment, choosing an action u_t in state x_t according to a policy π, and updating the Q-function by bootstrapping the immediate reward r_{t+1} received in state x_{t+1}, plus the expected future reward from the next state (as given by the Q-function) [128, 129, 203]:

$$\theta_{t+1} = \theta_t + \alpha\big(Y_t^Q - Q\left(x_t, u_t; \theta_t\right)\big)\nabla_{\theta_t} Q\left(x_t, u_t; \theta_t\right). \tag{2.7}$$

Here, Y_t^Q is the bootstrap target which sums the immediate reward r_{t+1} and the current estimate of the return obtainable from the next state x_{t+1} assuming optimal behaviour, discounted by $\gamma \in (0, 1]$ (Eq. (2.8)). The Q-value $Q\left(x_t, u_t; \theta_t\right)$ therefore moves towards the target by following the gradient $\nabla_{\theta_t} Q\left(x_t, u_t; \theta_t\right)$.

$$Y_t^Q \equiv r_{t+1} + \gamma \max_{u \in U} Q\left(x_{t+1}, u; \theta_t\right). \tag{2.8}$$

2.4 Exploration

Regarding the exploration/exploitation trade-off that is critical for reinforcement learning agents to converge, *ϵ-greedy* and *soft-max exploration* are two popular strategies frequently referred to in reinforcement learning literature [15, 122, 184, 200]. For ϵ-Greedy exploration the agent selects action $u = \text{argmax}_u Q(x, u)$ with a probability $1 - \epsilon$ and a uniformly random action with probability ϵ [129, 184, 193, 200, 216]. The exploration rate ϵ can either remain constant or be decayed over time using a decay factor μ. Soft-max action selection policies meanwhile compute the probability with which each action should be selected using a Boltzmann distribution [15, 33, 170, 200]:

$$P(u|x) = \frac{\exp\left(\frac{Q(x,u)}{\tau}\right)}{\sum_{u' \in \mathcal{U}} \exp\left(\frac{Q(x,u')}{\tau}\right)}, \tag{2.9}$$

where τ can be decayed to ensure that the action selection becomes greedier over time [14, 111]. Out of the two methods ϵ-greedy is more standard in Q-learning. Tuning the Boltzmann strategy is frequently a non-trivial task [**?**]. Nonetheless, the Boltzmann strategy is frequently combined with Q-learning, as we shall see in Chapter 3. In deployment agents trained using ϵ-greedy typically use an argmax action selection policy, where the action with the largest Q-value is selected at each stage.

2.5 Deep Learning

Deep neural networks are trained to extract compact features from complex high dimensional inputs, combining layers of hierarchical features into ever more complex concepts [55, 103, 133]. In this thesis we shall be working with Convolutional Neural Networks (ConvNets), which are geared towards extracting features from inputs in the form of arrays and tensors [103]. For instance, a ConvNet trained to classify images consists of layers of neurons, with the first layer extracting edges, which are combined into corners and contours by the next layers, before subsequently being combined to form the object parts that enable a classification [55, 103]. Traditional ConvNet architectures consist of multiple linear convolution and pooling layers stacked up on top of each other, followed by fully connected layers, which precede the output layer [55, 103, 180]. The convolutional layers are banks of filters which are convoluted with an input to produce an output map [55, 84, 103]. A non-linear activation function is subsequently applied to the output map such as the Rectified Linear Unit (ReLU) [135]. Through stacking multiple non-linear layers the network can be trained to implement complex functions, that are sensitive towards minute details within inputs, while simultaneously being able to ignore less relevant features [103]. Providing the individual modules of the network are smooth functions of their respective inputs, the network can be trained to minimize an objective

function by computing gradients via the back-propagation procedure [55, 103]. Deep neural networks can therefore be trained end-to-end using stochastic gradient descent.

ConvNet's strength lies in their large learning capacity, which can be adjusted through changing the network's depth and breadth [95, 215]. Furthermore ConvNets take advantage of assumptions regarding the location of pixel dependencies within images, reducing the number of weighted connections compared to a fully-connected neural network [95]. This ability of deep neural networks to learn hierarchies of concepts has proven valuable in numerous fields, including computer vision [55, 92, 103], generative modeling [56, 65, 96, 158, 171], speech and language processing [34, 173, 220]. Deep neural networks have also been widely used as function approximators in the field of incentive based learning, into which both *reinforcement learning* and *evolutionary algorithms* fall, allowing agents to learn control policies directly for high-dimensional observations of their environment.

2.6 Deep Q-Learning

In deep reinforcement learning a multi-layer neural network is used as a function approximator, mapping a set of n-dimensional state variables to a set of m-dimensional Q-values $f : \mathbb{R}^n \to \mathbb{R}^m$, where m represents the number of actions available to the agent [128, 129]. Over the past decades neural networks have often been used as function approximators in reinforcement learning literature [107, 126, 182, 185]. However, as mentioned, the recent successes in the field of deep learning have enabled neural networks to identify intricate structures and extract compact features from complex high-dimensional samples, such as images and frame-sequences [55, 92, 103]. Using deep neural network architectures as function approximator has allowed reinforcement learning approaches to master complex environments with a high-dimensional state space [106, 127, 129, 168, 203]. The *Deep Q-network* (DQN) introduced by Mnih et al. [129] updates parameters θ using stochastic gradient descent, randomly sampling past transitions experienced by the agent that are stored within an experience replay memory $\mathcal{D}$ [106, 129, 133, 160]. Transitions are tuples (x, u, x', r) consisting of the original state x, the action u, the resulting state x' and the immediate reward r. The network is trained to minimize the time dependent loss function,

$$L_t(\theta_t) = \mathbb{E}_{(x,u,x',r)\sim U(\mathcal{D})}\Big[\left(Y_t - Q(x, u; \theta_t)\right)^2 \Big], \tag{2.10}$$

where $(x, u, x', r) \sim U(\mathcal{D})$ represents minibatches of experiences drawn uniformly at random from the set of samples stored inside the experience replay memory $\mathcal{D}$ [128, 129], t the current iteration, and Y_t is the target:

$$Y_t \equiv r + \gamma Q(x', \underset{u\in U}{\operatorname{argmax}}\, Q(x', u; \theta_t); \theta'_t). \tag{2.11}$$

Equation (2.11) is a form of double Q-learning [201] in which the target action is selected using weights θ, while the target value is computed using weights θ' from a

target network. The target network is a more stable version of the current network, with the weights being copied from the current to the target network after every n transitions [203]. Double-DQNs have been shown to reduce overoptimistic value estimates [203]. This is interesting for our current work, since we are interested in scaling optimistic independent learning approaches to *multi-agent deep reinforcement learning*, to allow the learning agents to converge towards an optimal joint policy.

2.7 Policy Gradient Methods

While a DQN is used to approximate a value function, it is worth noting that deep neural networks can also be trained to approximate a policy [6, 48, 78]. If we consider value function and policy methods as individual sets, then at the intersection we find actor-critic methods, where a critic learns a value function that is used to guide the training of the actor [185]. Policy based methods have a number of advantages over value based methods that use an implicit policy, such as being suitable for physical control tasks. Lillicrap et al. [105] for instance introduced *Deep Deterministic Policy Gradient* (DDPG), an off-policy actor-critic architecture that can master tasks with a high-dimensional continuous action space. DDPG also uses an experience replay memory $\mathcal{D}$ to train the two networks. In contrast Asynchronous Advantage Actor-Critic (A3C) replaces $\mathcal{D}$ with an asynchronous training scheme, obtaining uncorrelated state-transitions samples from multiple CPUs. The method is on policy, with each of the agents computing their respective gradients locally, before using the gradients to update the global network prior to synchronization. While the contributions in this thesis focus on scaling modified Q-learning algorithms that encourage cooperation, we shall discuss modified versions of the above actor-critic algorithms while providing a recap of the current state of the art of multi-agent deep reinforcement learning in Section 3.3.

2.8 Game Theory

Reinforcement learning was traditionally intended for single agent applications with stationary dynamics. However, this assumption no longer holds in the multi-agent case. As a result we must now consider the learning and interaction mechanisms of a group of agents inhabiting the environment [33, 86, 197, 200]. A popular approach towards multi-agent learning is to deploy independent learning agents that are each implemented with a single-agent reinforcement learning algorithm [15, 79, 90, 120–122, 197, 209]. Each agent independently learns a policy, and treats the other agents as part of the environment [15, 79]. However, convergence guarantees only exist for reinforcement learning algorithms in domains where: (i) state transition probabilities $\mathcal{P}$ remain stationary; and (ii) every state has the Markov property [101, 137, 199]. By ignoring each other the multi-agent learning problem effectively becomes a single-agent learning problem, where interactions with cohabiting agents are implicitly observed in what is considered a stochastic non-stationary environment [15, 38, 137]. The Markov property therefore no longer applies

to multi-agent systems inhabited by independent learning agents [101, 137, 199]. Due to learners updating their policy in parallel the state-transition probabilities are no longer guaranteed to remain stationary [15, 17, 66, 78, 90, 137, 157, 174, 181, 194, 197]. MDPs are therefore unable to capture the interdependencies of systems inhabited by independent learners. However, due to multiple agents interacting we have a *game*. Thus, in this section we introduce relevant models from game theory, that provide a suitable framework through which to describe and evaluate multi-agent learning [33, 90, 108, 137, 167].

To express ideas clearly game theory relies on a considerable amount of notations and definitions [53]. To ensure that interpretations remain consistent across disciplines, caution is necessary when applying terminology from game theory to multi-agent reinforcement learning. This section therefore provides a taxonomy of the game theoretic terminology and definitions used within the context of this thesis. We focus on independent learning within two types of games: stateless games that assume the environment remains stationary, and Markov games, a direct generalization of MPDs, which assume the agent interacts with a dynamic environment [108, 137, 167]. In the sections below we will formally define each of these game types. Furthermore, the focus of this thesis is on fully-cooperative settings. We shall therefore also introduce the notion of *team-games*. Although the strategic interactions in game theory typically take place between *players*, we shall use the terms *agent* and *player* interchangeably. In addition, to prevent confusion in later chapters, we refer to actions taken by agents in stateless games as $a \in \mathcal{A}$ while actions in Markov games are denoted as $u \in \mathcal{U}$. A summary of the correspondence of terminology between the domains of game theory and reinforcement learning is provided in the Table 2.1 [15].

Reinforcement Learning	**Game Theory**
Environment	Game
Agent	Player
Action	Action
Policy	Strategy
Reward	Payoff

TABLE 2.1: Corresponding terminology for reinforcement learning and game theory.

2.8.1 Strategic-Form Games

Game theory uses mathematical objects to define strategic interactions between players [137]. For simplicity research in multi-agent reinforcement learning initially focused on strategic-form (normal-form) games, which are known as both stateless and single-state games [33, 90, 120–122]. Formally:

Definition 2.8.1 (Strategic-Form Game)**.** A n-player strategic-form game (also known as a normal form game and single-stage game) is defined as a tuple $(n, \mathcal{A}_{1...n}, \mathcal{R}_{1...n})$ where n represents the number of players, $\mathcal{A}_{1...n}$ the joint action space $(\mathcal{A}_1 \times ... \times \mathcal{A}_n)$, with $\mathcal{A}_p$ being the set of actions available to player p, and $\mathcal{R}_p$ is the reward function $\mathcal{R}_p : \mathcal{A}_1 \times ... \times \mathcal{A}_n \to \mathbb{R}$ for each player p [20].

We note that in strategic-form games players make their choices *simultaneously*. Strategic-form games with $n = 2$ are commonly known as bimatrix games. As the name indicates, bimatrix games can conveniently be captured using a matrix. Figure 2.2 provides an example. Players I and II are referred to as row and column players respectively. Therefore, the rows represent the actions available to player I, while the columns capture the actions available to player II. The matrix cells contain the reward that each player receives upon applying a joint-action. For n-player strategic-form game a n-dimensional tensor can be used to capture strategic interactions.

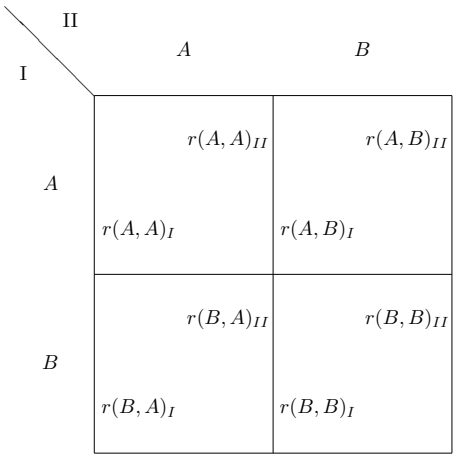


Figure 2.2: Two-player strategic-form game example, where players I and II are referred to as row and column players respectively. The matrix cells contain the reward that each player receives upon applying a joint-action.

Each player's behaviour is defined by a (mixed) strategy π, which maintains preferences over the available actions [15, 194, 194]. Formally:

Definition 2.8.2 ((Mixed) Strategy)**.** A strategy is a probability vector $\pi_p = (\pi_p^1, ..., \pi_p^k)$, which assigns a probability π_p^a to each action $a \in \mathcal{A}$ for player p, $\pi_p : \mathcal{A}_p \to [0, 1]$, such that the sum of the probability vector π_p is equal to 1, $\sum_{a \in \mathcal{A}_p} \pi_p^a = 1$. If there exists an action a where $\pi_p^a = 1$, then the strategy π_p is a *pure strategy*, where for all other actions $a' \in \mathcal{A}_p$, where $a' \neq a$, we have $\pi_p^{a'} = 0$. Otherwise we have a *mixed strategy*.

The outcome of a game is conditioned on the behaviour of each player, and is therefore determined by the players' joint-strategies profile:

Definition 2.8.3 (Strategy Profile)**.** A strategy profile is a vector $\overrightarrow{\pi} = (\pi_1, ..., \pi_n)$ that contains a strategy for each of the n players.

2.8.2 Markov Games

Strategic-form games provide a framework within which to study stateless multi-agent reinforcement learning algorithms. However, the MDP framework used for studying singe agent reinforcement learning also accounts for probabilistic transitions between states. As per the above definition strategic-form games are stateless games. Therefore we require a richer framework that generalizes both strategic-form games and MDPs [108, 137, 167, 200]. Here game theory also offers a solution. In 1953 Shapley extended strategic-form games to Markov games (also known as *stochastic games*) [165]. Formally:

Definition 2.8.4 (Markov Games)**.** A Markov game is defined as a tuple $(n, \mathcal{X}, \mathcal{U}, \mathcal{P}, \mathcal{R})$, that has a finite state space $\mathcal{X}$, for each state $x \in \mathcal{X}$ a joint action space $(\mathcal{U}_1 \times ... \times \mathcal{U}_n)$, with $\mathcal{U}_p$ being the number of actions available to player p, a state transition function $\mathcal{P} : \mathcal{X}_t \times \mathcal{U}_1 \times ... \times \mathcal{U}_n \times \mathcal{X}_{t+1} \to [0, 1]$, returning the probability of transitioning from a state x_t to x_{t+1} given an action profile $u_1 \times ... \times u_n$, and for each player p a reward function: $\mathcal{R}_p : \mathcal{X}_t \times \mathcal{U}_1 \times ... \times \mathcal{U}_n \times \mathcal{X}_{t+1} \to \mathbb{R}$ [165]. We allow *terminal states* (absorbing states) at which the game ends. Finally, each state $x \in \mathcal{X}$ is fully-observable.

Therefore, Markov games assume Markovian transitions that are conditioned on transition probabilities for the joint-actions of all players [108]. Reducing the number of players to one converts the Markov game into a MDP [137]. Furthermore, a strategic-form game can be thought of as a Markov game with one state, where all joint-action combinations result in a transition into an absorbing state [104, 137]. A player's strategy for Markov games is defined as follows:

Definition 2.8.5 (Strategies in Markov Games)**.** For each player p, the strategy π_p represents a mapping from the state space to a probability distribution over actions: $\pi_p : \mathcal{X}_p \to \Delta(\mathcal{U}_p)$.

Therefore, transitions within a Markov game are determined by a joint strategy:

Definition 2.8.6 (Joint Strategy)**.** The notation $\boldsymbol{\pi}$ refers to a joint strategy of all players. Joint strategies excluding player p are defined as $\boldsymbol{\pi}_{-p}$. The notation $\langle \pi_p, \boldsymbol{\pi}_{-p} \rangle$ refers to a joint strategy with player p following π_p while the other players follow $\boldsymbol{\pi}_{-p}$.

2.8.3 Partially Observable Markov Games

Over the past decades Markov games have established themselves as a mathematical framework for studying multi-agent learning [104, 108, 122, 137]. However, many multi-agent environments are in-fact partially observable. A famous example of a domain with partial observability is multiple agents playing football (soccer), where each player only receives a local (potentially noisy) observation using their sensors [138, 177]. Formally, games where learners only receive noisy partial observations of their environment can be defined as partially observable Markov games [104]:

Definition 2.8.7 (Partially Observable Markov Games)**.** A Partially observable Markov game is defined as a tuple $(n, \mathcal{X}, \mathcal{O}, \mathcal{U}, \mathcal{P}, \mathcal{R})$ that has a finite state space $\mathcal{X}$, an observation function $\mathcal{O}_p : \mathcal{X} \rightarrow \mathbb{R}^d$, which returns a d-dimensional observation for player p, for each state $x \in \mathcal{X}$ a joint action space $(\mathcal{U}_1 \times ... \times \mathcal{U}_n)$, with $\mathcal{U}_p$ being the number of actions available to player p, a transition function $\mathcal{P} : \mathcal{X}_t \times \mathcal{U}_1 \times ... \times \mathcal{U}_n \times \mathcal{X}_{t+1} \rightarrow [0, 1]$, returning the probability of transitioning from a state x_t to x_{t+1} given an action profile $u_1 \times ... \times u_n$, and a reward function: $\mathcal{R}_p : \mathcal{X} \times \mathcal{U}_1 \times ... \times \mathcal{U}_n \times \mathcal{X}_{t+1} \rightarrow \mathbb{R}$ for each player p [104, 165]. We allow *terminal states* (absorbing states) at which the game ends.

2.8.4 Repeated Games

Repeated games are frequently used as a test bed for novel multi-agent reinforcement learning algorithms [33, 90, 120–122, 197, 197, 200]. In a *repeated strategic-form game* the agents repeatedly play the same strategic-form game while being allowed to choose a different action during each iteration. We observe that in the multi-agent reinforcement learning literature the term *repeated game* has become synonymous with repeated strategic-form game [120, 121, 137, 209]. However, a Markov game can also be repeated [35, 35, 44, 77]. A *repeated Markov game* (also known as a *repeated stochastic game*) is a Markov game where there exists at least one state $x \in \mathcal{X}$ that is absorbing (terminal) [35, 35, 44, 77]. Upon entering an absorbing state the game is reset, and the players play the next iteration [35, 35, 44, 77]. Therefore, we argue that *repeated strategic-form games* should be distinguished from *repeated Markov games* (or *repeated stochastic games*) to promote consistency across disciplines.

The criteria used to evaluate repeated games include the total reward (for games that are not played infinitely), discounted future payoffs and the expected average reward [206]. In Markov games for instance we can compute the expected gain (also known as the expected sum of future rewards) for each player:

Definition 2.8.8 (Expected Gain)**.** Given a joint policy $\boldsymbol{\pi}$ the gain for each player p starting from a state x is defined in Equation (2.12), where $r_{p,t}$ refers to the reward received by player p at time-step t, while $\gamma \in [0, 1)$ is a discount factor [122]:

$$\mathcal{G}_{p,\boldsymbol{\pi}}(x) = \mathbb{E}_{\boldsymbol{\pi}}\left\{\sum_{k=0}^{\infty} \gamma^k r_{p,t+k+1} | x_t = x\right\}. \tag{2.12}$$

2.8.5 Incomplete Information and Bayesian Games

In traditional game theory the players are aware of each others' utility functions [197]. However, in multi-agent reinforcement learning the agents frequently only have limited information regarding the game (system) within which they are interacting, often being unable to observe the actions and rewards of the other agents [137, 195]. Instead the learners adapt their policy over time by updating their utility value estimates, thereby adjusting to the other agents' non-stationary policies [38, 137, 195]. Multi-agent reinforcement learning therefore often views (or uses) strategic-form games as a *distributed*

bandit problem [33]. This is in contrast to classical game theory, where the players consider the consequences of their own choices in relation to rational decisions by the other players, where factors that determine the agent's choices are [194]:

1. the preference over the stability of outcomes resulting from each strategy;
2. the strategic choices that the other players are likely to make in response.

Rational decisions are also assumed in *repeated games* with *incomplete information*, where a lack of full information can result from:

1. not being able to observe the exact state (e.g., the hand of an opponent in poker);
2. or not knowing the type of player (e.g., their exact reward function). Although we note that each player maintains a belief (distribution) over the types of the other players.

This set of games is frequently referred to as *Bayesian games*, since the players typically employ a Bayesian approach towards unknown variables, e.g., the belief (distribution) over which type the other player is (i.e. what their payoff matrix is) [69].

2.8.6 Monitoring Conditions in Repeated Games

If the repeated games have *imperfect monitoring* conditions, where the agents cannot observe each others' action choices, the agents must learn a policy via an observable signal from previous encounters, for instance using the reward signal [1, 89]. However, the players can maintain beliefs over what type the other player is, with regards to the payoff matrix [69]. This is in contrast to the majority of multi-agent reinforcement learning algorithms that are applied to strategic-form games, including those discussed within the main contribution chapters of this thesis. Typically independent learners choose actions based on utility values computed via the reward signal without reasoning about the other agents' available actions and decisions [33, 77, 90, 120–122, 137]. However, while the majority of the state-of the-art multi-agent reinforcement learning algorithms are designed for repeated interactions with fixed opponents (or teammates), there have been efforts towards learning best responses for arbitrary opponents in repeated games [29, 35, 36, 44, 77]. For example, similar to work on *Stochastic Bayesian Games*, which focus on incorporating beliefs over opponent types into observations, Hernandez-Leal and Kaisers [77] utilize Bayesian policy reuse for a fast detection of opponents in repeated Markov games, enabling players to choose from best response policies learned during an offline training phase.

2.8.7 Game Types

The reward function of both strategic-from and (partially observable) Markov games can help us differentiate between game types. Below we outline three types of games

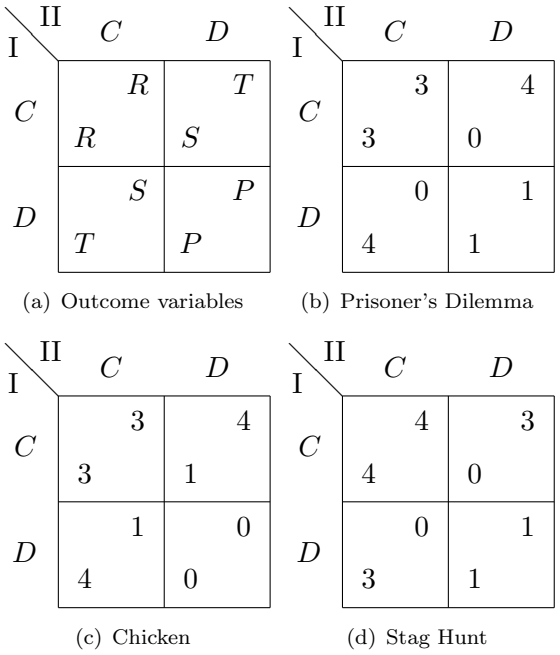
frequently discussed in multi-agent reinforcement learning literature: *purely competitive (zero-sum) games*, *team-games* and *general sum games*:

- **Purely Competitive (Zero-Sum) Games:** A game is a zero-sum game if the total of all players rewards adds up to 0.
- **Team-Games:** A game is a *team game* if all n players receive the same reward, i.e., $R_1 = R_2 = \cdots = R_n = R$. Thus, team-games can be thought of as *fully cooperative* settings, where players have a shared objective to maximize their common return [26, 33, 122]. We note that team-games are not to be confused with the coalition games discussed in *cooperative game theory*, where groups of players form coalitions to compete against other players [219].
- **General-Sum Games:** The payoffs in general-sum games can be arbitrary [137]. A famous class of general sum games are social dilemmas. We illustrate three famous examples of social dilemmas in Figure 2.3: the prisoner's dilemma, chicken and stag hunt games [104]. In these games the actions available to the agents can be interpreted as to either *cooperate* or *defect*. Therefore, for the bimatrix games outlined in Figure 2.3, each stage game has four possible outcomes: R (mutual cooperation); S (cooperating when the other player defects); T (giving in to temptation and defecting when the other player cooperates) and P (both players defect) [104]. A general sum game is considered a social dilemma if the following four inequalities hold [104, 112]:
  1. Mutual cooperation yields a higher payoff than mutual defection: $R > P$.
  2. Mutual cooperation yields a higher payoff than being exploited: $R > S$.
  3. Mutual cooperation is preferred to an equal probability of unilateral cooperation and defection: $2R > T + S$.
  4. Either exploitation is more profitable than mutual cooperation, or mutual defection is preferable to being exploited: $T > R$ or $P > S$ respectively.

In this thesis we empirically evaluate the extent to which independent learners can overcome multi-agent learning pathologies within team-games. However, in Chapter 3 we conduct a literature review of the current state of the art of multi-agent reinforcement learning, providing the necessary context for our contributions. The literature review also includes recent work conducted using zero-sum and general-sum games.

2.8.8 Equilibrium Concepts

In multi-agent learning the interdependence of agents' policies frequently limits the extent to which each agent can maximize their individual payoff. As a result defining a desired outcome in multi-agent reinforcement learning can be challenging, e.g., in settings where agents have conflicting goals it is impossible for each agent to reach their respective maximum expected gain [137]. Since the agents are computing best



(a) Outcome variables (b) Prisoner's Dilemma

(c) Chicken (d) Stag Hunt

Figure 2.3: Social Dilemmas: (a) Illustrates outcome variables R, P, S, and T, whose inequalities can be used to determine if a general sum game is in-fact a social dilemma [104, 112]; (b) The Prisoner's Dilemma; (c) Chicken and (d) Stag Hunt. Actions are to **C**ooperate or **D**efect.

responses to each others' actions, multi-agent reinforcement learning literature often relies on the equilibria concepts defined in game theory to evaluate the outcome of games [122, 137]. Two equilibrium concepts commonly used in game theory to define solutions in games are the Nash equilibrium [136] and Pareto optimality [122]. A group of agents have converged upon a Nash equilibrium if no agent can improve it's long-term gain by unilaterally deviating from its current strategy [122, 136]. Formally:

Definition 2.8.9 (Nash Equilibrium)**.** For a Markov game, a joint policy $\boldsymbol{\pi}^*$ is a Nash equilibrium *iff* no player i can improve it's gain through unilaterally deviating from $\boldsymbol{\pi}^*$:

$$\forall i, \forall \pi_i \in \Delta(\mathcal{X}, \mathcal{U}_i), \forall x \in \mathcal{X}, \mathcal{G}_{i,\langle \pi_i^*, \boldsymbol{\pi}_{-i}^* \rangle}(x) \geq \mathcal{G}_{i,\langle \pi_i, \boldsymbol{\pi}_{-i}^* \rangle}(x). \tag{2.13}$$

Therefore, no player will observe an increase in expected gain upon unilaterally deviating from a Nash equilibrium. While Markov and strategic form games may have more than one Nash equilibrium, from a group perspective Nash equilibria are often sub-optimal [122]. In contrast Pareto-optimality defines a joint strategy $\hat{\boldsymbol{\pi}}$ from which no player i can deviate without making at least one other agent worse off [122].

Definition 2.8.10 (Pareto Optimality)**.** A joint-strategy $\boldsymbol{\pi}$ is Pareto-dominated by $\hat{\boldsymbol{\pi}}$ *iff*:

$$\forall i, \forall x \in \mathcal{X}, \mathcal{G}_{i,\hat{\boldsymbol{\pi}}}(x) \geq \mathcal{G}_{i,\boldsymbol{\pi}}(x) \text{ and } \exists j, \exists x \in \mathcal{X}, \mathcal{G}_{j,\hat{\boldsymbol{\pi}}}(x) > \mathcal{G}_{j,\boldsymbol{\pi}}(x). \tag{2.14}$$

A joint policy $\hat{\boldsymbol{\pi}}^*$ is Pareto optimal if it is not Pareto-dominated by any other $\boldsymbol{\pi}$.

Traditional game theory makes assumptions that do not necessarily reflect the dynamics of the real world, e.g., hyper-ration players that are capable of correctly predicting other players in an equilibrium [53, 81, 159, 194]. Furthermore, determining whether players have converged upon a Nash equilibrium remains on open challenge for many multi-agent learning problems. These notions inspired John Maynard Smith [172] to apply evolution concepts from biology to game theory, resulting in the paradigm of evolutionary game theory. In contrast to traditional game theory, the question at the core of evolutionary game theory is to what extent can a player learn to optimize their behaviour in-order to maximize their return [194]. Learning in evolutionary game theory occurs by conducting repeated games, where players with strategic preferences are randomly drawn from large populations to interact with other players while having no information regarding their preferences [211].

Evolutionary game theory provides solid foundations for modeling decision making under uncertain conditions within complex domains, and is suitable for modeling learning agents within the context of multi-agent systems. For instance, Wiegand [212] introduced an evolutionary game theoretic model for cooperative co-evolutionary algorithms, while evolutionary game theory's replicator dynamics have been extended to study the convergence guarantees and visualize the basins of attraction for numerous multi-agent reinforcement learning algorithms [17, 88, 145, 198].

More recent work in this area has shown that asymmetric games can be decomposed into symmetric counterparts, enabling an evolutionary game theoretic analysis of the original asymmetric game [196], and the introduction of α-Rank [140], a principled evolutionary dynamics methodology which can be used to rank agents within *meta games* (empirical games), while providing insights into learning dynamics and basins of attractions. However, given that the independent learners studied in this thesis are situated within team-games, our aim is to evaluate to what extent we can enable independent learners to converge upon joint policies that are Pareto optimal. Identifying the Pareto optimal solutions and Nash equilibria is relatively trivial for the games within which our evaluations are conducted. Nevertheless, despite the simplicity of these games, considerations are required to enable independent learners to converge upon a Pareto optimal joint-policy.

2.9 Summary

In this chapter we outline the learning algorithms that serve as the foundations for the methods introduced and evaluated throughout this thesis. We also provide a recap of the necessary terminology from game theory to describe the multi-agent learning problem. In the next chapter we shall use this terminology to formally define the multi-agent learning pathologies that confront independent learners within team-games, before providing a summary of independent learning algorithms designed to overcome these pathologies. We subsequently discuss the current state of the art of multi-agent deep

reinforcement learning, which includes extensions of the deep reinforcement learning algorithms discussed in Sections 2.6 and 2.7.

Chapter 3

Multi-Agent Reinforcement Learning

In this chapter we conduct a survey of the current state of the art of multi-agent reinforcement learning. First we shall discuss the multi-agent learning pathologies that confront fully-cooperative independent learners. This is followed by a recap of independent learning algorithms designed to mitigate the outlined pathologies. Finally, we give an overview of research conducted to date in the relatively young field of multi-agent *deep* reinforcement learning, providing the necessary context for our own contributions.

3.1 Multi-Agent Learning Pathologies

One of the benefits of studying team-games, compared to *zero-sum* and *general-sum* games, is that the proof of convergence on a global Nash-equilibria is relatively simple [146]. Identifying the optimal joint-action in these games is trivial for rational players when the reward space is observable. However, converging upon an optimal joint-policy is significantly harder when no information is available regarding the structure of the reward space and the actions chosen by other agents [25, 33, 90, 120, 120, 122, 137, 148, 149]. Independent learners attempt to overcome the challenge of receiving imperfect information through estimating utility values using a reward signal. However, due to multi-agent learning pathologies independent learners' utility value estimates are often noisy, increasing the likelihood of convergence upon a sub-optimal joint-policy. In this section we shall therefore define the independent learning problem along six axis, based on six pathologies frequently observed in multi-agent reinforcement learning literature: *miscoordination*, *relative overgeneralization*, *stochasticity*, *the alter-exploration problem*, *the moving target problem* and *deception*. We briefly mentioned these pathologies in Chapter 1. However, in this section we formally define each pathology. Addressing one pathology often leaves agents vulnerable towards others. We discuss this in detail, while considering the implications of tackling multi-agent learning pathologies in complex environments.

To understand the pathologies, we consider two types of independent learners i attempting to estimate the quality of an action a when paired with the actions $\mathcal{A}'$ available to the other agent [90, 100, 209]:

- *Average based learners:* Estimate the quality of a based on the average return:

$$quality(a) = \sum_{a' \in \mathcal{A}'} \frac{R_i(a, a')}{|\mathcal{A}'|}. \tag{3.1}$$

- *Maximum based learners:* Estimate the quality of a based on the maximum return observed:

$$quality(a) = \max_{a' \in \mathcal{A}'} R_i(a, a'). \tag{3.2}$$

3.1.1 Miscoordination

Miscoordination (also known as the Pareto-selection problem) is a common pathology in repeated games [33, 90, 122]. It can occur when two or more incompatible Pareto-optimal equilibria are present [122]. As a consequence of the equilibria being incompatible, one agent choosing an action from an alternative equilibria is sufficient to lower the overall utility. For example, there are two Pareto optimal equilibria in the Bimatrix games in Figure 3.1, $\langle A, A\rangle$ and $\langle B, B\rangle$. Both joint actions result in a reward of 10 for each agent. However, mixing actions from the two equilibria reduces the utility to 0. Formally we can define miscoordination as follows [122]:

Definition 3.1.1 (Miscoordination)**.** Two equilibria $\boldsymbol{\pi}$ and $\hat{\boldsymbol{\pi}}$ are incompatible iff,

$$\exists i, \boldsymbol{\pi}_i \neq \hat{\boldsymbol{\pi}}_i, \mathcal{G}_{i,\langle \hat{\boldsymbol{\pi}}_i, \boldsymbol{\pi}_{-i}\rangle} < \mathcal{G}_{i,\boldsymbol{\pi}} \tag{3.3}$$

I \ II	A	B
A	10, 10	0, 0
B	0, 0	10, 10

Figure 3.1: Bimatrix game example with two Pareto optimal equilibria [209]

3.1.2 Relative Overgeneralization

Relative overgeneralization occurs when agents gravitate towards a robust but sub-optimal joint policy, due to noise induced by the mutual influence of each agent's exploration strategy on the other agents' learning updates [212]. It is a type of *action shadowing*, occurring in games where a sub-optimal policy yields a higher payoff on average when each selected action is paired with an arbitrary action chosen by the other

agent [147]. A *shadowed equilibrium* is an equilibrium defined by a policy $\overline{\boldsymbol{\pi}}$ that is shadowed by a policy $\hat{\boldsymbol{\pi}}$ in a state x, where at least one agent exists who when unilaterally deviating from $\overline{\boldsymbol{\pi}}$, will receive a gain $\mathcal{G}_{\langle \pi_i, \overline{\boldsymbol{\pi}}_{-i} \rangle}(x)$ less than the minimum gain that can be obtained for deviating from $\hat{\boldsymbol{\pi}}$ [122]. Formally:

$$\exists i \exists \pi_i \mathcal{G}_{\langle \pi_i, \overline{\boldsymbol{\pi}}_{-i} \rangle}(x) < \min_{j, \pi_j} \mathcal{G}_{\langle \pi_j, \hat{\boldsymbol{\pi}}_{-j} \rangle}(x). \tag{3.4}$$

Relative overgeneralization occurs in games where, as a result of a *shadowed equilibrium*, the agents converge upon a sub-optimal Nash Equilibrium that is Pareto-dominated by at least one other Nash Equilibrium [111, 122]. As a result of relative overgeneralization, independent learners can be drawn to sub-optimal but wide peaks in the reward space, due to a greater likelihood of achieving collaboration there [147]. This problem is illustrated in Figure 3.2, which depicts a reward space for continuous actions. The x and y axis represent the continuous actions for agents i and j, while the z axis illustrates the reward for each joint-action $\langle a_i, a_j \rangle$. While M represents the optimal action for agent i, pairing M with arbitrary actions by agent j results in a lower utility on average compared to when agent i chooses action N.

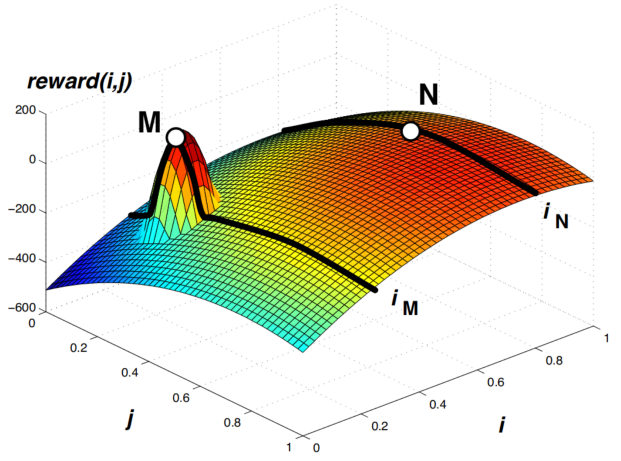


Figure 3.2: An illustration of a reward space for continuous actions where the relative overgeneralization pathology is present. The x and y axis represent the continuous actions for agents i and j, while the z axis illustrates the reward for each joint-action $\langle a_i, a_j \rangle$. For agent i action M can lead to the optimal reward, providing agent j chooses the correct response. However, due to miscoordination being less severely punished for actions approaching N, the agents are drawn towards a sub-optimal Nash equilibrium. This illustration is taken from Wei and Luke [209].

3.1.3 Stochasticity of Rewards and Transitions

A deviation of multi-agent reinforcement learning literature from traditional economic game theory is the assumption that the payoffs received can be stochastic, with joint-actions not always resulting in a deterministic reward for each agent [90, 137]. Given that

stochasticity (of both rewards and transitions) are a central pathology of multi-agent reinforcement learning, in this thesis we consider both *partially* and *fully stochastic* rewards. Deterministic and fully stochastic reward functions exclusively return deterministic and stochastic rewards respectively for each $\mathcal{A}_{1...n}$. For a partially stochastic reward function there exists up to $|\mathcal{A}| - 1$ joint actions $\mathcal{A}_{1...n}$ for which a stochastic reward is returned, while the remaining joint actions return deterministic rewards [90].

We illustrate this problem with two variations of a bimatrix-game in Figure 3.3. In the deterministic reward variation, Sub-Figure 3.3(a), *relative overgeneralization* can be overcome with *maximum-based learning*, where learners consider each action i based on the observed $max_j(i, j)$. However, this approach leaves learners vulnerable towards misleading stochastic rewards. For example, by making the game partially stochastic, Sub-Figure 3.3(b), the joint action $\langle C, C \rangle$ yields stochastic rewards of 14 and 0 with 50% probability. Therefore, maximum based learners are drawn towards $\langle C, C \rangle$, despite each agent only receiving a reward of 7 on average. In temporally extended games additional stochasticity can emerge as a result of environmental factors such as noisy observations [**?**] and probabilistic state transitions [**?**]. Meanwhile, independent learners facing the curse of dimensionality must overcome challenges introduced by noisy approximated utility estimates backed-up from stochastic follow-on state-transitions or rewards [122].

I \ II	*A*	*B*	*C*
A	11 / 11	−30 / −30	5 / 5
B	−30 / −30	5 / 5	6 / 6
C	5 / 5	6 / 6	7 / 7

(a) Deterministic

I \ II	*A*	*B*	C
A	11 / 11	−30 / −30	5 / 5
B	−30 / −30	5 / 5	6 / 6
C	5 / 5	6 / 6	14/0 / 14/0

(b) Partially Stochastic

Figure 3.3: Two variations of a bimatrix game that confronts independent learners with relative overgeneralization. For the deterministic variation (a) maximum based learners will converge upon the optimal joint-action $\langle A, A \rangle$, by ignoring the miscoordination penalties. For (b) joint-action $\langle C, C \rangle$ yields stochastic rewards of 14 and 0 with 50% probability, towards which maximum based learners are drawn.

3.1.4 The Alter-Exploration Problem

The exploration-exploitation trade-off required by reinforcement learners adds to the challenge of learning noise-free utility estimates. Matignon et al. [122] define global exploration, the probability of at least one of n agents exploring as $1 - (1 - \epsilon)^n$, where each agent explores according to a probability ϵ. In environments with a *shadowed*

equilibrium, as defined in Section 3.1.2, higher global exploration can result in agents converging upon a sub-optimal joint policy, as exploration can lead to penalties [122].

3.1.5 The Moving Target Problem

As mentioned, an environment can no longer be considered Markovian when multiple independent learners update their policies in parallel, thereby losing the property that guarantees convergence for a large number of single-agent learning algorithms [20, 183, 195, 200]. This problem is amplified in multi-agent deep reinforcement learning, where using an experience replay memory $\mathcal{D}$ often results in deprecated transitions being sampled during training [46, 141, 144].

3.1.6 Deception

Deception can occur in Markov games with $|\mathcal{X}| > 1$, where utility values are calculated by incorporating backed up rewards from follow-on states from which pathologies such as miscoordination and relative overgeneralization can also be back-propagated [209]. This pathology leaves maximum based learners vulnerable towards overestimating the expected utility of actions that result in state-transitions towards high utility states that occur with a low probability. Furthermore, independent learners can be drawn away from optimal state-transition trajectories in the presence of states with high local rewards that lead to states with low future rewards [209].

3.2 Independent Learning Approaches

A number of games exist where independent learners using standard Q-learning are unable to converge upon equilibrium play [33, 90, 120, 137]. In team-games this is often due to the learners being confronted with the learning pathologies outlined above. However, an increased interest in independent learners over the past decades has led to the development of algorithms that have proven effective within a range of challenging repeated strategic-form and Markov games [35]. Despite these developments independent learning still lacks a *silver bullet* approach, even for team bimatrix games [28]. Therefore, before we discuss algorithms designed to facilitate cooperation within high-dimensional settings (Section 3.3.1), in this section we provide a recap of independent learning algorithms that have proven robust in low-dimensional settings, including: decentralized Q-learning, distributed Q-learning [100], hysteretic Q-learning [120], Frequency Maximum Q-value (FMQ) [90], Recursive Frequency Maximum Q-value (Recursive-FMQ) [121] and Lenient Multi-Agent Reinforcement Learning 2 (LMRL2) [209].

LMRL2 is currently considered the state of the art independent learning approach for mitigating the above pathologies within two-player strategic-form games and Markov games with a small low-dimensional state space, enabling a high convergence rate even in the presence of relative overgeneralization and stochasticity [209]. We verify this claim in Chapter 4, evaluating to what extent LMRL2 (and the remaining approaches) can

converge upon optimal joint-policies upon increasing the scale of penalty values and the number of learners. During this process we identify a number of LMRL2's weaknesses, which we subsequently address in Chapter 5. Finally, we scale and evaluate leniency within a *deep* multi-agent reinforcement learning context in Chapters 6 and 7.

While LMRL2 represents the current state of the art, the approach requires carefully tuned hyperparameters that rarely translate across domains [209]. This finding is worrying, as tuning hyperparameters is notoriously expensive within deep reinforcement learning [79]. In contrast, decentralized and hysteretic Q-learning have less overheads [120]. Furthermore, despite being less robust towards a combination of stochasticity and relative overgeneralization, both approaches have been scaled to multi-agent deep reinforcement learning [104, 110, 141, 187]. Decentralized and hysteretic Q-learning therefore provide valuable baselines against which to compare our contributions.

In contrast to hysteretic and decentralized Q-learning, no equivalent *deep* approaches currently exist for FMQ and Recursive-FMQ. Nevertheless, the two approaches are relevant to our current work, as they both modify the learner's exploration policy. They therefore provide a means to gain valuable insights regarding the choice of exploration method, e.g., helping us better understand the challenges regarding the tuning of LMRL2's own modified Boltzmann exploration strategy.

3.2.1 Decentralized Q-learning

Implementing learning agents with standard Q-learning is often referred to in multi-agent reinforcement learning literature as *decentralized Q-learning*. While decentralized Q-learning has a poor convergence rate when confronted with domains suffering from the pathologies discussed in this chapter, it does provide a valuable baseline against which to compare other approaches, and is therefore frequently featured in multi-agent reinforcement learning literature. We introduced the full version of Q-learning in Section 2.3. The stateless version of Q-learning used in previous work on strategic-form games computes Q-value updates using an exponential weighted moving average [33, 90, 120, 121, 137], where the utility value after each episode is updated with the reward r as follows [1]:

$$Q(a) = (1 - \alpha)Q(a) + \alpha r. \tag{3.5}$$

Decentralized Q-learners are therefore by definition average reward learners. This underlines why Q-learning struggles when confronted with multi-agent learning pathologies, such as relative overgeneralization and miscoordination [33, 90].

3.2.2 Distributed Q-learning

Lauer and Riedmiller's [100] distributed Q-learning is believed to be one of the earliest attempts at designing an algorithm to help cooperative learners prevent relative overgeneralization. In contrast to decentralized Q-learning the algorithm maintains two tables,

[1] Q-values Q_i are computed independently for each agent i. For simplicity we drop the subscript i.

a Q-table and a policy table π. In regular Q-learning a learning rate $\alpha < 1.0$ ensures that Q-value updates only incorporate a portion of the observed reward and follow-on utility estimate. In contrast distributed Q-learning uses $\alpha = 1.0$ for positive updates that increase a Q-value estimate. Therefore, the current Q-value is completely replaced with the observed reward and follow-on utility estimate [100]:

$$Q(x_t, u_t) \leftarrow max\{Q(x_t, u_t), r_{t+1} + \gamma \max_{u \in \mathcal{U}} Q(x_{t+1}, u)\}. \tag{3.6}$$

For the stateless version applied to strategic-form games Lauer and Riedmiller [100] use $\gamma = 0$. The current policy table π is updated *iff* an improvement occurs with regards to the maximum Q-value, allowing the agents to overcome the Pareto selection problem [100]. At time step $t = 0$ actions are selected arbitrarily: $\pi_0(x) \in \mathcal{U}$. At each time-step the policy table π is updated as follows:

$$\pi_{t+1}(x) \leftarrow \begin{cases} \pi_{t+1}(x), & \text{if } x \neq x_t \text{ or } \max_{u \in \mathcal{U}} Q_t(x, u) = \max_{u \in \mathcal{U}} Q_{t+1}(x, u). \\ u_t, & \text{otherwise.} \end{cases} \tag{3.7}$$

Distributed Q-learners perform well in deterministic games, but are vulnerable towards misleading stochasticity due to being *maximum based learners* [90].

3.2.3 Hysteretic Q-learning

By setting the learning rate $alpha = 1.0$ for positive updates, distributed Q-learners are effectively maximum based learners. Hysteretic Q-learning was introduced to address distributed Q-learners' tendency of gravitating towards sub-optimal equilibria, as a result of being blind towards stochastic reward signals that would result in a lowering of a utility estimate [120]. To address this issue hysteretic Q-learning introduces a second learning rate $\beta < \alpha$. Given a temporal difference error δ,

$$\delta \leftarrow r - Q(a), \tag{3.8}$$

learning rate α is applied to updates where δ is positive, meaning the utility of an action is currently being underestimated. Learning rate β, on the other hand, is applied when δ is negative, giving the agent an optimistic disposition without entirely ignoring rewards that would lower the agent's utility estimate:

$$Q(a) \leftarrow \begin{cases} Q(a) + \alpha\delta, & \text{if } \delta \geq 0. \\ Q(a) + \beta\delta, & \text{otherwise.} \end{cases} \tag{3.9}$$

For Markov games *delta* is computed for state-action pairs:

$$\delta = r_{t+1} + \gamma \max_{u \in \mathcal{U}} Q(x_{t+1}, u) - Q(x_t, u_t), \tag{3.10}$$

and subsequently scaled using either α or β accordingly:

$$Q(x_t, u_t) \leftarrow Q(x_t, u_t) + \begin{cases} \alpha\delta, & \text{if } \delta \geq 0. \\ \beta\delta, & \text{otherwise.} \end{cases} \tag{3.11}$$

Hysteretic Q-learning is a form of optimistic learning with a strong empirical track record in fully-observable environments, requiring less overhead than the majority of independent learning approaches [9, 120, 122, 217]. However, hysteretic Q-learners still incorporate maximum-based learning traits, due to their Q-value estimates being more likely to reflect new superior results. Therefore, despite being introduced to address distributed Q-learners' vulnerability towards misleading stochastic rewards, depending on the values chosen for β, hysteretic Q-learners often converge on sub-optimal joint policies when simultaneously confronted with the pathologies of relative over-generalization and stochasticity with regards to both rewards and stochastic transitions [209].

3.2.4 Frequency Maximum Q-value

The methods outlined above focus on modifying the Q-value update function. In contrast, Frequency Maximum Q-value (FMQ) [90] takes an alternative approach by modifying the exploration strategy. Using a modified Boltzmann exploration (see Section 2.3), FMQ applies an addition estimated value term EV to each Q-value while computing the action selection probabilities:

$$P(a) \leftarrow \frac{\exp\left(\frac{EV(a)}{\mathcal{T}}\right)}{\sum_{a' \in \mathcal{A}} \exp\left(\frac{EV(a')}{\mathcal{T}}\right)}, \tag{3.12}$$

where:

$$EV(a) \leftarrow Q(a) + c \times freq(maxR(a)) \times maxR(a). \tag{3.13}$$

In the above equation

- $maxR(a)$ represents the maximum reward observed for action a;
- $freq(maxR(a))$ the likelihood of $maxR(a)$ occurring, based on the number of times $maxR(a)$ was received divided by the number of times action a was selected;
- c determines the weighting of the FMQ heuristic. Therefore, c determines the extent to which action selection should rely on high instead of average rewards.

The temperature $\mathcal{T}$ is determined using Equation 3.14:

$$\mathcal{T} \leftarrow \exp\left(-s \times t\right) \times MaxTemp + 1, \tag{3.14}$$

where t represents the current time-step; s controls the rate of exponential decay; and $MaxTemp$ is the initial temperature. Two limitations of FMQ are that it only

marginally outperforms decentralized Q-learning when the variance of the reward function is high [90], and that it is limited to repeated strategic-form games [121].

3.2.5 Recursive Frequency Maximum Q-value

FMQ inspired Matignon et al. [121] to develop Recursive-FMQ (RFMQ), which the authors subsequently scaled to Markov games [2]. Matignon et al. [121] developed Recursive-FMQ to address FMQ's weaknesses. For instance, the authors find that if the first observation of the optimal joint-action is delayed, despite one of the agents frequently choosing the optimal action, then the FMQ heuristic will enforce that the optimal action remains an exploratory step. Thus the agents can only recover through repeatedly executing the optimal join-action, enabling the frequency term to increase sufficiently for both agents to converge upon the optimal joint-action. The first phase of exploration is therefore critical for FMQ to succeed [121]. The authors' mitigate this problem by replacing the count based frequency used in FMQ with a recursively computed frequency term $F(a)$:

$$F(a) \leftarrow \begin{cases} 1, & \text{if } r > Q_{max}(a). \\ (1-\alpha_f)F(a) + \alpha_f, & \text{if } r = Q_{max}(a). \\ (1-\alpha_f)F(a), & \text{otherwise.} \end{cases} \tag{3.15}$$

The recursive frequency term $F(a)$ for action a is therefore reset to 1 when receiving a reward r greater than the observed $Q_{max}(a)$, with $Q_{max}(a)$ subsequently set to r. If r is not greater than $Q_{max}(a)$, then $F(a)$ is updated using a frequency learning rate α_f as outlined above in Equation (3.15). The frequency term $F(a)$ therefore only decreases in cases of miscoordination, due to the alter exploration problem, or due to a noisy reward [121]. Similar to FMQ, Recursive-FMQ also modifies the exploration strategy. However, instead of using Boltzmann Exploration, ϵ-greedy exploration is applied to a policy vector π. A parameter free linear interpolation heuristic for evaluating the actions determines when to update π (Equation (3.16)). The complete Recursive-FMQ algorithm as specified by Matignon et al. [121] is outline in Algorithm 1.

$$E(a) \leftarrow [1 - F(a)]Q(a) + F(a)Q_{max}(a). \tag{3.16}$$

RFMQ can *swing between optimist or neutral* depending on whether the rewards received are deterministic or stochastic respectively [121]. Therefore, in deterministic strategic-form games the agents predominately rely on $Q_{max}(a)$, while an increase in stochasticity will result in the agents being guided by the average reward $Q(a)$ when choosing an exploration step [121].

[2] The full version of RFMQ is named Swing between Optimal or Neutral (SOoN)[121].

Algorithm 1 Recursive-FMQ

Input: Max steps T, learning rate α, frequency learning rate α_f
Init: $\forall a \in \mathcal{A}$, $Q(a) \leftarrow 0$, $Q_{max}(a) \leftarrow 0$, $F(a) \leftarrow 1$, $E(a) \leftarrow 0$, $\pi(a)$ arbitrarily
for $t = 0$ to T **do**
    Select a according to the ϵ-greedy selection method based on π
    Apply a and observe reward r
    $Q(a) \leftarrow (1 - \alpha)Q(a) + \alpha r$
    if $r > Q_{max}(a)$ **then**
        $Q_{max}(a) \leftarrow r$
        $F(a) \leftarrow 1$
    else if $r = Q_{max}(a)$ **then**
        $F(a) \leftarrow (1 - \alpha_f)F(a) + \alpha_f$
    else
        $F(a) \leftarrow (1 - \alpha_f)F(a)$
    $E(a) \leftarrow [1 - F(a)]Q(a) + F(a)Q_{max}(a)$
    if $E(\text{argmax}_{o \in \mathcal{A}}\, \pi(o)) \neq \max_{o \in \mathcal{A}} E(o)$ **then**
        Select a random action $a_{max} \in \text{argmax}_{o \in \mathcal{A}}\, E(o)$
        $\forall b \in \mathcal{A}\ \pi(b) \leftarrow \begin{cases} 1, & \text{if } b = a_{max}. \\ 0, & \text{otherwise.} \end{cases}$

3.2.6 Lenient Multi-Agent Reinforcement Learning

Lenient learning was originally introduced by Potter and De Jong [152] to help cooperative co-evolutionary algorithms converge towards an optimal joint-policy, and was later applied to multi-agent reinforcement learning as well [149]. It was designed to prevent relative overgeneralization [212], and has been shown to increase the likelihood of convergence towards the globally optimal solution in stateless coordination games for reinforcement learning agents [17, 18, 148, 149]. Lenient learners do so by effectively forgiving (ignoring) sub-optimal actions by teammates that lead to low rewards during the initial exploration phase.

While initially adopting an optimistic disposition, the amount of leniency displayed is typically decayed each time a state-action pair is visited [17, 18, 148, 149]. As a result the agents become less lenient over time for frequently visited state-action pairs, while remaining optimistic within unexplored areas. This transition to average reward learners helps lenient agents avoid sub-optimal joint policies in environments that yield stochastic rewards. Therefore, leniency is less vulnerable towards misleading stochastic rewards than distributed and hysteretic Q-learning [209].

During training the frequency with which lenient reinforcement learning agents perform updates that result in lowering the Q-value of an action a is determined by leniency and temperature functions $L : \mathcal{A} \rightarrow \mathbb{R}$ and $\mathcal{T} : \mathcal{A} \rightarrow \mathbb{R}$ respectively [17, 18, 148, 149]. The stateless version of LMRL2 used on strategic-form games maintains temperature values $\mathcal{T}(a)$ for each action $a \in \mathcal{A}$. The mapping between actions and temperature values for the function $\mathcal{T}$ is *one to one*, with each action being assigned a real-valued temperature that is initially set to a defined maximum value. As in the original version

of leniency the temperature value associated with the selected action is decayed using a decay rate ν following a utility value update [147, 148]:

$$\mathcal{T}(a) \leftarrow \nu\mathcal{T}(a). \tag{3.17}$$

The amount of leniency that should be applied during a utility value update is computed using equation 3.18:

$$L\left(a_t\right) = \exp\left(\frac{-1}{k\mathcal{T}_t\left(a_t\right)}\right). \tag{3.18}$$

A constant k is used as a leniency moderation factor to determine how the temperature value affects the drop-off in lenience.

A Q-value update is performed *iff* the temporal difference error δ is positive, or a random variable $z \in [0, 1]$ is greater than the amount of leniency $L(a)$ computed for action a:

$$Q(a) \leftarrow \begin{cases} r, & \text{if } Q(a) = \inf \text{ (Only if initialization was to infinity).} \\ Q(a) + \alpha\delta, & \text{if } \delta > 0 \text{ or } z > L(a). \\ Q(a), & \text{otherwise.} \end{cases} \tag{3.19}$$

With LMRL2 Wei and Luke [209] extend leniency [147, 148]: LMRL2's temperature values $\mathcal{T}$ are not only used to compute the amount of leniency that a learner should apply towards updates that would lower a utility value, but are also used to guide the exploration-exploitation trade-off. Similar to FMQ the Boltzmann action selection method is modified. The average temperature value,

$$\overline{\mathcal{T}} \leftarrow mean_a\mathcal{T}(a), \tag{3.20}$$

determines the action selection probabilities returned by the modified Boltzmann exploration method:

$$P(a) \leftarrow \frac{W_a}{\sum_{a' \in \mathcal{A}} W_{a'}}. \tag{3.21}$$

For each action $a \in \mathcal{A}$, W_a is computed using Equation (3.22), with ω being an action selection moderation factor.

$$W_a \leftarrow \exp\left(\frac{Q(a)}{\omega\overline{\mathcal{T}}}\right). \tag{3.22}$$

Wei and Luke [209] add a $MinTemp$ value for exploration to avoid floating point overflow errors, and thereby prevent the agents from becoming completely greedy. The Q-value update for LMRL2 are unaffected by the $MinTemp$ value, and remain identical to how leniency was described by Panait et al. [147, 148]. The complete algorithm for LMRL2 in repeated strategic-form games is outlined in Algorithm 2.

For Markov games leniency requires temperature values for state-action pairs:

$$L\left(x_t, u_t\right) = \exp\left(\frac{-1}{k\mathcal{T}_t\left(x_t, u_t\right)}\right). \tag{3.23}$$

Algorithm 2 LMRL2 for Repeated Strategic-Form Games

Input: Max steps T, $MaxTemp$, $MinTemp$, learning rate α, leniency moderation factor k, exploration moderation factor ω, temperature decay rate ν
for all $a \in \mathcal{A}$ **do**
    $Q(a) \leftarrow initialize(a)$, $\mathcal{T}(a) \leftarrow MaxTemp$
for $t = 0$ to T **do**
    if $\overline{\mathcal{T}} < MinTemp$ or $max_a Q(a) = \inf$ **then**
        $u \leftarrow \text{argmax}_a\, Q(a)$ (breaking ties randomly)
    else
        Choose a using probability distribution P obtained using Equation (3.21).
    Execute action a and observe r_t
    Update $Q(a)$ using Equation (3.19)
    $\mathcal{T}(a) \leftarrow \nu\mathcal{T}(a)$

For the temperature based exploration the average temperature value for the current state, $\overline{\mathcal{T}}(x)$ determines the action selection probabilities returned by the modified Boltzmann exploration method:

$$\overline{\mathcal{T}}(x) \leftarrow mean_u \mathcal{T}(x, u). \tag{3.24}$$

States are therefore also considered when computing the action selection probabilities,

$$P(u) \leftarrow \frac{W_u}{\sum_{u' \in \mathcal{U}} W_{u'}}, \tag{3.25}$$

using

$$W_u \leftarrow \exp\left(\frac{Q(x, u)}{\omega \overline{\mathcal{T}}(x)}\right). \tag{3.26}$$

As a result agents are more likely to choose a greedy action within frequently visited states while remaining exploratory for less-frequented areas of the environment. However, Wei and Luke [209] note that the choice of the moderation factor ω is a non-trivial task, as Boltzmann selection struggles to distinguish between similar Q-values [87].

After each transition the temperature value for the current state-action pair are decayed, where t is the current time-step. If the agents find themselves in the same initial state at the beginning of each episode, then after repeated interactions the temperature values for state-action pairs close to the initial state can decay rapidly as they are visited more frequently. However, it is crucial for the success of the lenient learners that the temperatures for these state-action pairs remain sufficiently high for the rewards to propagate back from later stages, and to prevent the agents from converging upon a sub-optimal policy. Wei and Luke [209] attempt to mitigate the premature decay of temperature values by folding the average temperature for the n actions available to the agent in x_{t+1} into the temperature that is being decayed for (x_t, a_t). The extent to which this average temperature $\overline{T}_t\,(x_{t+1})$ is folded in is determined by a constant υ as

Algorithm 3 LMRL2 for Markov Games

Input: Max steps T, $MaxTemp$, $MinTemp$, learning rate α, leniency moderation factor k, exploration moderation factor ω, temperature decay rate ν
for all $x \in \mathcal{X}$ and $u \in \mathcal{U}$ **do**
    $Q(x,a) \leftarrow initialize(x,u)$, $\mathcal{T}(x,u) \leftarrow MaxTemp$
$x \leftarrow$ initial state
for $t = 0$ to T **do**
    if $\overline{\mathcal{T}}(x) < MinTemp$ or $max_a Q(x,u) = \inf$ **then**
        $u \leftarrow \text{argmax}_a\, Q(x,u)$ (breaking ties randomly)
    else
        Choose u using probability distribution P obtained using Equation (3.25).
    Execute action u and observe x_{t+1}, r_t
    Update $Q(x,u)$ and $\mathcal{T}(x,u)$ using Equations (3.28) and (3.27) respectively.

follows:

$$\mathcal{T}_{t+1}(x_t,u_t) = \beta \begin{cases} \mathcal{T}_t(x_t,u_t) & \text{if } x_{t+1} \text{ is terminal.} \\ (1-\upsilon)\,\mathcal{T}_t(x_t,u_t) + \upsilon \overline{\mathcal{T}}_t(x_{t+1}) & \text{otherwise.} \end{cases} \tag{3.27}$$

The leniency Q-value update strategy now uses Equation (3.10) to calculate temporal difference error δ. As above the Q-value updates are carried out if the δ is positive, or a random variable $z \in [0,1]$ is greater than the leniency value $L(x,u)$ computed for the state-action pair (x,u):

$$Q(x,u) \leftarrow \begin{cases} r, & \text{if } Q(x,u) = \inf \text{ (Only if initialization was to infinity).} \\ Q(x,u) + \alpha\delta, & \text{if } \delta > 0 \text{ or } z > L(x,u). \\ Q(x,u), & \text{otherwise.} \end{cases} \tag{3.28}$$

The complete algorithm for LMRL2 in repeated games is outlined in Algorithm 3.

3.2.7 Comparison

Wei and Luke [209] provide a comprehensive evaluation of each of the approaches outlined above. Evaluations took place in four strategic-from games from multi-agent reinforcement learning literature and eight Markov games (six of which were designed by the authors). Approaches were compared based on their ability to converge upon *correct* and *complete* policies in self-play. Learners have converged upon a correct policy if they behave optimally when following the policy from a designated initial state. Learners who have converged upon a complete policy meanwhile behave correctly in every state.

As mentioned, the authors found that LMRL2 outperforms the other methods, being placed in top statistical tier for correct joint-policies for eleven out of the twelve games, while also being in the top tier for finding complete solutions. Surprisingly decentralized Q-learning performed reasonably well in games where the learners were not confronted with relative overgeneralization. However, as we shall see in Chapter 4, these findings do not scale as the penalty for miscoordination is increased [90]. Both distributed

and hysteretic Q-learning meanwhile were able to prevent relative overgeneralization in deterministic domains, but struggled in games with stochastic rewards and transitions. For FMQ the authors replicate previous results where the percentage of correct runs decreases for domains where the variance of the reward function is high [90]. RFMQ meanwhile was the second most consistent performer on strategic-form games. However, SOoN, the scaled version of RFMQ, lacked consistency in Markov games.

Despite conducting evaluations using tuned hyperparameters for each domain, the authors find that even LMRL2 fails to consistently converge upon correct joint-policy in all of the games. This raises the question to what extent hyperparameter settings can be found that deliver a consistent performance across each game. Our work in Chapter 4 aims to answer this question within repeated n-player strategic-form games, while in Chapter 5 we propose our own extensions to improve the convergence rate of lenient learners within repeated strategic-form and Markov games with a low-dimensional state space. In Chapters 6 and 7 we turn to multi-agent deep reinforcement learning, and evaluate the extent to which findings from strategic-form and Markov games with a low dimensional state-space scale to domains suffering from the curse of dimensionality. We provide the necessary background regarding the current state of the art in multi-agent deep reinforcement learning research below, to provide sufficient context for our contributions in the later chapters.

3.3 Multi-Agent Deep Reinforcement Learning

The emergence of deep reinforcement learning has opened up new possibilities with regards to scaling multi-agent reinforcement learning to complex high-dimensional environments. However, considerations are required when applying single-agent architectures to multi-agent deep reinforcement learning. For example, independent learners sampling from an experience replay memory $\mathcal{D}$ will be confronted with *sample obsolescence*, where, due to the non-stationarity pathology, state-transitions stored within $\mathcal{D}$ become obsolete and thereby misleading [45]. Initial solutions to this problem were to either disable the experience replay memory or reduce the sample capacity [45, 104]. However, these solutions both limit sampling efficiency and threaten the stability of the function approximator [46]. A further issue that becomes more noticeable in partially observable Markov games is the *credit assignment problem*, where agents receive spurious reward signals following unobserved actions performed by teammates [181]. Sunehag et al. [181] hypothesize that increasing the number of agents within a multi-agent system will increase the credit assignment problem.

In this section we first discuss methods that have recently been proposed in multi-agent deep reinforcement learning literature to address the above challenges and facility cooperation between agents. We subsequently provide an overview of related topics, before briefly discussing the practical challenges of multi-agent deep reinforcement learning.

Finally we shall conclude this chapter with a discussion regarding the limitations of the current state of the art.

3.3.1 Facilitating Cooperation

Multi-agent deep reinforcement learning literature has put forward a number of solutions for addressing the sample obsolescence and credit assignment problems. For instance, a well studied approach towards addressing the credit assignment problem is to use reward shaping through handcrafted reward functions, thereby providing agents with a personal reward signal for each observation [23, 24, 38–40, 60, 62, 113, 181]. However, reward shaping can deviate the learner from their true objective if poorly designed [38, 181]. Therefore, a more general solution is required. One approach is to mitigate both the credit assignment and sample obsolescence problems through centralized learning [79]. Centralized learning reduces the multi-agent learning problem to a single agent-learning problem through concatenating the observations from each of the agents in the system, before feeding them to a network that outputs values (be it Q-values, or action values) for each of the $|\mathcal{A}|^n$ actions, where n represents the number of agents [15, 66, 181]. However, centralized approaches are impractical, as increasing the number of agents leads to an exponential increase in the size of the state-action space [15, 38, 66, 79, 120, 157, 174, 181].

Gupta et al. [66] address this intractability by factoring the action space of the policy to capture the action distributions for each agent, reducing the size of the action space from $|\mathcal{A}|^n$ to $n|\mathcal{A}|$. However, this approach assumes that the agents are homogeneous with regards to their action-space. Further drawbacks include: (i) the approach does not address the issue of an exponential growth in the observation space upon increasing n, and (ii) the model is centralized both during training and execution [66]. Finally, Sunehag et al.[181] observe that in practice centralized approaches can fail to converge on an optimal policy due to the *lazy agent problem*, occurring when a useful behaviour is learned for one of the agents, resulting in the other agents being marginalized to prevent interference.

Hybrid approaches using centralized training for decentralized execution (CTDE) have emerged as an alternative to centralized and decentralized methods [139]. The concept behind CTDE is to optimize a joint action-value function at training time that in turn optimizes at an individual level, thereby enabling agents to learn individual action-value functions [139, 174]. Therefore, in contrast to centralized approaches, once deployed CTDE allows agents to choose actions based on individual observations using their own action-value function, without having to refer to the joint-action value function [157, 174, 181]. CTDE is suitable for domains with partial observability where sufficient information can be made available at training time to enable centralized training [174]. In the paragraphs below we provide a summary of CTDE approaches outlined in literature, before turning to decentralized learning approaches.

PS-TRPO: Gupta et al. [66] proposed one of the first CTDE algorithms for multi-agent deep reinforcement learning: a parameter sharing approach where each agent has access to the same policy network. The network is trained using a replay memory $\mathcal{D}$, which stores state-transition tuples obtained from all agents within the system. Therefore, the observation-action space for each agent is reduced to the same size as for concurrent learners [66]. The authors find that parameter sharing outperforms centralized and concurrent learning on one discrete task (*Multi-Agent Pursuit*) and two continuous control tasks (*Waterworld* and *Multi-Walker*). Parameter sharing has two considerable advantages: (i) it reduces the number of learn-able parameters, and (ii) having invariant agents mitigates the lazy agent problem [181]. However, further considerations are required in domains with specialized roles or for heterogeneous agents [181].

COMA: Inspired by difference rewards [40, 117, 213], Foerster et al. [47] propose an actor-critic architecture with a centralized critic that is used to train actors using Counterfactual Multi-Agent Policy Gradients (COMA). The approach tackles the credit assignments problem, and attempts to estimate the contributions made by each agent, by computing a counter-factual baseline by marginalizing the impact of each actor in turn.

MADDPG: Lowe et al. [109] introduce Multi-Agent Deep Deterministic Policy Gradients (MADDPG) which extends the deep deterministic policy gradient algorithm [105] by adding a centralised critic network to train independent actor networks. However, Rashid et al. [157] question the practicality of a fully centralized critic as the number of agents increases.

Multiagent Soft Q-learning: Wei et al. [210] introduce Multiagent Soft Q-learning, which converges towards a superior local optima compared to MADDPG [109] within continuous action domains where learners must avoid relative overgeneralization. Multiagent Soft Q-learning augments rewards with an entropy term, thereby increasing the likelihood of learners discovering multiple modes within a continuous action space [67, 210]. However, the approach is currently limited to single state continuous games.

Further successful approaches include training networks to approximate centralized but factored Q-value functions [79]. Factorization in fully-cooperative games takes advantage of the fact that optimal actions across the agents are equivalent to the set of optimal actions for each individual agent [174]. *Value Decomposition Network* (VDN) [181], *QMIX* [157] and *QTRAN* [174] are recent examples of applying factorized joint-action values into individual action-values for decentralized execution in domains with a high-dimensional state space:

VDN: Sunehag et al. [181] introduce a value decomposition network (VDN) architecture, an approach that learns an optimal value decomposition from the reward signal. During training the gradient for the additive Q-value is back-propagated through the individual DQN architectures, thereby mitigating spurious reward signals. Each agent is therefore trained using it's own observations, and can subsequently be deployed independently of the other agents. Furthermore, the authors find that agents trained using

VDN outperform decentralized learners, and those trained using a centralized approach by a large margin [181].

QMIX: A more recent approach, QMIX, estimates joint action-values as a non-linear combination of agent-values, thereby helping agents learn optimal joint-action values based on the additional information that centralized learning provides [157]. The authors observe that the full factorisation of VDN is not necessary, and instead use a mixing network that combines the individual Q values into a Q_{tot} in a complex non-linear way. The approach can therefore be applied to an increased number of potential action-value functions compared to VDN, and outperforms VDN on a number of micromanagement tasks built in StarCraft II [205]. Furthermore, the approach performs well in domains with heterogeneous agents. However, it can only be applied to problems where a monotonicity assumption holds [157].

QTRAN: Son et al. [174] show that despite both VDN and QMIX being value based approaches, their respective additivity and monotonicity assumptions limit the set of games that each approach can be applied to. The authors demonstrate this using a simple matrix game, and go on to propose a new method that is not conditioned on these structural constraints, thereby being applicable to far wider range of domains: QTRAN. An affine transformation transforms the original joint-action-value function Q_{jt} into a new one Q'_{jt}, that shares the optimal joint action with Q_{jt} and is factorized by additive decomposition. The authors accomplish this task via training three networks: a joint action-value network; individual action-value networks and a state-value network which is used to address the impact of partial observability. The authors go on to demonstrate the superiority of their method over VDN and QMIX in predator-prey and a Gaussian-squeeze task.

Despite the empirical success of VDN, QMIX and QTRAN, training networks to approximate factorized value-functions remain an open challenge for complex coordination problems [79]. Castellini et al. [28] empirically evaluate the representational power of neural network architectures, using a factorization to represent the joint-action-value function based on the sum of smaller action-value functions defined over a coordination graph [64]. A number of factorizations are evaluated, including single agent decompositions (each agent is represented by a single neural network); random partitions (each agent is only involved in one factor); overlapping factors (a fixed number of factors is picked at random from the set of all possible factors); and complete factorization. Evaluations were conducted using one-shot games with six agents, thereby capturing the exponentially large joint-action space problem, while at the same time minimizing noise and other confounding factors. A number of games were identified where all factorizations are unable to overcome multi-agent learning pathologies, in particular relative overgeneralization. The authors note that only joint-action learners are able to currently address these pathologies. For more benign domains, not suffering from pathologies such as relative overgeneralization, the authors achieve near perfect reconstruction for both

complete factorizations of a modest factor size, random overlapping factors, and non-factored action-value functions.

The above finding highlights the need for deep approaches capable of mitigating the pathologies outline in Section 3.1. An alternative approach towards overcoming the challenges outlined above is to use decentralized learning agents with modified deep reinforcement learning architectures [46, 109, 141, 143, 144, 221]. Given the amount of work that has been conducted on concurrent and independent learning, a subset of which we outline in Section 3.2, it should not come as a surprise that many of the recent deep decentralized approaches draw inspiration from multi-agent reinforcement learning literature.

For example, inspired by work on off-environment reinforcement learning [2, 49], Foerster et al. [46] use importance sampling to estimate the probability of a joint-action and subsequently apply a correction when a state transition tuple is sampled. However, this approach requires the agents to maintain an action observation history for each agent. The authors also introduce a second approach that uses *fingerprints* (consisting of the iteration and exploration rate) to disambiguate the age of the state-transition tuples. The authors' fingerprints approach also draws inspiration from previous work, in particular from hyper Q-learning [191], an approach that attempts to mitigate the non-stationarity problem through each agent learning a policy conditioned on estimates of the policies of other agents, based on their behaviour. The authors evaluate their approaches on a decentralized variation of the StarCraft unit micromanagement task, finding fingerprints to be the more effective method for resolving the non-stationarity problem.

While Foerster et al.'s [46] contributions focus on stabilizing the experience replay memory, Omidshafiei et al. [141] consider multi-task reinforcement learning, where the goal is for agents to master a set of related tasks that have shared characteristics. Their approach is to extract knowledge from previous tasks to accelerate learning on a novel task. The authors introduce multi-task multi-agent reinforcement learning (MT-MARL), and propose a two-phase approach towards mastering a partially-observable multi-agent multi-target capture domains, where all agents must capture their targets simultaneously. During the first learning phase hysteretic Q-learning [120] is combined with a Recurrent-DQN [71] and Concurrent Experience Replay Trajectories (CERTS) to enable coordination in single task domains. During the second phase the knowledge from specialized network is distilled into a generalized recurrent multi-task network, using a supervised learning approach (regression), where the specialist networks generate sequences of experiences $\langle o_{i,t}, Q(o_{i,t}; \Theta(i))_{i,t} \rangle$, consisting of observations o for agent i at time-step t and the corresponding Q-values based on the specialized parameters Θ.

The majority of the approaches outlined in this section focus on mitigating the non-stationarity and credit assignment problems. However, as outlined in Section 3.1, a large number of multi-agent learning pathologies exist that can prevent multi-agent deep reinforcement learning agents from converging upon optimal joint-policies. Furthermore,

the finding that factored approaches fail to overcome pathologies such as relative overgeneralization in stateless games is concerning [28]. In Chapters 6 and 7 we therefore evaluate the extent to which we can modify the (Double) DQN architecture to enable independent learners to overcome relative-overgeneralization and the alter-exploration problem in addition to the sample obsolescence problem [143, 144].

We wish to highlight at this point that the literature on multi-agent deep reinforcement learning is diverse and does not exclusively focus on the challenges discussed above. Indeed, in a recent survey Hernandez-Leal et al. [79] identify four (non-disjoint) categories within which work conducted to date can be placed: (i) learning cooperation; (ii) learning communication; (iii) agents modeling agents; and (iv) analysis of emergent behaviors. The literature discussed above has focused on learning cooperation. In the remainder of this section we provide a brief summary of the work conducted in each of the other three areas. Finally, we discuss the limitations of the current state of the art.

3.3.2 Enabling Communication

While this thesis focuses on agents learning implicit coordination strategies, enabling agents to communicate has been one of the long term goals of artificial intelligence research. Agents capable of explicit communication are hypothesized to be more likely to achieve coordination while interacting with other agents, provide a means through which to allow humans to interpret agent behaviour, and potentially even provide answers regarding how language is developed [131]. In contrast to research from natural language processing, e.g., machine translation, answering questions, and sentiment analysis, research on learning communication has predominately focused on agents learning their own communication protocols to achieve coordination while solving cooperative tasks [79]. Agents are judged to possess an understanding of language when they can utilize communication protocols to accomplish their goals in the domain within which they are situated [52, 131]. The emergence of deep reinforcement learning has enabled significant progress in this area [45, 93, 102, 131, 151, 179].

Foerster et al. [45] introducing Reinforced Inter-Agent Learning (RIAL) and Differentiable Inter-Agent Learning (DIAL). Both approaches use deep neural networks to approximate Q-values and messages to send to the other agent. RIAL uses CTDE. DIAL meanwhile takes advantage of the fact that the communication channels are differentiable, allowing gradients to be computed for the messages transmitted during training. Therefore, DIAL is fully-differentiable across agents [45]. While the work conducted by Foerster et al. [45] focuses on value based methods, other approaches have utilized policy gradient approaches (where parameterized policies are trained directly with respect to the expected return) [93, 102, 131, 151, 179]. For example, Mordatch and Abbeel [131] investigate to what extent a grounded composition language can emerge, using streams of abstract discrete symbols (to which no meaning has been assigned) within a cooperative partially observable Markov game. Upon adding vocabulary size penalties that discourage synonyms the authors find that the agents settle on using a consistent

set of symbols for each meaning. Interestingly the authors also observe the emergence of nonverbal communication cues when symbolic communication is disabled [131].

Malysheva [114] et al. introduce MAGnet, a network that maintains a relevance graph neural network, trained to represent the relationship between agents and objects within their environment. MAGnet is also implemented with a message generation module, capable of passing messages based on the relevance graph. The authors applied their architecture to the popular Pommerman environment [123], finding that MAGnet outperforms DQN and MADDPG.

In contrast to the research outlined in this subsection all the agents evaluated in this thesis are unable to communicate with each other via symbolic means. However, for our *deep* experiments in Chapter 7 we also observe nonverbal communication. Our motivation for focusing on implicit coordination is that communication can be expensive in practical applications, and requires efficient protocols [8, 122, 188].

3.3.3 Agents Modelling Agents

This thesis focuses on evaluating the extent to which independent learners can converge upon optimal joint-policies within repeated team-games. Our learners are unaware of the presence of other learning agents in the strategic-form and Markov games that we shall use for our evaluations in Chapters 4 and 5. Meanwhile, although the learners can observe other agents in the Markov games that we shall discuss in Chapters 6 and 7, the algorithms do not attempt to explicitly model the other agents in the system. However, a significant number of multi-agent deep reinforcement learning publications focus on agents trained to model other agents and predict their actions and goals [79].

Approaches range from training separate networks to predict the policies of other agents using hand crafted features [73] to learning them from raw observations using auxiliary loss functions [82]. Meanwhile a variety of methods have been used to predict the goals and actions of other agents. *Self other modeling* (SOM) [156] for instance can be applied to settings where each agent is assigned a goal at the beginning of an episode. SOM predict the other agent's actions using the agent's own policy network, with a separate network subsequently inferring the other agent's goals based on the actions taken. Network parameters are updated at the end of each episode. However, one of the disadvantages of this approach is that optimization takes longer, as additional optimization steps are performed based on the actions that have been observed [79].

In contrast *theory of mind* approaches attempt to estimate the beliefs and mental states of other agents [79]. For instance, Theory of Mind Network (ToMnet) [155] is composed of three modules: a character network, a mental state network, and an action prediction network. The character network uses trajectories from previous episodes to estimate the agent type, whereas the mental state is inferred via observations from the current episode. Outputs from both these networks are fed to the prediction network together with the current observation, in order to predict the agent's next action.

Other approaches in this area have been inspired by literature from game theory and multi-agent learning [79], e.g., using networks to find approximate Nash equilibria in two-player imperfect information games using Neural Fictitious Self-Play (NFSP) [75], or Policy-Space Response Oracles (PSRO) [99], a meta-algorithm for independent reinforcement learners that returns mixtures of approximate best response policies using an empirical game theoretic analysis. PSRO addresses the fact that independent learning agents implemented with approximators such as a DQN are likely to over-fit on each other in repeated games. While the above approaches can be used in cooperative and mixed settings, they are particularly useful in general sum games when an agent wants to estimate whether the opponent/teammate is a cooperator or defector.

3.3.4 Analysis of Emergent Behaviors

Work on emergent behaviour primarily focuses on the extent to which modifying environment impacts the learning dynamics of deep reinforcement learning algorithms [79]. For example, one of the first works on multi-agent deep reinforcement learning evaluated decentralized DQNs learning to play pong, and the extent to which modifying the reward function results in the agents cooperating or competing [187].

Leibo et al. [104] introduced a specific type of Markov game for studying multi-agent deep reinforcement learning agents: the *sequential social dilemma* (SSD). In SSDs inequalities in the reward space reflect those from social dilemmas (general-sum games) in the form of strategic-form games. The paper focuses on the extent to which policies implementing cooperate and defect strategies emerge depending on on environmental factors (with regards to an abundance of resources).

A proportion of our work in Chapter 7 is also concerned with emergent behaviours. However, instead of being placed within SSDs our learners are situated within temporally extended versions of team strategic form games, with the motivation of studying the susceptibility of agents towards the learning pathologies outlined above within temporally extended fully cooperative high dimensional domains.

3.3.5 Practical Challenges

Hernandez-Leal et al. [79] discuss two practical challenges of multi-agent deep reinforcement learning research: hyperparameter tuning and coping with limited computational resources. The authors note that optimizing deep learning architectures is far from trivial, and there is a danger that a choice of sub-optimal hyperparameters can result in a state of the art approach under-performing [124]. This result has been attributed to the difficulty in training deep learning architectures, and the fact that the deep learning community needs to learn more about hyperparameter tuning [79]. However, for smaller research institutions limitations with regards to resources can add to the challenge. Many deep learning architectures require GPUs in order for learners to converge within a reasonable amount of time [133, 161, 162]. Training deep learning architectures to converge typically requires hours, even on moderately complex domains. Furthermore,

memory limitations constrain the number of runs that can be launched in parallel, as well as the number of (decentralized) agents that can inhabit a multi-agent system. This makes hyperparameter tuning and gathering sufficient runs for bench-marking a challenging task [66, 79], and helps explain why a significant number of multi-agent deep reinforcement learning experiments are conducted in simplistic grid-world like domains, reducing the computational cost of conducting experiments [104].

3.3.6 Limitations

The majority of the research discussed in this section focuses on a small subset of the pathologies outline in Section 3.1, such as the impact of stochasticity and the sample obsolescence problem [46]. One of the few exceptions is work conducted by Wei et al. [210] on multi-agent soft Q-learning. However, multi-agent soft Q-learning is currently a centralized approach that has only been tested within a single state continuous game for two agents, with the authors investigating the algorithm's scalability to independent learners within sequential continuous games. Rashid et al. [157] note that the simplest approach to multi-agent reinforcement learning is to forgo centralised learning and CTDE in favour of concurrent learning. However, as discussed in this section, the likelihood of convergence within multi-agent deep reinforcement learning increases significantly when using centralized and factorized approaches, as the non-stationarity issue prevents decentralized agents from converging upon an optimal joint-policy [157]. Though even for CTDE approaches convergence upon optimal joint-policies is also not guaranteed. As noted by Castellini et al. [28], overcoming multi-agent learning pathologies such as relative overgeneralization is far from trivial, even for factorized approaches. Meanwhile there exists a multitude of independent learning approaches designed to help multi-agent reinforcement learning agents overcome multi-agent learning pathologies, as discussed in Section 3.2. This raises the question to what extent these approaches can be scaled to multi-agent deep reinforcement learning.

Chapter 4

Evaluating Independent Reinforcement Learning

The work presented in this chapter is in preparation for a submission to the *Journal of Machine Learning Research*.

Recent years have seen an increase in the number of *multi-agent deep reinforcement learning* publications in high ranking artificial intelligence conferences, including AAAI, ICML, ICLR, IJCAI, NeurIPS, and AAMAS [79]. However, *multi-agent reinforcement learning* is a topic that has been studied for decades. For instance, Lauer and Riedmiller's [100] distributed Q-learning is widely believed to be one of the first independent learning approaches to address cooperative learning pathologies. The authors introduced distributed Q-learning in the year 2000, thirteen years prior to Mnih et al. [128] establishing the field of *deep reinforcement learning*.

In the years that followed a large number of independent learning algorithms were introduced, each designed to address weaknesses identified in previous approaches [17, 90, 120–122, 145, 148, 149, 194, 209]. Therefore, multi-agent deep reinforcement learning research can draw upon a wealth of past literature for both inspiration and guidance. However, Hernandez-Leal et al. [79] warn of a *deep learning amnesia*, where researchers either pay insufficient attention to findings discussed in existing literature, or where past literature has not been cited.

The aim of this chapter is to address this amnesia by re-evaluating traditional independent learning approaches with the criteria of scalability to multi-agent deep reinforcement learning in mind. More specifically, we evaluate to what extent decentralized Q-learning [33], hysteretic Q-learning [120], Frequency Maximum Q-value (FMQ) [90], Recursive Frequency Maximum Q-value (RFMQ) [121] and Lenient Multi-agent Reinforcement Learning 2 (LMRL2) [209] can overcome relative overgeneralization and miscoordination in extended versions of four well studied strategic-form games (introduced in Section 4.2). We establish the robustness of each approach towards an increase in the number of learning agents, and the scale of the penalty values following miscoordination.

Out of the listed approaches LMRL2 is considered the most robust method for preventing relative overgeneralization from occurring within a stochastic reward space [209]. We conduct an extensive empirical evaluation in Section 4.4 to verify these claims, motivated by a *cherry picking* tendency that has emerged in multi-agent reinforcement learning research, where only positive results are reported out of fear that reporting negative findings can lead to a publication being rejected [79].

In addition, inspired by the finding that with sufficient hyperparameter tuning older methods can outperform more recent approaches [79, 124], we consider to what extent the performance of independent learners can be improved with carefully tuned hyperparameters. Furthermore, we consider the scalability of each approach to multi-agent deep reinforcement learning (with the exception of decentralized and hysteretic Q-learning, which have already been scaled to high-dimensional domains [110, 141]). In the next section we discuss traits that we consider desirable in order to successfully scale an independent learning approach to multi-agent deep reinforcement learning.

4.1 Desirable Traits of Independent Learners

The convergence properties for model-free deep reinforcement learning algorithms are known to be brittle [68]. Applying a (potentially novel) approach to a domain for the first time can therefore require a time consuming and meticulous hyperparameter tuning process [79]. Deep learning's hardware requirements have further implications for this tuning process [161, 162]. Only a limited number of runs can be gathered in parallel depending on the number of GPUs available (for approaches using an experience replay buffer), or CPUs (when gathering samples in parallel that are communicated to a centralized learner) [127]. Furthermore, the optimization of deep learning architectures can require hours of training time [79]. Therefore, given that hyperparameter tuning is expensive in multi-agent deep reinforcement learning, our aim is to identify independent reinforcement learning approaches, that:

i require a limited amount of hyperparameter tuning;

ii are robust towards the multi-agent learning pathologies outlined in Section 3.1;

iii are scalable with respect to the number of learning agents;

iv are invariant towards the scale and variance of the reward function;

v use an efficient exploration strategy.

An idealized independent learning approach can therefore overcome multi-agent learning pathologies within a wide range of domains, using a (near) identical hyperparameter configuration. However, achieving consistent convergence upon optimal joinpolicies using a single hyperparameter configuration has proven challenging, even within stateless games with a small action space [90, 121, 209]. This observation is concerning,

given that one of the goals of multi-agent reinforcement learning is to deploy cooperative independent learners within complex settings with unfamiliar dynamics, e.g., space exploration, where autonomous devices such as rovers must learn to cooperate within uncertain and unsafe environments [218]. The aim of this chapter is therefore to identify hyperparameter configurations that increase the likelihood of independent learners converging upon optimal joint-policies within the largest possible number of n-player strategic-form games. Robust configurations are identified via an extensive hyperparameter sweep, allowing us to visualize the inter-dependencies of each algorithm's hyperparameters.

The remainder of this chapter will proceed as follows: first we outline the four strategic-form games that we shall use for our evaluations. We subsequently empirically re-evaluate the independent learning algorithms outlined above. However, with scalability to complex temporally-extended, high-dimensional state-spaces in mind, our aim is to identify robust hyperparameters for each algorithm to enable a consistent performance across settings. Furthermore, we evaluate the scalability of each approach with regards to coping with increasing penalty values (as in Kapetanakis and Kudenko [90]) and number of agents. We focus on the more robust algorithms identified by Wei and Luke's [209] empirical evaluation, selecting approaches that were among the statistically significant best performers. We therefore exclude distributed Q-Learning, which struggles when confronted with stochasticity. We shall however include decentralized Q-learning, as it provides a valuable baseline.

4.2 n-Player Strategic-Form Games

Over the past two decades repeated single-stage strategic-form team-games have often been used as a test-bed for independent learning algorithms [17, 90, 120–122, 145, 148, 149, 194, 209]. At each time-step every agent simultaneously chooses and executes an action [90, 120, 121]. In team-games the learners subsequently receive an identical reward signal corresponding to the joint-actions [90, 120, 121]. This signal is used to update their respective utility values. We conduct our empirical evaluation using n-player strategic-form game versions of the four bimatrix games studied in past multi-agent reinforcement learning literature [90, 120, 121, 149]: (i) *the Climb Game* [33]; (ii) *the Partially Stochastic Climb Game* [90]; (iii) *the Fully Stochastic Climb Game* [90]; (iv) *the Penalty Game* [33]. Below we provide a definition for each game, and discuss the variations used to evaluate the independent learning algorithms.

4.2.1 The Penalty Game

The penalty game introduced by Claus and Boutilier [33] confronts independent learners with the miscoordination pathology. We outline the bimatrix game version of the penalty game in Figure 4.1. Increasing the magnitude of the penalty value p results in lowering utility value estimates for actions A and C for average reward learners using random

exploration. For example, $p = -100$ will result in agents being less likely to choose actions A and C compared to when $p = 0$ [33]. The magnitude of p can therefore increase the likelihood of convergence on joint-action $\langle B, B\rangle$, despite joint actions $\langle A, A\rangle$ and $\langle C, C\rangle$ yielding higher rewards. As a result the game has three deterministic equilibria $\langle A, A\rangle$, $\langle B, B\rangle$ and $\langle C, C\rangle$, with both $\langle A, A\rangle$ and $\langle C, C\rangle$ being Pareto optimal. In our empirical evaluation in Section 4.4 we shall investigate the extent to which scaling the penalty value p causes the percentage of optimal joint-policies to decrease. Furthermore, we conduct evaluations using n-player variations of the penalty game [91]. The rewards in the n-player version are determined using Equation (4.1), where i and j represent agent indexes.

I \ II	A	B	C
A	10 10	0 0	p p
B	0 0	2 2	0 0
C	p p	0 0	10 10

Figure 4.1: The Penalty Game [33]

$$r \leftarrow \begin{cases} 10, & \text{iff } \forall i, a_i = A \vee \forall i, a_i = C, \\ 2, & \text{iff } \forall i, a_i = B, \\ p, & \text{iff } \exists i, a_i = A \wedge \exists j, a_j = C, \\ 0, & \text{otherwise.} \end{cases} \tag{4.1}$$

4.2.2 The Climb Game

Variations of the Climb Game [33] are frequently used to study the susceptibility of independent learners towards relative overgeneralization. The bimatrix version of the Climb Game is outlined in Figure 4.2. The Pareto-Optimal Nash Equilibrium is $\langle A, A\rangle$. However, assuming two independent learners initially choose each of the actions available with equal probability, using an *average based* algorithm, and a sufficiently large penalty p, *Player I* will estimate that C should be preferred over A and B, since $\sum\langle A, j\rangle < \sum\langle C, j\rangle$ and $\sum\langle B, j\rangle < \sum\langle C, j\rangle$ for each of *Player II*'s actions j [209]. *Player II* will come to the same conclusion, resulting in the players gravitating towards the *shadow equilibrium* $\langle C, C\rangle$. If an alternative action is still being played with a small probability, then *Player I* will move from action C to B. Subsequently *Player II* will also *climb* from C to B. At this point the agents will climb no further, having reached a Pareto dominated sub-optimal Nash equilibrium $\langle B, B\rangle$ [33, 90, 149]. As with the penalty game we shall study the impact of increasing the penalty p on the independent

learning algorithms, and shall also conduct experiments with n-player versions of the Climb Game. Learners receive a reward determined by Equation (4.2), where i and j represent agent indexes, and n_a represents the number of players that chose action a.

$$r \leftarrow \begin{cases} 9 + n_A, & \text{iff } \forall i, a_i = A, \\ 5 + n_B, & \text{iff } \nexists i, a_i = A \wedge \forall a_i = B, \forall a_j = C, j > i, \\ p, & \text{iff } \exists i, a_i = A \wedge \exists j, a_j = B, \\ 0, & \text{otherwise.} \end{cases} \tag{4.2}$$

I \ II	A	B	C
A	11 11	p p	0 0
B	p p	7 7	6 6
C	0 0	0 0	5 5

Figure 4.2: The Climb Game [33]

4.2.3 The Partially Stochastic Climb Game

In the Climb Game outlined in Figure 4.2, relative overgeneralization can be overcome with *maximum-based learning*, where agents consider each action i based on the observed $max_j(i, j)$. However, this approach leaves agents vulnerable towards misleading stochastic rewards. Kapetanakis and Kudenko [90], for instance, introduce stochastic variations of the Climb Game, where overoptimistic independent learners can be led astray by misleading stochastic rewards. For example, in the Partially Stochastic Climb Game described in Figure 4.3, the joint action $\langle B, B \rangle$ yields stochastic rewards of 14 and 0 with 50% probability. Therefore maximum based learners are drawn towards $\langle B, B \rangle$, despite each agent only receiving a reward of 7 on average. For the n-player Partially Stochastic Climb Game we add a stochastic reward to the case where all n players choose B in Equation (4.3), where i and j represent agent indexes, and n_a represents the number of players that chose action a, and rewards x/y means rewards x and y are yielded with 50% probability.

$$r \leftarrow \begin{cases} 9 + n_A, & \text{iff } \forall i, a_i = A, \\ 2(5 + n_B)/0, & \text{iff } \forall i, a_i = B, \\ 5 + n_B, & \text{iff } \nexists i, a_i = A \wedge \exists i, a_i = C \wedge \forall a_i = B, \forall a_j = C, j > i, \\ p, & \text{iff } \exists i, a_i = A \wedge \exists j, a_j = B, \\ 0, & \text{otherwise.} \end{cases} \tag{4.3}$$

4.2.4 The Fully Stochastic Climb Game

The fully stochastic Climb Game variation yields stochastic x/y rewards with 50% probability for each joint-action (a_i, a_j) for agents i and j (see Figure 4.4) [90]. For the n-player Fully Stochastic Climb Game (Equation (4.4)) we add a stochastic reward to each joint-action, where i and j represent agent indexes, and n_a represents the number of players that chose action a, and rewards x/y means rewards x and y were yielded with 50% probability. In addition to the penalty value p for miscoordination, we also add a scalable penalty value l to lower the average utility for $\langle A, C\rangle$, $\langle C, A\rangle$ and $\langle C, B\rangle$.

$$r \leftarrow \begin{cases} 10 + n_A/8 + n_A, & \text{iff } \forall i, a_i = A, \\ 2(5 + n_B)/0, & \text{iff } \nexists i, a_i = A \wedge \forall a_i = B, \forall a_j = C, j > i, \\ 5/p, & \text{iff } \exists i, a_i = A \wedge \exists j, a_j = B, \\ 5/l, & \text{otherwise.} \end{cases} \tag{4.4}$$

I \ II	A	B	C
A	11 / 11	p / p	0 / 0
B	p / p	14/0 / 14/0	6 / 6
C	0 / 0	0 / 0	5 / 5

Figure 4.3: The Partially Stochastic Climb Game: The joint-action $(\boldsymbol{B}, \boldsymbol{B})$ yields stochastic rewards of 14 and 0 with 50% probability [90].

I \ II	A	B	C
A	12/10 / 12/10	$5/p$ / $5/p$	$5/l$ / $5/l$
B	$5/p$ / $5/p$	14/0 / 14/0	12/0 / 12/0
C	$5/l$ / $5/l$	$5/l$ / $5/l$	10/0 / 10/0

Figure 4.4: The Fully Stochastic Climb Game: Each joint-action yields stochastic rewards x/y, yielding rewards of x and y with 50% probability [90].

4.3 Previous Findings

Table 4.1 provides a recap of Wei and Luke's [209] findings for each of the bimatrix games outlined in Section 4.2. We observe that none of the approaches achieved a 100% convergence on the optimal joint-policy across all games. However, LMRL2 is among the statistically significant highest performing approaches in all games with the exception of the Partially Stochastic Climb Game (verified using the Marasquilo procedure for χ^2). Meanwhile, Recursive-FMQ (RFMQ) outperformed all other approaches in the Partially Stochastic Climb Game, while converging on the second highest correct run total for the Fully Stochastic Climb Game. Interestingly (decentralized) Q-learning struggles in the three Climb Game variations, while converging on the correct policy in 99.97% of runs in the penalty game. However, this is due to the selection of a rather benign penalty value, $p = -10$. Scaling the penalty significantly changes the results for decentralized Q-learning [90]. The results also reiterate distributed and hysteretic Q-learner's vulnerability towards misleading stochastic rewards, as evident from the low convergence rates within the Partially and Fully Stochastic Climb Games. The results outlined in Table 4.1 were achieved using tuned hyperparameters for each game. For example, for each game LMRL2 was evaluated using a different leniency moderation factor k. We list the default and tuned hyperparameters used by Wei and Luke [209] in Tables 4.2 and 4.3 respectively.

Strategic Game	LMRL2	Q-learning	Distributed Q	Hysteretic Q	FMQ	RFMQ*
Climb Game (DET)	**99.99%**	16.61%	**100%**	**100%**	99.56%	**100%**
Climb Game (PS)	9,930%	18.20%	28.21%	74.54%	98.57%	**99.95%**
Climb Game (FS)	**90.16%**	17.63%	38.74%	25.58%	38.94%	87.23%
Penalty Game	**99.99%**	**99.97%**	**100%**	**100%**	**100%**	**100%**

TABLE 4.1: A summary of Wei and Luke's [209] strategic-form game results, where deterministic, partially stochastic and fully stochastic rewards are abbreviated to DET, PS and FS respectively. The authors verified the statistical significance of the results using the Marasquilo procedure for χ^2. A boldface was used to denote algorithms that did not significantly outperform the other highest performing approaches. *The authors provide a summary table for both strategic form and Markov games, and therefore denote the RFMQ column as SOoN. However, RFMQ is the stateless version of SOoN, designed for strategic form games.

4.4 Empirical Evaluation

In this section we visualize the results from an extensive hyperparameter sweep for LMRL2 [209], decentralized Q-learning [33], hysteretic Q-learning [120], FMQ [90] and RFMQ [121] within the n-player strategic-form games outlined above. We conduct 1,000 training runs for each hyperparameter combination. This allows us to visualize and identify the best performing hyperparameter configurations for each strategic-form

Algorithm	Default Parameters
LMRL2	$\alpha \leftarrow 0.1$, $\nu \leftarrow 0.995$, $MaxTemp \leftarrow 50$, $MinTemp \leftarrow 2$, $k \leftarrow 1$, $\omega \leftarrow 1$, Boltzmann
Q-learning	$\alpha \leftarrow 0.1$, $\epsilon \leftarrow 0.1$, $\mu \leftarrow 1$, ϵ-Greedy
Hysteretic	$\alpha \leftarrow 0.1$, $\epsilon \leftarrow 0.1$, $\beta \leftarrow 0.01$, $\mu \leftarrow 1$, ϵ-Greedy
RFMQ	$\alpha \leftarrow 0.1$, $\epsilon \leftarrow 0.1$, $\alpha_f \leftarrow 0.05$, $\mu \leftarrow 1$, ϵ-Greedy
FMQ	$\alpha \leftarrow 0.1$, $c \leftarrow 10$, $MaxTemp \leftarrow 500$, $MaxMove \leftarrow 2000$

Table 4.2: Default hyperparameters used by Wei and Luke [209].

Algorithm	**Climb Game**	**Climb Game (PS)**	**Climb Game (FS)**	**Penalty Game**
LMRL2	$k = 10^7$	$k = 10^3$	$k = 10^1$	$k = 10^0$
Q-learning	$\epsilon = 0$	$\epsilon = 0.01$	$\alpha = 0.05$	$\alpha = 0.05$ $\epsilon = 0.35$
Hysteretic	$\beta = 0.0001$	$\beta = 0.01$ $MinTemp = 2$ $MaxTemp = 40$ $\delta = 0.99$ Boltzmann	$\beta = 0.001$, $\nu = 0.99$	$\beta = 0.01$ $\epsilon = 0.12$
FMQ	-	-	$c = 200$	-
RFMQ	- -	$\alpha_f = 0.05$ $\epsilon = 0.05$	$\alpha_f = 0.03$ $\alpha = 0.03$	- -

Table 4.3: Tuned hyperparameter configurations used by Wei and Luke [209].

game, and evaluate to what extent optimal hyperparameter configurations are overfitting on the problem in question. With regards to default parameters, to remain inline with previous work [90, 209]:

- Unless specified otherwise we use learning rate $\alpha = 0.1$;
- Q-values are initialized to 0;
- With the exception of FMQ all training runs end after 15,000 iterations;
- To remain inline with past literature FMQ is trained for 2,000 iterations.

We implement the repeated strategic form games as outlined in Wei and Luke's [209] Appendix, where each iteration ends in a terminal (absorbing) state. Therefore, as in previous work [33, 90, 120–122], we compute utility values using an exponentially weighted moving average using the stateless version of each algorithm discussed in Section 3.2. Finally, we list the scaled penalty values for each game variation in Table 4.4.

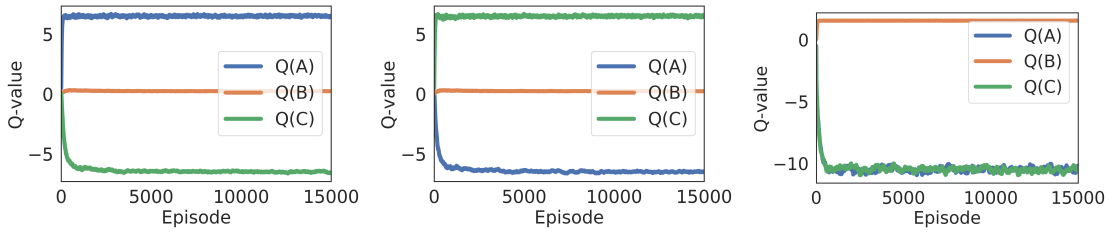
Reward Scale	Penalty Game	Climb Game	Partially Stochastic Climb Game	Fully Stochastic Climb Game (k, l)
Low	-10	-30	-30	(-65, -5)
Medium	-100	-300	-300	(-650, -50)
High	-1000	-3000	-3000	(-6500, -500)

TABLE 4.4: Strategic-Form Games Penalty Look-up Table.

4.4.1 Decentralized Q-learning

Despite decentralized Q-learners being average reward learners by definition (see Equation (3.5)), Wei and Luke's [209] empirical evaluation found that 99.97% of Penalty Game runs converged upon an optimal joint-policy when the penalty value $p = -10$. The agents were implemented with a learning rate $\alpha = 0.05$ and a fixed ϵ-Greedy exploration rate of $\epsilon = 0.35$. However, while this particular setting beats previous baseline convergence rates for $p = -10$, scaling the penalty value p has been shown to result in agents finding actions A and C less attractive [33, 90]. This raises the question of how robust this hyperparameter setting will prove upon scaling p.

Interestingly we also observe 99.97% optimal joint-policies upon conducting 10,000 training runs using the above settings. However, this number decreases to 0% upon setting the miscoordination penalty p to -100. Scaling p has a significant impact on the Q-value estimates for each action, as illustrated in Figure 4.5. In Sub-Figures 4.5(a) and 4.5(b) we observe for runs that converged on $\langle A, A\rangle$ and $\langle C, C\rangle$ respectively, the average Q-values for the respective actions were just above 6. Meanwhile, the estimated utility of actions belonging to the alternative Pareto optimal solution was -6. The utility estimates for actions upon which the agents converged being significantly lower than 10 can be explained by the increased global exploration resulting from using $\epsilon = 0.35$ with a decay rate of 1.0. However, the average utility values for both actions drop below -10 upon setting p to -100, as illustrated in Sub-Figure 4.5(c). Furthermore, we observe a significant decrease in the average reward (Figure 4.6). Therefore, the optimal hyperparameter setting identified by Wei and Luke [209] for the Penalty Game with $p = -10$ does not scale to $p = -100$.



(a) $p = -10$, $\langle A, A\rangle$ (b) $p = -10$, $\langle C, C\rangle$ (c) $p = -100$, $\langle B, B\rangle$

Figure 4.5: The Penalty Game: Average Q-value comparison for decentralized Q-learning agents for $p = -10$ and $p = -100$. For (a) and (b) we compute the averages for runs that converged upon joint-policies $\langle A, A\rangle$ and $\langle C, C\rangle$ respectively.

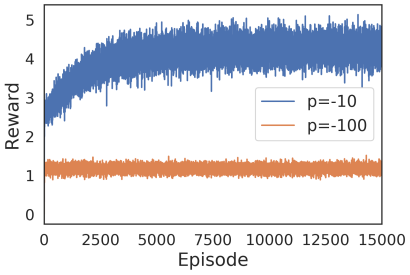
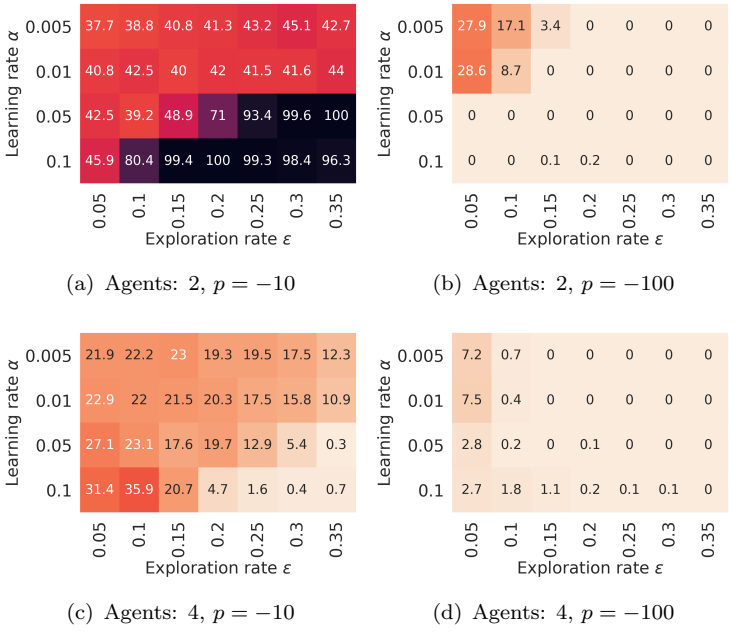


Figure 4.6: Mean reward comparison for $p = \{-10, -100\}$ in the Penalty Game. (Decentralized Q-learning)

For our first hyperparameter sweep we investigate if hyperparameter combinations exist that enable decentralized Q-learning to overcome an increased penalty $p = -100$, while also using a stationary exploration strategy. We also evaluate the impact of increasing the number of agents from two to four. Runs are gathered for each combination of learning rate $\alpha = \{0.1, 0.05, 0.01, 0.005\}$ and stationary exploration rate $\epsilon = \{0.35, 0.3, 0.25, 0.20, 0.15, 0.10, 0.05\}$. For each combination of the α and ϵ settings we gather 1,000 runs. In Figure 4.7 we use heat-maps to illustrate the percentage of runs that converge upon an optimal joint-policy for each setting.



(a) Agents: 2, $p = -10$

(b) Agents: 2, $p = -100$

(c) Agents: 4, $p = -10$

(d) Agents: 4, $p = -100$

Figure 4.7: Percentages of correct runs for decentralized Q-learning within variations of the penalty game using ϵ-Greedy exploration with stationary exploration rates.

As expected we observe that scaling the penalty value p leads to a reduction in the percentage of correct (optimal) runs for each setting (Sub-Figures 4.7(b) and 4.7(d)). While the tuned parameters $\alpha = 0.5$ and $\epsilon = 0.35$ used by Wei and Luke [209] converge on the optimal joint-policy for each of the 1,000 runs when $p = -10$, this number drops to 0 upon scaling the penalty to $p = -100$. Interestingly, when $p = -10$ we observe that in the two agent penalty game (Sub-Figure 4.7(a)) 100% of the runs converged on

the optimal joint-policy for $(\alpha = 0.05, \epsilon = 0.35)$ and $(\alpha = 0.1, \epsilon = 0.2)$; with a greater percentage of optimal joint-policies observed for larger learning rates α and exploration rates ϵ. However, in the heat-map for two agents with $p = -100$ (Sub-Figure 4.7(b)) we observe that agents using a lower learning rate α are more likely to converge on the optimal joint-policy, when combined with a low exploration rate ϵ. Only a small percentage of runs converge upon the correct joint-policy when $\alpha > 0.01$ and $\epsilon > 0.1$. The highest percentages of runs are achieved for $\alpha = \{0.01, 0.005\}$ and $\epsilon = 0.05$.

With regards to scaling the number of agents, the heat-maps illustrate a significant reduction in the percentage of correct runs upon increasing the number of agents from two to four, even when $p = -10$ (Sub-Figure 4.7(c)). We observe that $\alpha = 0.1$ and $\epsilon = 0.1$ deliver the highest convergence rate (36%). We hypothesize that, as a result of an increase in global exploration, agents utilizing a larger exploration rate ϵ rarely converge upon correct joint-policies in the four-agents setting. For instance, one of the strongest settings in the two agent scenario $(\alpha = 0.05, \epsilon = 0.35)$ only converges on correct joint polices on 0.3% of runs. As in the two agent setting, scaling the penalty to $p = -100$ results in learners with both low learning and exploration rates having the largest convergences percentage. More specifically, we observe that agents with $\alpha = \{0.01, 0.005\}$ and $\epsilon = 0.05$ outperform the other settings (Sub-Figure 4.7(d)).

We hypothesize that learners benefit from lower learning and exploration rates within the more challenging settings, due to a small percentage of agents choosing the optimal joint-actions $\langle A, A \rangle$ during the initial steps. Subsequently a combination of low global exploration and small update step size protect the agents from the alter exploration problem. We provide evidence to support this hypothesis in Figure 4.8, which illustrates the optimal joint-policy percentages for additional four-player, $p = -100$ runs, where the agents receive a demonstration of the optimal joint-actions during the first iteration: for each agent i action a was set to $a_i = A$.

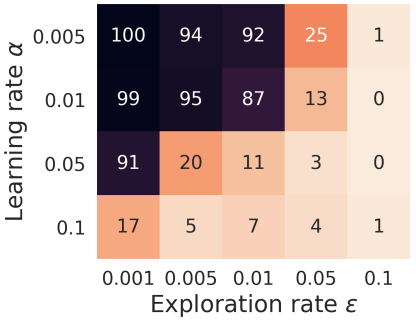
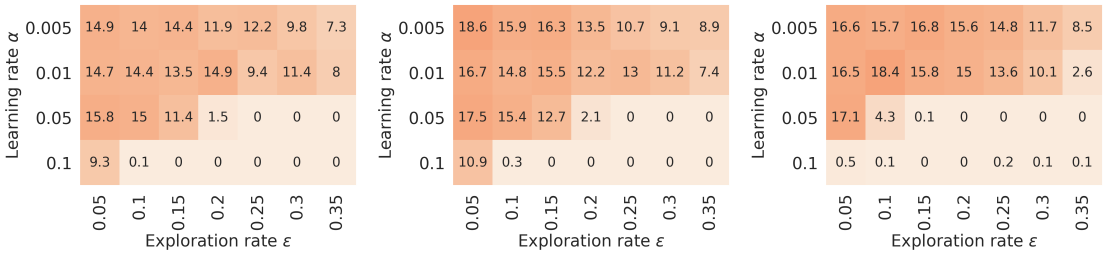


Figure 4.8: Percentage of optimal joint-policies upon giving decentralized Q-learners a supervised start within the four-player Penalty Game with penalty $p = -100$. During the first iteration action $a_i = A$ for each agent i.

We observe that a relatively low likelihood of each agent exploring, with $\epsilon = 0.1$, is sufficient for the agents to converge on a sub-optimal joint-policy. Furthermore, even with a small likelihood of exploration at each step, with $\epsilon = 0.001$, we observe a significant decrease in the number of optimal joint policies as learning rate α is increased

from 0.05 to 0.1. We find that decentralized Q-learners are also more likely to converge upon the optimal joint-policy within the three variations of the Climb Game when using lower learning rates α and exploration rates ϵ (Figure 4.9). However, the percentage of optimal join-policies remains low across all settings.



(a) Deterministic (b) Partially Stochastic (c) Fully Stochastic

Figure 4.9: Correct run percentages for decentralized Q-learners within the two-agent low-penalty Climb Game variations. We observe a higher convergence rate for learners using low exploration rates ϵ and learning rates α.

While Wei and Luke [209] evaluate decentralized Q-learning agents with a stationary exploration strategy, Matignon et al. [121] note that decentralized Q-learning can often benefit from a *greedy in the limit with infinite exploration* (GLIE) strategy, e.g. decreasing the exploration frequency throughout the training process. For instance, using Boltzmann exploration with a initial temperature of 5000 and a decay rate of 0.997, Matignon et al. [121] report a 96.6% convergence rate in the penalty game with $p = -100$. We conduct our own hyperparameter sweep using Boltzmann exploration for $MaxTemp = \{50, 500, 5000\}$ and decay rates $\{0.9, 0.99, 0.995, 0.996, 0.997, 0.998, 0.999\}$, conducting 1,000 training runs for each combination.

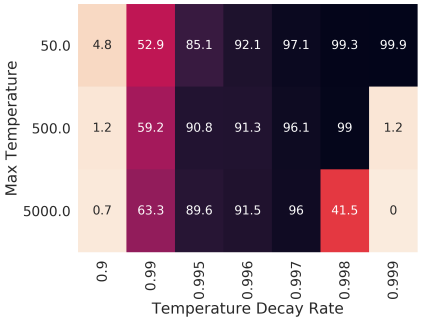
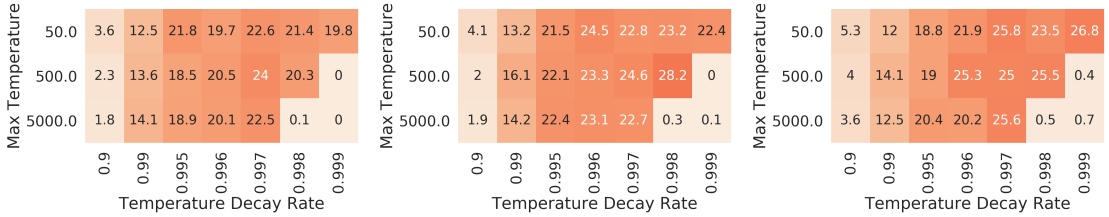


Figure 4.10: Convergence rates for decentralized Q-learners using Boltzmann exploration within the medium-penalty two-agent Penalty Game.

We illustrate the convergence rates in the heat-map in Figure 4.10. We replicate the 96% convergence rate achieved by Matignon et al. [121] for $MaxTemp = 5000$ and a decay rate of 0.997. Furthermore, we observe three configurations that converge on 99% of the runs conducted: $\{(MaxTemp = 500, Decay = 0.998), (MaxTemp = 50, Decay = 0.998)$ and $(MaxTemp = 50, Decay = 0.999)\}$. In fact, we observe a pattern where learners benefit from lower maximum temperature values and slower

temperature decay rates. Meanwhile, combining large maximum temperature values with slow temperature decay rates, i.e., remaining exploratory leads to a large percentage of sub-optimal joint-policies (for example $(MaxTemp = 500, Decay = 0.999)$ and $(MaxTemp = 5000, Decay > 0.997)$). Similarly we observe a drop in convergence when using a decay rate less than 0.997.

However, even with Boltzmann exploration decentralized Q-learners are unable to prevent relative overgeneralization from occurring in the three Climb Game variations (Figure 4.11). We do however identify settings that improve upon the results reported by Wei and Luke [209]: in all three Climb Game variations learners using a maximum temperature of either 50 or 500, and decay rates 0.997 or 0.998 converge upon optimal joint-policies on above 20% of runs.



(a) Deterministic (b) Partially Stochastic (c) Fully Stochastic

Figure 4.11: Decentralized Q-learning: Correct run percentages for two-agent low penalty Climb Game variations using Boltzmann exploration.

The above results provide further evidence that baseline methods against which novel approaches are benchmarked are often tuned insufficiently [78, 124]. While decentralized Q-learning has been found to perform worse than state of the art methods in the Penalty Game with $p = -100$ [90, 122], we show that competitive convergence rates can be achieved with a carefully tuned Boltzmann exploration. However, the tuned hyperparameters are unable to deliver a consistent performance across domains. Furthermore, hyperparameter tuning is more expensive within complex domains. Therefore, the above results reiterate the need for robust independent learning approaches. In particular, balancing the exploration-exploration trade-off within high-dimensional domains is a non-trivial problem [142, 176, 190]. Therefore, independent learning approaches are required that are capable of overcoming relative overgeneralization and miscoordination while mitigating the challenges introduced by scaled penalty values [90] and increased global exploration [122].

4.4.2 Frequency Maximum Q-value

In the previous section we establish that balancing the exploration-exploitation trade-off is critical for decentralized Q-learners to mitigate multi-agent learning pathologies. We now turn to Frequency Maximum Q-value (FMQ) [90]. While scaling FMQ to more complex domains requires considerable modifications [121], we consider that valuable lessons can be learned from this approach, especially with regards to modifying

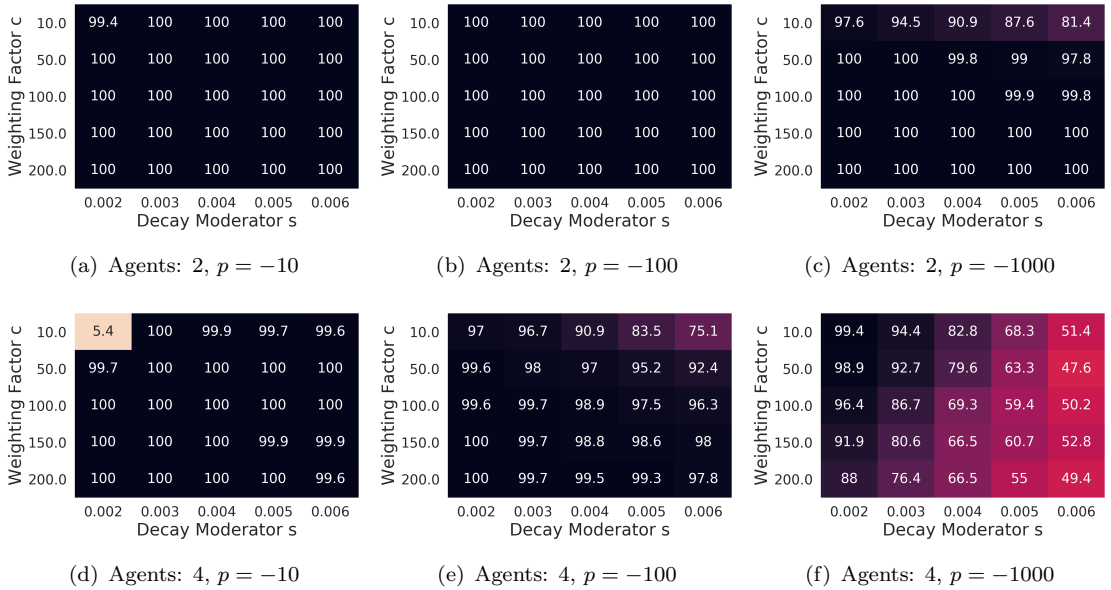
the Boltzmann exploration method, an approach Wei and Luke [209] also proposed for LMRL2. We evaluate inter-dependencies between the temperature decay moderator s and the EV term weighting factor c, by training agents implemented with FMQ using each combination of the following settings:

- $s = \{0.002, 0.003, 0.004, 0.005, 0.006\}$;
- $c = \{10, 50, 100, 150, 200\}$.

To remain inline with previous research [90, 209] we set:

- $MaxTemp = 500.0$;
- $MinTemp = 1.0$.

With the exception of the Fully Stochastic Climb Game, we identify numerous hyperparameter configurations that converge upon correct joint-policies within 100% of the runs for the low-penalty, two-player versions of each game [1]. Furthermore, we observe that FMQ scales well in the Penalty Game when increasing the scale of the penalty value and the number of agents, as evident from the heat-maps illustrating the correct policy percentages for each EV term weighting factor c and temperature decay moderator s configuration in Figure 4.12.



(a) Agents: 2, $p = -10$ (b) Agents: 2, $p = -100$ (c) Agents: 2, $p = -1000$

(d) Agents: 4, $p = -10$ (e) Agents: 4, $p = -100$ (f) Agents: 4, $p = -1000$

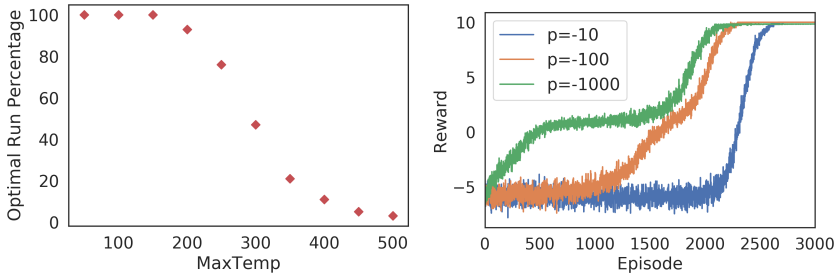
Figure 4.12: Heat-maps illustrating the correct run percentages for independent learners using FMQ within six variations of the Penalty Game.

In the two-player Penalty Game we find that learners benefit from an increased reliance on the EV heuristic as the penalty p increases. In Sub-Figure 4.12(c), for instance,

[1] We provide a detailed summary of our FMQ results in Appendix A, Section A.1.

where $p = -1000$, we observe 100% convergence when $c \geq 150$. Meanwhile, the percentages of correct join-policies decrease across a range of hyperparameter configurations in arguably the most challenging domain setting, with four-agents and high-penalty values, as illustrated in Sub-Figure 4.12(f). However, FMQ still delivers high convergence rates for $s = 0.002$, including 99.4% when $c = 10$. Furthermore, we observe that the learners benefit from lower decay moderators s in general in this setting, therefore preferring a slow transition from explorers to exploiters.

An interesting anomaly emerges within the four-player penalty game. When $p = -10$ we notice a sharp decrease in the percentage of optimal joint-policies for $c = 10$ and $s = 0.002$ (Sub-Figure 4.12(d)). Not only does this result stand out from the 100% convergence rate achieved by the other hyperparameter configurations, but upon inspecting the heat-maps in Sub-Figures 4.12(e) and 4.12(f) it becomes apparent that $c = 10$ and $s = 0.002$ achieved 97% and 99% for $p = -100$ and $p = -1000$ respectively. Initially the 5.4% convergence rate for $c = 10$, $s = 0.002$ and $p = -10$ appears to be an error. However, upon closer investigation we find there is a further critical inter-dependency between parameters c, s and the value chosen for $MaxTemp$. Choosing lower values for $MaxTemp$ increases the percentage of optimal joint policies, as the scatter plot in Sub-Figure 4.13(a) illustrates. However, we can also increase the number of training iterations and give the learners more time to converge upon one of the Pareto optimal equilibria, as the results of experiments conducted with $MaxMove = 3000$ in Sub-Figure 4.13(b) demonstrate.

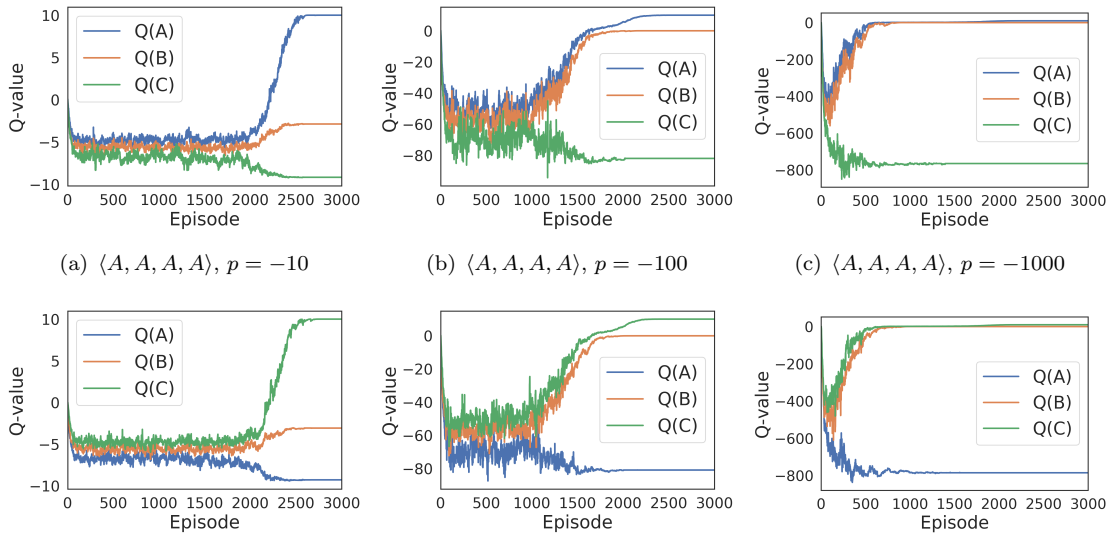


(a) $MaxTemp$ impact on optimal run % (b) Mean reward over 3000 iterations

Figure 4.13: The plots illustrate the interesting interdependence between FMQ hyperparameters. We observe a delayed convergence for FMQ learners implemented with $c = 10$, $s = 0.002$ and $MaxTemp = 500$ when confronted with the four-player Penalty Game with penalty $p = -10$. Sub-Figure (a) illustrates how choosing a lower $MaxTemp$ value increases the percentage of optimal joint-policies for this particular FMQ configuration, whereas Sub-Figure (b) illustrates that given more time, FMQ learners with $MaxTemp = 500$ will eventually converge upon an optimal joint-policy. The plot also illustrates, that due to the FMQ Boltzmann selection method the same setting requires less time when the scale of the penalty value p is increased.

Sub-Figure 4.13(b) raises the question why convergence requires less iterations for $p = -100$, and even fewer for $p = -1000$. Upon closer inspection, we observe that while actions from one of the Pareto optimal solutions obtain the highest Q-values after only a few iterations for each setting, including $p = -10$ (See Figure 4.14), due to the

Boltzmann selection method being able to distinguish between larger Q-values earlier, the point at which the selection probabilities are distinguishable occurs significantly earlier for larger penalty values p, as illustrated in Figure 4.15. Therefore, in this particular setting the agents actually benefit from receiving larger penalty values.



(a) $\langle A, A, A, A\rangle$, $p = -10$ (b) $\langle A, A, A, A\rangle$, $p = -100$ (c) $\langle A, A, A, A\rangle$, $p = -1000$

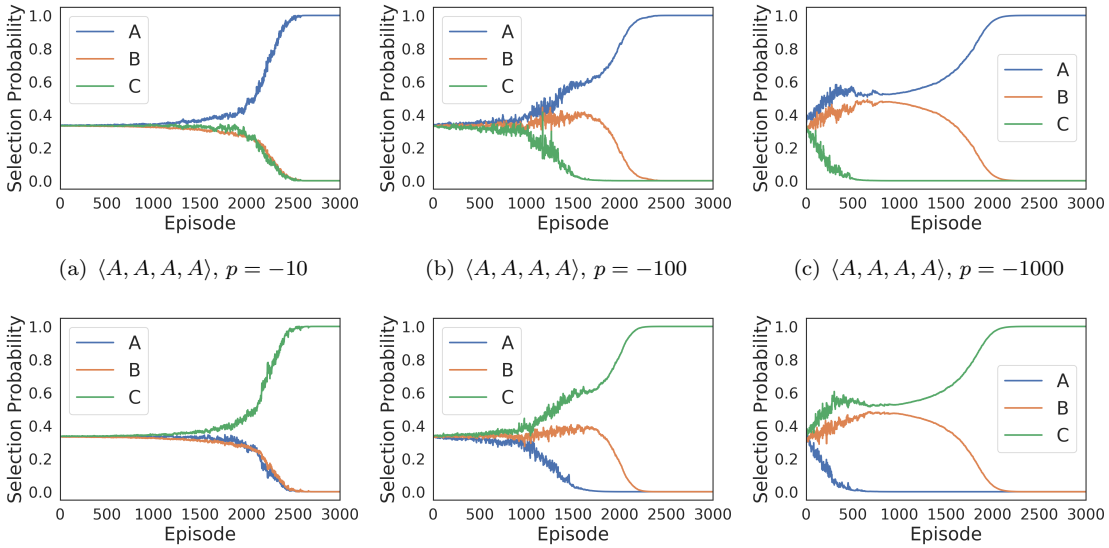
(d) $\langle C, C, C, C\rangle$, $p = -10$ (e) $\langle C, C, C, C\rangle$, $p = -100$ (f) $\langle C, C, C, C\rangle$, $p = -1000$

Figure 4.14: Q-values (averaged over 1,000 training runs) for FMQ in the four-player Penalty Game. Illustrations are provided for runs that have either converged upon the joint-actions $\langle A, A, AA\rangle$ or $\langle C, C, C, C\rangle$. The learners are implemented with an EV weighting factor $c = 10$, and a temperature decay moderator $s = 0.002$.

This result has implications, given that in arguably the most challenging setting (the four-agent penalty game with $p = -1000$), training agents with $c = 10$ and $s = 0.002$ resulted in the joint-best convergence rate (99%). Therefore, while FMQ has the potential to overcome miscoordination in domains with severe penalties and $n > 2$ agents, selecting optimal hyperparameters requires considerations regarding the scale of the rewards, the desired temperature decay factor combined with the $MaxTemp$, and the extent to which the EV term should be incorporated into the action selection mechanism.

This challenge increases significantly for the Climb Game variations, where no single hyperparameter configuration can be identified that scales well upon increasing the penalty values and the number of agents. While we observe 100% convergence on correct policies in the two-agent, low-penalty, deterministic and partially stochastic versions of the Climb Game, the percentage of correct policies drops significantly upon increasing the number of agents and the scale of the penalty values. Furthermore, identifying a hyperparameter configuration that enables a high-percentage of correct policies across domain settings is non-trivial.

The heat-maps in Figure 4.16 illustrate the correct run percentages for the three Climb Game variations in two and four-agents settings with medium sized penalty values. We observe that the preference in the EV term weighting factor c shifts depending on the



(a) $\langle A, A, A, A\rangle$, $p = -10$ (b) $\langle A, A, A, A\rangle$, $p = -100$ (c) $\langle A, A, A, A\rangle$, $p = -1000$

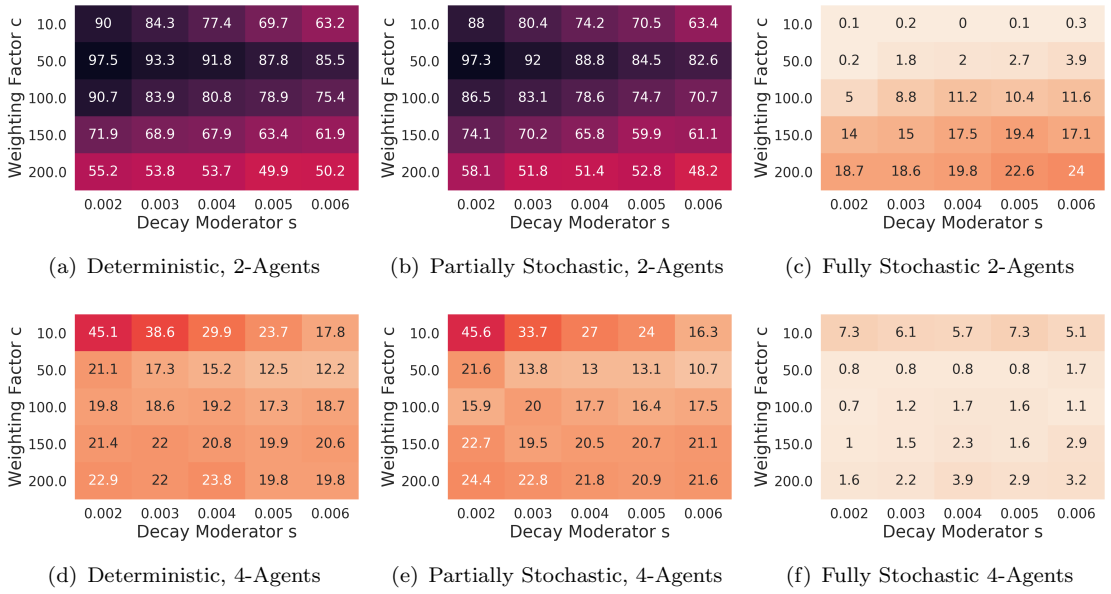
(d) $\langle C, C, C, C\rangle$, $p = -10$ (e) $\langle C, C, C, C\rangle$, $p = -100$ (f) $\langle C, C, C, C\rangle$, $p = -1000$

Figure 4.15: FMQ Boltzmann selection probabilities for the four-player Penalty Game. The learners are implemented with an EV weighting factor $c = 10$, and a temperature decay moderator $s = 0.002$. The selection probabilities (averaged over 1,000 training runs) are illustrated for runs that converge upon the joint-actions $\langle A, A, A, A\rangle$ or $\langle C, C, C, C\rangle$. Convergence requires fewer iterations for larger penalty values p.

game type and the number of agents. For instance, in the deterministic and partially stochastic two-agent variations, learners using $c = 50$ consistently outperform other settings, regardless of the value chosen for the decay moderator s. However, upon scaling to the four-agent variations $c = 10$ outperforms agents using $c = 50$ and all other configurations for $0.002 \leq s < 0.006$. Using $c = 10$ also enables the highest percentage of correct runs on the four-agent Fully Stochastic Climb Game. However, for the two-agent variation we observe that the agents prefer larger weighting factors c, with $c = 200$ resulting in the highest convergence rates across all s settings. Interestingly Wei and Luke [209] also choose $c = 200$ for the Fully Stochastic Climb Game. Finally, we observe that learners in the two-agent Fully Stochastic Climb Game achieve higher convergence rates using larger decay moderators s, in contrast to the other setting (including the penalty game), where the learners generally perform better with lower decay-moderators s. We are therefore unable to identify a single hyperparameter setting for FMQ that enables consistent convergence across games.

4.4.3 Recursive Frequency Maximum Q-value

As with all the independent learning approaches discussed in this chapter, for Recursive Frequency Maximum Q-value (RFMQ) considerations are required regarding the exploration-exploitation trade-off. The success of RFMQ hinges on learners maintaining accurate estimates of the frequency with which $Q_{max}(a)$ can be observed for each action $a \in \mathcal{A}$ for coordinated outcomes. However, depending on the value chosen for



(a) Deterministic, 2-Agents (b) Partially Stochastic, 2-Agents (c) Fully Stochastic 2-Agents

(d) Deterministic, 4-Agents (e) Partially Stochastic, 4-Agents (f) Fully Stochastic 4-Agents

Figure 4.16: FMQ: Comparison of the percentage of runs that converge upon optimal outcomes for two and four-agent versions of each Climb Game variation.

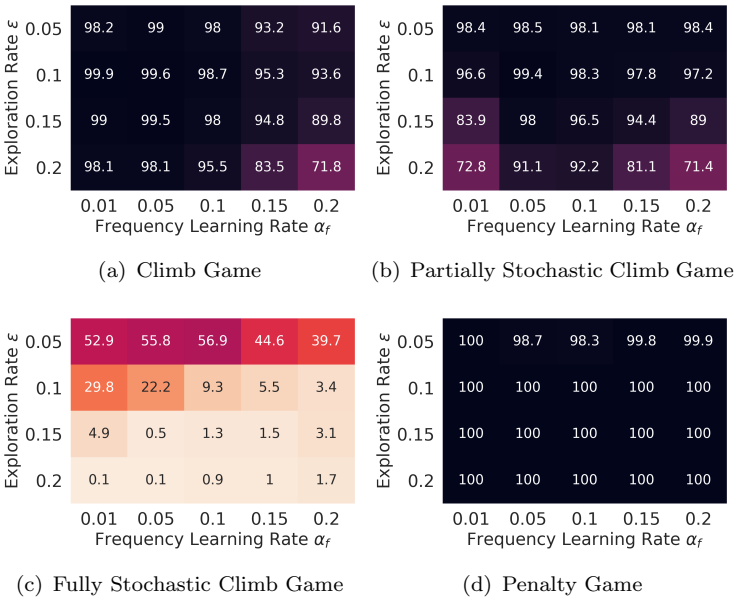
the frequency learning rate α_f, frequent miscoordination can lead to deterioration of the frequency terms. Therefore, high global exploration can result in RFMQ relying exclusively on the Q-value estimates. One approach towards mitigating the deterioration of the frequency terms is to choose a sufficiently small frequency learning rate α_f [121]. However, care is required when choosing α_f, due to RFMQ implementing maximum based learning as α_f approaches zero, with $\alpha_f = 0$ implementing distributed Q-learning [121]. To minimize the noise introduced by miscoordination Matignon et al. [121] use a stationary policy, namely ϵ-greedy with a fixed ϵ. As a result we conduct our hyperparameter sweep for RFMQ using the following parameters combinations:

- $\epsilon = \{0.2, 0.15, 0.1, 0.005\}$;
- $\alpha_f = \{0.01, 0.05, 0.1, 0.15, 0.2\}$.

We illustrate the optimal joint-policy percentage for each configuration in the low-penalty bimatrix games in Figure 4.17. In Sub-Figure 4.17(d) we identify a number of settings for the Penalty Game ($p = -10$) where 100% of training runs converged upon the correct joint policy, only observing a small decrease in percentage of correct joint-policies when learners use a low exploration rate $\epsilon = 0.05$.

However, the convergence rates for the three Climb Game variations are below the results reported in previous work [121, 209]:

Deterministic Climb Game: We are unable to replicate the 100% correct runs for the Climb Game [121, 209]. For the configuration used by Wei and Luke [209] ($\epsilon = 0.1$, $\alpha_f = 0.05$) we observe a convergence rate of 99.6%. We find this number increases to 99.9% for the configuration used by Matignon et al. [121] ($\alpha_f = 0.01$ and $\epsilon = 0.05$), who observed 100% correct joint-polices over 500 training runs.



(a) Climb Game (b) Partially Stochastic Climb Game

(c) Fully Stochastic Climb Game (d) Penalty Game

Figure 4.17: RFMQ: Correct run percentages for two-agent, low-penalty implementations of the penalty game and the three variations of the Climb Game.

Partially Stochastic Climb Game: Matignon et al. [121] report a 100% convergence rate for the Partially Stochastic Climb Game, while Wei and Luke [209] report 99.95%. Our convergence rate for the setting used by Wei and Luke [209] ($\alpha_f = 0.05$, $\epsilon = 0.05$) is 98.5%. However, we do observe 99.4% correct policies for $\alpha_f = 0.05$ and $\epsilon = 0.05$.

Fully Stochastic Climb Game: Wei and Luke [209] observe a 87.23% convergence rate using $\alpha_f = 0.3$, $\alpha = 0.03$, and $\epsilon = 0.1$. In contrast we observe a convergence rate of 37%, with $\alpha_f = 0.3$ leaving the recursive frequency estimates vulnerable towards miscoordination. Meanwhile, Matignon et al. [121] report a 56% convergence rate using $\alpha_f = 0.01$ and $\epsilon = 0.0.5$. Our results in Sub-Figure 4.17(c) are approximately in-line with those reported by Matignon et al. [121]. We observe an increase in the correct run percentage for $\epsilon = 0.05$. While we only observe 52.9% for $\alpha_f = 0.01$, we do obtain 56.9% for $\alpha_f = 0.1$. In Figure 4.18 we show that the convergence rate can be improved to 64.7% by lowering the exploration rate ϵ. However, for $\epsilon < 0.03$ we observe a decrease in percentage of optimal runs.

Figure 4.19 illustrates the consequences of using a large frequency learning rate on the Q-values, action evaluations $E(a)$ and frequency estimates $F(a)$. We compare Wei and Luke's [209] configuration ($\alpha = 0.03$, $\alpha_f = 0.3$, $\epsilon = 0.1$) against the best configuration encountered during our evaluation ($\alpha = 0.1$, $\alpha_f = 0.05$, $\epsilon = 0.03$). For the runs gathered using the large $\alpha_f = 0.3$ we observe a faster decay of the $F(a)$ values, which drop below 0.5 after only a few episodes. This decay is significantly slower for $\alpha_f = 0.05$, resulting in $E(A)$ and $Q(A)$ having the highest values on average. Using $\alpha_f = 0.3$ meanwhile results in action C having the highest action evaluation value.

The convergences rates illustrated in Figures 4.17 and 4.18 introduce a dilemma regarding selecting a hyperparameter configuration that achieves an optimal performance

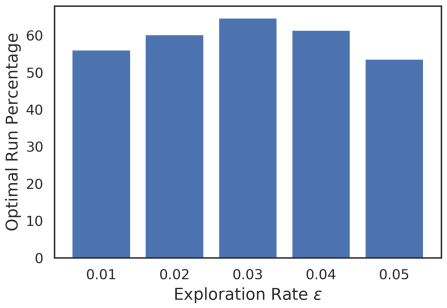


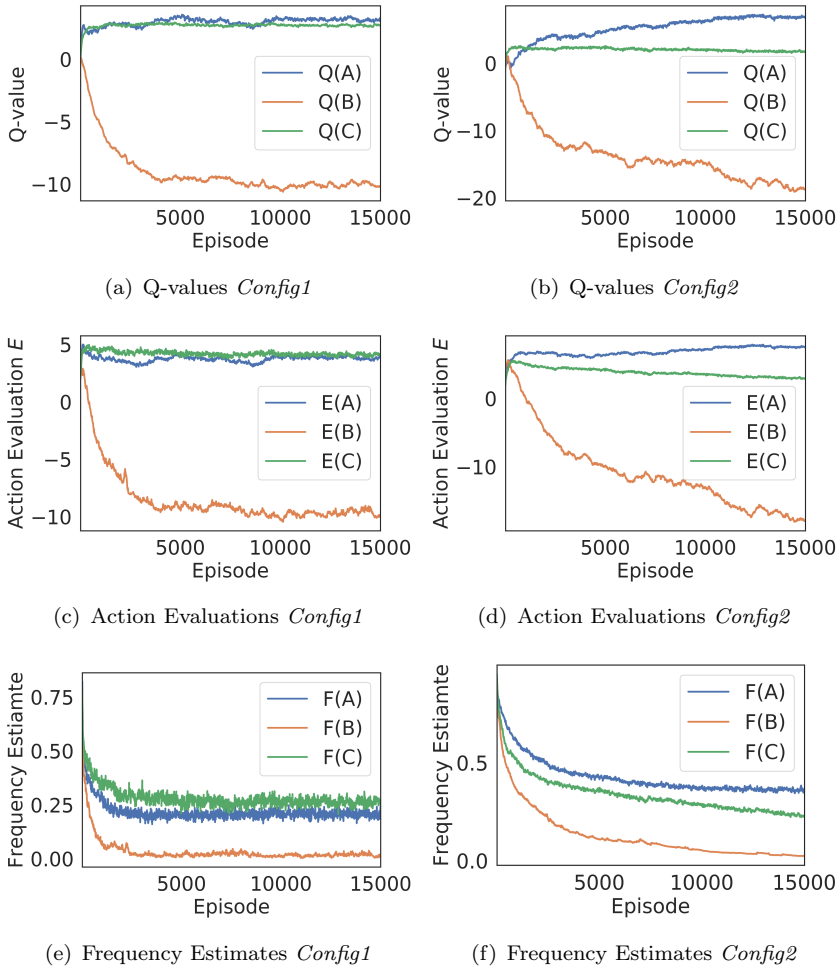
Figure 4.18: RFMQ: Correct run percentages for the two-agent low-penalty Fully Stochastic Climb Game using frequency learning rate $\alpha_f = 0.05$.

across domains. The Fully Stochastic Climb Game evidently requires an exploration rate $\epsilon < 0.1$. In contrast in the remaining games learners achieve their respective highest convergence rates across α_f settings using $\epsilon = 0.1$. However, all three games benefit from a $\epsilon < 0.1$ upon increasing the scale of the penalty value, while suffering significant decrease in the percentage of optimal policies across each hyperparameter combination (Figure 4.20). Furthermore, we observe that upon increasing the penalty values the learners benefit from maintaining an optimistic disposition for longer, via a low frequency learning rate α_f.

In addition, we find that choosing an optimal exploration rate ϵ depends on the number of learners present in the system. We demonstrate this by conducting additional experiments for a range of ϵ settings within the four-agent, low-penalty variation of each strategic-form game. For each ϵ configuration we collect 1,000 runs. Each run consists of 150,000 iterations, ensuring that the learners are given sufficient time to observe the optimal joint action. We illustrate the correct run percentages in Figure 4.21.

We therefore find that, with the exception of the Fully Stochastic Climb Game, an increase in global exploration is required to enable convergence on an optimal joint-policy. For each game $\epsilon = 0.2$ results in the highest convergence rate. For the Penalty Game we observe a 99% convergence rate. In contrast, Matignon et al. [121] achieve 91% in a different version of the four-agent Penalty Game, where a positive reward only requires over half of the agents to choose either A or C, with the remaining agents choosing any action other than B. The authors observed this result using fewer iterations (50,000), and modifying their Q_{max} initialization, setting the max Q estimates to -100.

RFMQ requires an excessive number of iterations in order to achieve convergence due to using a fixed exploration rate ϵ. In Figure 4.22 we illustrate the $Q_{max}(a)$ estimates for each action $a \in \mathcal{A}$ within the four-agent, low-penalty deterministic and Partially Stochastic Climb Games. We divide 100 runs into correct and incorrect groups depending on the joint-policy, and subsequently plot the running average Q_{max} values. We observe that for the correct runs only a few iterations were required for the learners to establish the maximum Q-value that could be obtained for the joint-action $\langle B, B, B, B \rangle$. Obtaining $Q_{max}(A)$ meanwhile requires a large number of iterations on average.



(a) Q-values *Config1*

(b) Q-values *Config2*

(c) Action Evaluations *Config1*

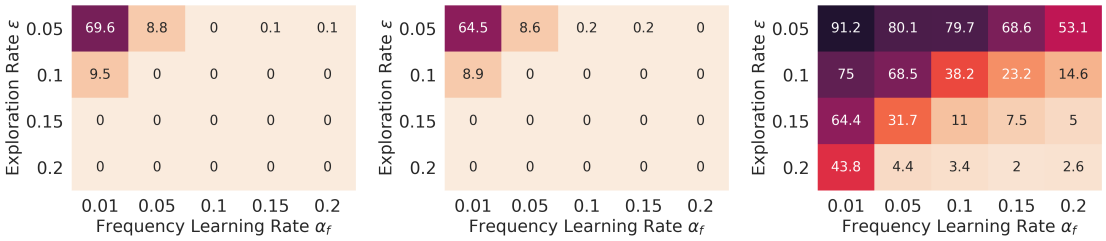
(d) Action Evaluations *Config2*

(e) Frequency Estimates *Config1*

(f) Frequency Estimates *Config2*

Figure 4.19: A comparison of Q-values, action evaluation values $E(a)$ and the frequency estimates $F(a)$ averaged over 100 runs for RFMQ in the Fully-Stochastic Climb Game. We compare the configuration used by Wei and Luke [209] (*Config1*$= \{\alpha \leftarrow 0.03, \alpha_f \leftarrow 0.3, \epsilon \leftarrow 0.1\}$), against the best configuration encountered during our evaluation (*Config2* $= \{\alpha \leftarrow 0.1, \alpha_f \leftarrow 0.05, \epsilon \leftarrow 0.03\}$). In Sub-Figures 4.19(e) and 4.19(f) we observe that the frequency values deteriorate significantly faster for $Config1$. As a result action C has the largest action evaluation value E (Sub-Figure 4.19(c)), which in turn impacts the Q-values (Sub-Figure 4.19(a)).

For the correct runs in the Partially Stochastic Climb Game (Sub-Figure 4.22(c)) the estimate for $Q_{max}(A)$ only stops increasing after 120,000 episodes. For the runs that converged upon sub-optimal joint policies meanwhile (Sub-Figures 4.22(b) and 4.22(d)) we observe that on average the joint-action $\langle B, B, B, B\rangle$ is also underestimated throughout each run. As a result the Q-value estimates for actions A and B are underestimated throughout the incorrect runs (Sub-Figure 4.23(b) and 4.23(d)), while for the correct runs the Q-value estimate for A rises above C after approximately 40,000 iterations. This finding is worrying, given that the likelihood of observing the optimal joint-action in an n-player Climb Game is $1/3^n$. Therefore, to mitigate an exponential increase in the iterations needed to observe the optimal joint-action, independent learners clearly require a more efficient exploration strategy than using a fixed exploration rate.



(a) Climb Game (b) Partially Stochastic Climb Game (c) Penalty Game

Figure 4.20: Correct run percentages for RFMQ in the two-agent, medium-penalty deterministic and partially stochastic Climb Games, and the Penalty Game.

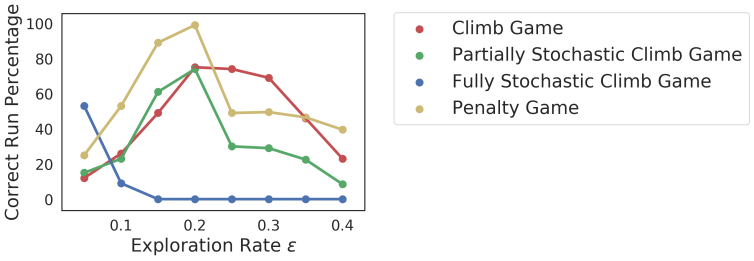
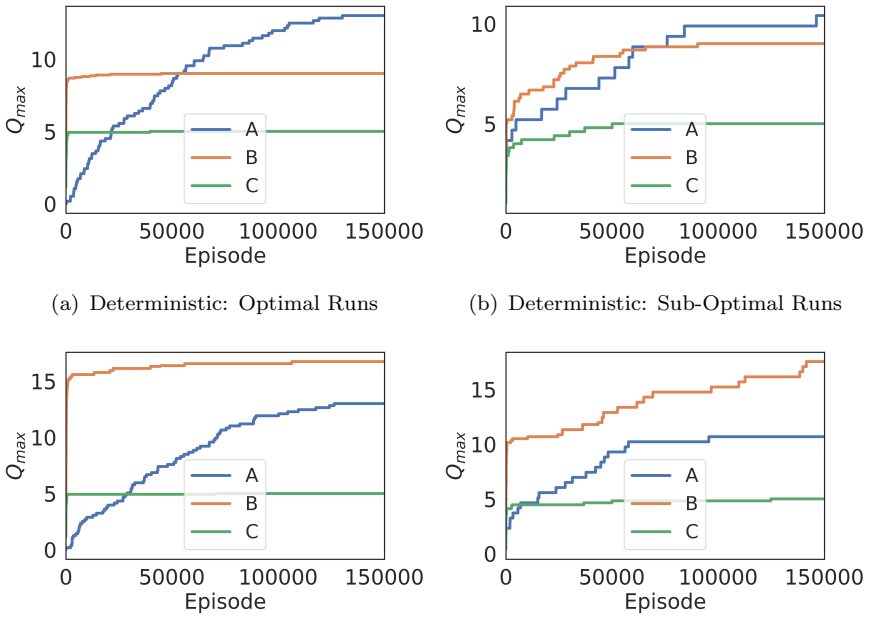


Figure 4.21: Correct run percentages for four-agent, low-penalty variations of each strategic form game, when using $\alpha_f = 0.01$.

4.4.4 Hysteretic Q-learning

The Partially and Fully Stochastic Climb Game results from Wei and Luke's [209] analysis (Table 4.1) serve as a reminder of hysteretic Q-learners vulnerability towards misleading stochastic rewards. However, for the Partially Stochastic Climb Game the 74% convergence rate observed by Wei and Luke [209] can be improved by using Boltzmann exploration. Indeed, seven years prior to Wei and Luke's [209] publication, Matignon et al. [121] achieved a convergence rate of 82%, computing the Boltzmann temperature value using $\tau = 5000e^{-0.003t}$, where t is the current time step, and using learning rates $\alpha = 0.1$ and $\beta = 0.01$, $MinTemp = 2$, $MaxTemp = 40$, and a temperature decay rate of 0.99. Furthermore, whereas the results achieved by Wei and Luke [209] used a different hyperparameter configuration for each strategic-form game (Table 4.3), Matignon et al.'s [121] configuration also achieves 100% convergence upon correct policies in both the (deterministic) Climb Game and the Penalty Game with $p = -100$. Therefore, the authors are able to overcome relative overgeneralization within the Climb Game with a relatively low amount of optimism (in contrast to the $\beta = 0.0001$ used by Wei and Luke [209]). As a result we conduct our hyperparameter sweep using Matignon et al.'s [121] configuration as a guide, using the following hyperparameter combinations:

- $\beta = \{0.01, 0.001, 0.0001\}$;
- $MaxTemp = \{50, 500, 5000\}$;
- $s = \{0.002, 0.003, 0.004, 0.005, 0.006\}$.



(a) Deterministic: Optimal Runs

(b) Deterministic: Sub-Optimal Runs

(c) Partially Stochastic: Optimal Runs

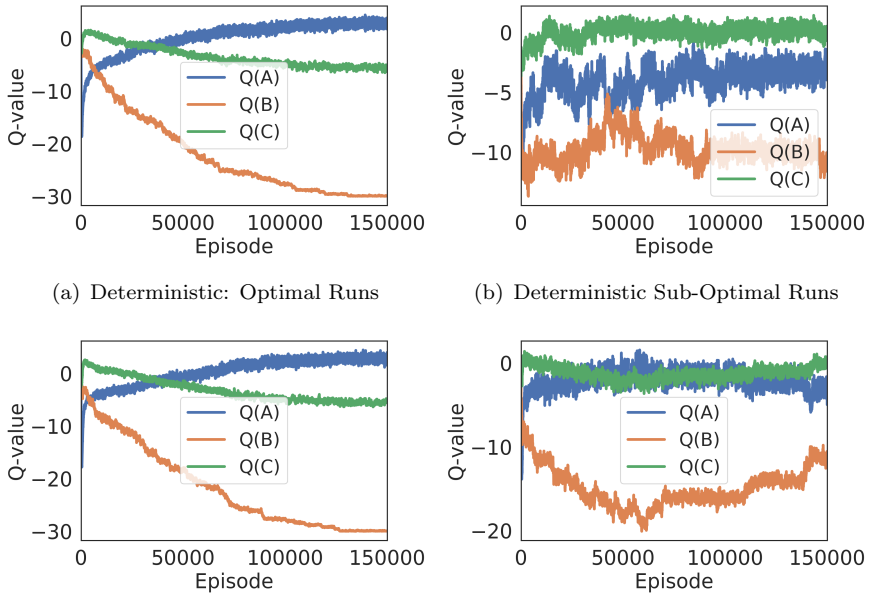
(d) Partially Stochastic: Sub-Optimal Runs

Figure 4.22: RFMQ Four-Agent Deterministic and Partially Stochastic Climb Game average Q_{max} for optimal and sub-optimal runs. We observe that for the sub-optimal runs more steps are required to establish the Q-max for action B.

For the Partially Stochastic Climb Game we identify a number of configurations that improve upon the percentage of correct joint-policies reported by Matignon et al. [122], as illustrated in Figure 4.24. The highest convergence rate that we observe is 87%, obtained by setting $\beta = 0.01$, $s = 0.002$ and $MaxTemp = 500$. However, the plots in Figure 4.24 also illustrate the impact of hysteretic Q-learners using too much optimism, with the majority of runs conducted with $\beta = \{0.001, 0.0001\}$ converging upon sub-optimal joint policies.

Regarding the performance of hysteretic Q-learners in the reaming games, we observe 100% convergence on correct joint-policies within the (deterministic) Climb Game and the Penalty Game for $\beta = 0.01$, $s = 0.002$ and $MaxTemp = 500$ (See Figure 4.25). In the Fully-Stochastic Climb Game meanwhile learners achieve a convergence rate of over 30% for each configuration s when $\beta = 0.01$ and $MaxTemp = 500$, which is an improvement upon the 25.58% reported by Wei and Luke [209] (achieved using $\beta = 0.001$, and ϵ-greedy exploration with a ϵ decay rate of 0.99).

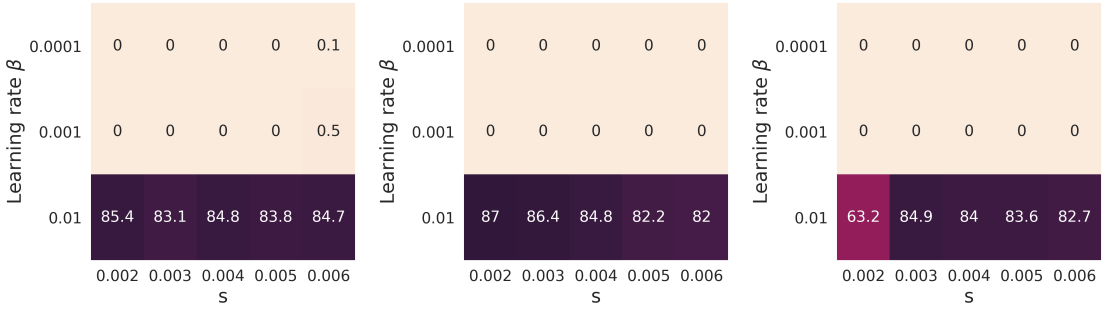
Upon scaling the penalty values we observe that $\beta = 0.01$ provides insufficient optimism for hysteretic Q-learners to prevent relative overgeneralization consistently in the (deterministic) Climb Game. Indeed, each time the penalty value is increased, more optimism is required (in the form of lower values for β) in order to prevent relative overgeneralization (Sub-Figures 4.26(a) and 4.27(a) respectively). Similarly we observe both an increase in the number of runs that converge upon an optimal joint-policy for lower β values in the Partially and Fully Stochastic Climb Games, while observing a reduction with regards to the highest convergence rates achieved. For the Penalty Game



(a) Deterministic: Optimal Runs (b) Deterministic Sub-Optimal Runs

(c) Partially Stochastic: Optimal Runs (d) Partially Stochastic: Sub-Optimal Runs

Figure 4.23: RFMQ Four-Agent Deterministic and Partially Stochastic Climb Game average Q-Values for optimal and sub-optimal runs. We observe that for optimal runs approximately 40,000 iterations are required until $Q(A)$ is larger than $Q(B)$.



(a) $MaxTemp = 50$ (b) $MaxTemp = 500$ (c) $MaxTemp = 5000$

Figure 4.24: Low-penalty two-agent Partially Stochastic Climb Game convergence rates for hysteretic Q-learning using Boltzmann exploration .

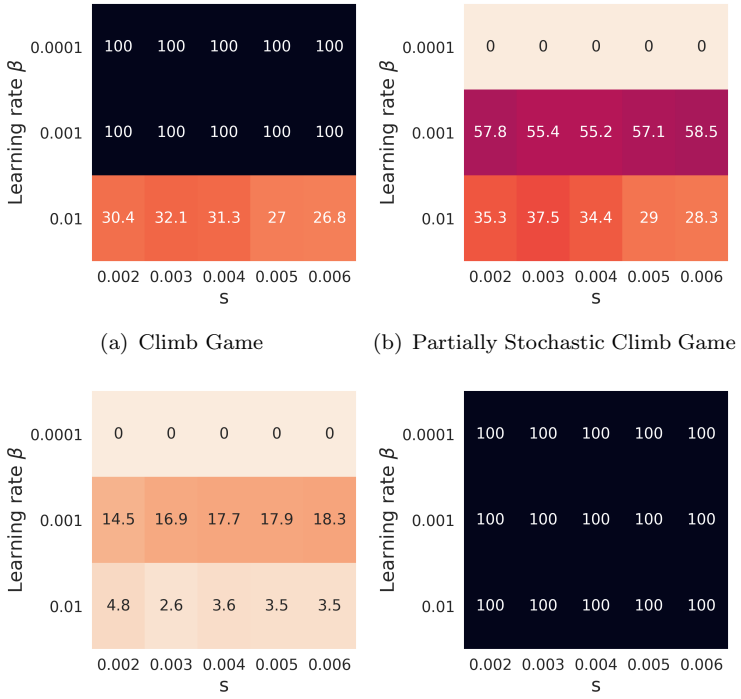
meanwhile we observe a slight lowering of the percentage of optimal joint-policies for $\beta = 0.01$ upon scaling p to -1000 (Sub-Figure 4.27(d)).

Next we consider hysteretic Q-learners scalability with regards to the number of agents. In the four-agent Penalty Game we identify a multitude of configurations that result in a 100% convergence rate during the runs conducted, even when confronted with high penalty values (See Figure 4.28). However, we do observe more consistent convergence across β and decay parameter s settings for the two larger initial temperature values 500 and 5000.

Convergence in the four-agent Climb Game variations is not as consistent (see Appendix A). In Table 4.5 we provide a summary of the best performing configurations. In particular, we observe that in the low and medium reward (deterministic) Climb Game,

(a) Climb Game (b) Fully Stochastic Climb Game (c) Penalty Game

Figure 4.25: Results for the low-penalty two-agent Penalty Game plus the Deterministic and Fully Stochastic Climb Game variations for hysteretic Q-learning using Boltzmann exploration ($MaxTemp = 500$).



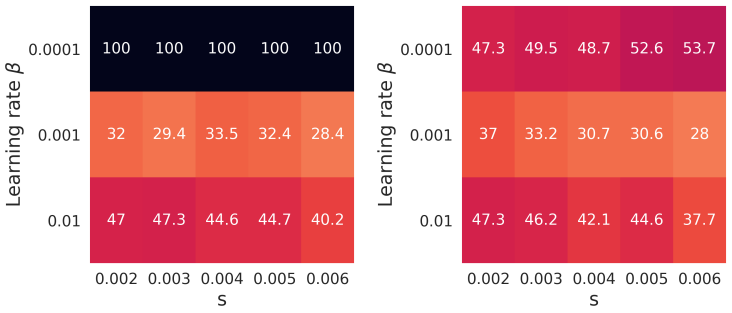
(a) Climb Game (b) Partially Stochastic Climb Game

(c) Fully Stochastic Climb Game (d) Penalty Game

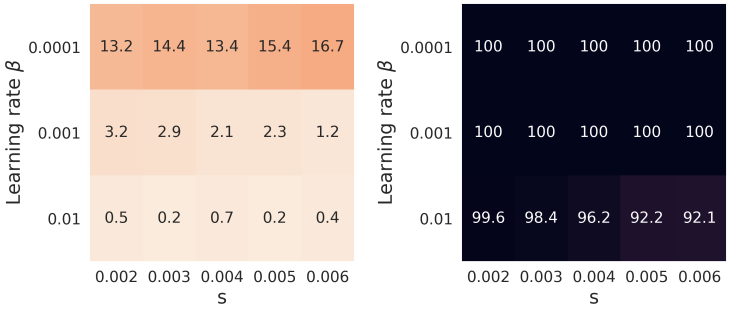
Figure 4.26: Repeated bimatrix game convergence rates for hysteretic Q-learning: medium-penalty and $MaxTemp = 500$.

relative overgeneralization can be prevented through using low settings for learning rate β. For the Partially Stochastic Climb Game the convergence rates for the low, medium and high penalty values decrease to 63.9% and 62.3% and 50.9% respectively, requiring more optimism than in the two-agent setting. Furthermore, as in the two-agent Partially Stochastic Climb Game these percentages decrease significantly upon deviating from the listed hyperparameters.

Therefore the amount of optimism required by hysteretic Q-learners is dependent on the stochasticity within the reward space, the number of agents, and the scale of

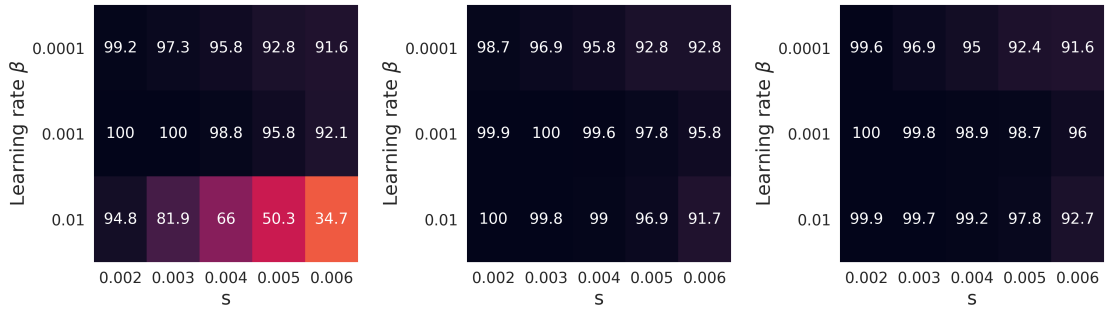


(a) Climb Game (b) Partially Stochastic Climb Game



(c) Fully Stochastic Climb Game (d) Penalty Game

Figure 4.27: Repeated bimatrix game convergence rates for hysteretic Q-learning: high-penalty and $MaxTemp = 500$.



(a) $MaxTemp = 50$ (b) $MaxTemp = 500$ (c) $MaxTemp = 5000$

Figure 4.28: Hysteretic Q-learning convergence rates for the four-agent Penalty Game with high-penalty values.

the penalty values in the presence of the miscoordination and relative overgeneralization pathologies. Despite these challenges we will return to hysteretic Q-learning in the chapters that follow. Given that hysteretic Q-learning is a relatively simple concept, that only introduces one additional hyperparameter, it should not come as a surprise that it was one of the first independent learning approaches to be scaled to multi-agent deep reinforcement learning [141]. However, the findings discussed in this section do illustrates the need for a mechanism capable of adapting the amount of optimism applied to utility value updates by independent learners, which brings us to *leniency*.

	Deterministic				Partially Stochastic				Fully Stochastic			
Penalty	β	s	$MaxTemp$	%	β	s	$MaxTemp$	%	β	s	$MaxTemp$	%
Low	0.0001	0.002	500	100%	0.001	0.003	5000	63.9%	0.0001	0.004	50	33.8%
Medium	0.0001	0.002	500	95.2%	0.0001	0.002	500	62.3%	0.0001	0.006	50	26.5%
High	0.01	0.002	5000	50.3%	0.01	0.002	5000	50.9%	0.0001	0.005	500	13.7%

TABLE 4.5: Summary of the hyperparameters for hysteretic Q-learning that led to the highest correct joint-policy percentages in the four-agent Climb Game variations.

4.4.5 Lenient Multi-Agent Reinforcement Learning

Lenient Multi-Agent Reinforcement Learning 2 (LMRL2) emerged as the most robust independent learning approach from Wei and Luke's [209] analysis. However, we observe that a different leniency moderation factor k was used for each strategic-form game (see Table 4.3). In Figure 4.29 we illustrate how the choice of leniency moderation factor can either impede or accelerate an agent's transition from maximum based to average reward learner (assuming the same action is chosen at each time-step). Upon closer inspection we observe that lenient learners implemented with the hyperparameters chosen by Wei and Luke [209] for the (deterministic) Climb Game, a leniency moderation factor $k = 10^7$ combined with a temperature decay rate of $\nu = 0.995$, are maintaining a maximum-reward learner's disposition for over 4,000 episodes. Therefore, any success achieved using this configuration with deterministic reward functions is unlikely to translate to a stochastic reward space.

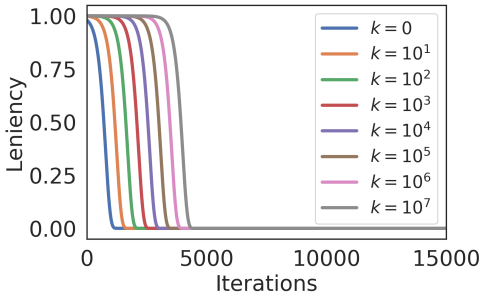


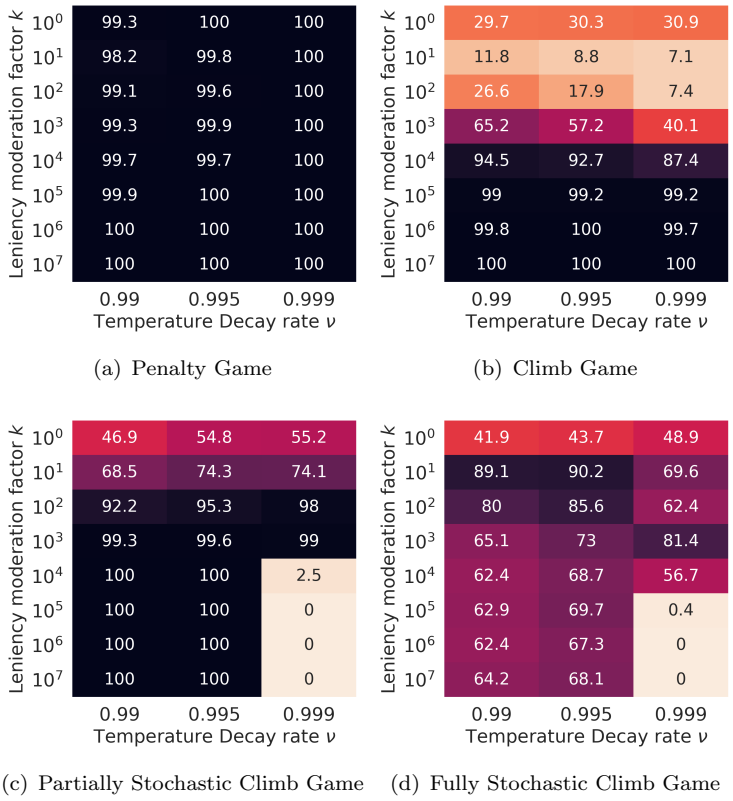
Figure 4.29: An illustration of the impact of the moderation factor k on the leniency function using $MaxTemp = 50$ and a temperature decay rate $\nu = 0.995$.

The above observation raises the question as to what extent we can identify values for the leniency moderation factor k that allow LMRL2 to consistently converge upon the optimal joint-policy across settings. We therefore evaluate the interdependencies between the following leniency moderation factors k and temperature decay rates ν:

- $k = \{10^0, ..., 10^7\}$;
- $\nu = \{0.99, 0.995, 0.999\}$.

Using the same maximum and minimum temperature setting as Wei and Luke [209] ($MaxTemp = 50$, $MinTemp = 2$) we are able to replicate the authors' findings for each

of the tuned hyperparameter configurations. We illustrate the results for our hyperparameter sweep for the two-agent low-penalty version of each game in Figure 4.30.



(a) Penalty Game (b) Climb Game

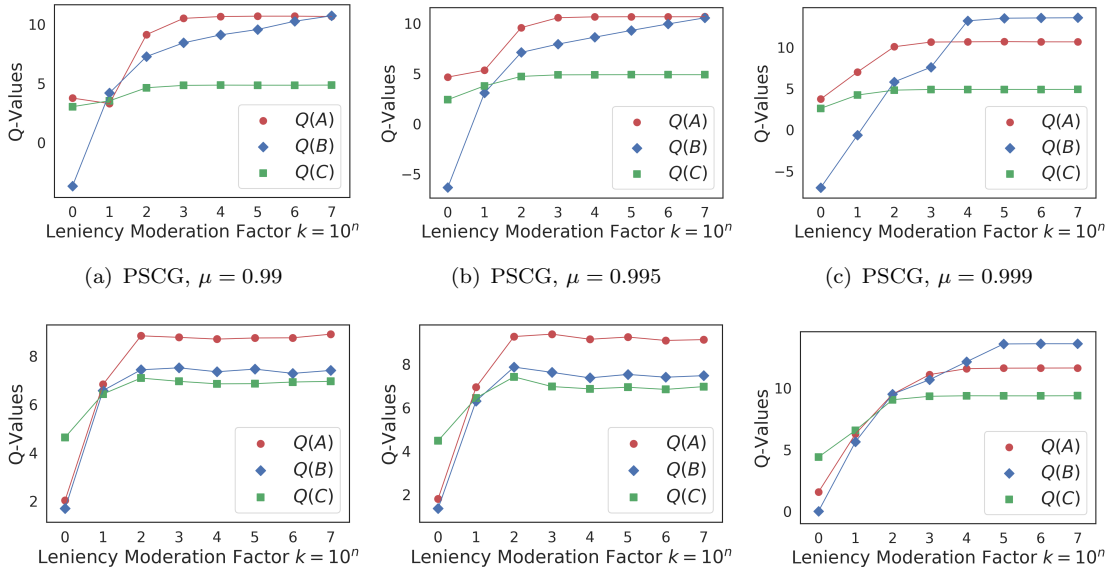
(c) Partially Stochastic Climb Game (d) Fully Stochastic Climb Game

Figure 4.30: LMRL2 convergence rates within the two-agent low-penalty versions of the Climb and Penalty Games.

As with the other algorithms discussed in this chapter, we identify a number of hyperparameter configurations where LMRL2 achieves a 100% convergence rate within the Penalty Game when $p = -10$ (Sub-Figure 4.30(a)). We also verify that LMRL2 achieves the highest convergence rate within the Fully Stochastic Climb Game [209] (Sub-Figure 4.30(d)). However, while we identify a number of configurations that result in 100% convergence upon optimal joint-policies within the Deterministic and Partially Stochastic Climb Games (Sub-Figures 4.30(b) and 4.30(c) respectively), none of these settings overlap with the best performing configurations for the Fully Stochastic Climb Game. While LMRL2 appears capable of consistent convergence using lower leniency moderation factors k and faster decay rates ν, we observe a significant drop in performance upon using $k \leq 10^3$. Meanwhile, LMRL2 is more likely to converge in the Fully Stochastic Climb Game when using lower leniency moderation factors $k = \{10^1, 10^2\}$.

However, for LMRL2 there does exist a compromise hyperparameter configuration that enables convergence with a high-likelihood across low-penalty two-agent settings: $k = 10^5$ and $\nu = 0.995$, where we observe 100% convergence within the Penalty Game and the Partially Stochastic Climb Game, 99% in the Deterministic Climb Game, and 70% in the Fully Stochastic Climb Game. We illustrate the need for this compromise

setting in Figure 4.31, which provides scatter plots illustrating the average Q-values for each action within the Partially and Fully Stochastic Climb Games. We observe that using large leniency moderation factors k combined with a slow temperature decay rate within the two-agent, low reward versions of these games leaves the learners vulnerable towards the very pathology combination that leniency was designed to address: relative-overgeneralization combined with misleading stochastic rewards. Specifically, we observe that LMRL2 agents using large leniency moderation factors $k \geq 4$ combined with a slow temperature decay rate $\mu = 0.999$ overestimate the utility values for the sub-optimal action B (Sub-Figures 4.31(c) and 4.31(f)). In contrast, we find that faster temperature decay rates can compensate for larger leniency moderation factors k, with the average Q-values for the optimal action A being higher than those belonging to actions B and C (Sub-Figures 4.31(a), 4.31(b), 4.31(d) and 4.31(e)).



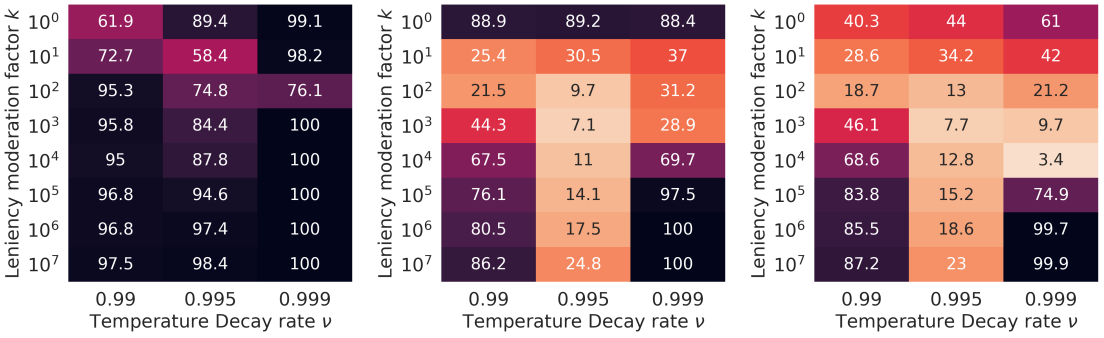
(a) PSCG, $\mu = 0.99$ (b) PSCG, $\mu = 0.995$ (c) PSCG, $\mu = 0.999$

(d) FSCG, $\mu = 0.99$ (e) FSCG, $\mu = 0.995$ (f) FSCG, $\mu = 0.999$

Figure 4.31: Average Q-Values for LMRL2 within the low-penalty two-player Partially and Fully Stochastic Climb Games (PSCG and FSCG respectively) using temperature decay rates $\mu = \{0.99, 0.995, 0.999\}$.

However, the compromise setting identified above does not scale with regards to the number of agents and the size of the penalty values. In the four-agent Penalty Game for instance we observe that as the scale of the penalty increases, LMRL2 is more likely to converge upon the optimal joint-policy using a large leniency moderation factors $k = \{10^6, 10^7\}$ and a slow temperature decay $\nu = 0.999$ (Figure 4.32). Interestingly the learners also achieve high convergence rates using the fastest temperature decay rate $\nu = 0.99$. In contrast learners using the medium temperature decay rate $\nu = 0.995$ increasingly fail to converge with larger penalty values p.

We also observe an interesting anomaly for LMRL2 within the four-agent Fully Stochastic Climb Game. Despite a high penalty the learners converge upon a correct joint-policy in 90.94% of the runs conducted when using $k = 10^7$ and a decay rate of



(a) $p = -10$ (b) $p = -100$ (c) $p = -1000$

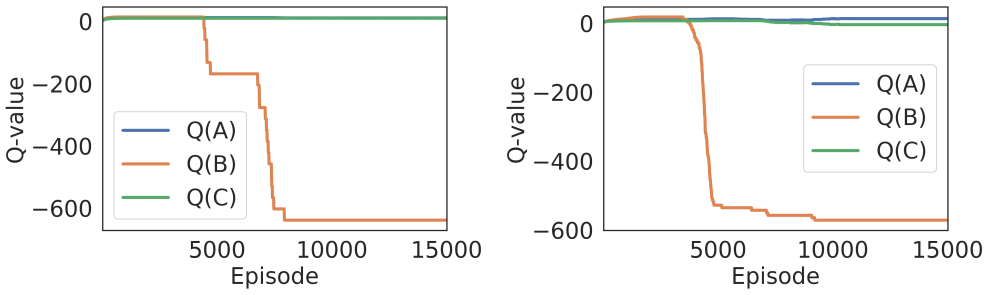
Figure 4.32: Performance of LMRL2 within the four-agent Penalty Game.

0.995. In contrast LMRL2 only converges on 56.8% of runs in the equivalent two-agent setting. A closer look at the Q-values provides answers as to why this is the case (Figure 4.33). In both the two and four agent scenarios the agents initially overestimate the Q-values for the misleading action B. However, upon sufficiently decaying the temperature value for action B we observe a rapid decay in the corresponding Q-value. In both cases the agents subsequently converge upon the optimal action A. However, this also leads to decay in the temperature value for A, thereby making the learners vulnerable towards noise introduced through miscoordination. However, the average reward for the joint-action computed using Equation (4.4) is larger for $\langle A, A, A, A\rangle$ than $\langle A, A\rangle$, as evident from the increased separation between the Q-values for A and C in the four-agent case. Therefore, due to using a modified Boltzmann exploration lenient learners are more likely to maintain the optimal joint-policy within the four-agent reward space compared to the two-agent setting.

In summary, we find that with sufficient hyperparameter tuning LMRL2 can achieve a high convergence rate within the majority of settings used in our evaluation. However, we are unable to identify a single hyperparameter configuration that results in a high convergence rate across settings. Furthermore, in Figure 4.33 we observe the deterioration of Q-values belonging to actions with insufficient leniency. We hypothesize that this deterioration is caused by the moving target problem. This finding is worrying, as the deterioration of Q-values eliminates the possibility of learners returning to a previous equilibrium, should the current one prove sub-optimal. In Chapter 5 we shall consider approaches towards reducing the likelihood of miscoordination occurring as lenient learners transition between equilibria.

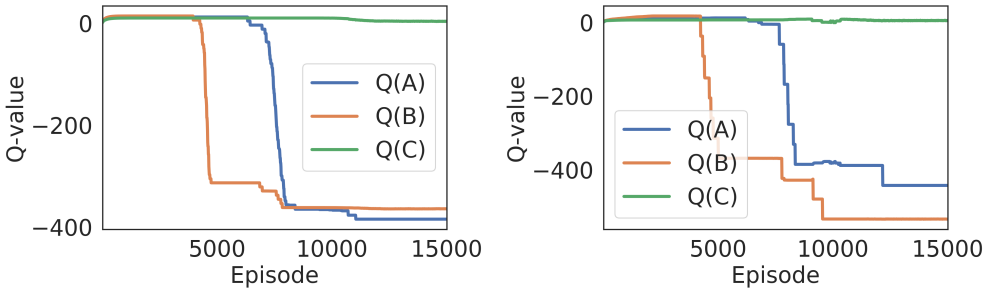
4.5 Summary

In this chapter we evaluate to what extent existing independent learning approaches can mitigate the multi-agent learning pathologies outlined in Section 3.1 within four challenging n-player strategic-form games. More specifically we evaluate decentralized Q-learning [33], hysteretic Q-learning [120], Frequency Maximum Q-value (FMQ) [90],



(a) Two Agent Q-values (Correct Runs)

(b) Four Agent Q-values (Correct Runs)



(c) Two Agent Q-values (Incorrect Runs)

(d) Four Agent Q-values (Incorrect Runs)

Figure 4.33: LMRL2 Q-values within the two and four agent high-penalty Fully Stochastic Climb Game. Due to the reward function we observe a larger separation between Q-values for actions A and C for correct runs in the four agent setting.

Recursive-FMQ [121] and LMRL2 [209] in extended versions of the well studied Climb and Penalty games [33, 90]. Below we provide a brief recap of our findings for each approach:

- **Decentralized Q-learning:** We find that through hyperparameter tuning decentralized Q-learners implemented with Boltzmann exploration can deliver competitive convergence rates within the Penalty Game when the penalty $p = -100$. We identify a number of configurations which enable a convergence upon a correct joint-policy on 99.9% of the 1'000 training runs conducted, exceeding convergence rates reported in previous literature [90, 120], and providing further evidence that baseline methods against which novel approaches are benchmarked can be tuned insufficiently [79, 124]. However, while we do find hyperparameter configurations for the Climb Game variations that exceed those reported from previous evaluations, decentralized Q-learners are unable to prevent relative overgeneralization from occurring for the majority of the runs conducted. Finally, we provide evidence that decentralized Q-learners implemented with low exploration and learning rates can maintain an optimal joint-policy in the Penalty Game when given a supervised start during the first iteration.

- **FMQ:** We find interesting interdependencies between the reward space and the FMQ Boltzmann exploration method. FMQ requires less iterations to converge

within a four-player penalty game when receiving larger penalty values for miscoordination. This is due to the ratio of probabilities produced by Boltzmann exploration depending on the difference between Q-values. Our finding reiterates the difficulties involved with tuning the Boltzmann exploration method [**?**]. However, while FMQ can overcome miscoordination in the six variations of the Penalty Game used during our evaluation, the approach does not scale well with regards to the number of agents and the size of penalty values within the Climb Game variations.

- **RFMQ:** For RFMQ we are unable to replicate the results reported by Wei and Luke [209]. We achieve a significantly lower convergence rate within the Full Stochastic Climb Game, due to a large frequency learning rate α_f leaving the learners vulnerable towards miscoordination. We do, however, achieve convergence rates in-line with those reported by Matignon et al. [121] when using a lower frequency learning rate α_f, and identify a number of hyperparameter configurations which improve upon the authors' results. Finally, RFMQ attempts to mitigate the alter-exploration problem via choosing a constant low exploration rate ϵ. However, we find that for the n-player Climb Game this approach requires an exponential increase in the number of iterations until the optimal reward is observed as the number of learners is increased. Nevertheless, we shall come back to Matignon et al.'s [121] considerations regarding limiting miscoordination in Chapter 5, where we propose a number of extensions for improving lenient learners.

- **Hysteretic Q-learning:** In contrast to Wei and Luke [209], and in-line with Matignon et al. [120], we find that hysteretic Q-learning can benefit from using Boltzmann exploration strategy. We furthermore identify hyperparameter configurations that improve on the previously reported benchmarks for hysteretic Q-learning within the Partially Stochastic Climb Game [120, 209]. However, we are unable to identify a configuration that enables a high convergence rate within the Fully Stochastic Climb Game.

- **LMRL2:** We find that out of the approaches evaluated LMRL2 is capable of delivering the highest convergence rates across domain configurations. However, hyperparameter configurations that perform well in the deterministic and partially stochastic Climb Games lead to a poor convergence rate within the Fully Stochastic Climb Game. Furthermore, we observe that the Q-values belonging to actions with temperature values that have been cooled down are vulnerable towards the alter-exploration and moving target problems, especially during periods where the agents are transitioning between equilibria.

Therefore, we find that LMRL2 remains the most robust out of the independent learning methods evaluated, even within the extended variations of the four strategic-form games used by Wei and Luke [209]. However, we observe that LMRL2 is vulnerable towards the moving-target and alter-exploration problem following the cooling of

temperature values, leading to the deterioration of Q-values. Furthermore, LMRL2 requires a significant amount of hyperparameter tuning, and considerations regarding the impact of temperature values on the Boltzmann exploration. In Chapter 5 we shall consider approaches toward mitigating the consequent destruction of Q-values, and reducing the amount of hyperparameter tuning required, by introducing *Distributed-Lenient Q-learning*.

Chapter 5

Towards Improved Lenient Learners

The work presented in this chapter is in preparation for a submission to the *Journal of Machine Learning Research*.

In the previous chapter we find that to enable a high convergence rate upon correct (optimal) joint-policies within repeated n-player strategic-form games, Lenient Multi-Agent Reinforcement Learning 2 (LMRL2) [209] requires a significant amount of domain specific hyperparameter tuning. We observe that tuning the decay rate of temperature values that are used for both lenient Q-value updates and Boltzmann exploration, while minimizing the impact of the alter-exploration problem, is far from trivial. Furthermore, we hypothesize that for LMRL2 the moving target problem is aggravated by asynchronous lenient Q-value updates, resulting in lenient learners switching between equilibria during different time-steps. This leaves lenient learners vulnerable towards miscoordination, resulting in the destruction of utility values belonging to actions with insufficient leniency. We hypothesize that lenient learners can benefit from returning to a previous policy, should the new join-action(s) also prove misleading. In this chapter we introduce *Distributed-Lenient Q-learning* (DLQ), a novel leniency algorithm designed to mitigate the moving-target and alter-exploration problems.

Our contributions can be summarized as follows:
1) We introduce Distributed-Lenient Q-learning (DLQ), a novel variation of leniency, which splits learning into two distinct phases: (i) An initial maximum reward learner phase with a uniform action selection policy, during which peaks in the reward space are discovered; (ii) A greedy action selection phase combined with lenient Q-value updates. Using a greedy action selection policy during the later training phase mitigates the alter-exploration problem.
2) We introduce *synchronized leniency updates*, designed to increase the likelihood of lenient learners switching between equilibria during the same time-steps.
3) We empirically show that synchronized leniency updates can reduce the likelihood of

miscoordination during phases where lenient learners frequently switch between equilibria. Our empirical evaluation finds that Synchronized DLQ can deliver state of the art convergence rates in the repeated n-player strategic-form games used for our empirical evaluation of existing approaches in Chapter 4.
4) We scale DLQ to Markov games, and show that the scaled variation can deliver state of the art performances in two challenging domains proposed by Wei and Luke [209]: the *Gradient 2* and *Relative Overgeneralization 3* games.

5.1 Algorithmic Definition

In Chapter 4 we provide an extensive evaluation of LMRL2 and other independent learning approaches. We observe that the following weaknesses of LMRL2 must be addressed to further reduce the likelihood of relative overgeneralization occurring in domains with a stochastic reward space:

1. Decaying the leniency related temperature values leaves LMRL2 vulnerable towards the alter-exploration problem. As a result utility value estimates for an optimal action can be significantly underestimated following only a few iterations of agents performing sub-optimal joint-actions.

2. Even if lenient learners utilize a greedy action selection strategy to mitigate the alter exploration problem, lenient utility value updates that result in lowering a utility estimate are conducted asynchronously with a probability $z \in [0, 1]$ (See Equation (3.19)). Therefore we have no guarantee that agents will establish that a joint-action is suboptimal during the same time-step t. Using the two-agent Fully Stochastic Climb Game as an example: we shall assume that at time step t for both agents i: $\forall_i Q_i(B) > Q_i(A)$. However, if *agent 1* estimates that $Q_1(A) > Q_1(B)$ following the subsequent Q-value update, then there is no guarantee that this will also be the case for *agent 2*. This can result in a number of iterations where the joint-action is $\langle A, B\rangle$. For actions with insufficient leniency the utility value estimates will subsequently be destroyed.

DLQ addresses both of these weaknesses. The intuition behind our approach is as follows: upon conducting an extensive random exploration phase (*learning phase 1*) to establish the maximum utility for each action, we employ a greedy exploration method to address the alter-exploration problem (*learning phase 2*). Secondly, we introduce *synchronized leniency updates* to enable the learners to switch between equilibria during the same time-step, where the random variable $z \in [0, 1]$ from Equation (3.19) used to determine if updates using a negative δ should be performed is identical for each agent. Therefore, our approach reduces stochasticity with regards to exploration and the lowering of Q-values.

Temperature values $\mathcal{T}$ are not decayed until the random exploration phase is complete, thereby preventing global-exploration from impacting the utility values. Therefore, DLQ learners must discover the maximum reward for each action during random exploration. To maximize the likelihood of the agents observing each joint-action combination, DLQ uses a uniform action selection strategy during the random exploration phase. Furthermore, a learning rate $\alpha = 1.0$ ensures that Q-value estimates are set to the maximum reward encountered. During this random exploration phase the learners are therefore maximum reward learners. Furthermore, to overcome miscoordination we adopt the policy table from distributed Q-learning [100]. Policy table updates only occur when the current argmax action no longer has the highest utility value (See Section 3.2.2). Therefore DLQ is equal to distributed Q-learning with $\epsilon = 1.0$ during *learning phase 1*.

Following a predefined number of exploratory steps, DLQ switches to an argmax action selection method using the policy table. During this phase the temperature values are decayed and utilized by the leniency function to determine the frequency with which utility values are lowered. However, the leniency function can be synchronised, with each learner using a random value function with an identical seed value. Furthermore, during the second learning phase Q-value updates are performed with a more cautious learning rate α [1].

We note that in stochastic domains a sub-optimal joint-action can yield an identical maximum reward as an optimal joint-action. We shall encounter two domains with this property in Section 5.4. As a consequence the random exploration phase can yield a sub-optimal joint policy consisting of an optimal action being paired with a sub-optimal action. However, if a utility value update subsequently requires a change to the policy table for the agent using the optimal action, then, due to the update only considering $max_{a \in \mathcal{A}} Q(a)$, this action will no longer be considered, thereby reducing the likelihood of convergence upon the optimal joint-policy. To increase the likelihood of the learners selecting optimal joint-actions we therefore restrict the policy updates to actions with the highest utility estimates within a boundary ε. We define Synchronized DLQ in Algorithm 4.

5.2 Strategic-Form Game Evaluation

We compare the performance of Synchronized DLQ (SDLQ) against Asynchronized DLQ (ADLQ) for which the synchronized leniency updates are disabled. Therefore, for each agent the random value z, which determines whether a negative update using $\delta < 0$ takes place (See Equation (3.19) in Chapter 3), is drawn from random number generators using distinct seed values. During preliminary trials SDLQ proved robust in each of the repeated n-player strategic-form games used for the empirical evaluation of existing approaches in Chapter 4, irrespective of the choice of leniency moderation factor k and

[1] From this point forward α refers to the learning rate used during *learning phase 2*.

Algorithm 4 Synchronized DLQ for Strategic-Form Games

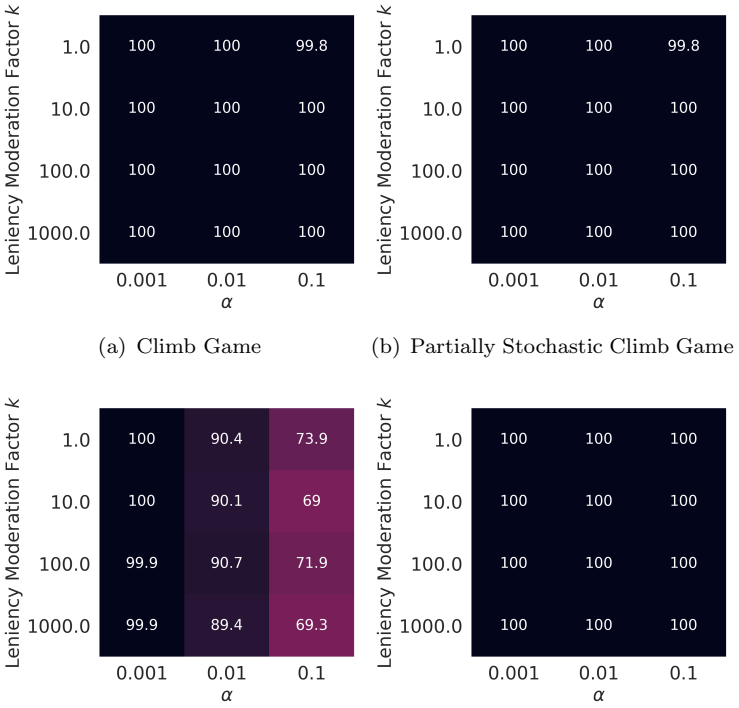
Input: Max steps T, $MaxTemp$, learning rate α, leniency moderation factor k, temperature decay rate ν, $ExplorationSteps$
for all $a \in \mathcal{A}$ **do**
    $Q(a) \leftarrow initialize(a)$, $\mathcal{T}(a) \leftarrow MaxTemp$, $\pi(a)$ arbitrarily
for $t = 0$ to T **do**
    if $t < ExplorationSteps$ **then**
        Choose a using a uniform probability distribution
        Execute action a and observe r
        $\delta \leftarrow r - Q(a)$
        if $\delta > 0$ **then**
            $Q(a) \leftarrow Q(a) + \delta$
    else
        Choose $\text{argmax}_{a \in \mathcal{A}} \pi(a)$
        Execute action a and observe r
        $\delta \leftarrow r - Q(a)$
        $L(a_t) = \exp\left(\frac{-1}{k\mathcal{T}_t(a_t)}\right)$
        $Q(a) \leftarrow \begin{cases} Q(a) + \alpha\delta, & \text{if } \delta \geq 0 \text{ or synchronized } z > L(a) \\ Q(a), & \text{otherwise} \end{cases}$
        $\mathcal{T}(a) \leftarrow \nu\mathcal{T}(a)$
    if $|Q(\text{argmax}_{o \in \mathcal{A}} \pi(o)) - \max_{o \in \mathcal{A}} Q(o)| > \varepsilon$ **then**
        Select a random action $a_{max} \in \text{argmax}_{o \in \mathcal{A}} Q(o)$ within bounds ε
        $\forall b \in \mathcal{A}\ \pi(b) \leftarrow \begin{cases} 1, & \text{if } b = a_{max} \\ 0, & \text{otherwise} \end{cases}$

the learning rate α (as we shall discuss below). In contrast, the success of ADLQ depends on the leniency moderation factor k and the learning rate α. Therefore, we evaluate DLQ using the following hyperparameter configurations with a temperature decay rate $\nu = 0.995$ and 500/1000 exploration steps for two and four agent experiments respectively:

- $k = \{10^0, ..., 10^3\}$;
- $\alpha = \{0.1, 0.01, 0.001\}$.

As in previous experiments we gather 1,000 training runs for each hyperparameter configuration. We illustrate the ADLQ convergence rates for each game in Figure 5.1. In Section 4.4.5 we observed that even in the low-penalty bimatrix game version of the Fully-Stochastic Climb Game, LMRL2 converges upon an optimal joint policy on 90.2% of the runs conducted. In contrast, even with the synchronized property disabled, ADLQ achieves a 100% convergence rate for a number of settings within the two-agent low penalty Fully Stochastic Climb Game. However, this result only occurs for the lowest learning rate $\alpha = 0.001$ (See Sub-Figure 5.1(c)). Furthermore, while we observe a 100% convergence rate for ADLQ in each variation of the remaining games, only SDLQ

is able to overcome scaled penalty values and an increase in the number of agents in the Fully Stochastic Climb Game [2]. In both the two and four-agent variation of the Fully Stochastic Climb Game we observe a further interesting interdependency between the learning rate α and the scale of the penalty values for ADLQ. For medium penalty values learners using $\alpha = 0.01$ achieve the highest convergence rates across each of the leniency moderation factor k settings. However, upon increasing the penalty values the learners prefer $\alpha = 0.001$ (see Figure 5.2).



(a) Climb Game (b) Partially Stochastic Climb Game

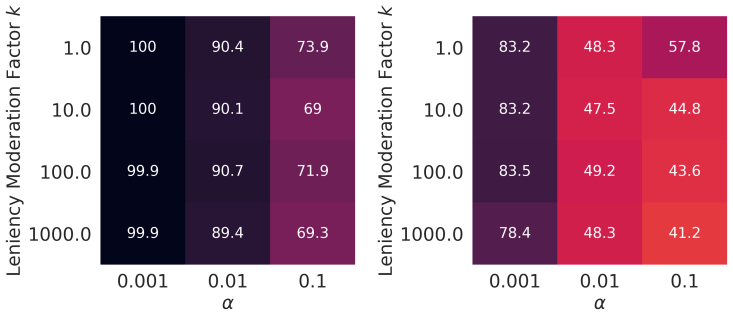
(c) Fully Stochastic Climb Game (d) The Penalty Game

Figure 5.1: Convergence rates from ADLQ runs within the two-player low-penalty Climb and Penalty Games.

In contrast, we observe no significant difference between the results gathered for SDLQ using $\alpha = 0.01$ and $\alpha = 0.1$ across leniency moderation factors k. SDLQ converged on the correct policy on 100% of the runs gathered within each version of the (deterministic) Climb Game, Partially Stochastic Climb Games and the Penalty Game, irrespective of the scale of the penalty values and the number of agents. Within the Fully Stochastic Climb Game meanwhile SDLQ hyperparameter configurations exist for each domain configuration that result in a 100% convergence rate, with the worst performing configurations being 99.9%.

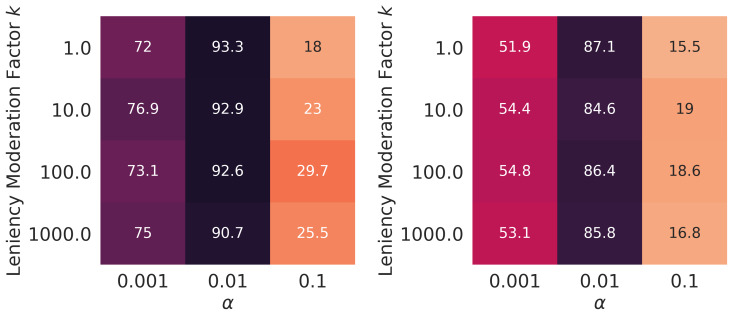
To evaluate the extent to which using synchronized updates help when using a larger learning rate $\alpha = 0.01$, we gather an additional 100 runs for SDLQ and ADLQ within the two-agent low-penalty Fully Stochastic Climb Game. We evaluate the actions performed, Q-values and leniency temperature values $\mathcal{T}$ for every episode during each training run.

[2] See heat-maps in Sections A.6 and A.5.



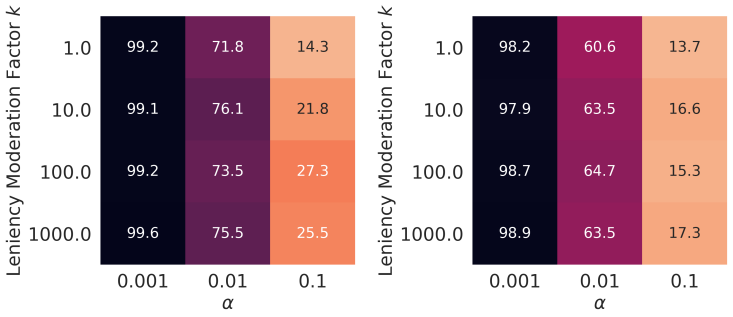
(a) Penalty: Low, Agents: 2

(b) Penalty: Low, Agents: 4



(c) Penalty: Medium, Agents: 2

(d) Penalty: Medium, Agents: 4



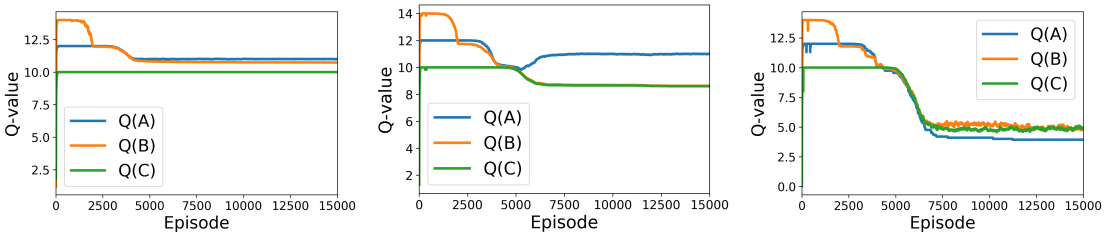
(e) Penalty: High, Agents: 2

(f) Penalty: High, Agents: 4

Figure 5.2: Heat-maps illustrating the convergence rates of ADLQ within the six Fully Stochastic Climb Game variations.

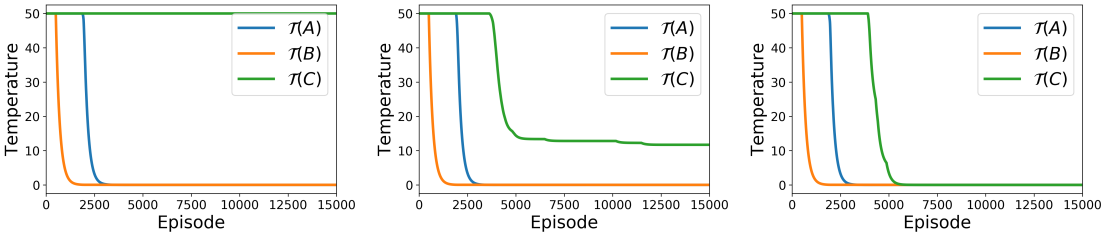
We observe interesting differences between the Q-values learned by each algorithm in Figure 5.3. Sub-Figure 5.3(a) illustrates the Q-values resulting from synchronous updates, while Sub-Figures 5.3(b) and 5.3(c) illustrate the Q-values for ADLQ runs that resulted in optimal and suboptimal policies respectively. For both approaches we observe that the initial utility value estimates for each action are based on the respective R_{max}. For SDLQ (Sub-Figure 5.3(a)) we observe that following the random exploration phase (500 iterations) the utility values estimate for action B decreases rapidly until being on par with the utility estimate for action A. From around episodes 2000 to 5000 the Q-values for A and B are closely aligned, with agents switching between the two equilibria as the temperature values are decayed. Around the 4000 episode mark action A emerges with a slightly higher average utility estimate. At this point the Q-value for

action B is no longer modified, as the learners have no reason to deviate from action A. The Q-value and temperature value (Figure 5.4) for action C meanwhile remain unchanged following the exploration phase.



(a) SDLQ (b) ADLQ: Optimal Runs (c) ADLQ: Sub-Optimal Runs

Figure 5.3: Average Q-value comparison for the two-player low-penalty Fully Stochastic Climb Game. For SDLQ and ADLQ 100 runs were gathered. For ADLQ we separate optimal and sub-optimal runs prior to plotting the Q-values. We observe that for SDLQ $Q(A)$ and $Q(B)$ are closely aligned between episodes 2000 and 5000, before action A emerges with the highest Q-value. For ADLQ the Q-value for action B is vulnerable towards instances of miscoordination. Furthermore, while $Q(C)$ remains unchanged for SDLQ following *learning phase 1*, this is not the case for ADLQ.

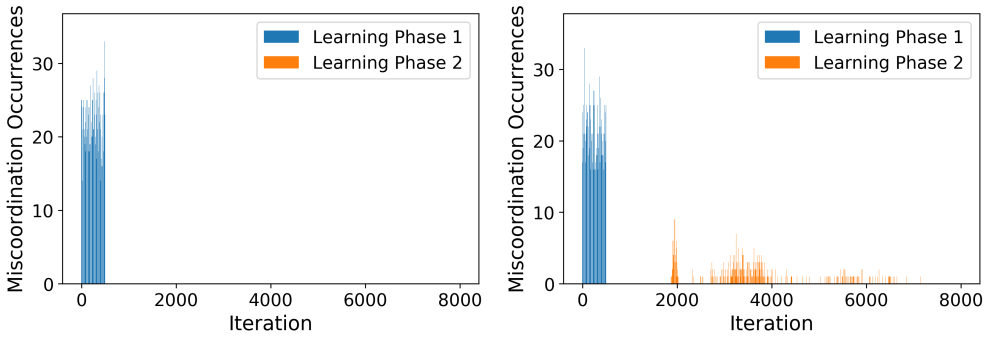


(a) SDLQ (b) ADLQ: Optimal Runs (c) ADLQ: Sub-Optimal Runs

Figure 5.4: Average leniency temperature value comparison within the two-player low-penalty Fully Stochastic Climb Game. For both SDLQ and ADLQ 100 runs were gathered. For ADLQ we separate optimal and sub-optimal runs. We observe that the temperature value for action C remains unchanged for SDLQ. For optimal ADLQ runs meanwhile $\mathcal{T}(C)$ approaches 0 for sub-optimal runs, while also being decayed during optimal runs.

In contrast, even during successful ADLQ runs, the Q-values for actions A and B are not as closely aligned. We hypothesize that this is due to miscoordination transitions where one of the agents changes policy before the other. For ADLQ runs we observe that 0.8525% of transitions result in miscoordination between iterations 2000 and 5000. In contrast, for SDLQ we do not observe any miscoordination. The barcode plots in Figure 5.5 illustrate instances of the joint-actions $\langle A, B\rangle$ and $\langle B, A\rangle$ over 100 training runs for SDLQ and ADLQ within the two agent low-penalty Fully Stochastic Climb Game. We observe that while for SDLQ miscoordination only occurs during the 500 exploration iterations (*learning phase 1*), agents using ADLQ often receive penalty values between iterations 2,000 and 6,000. Furthermore, ADLQ overshoots the average reward for the joint-action $\langle A, A\rangle$, reaching an average utility estimate of 10.0. This explains why

ADLQ benefits from using lower learning rates. Action A emerges as the optimal action on 90% of runs (Sub-Figures 5.3(b)), while being significantly underestimated in the remaining 10% (Sub-Figures 5.3(c)).



(a) Synchronized-DLQ (b) Aynchronized-DLQ

Figure 5.5: Histograms illustrating the number of occurrences of joint-actions $\langle A, B\rangle$ and $\langle B, A\rangle$ during 100 SDLQ and ADLQ training runs conducted in the low-penalty Fully Stochastic Climb Game. For both approaches miscoordination occurs during *Learning Phase 1* (the initial 500 iterations) where random exploration is combined with maximum reward updates. During *Learning Phase 2* we initially observe a reduction in miscoordination due to learners overestimating the utility of action B (see Figure 5.3). However, upon learners applying less leniency towards updates involving B we observe frequent miscoordination for ADLQ. For SDLQ meanwhile we observe no miscoordination occurrences during *Learning Phase 2*.

The results for ADLQ can be improved by lowering the learning rate for each agent to $\alpha = 0.001$. However, we still observe a significant amount of miscoordination, as the time-series plot depicting the average rewards for runs conducted with SDLQ ($\alpha = 0.1$) and ADLQ ($\alpha = 0.001$) in Figure 5.6 illustrates. Furthermore, we observe that due to being able to use a larger learning rate, SDLQ requires fewer iterations to converge on a joint-policy with an optimal average reward (while achieving the same convergence rate).

5.3 Learning Complete Policies in Markov Games

The stateless version of DLQ discussed in the previous section is limited to n-player strategic-form games. In this section we extend DLQ to Markov games with $|\mathcal{X}| > 1$, where independent learners may encounter an additional pathology: *deception*. LMRL2 addresses deception by using *Average Temperature Folding* to prevent a premature temperature decay for actions belonging to frequently visited early transitions in episodic tasks (see Section 3.2.6). Wei and Luke [209] hypothesize that average temperature folding allows the learners to remain lenient long enough to allow the average rewards from follow-on states to be back-propagated. Given that LMRL2 also uses the temperature value to control the exploration-exploitation trade-off, average temperature folding should result in the learners remaining exploratory in early states while the optimal joint-policy in each of the follow-on states is established. Indeed, enabling learners to

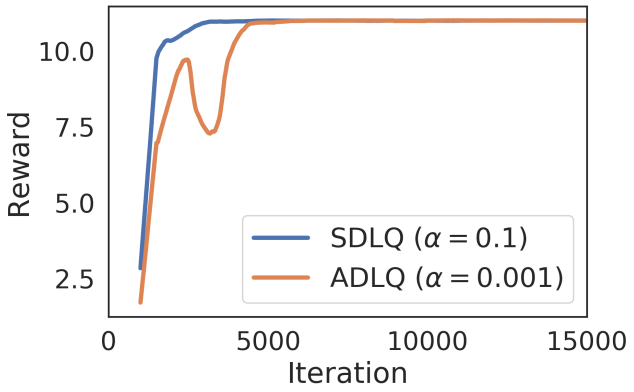


Figure 5.6: Running average reward comparison (window=1000 iterations) for the two-player, low-penalty Fully Stochastic Climb Game. We compare SDLQ with $\alpha = 0.1$ against ADLQ with $\alpha = 0.001$. We observe that due to ADLQ using the lower learning rate there is a significant delay in convergence. Furthermore, we observe a dip in the average reward due to miscoordination frequently occurring between iterations 2000 and 5000.

converge on policies that behave correctly within any state $x \in \mathcal{X}$ of a repeated Markov game is one of the goals of independent learning, offering advantages such as being deployable from any given initial state within the environment. As a result Wei and Luke [209] introduce two measures of successful convergence within Markov games:

1. **Correct Policies:** Agents perform optimally when following the learned policy from the initial state onwards;
2. **Complete Policies:** Agents behave optimally within all states $x \in \mathcal{X}$.

Therefore, while every complete policy is also a correct policy, a correct policy that behaves optimally when starting from an initial state x, may behave incorrectly if an alternative state x' is designated as the initial state. It is worth noting that for Markov games with stochastic transitions every correct policy is also a complete policy.

To enable independent learners to converge upon a complete joint-policy, sufficient exploration is necessary to allow the learners to discover the optimal action for each state $x \in \mathcal{X}$. In this section we shall observe that even domains with a relatively small state-action space can require a considerable amount of exploration, in order for independent learners to consistently converge upon complete policies. However, a misguided random exploration strategy is likely to result in frequent miscoordination. In contrast, for repeated Markov games with a significant amount of branching in the state-action space, a more robust approach towards exploration is to initially remain exploratory in early states, while acting greedy in follow-on states that lead to absorbing (terminal) states [209]. Acting greedy in follow-on states allows the learners to limit the noise introduced by miscoordination. Meanwhile, by remaining exploratory in earlier states the learners are more likely to explore different paths in the state-space. Using a staggered approach the learners can increasingly apply less exploration from the penultimate states backwards, until finally acting greedy in the initial state(s).

LMRL2 uses the average temperature values within each state to enable a staggered exploration of the state-space [209]. However, in the previous chapter we observed that tuning the leniency related hyperparameters to enable the temperature values to correctly balance the exploration-exploitation trade-off is non-trivial. Furthermore, if we consider the temperature value for action C in Sub-Figure 5.4(b), then we observe that remaining exploratory to sufficiently decay each of the actions' values may introduce additional noise through increased global exploration. However, we consider that a state's temperature values can provide valuable information regarding whether the agents have converged within a given state. Instead of using the average temperature for guidance, we can use the discrete derivative obtained during the temperature updates $\mathcal{T}(x,u) \leftarrow \nu\mathcal{T}(x,u)$ to estimate if a learner's policy has converged for a state x. More formally, we assume that learners have converged in a state x if the exponentially weighted moving average of the discrete derivate $d \leftarrow \mathcal{T}(x,u)_t - \mathcal{T}(x,u)_{t+1}$ following a temperature value update,

$$\overline{d}(x) = (1.0 - \tau) \times \overline{d}(x) + \tau \times d, \tag{5.1}$$

is below a threshold ϱ. We assume that if $\overline{d}(x) < \varrho$, then the learner has either:

i Decayed the temperature values for all actions to the point where the agent has become an average reward learner;

ii Identified an optimal action, meaning the temperature values for all other actions are no longer being decayed.

Therefore, *iff* $\overline{d}(x) < \varrho$, then we assume that the learner has converged in state x. We use a vector $C(x) = \{1, 0\}$ for each $x \in \mathcal{X}$ to indicate whether a state x has converged, where $C(x) \leftarrow 1$ if $\overline{d}(x) < \varrho$. Therefore, while the stateless version of DLQ switches from *learning phase 1* to *phase 2* after a specified number of iterations, in the full version of DLQ we use $C(x')$ from each follow-on state x' to control this transition. However, as $|\mathcal{X}|$ increases so will the memory requirements for each state x maintaining the respective set of follow-on states x'. To reduce the memory requirements we resort to bootstrapping:

$$\overline{C}(x) = (1.0 - \tau) \times \overline{C}(x) + \tau \times C(x_{t+1}) \tag{5.2}$$

We therefore switch from explore to exploit, and max reward to lenient learning, if $\overline{C}(x) > 1.0 - \varepsilon$, for some small $\varepsilon > 0$. We set $C(x) \leftarrow 1$ for absorbing (terminal) states by default. Furthermore, we keep the n exploration steps, allowing agents to establish the max rewards for non-absorbing states. Therefore, DLQ learners continue to use a uniform action selection strategy combined with maximum reward learner updates in frequently visited early transitions until the agents have converged upon a joint-policy in the direct follow-on states. This process begins in the penultimate states where actions result in a transition into an absorbing state. However, we consider that the number of steps required to reach convergence in each of the follow-on states will

Algorithm 5 Synchronized DLQ for Markov Games

Input: Max steps T, $MaxTemp$, learning rate α, leniency moderation factor k, temperature decay rate ν, $ExplorationSteps$, temperature threshold ϱ, τ.
for all $x \in \mathcal{X}$ and $u \in \mathcal{U}$ **do**
    $Q(x,u) \leftarrow initialize(x,u)$, $\mathcal{T}(x,u) \leftarrow MaxTemp$, $\pi(x,u)$ arbitrarily.
$\overline{d}(x) \leftarrow 0$, $\overline{C}(x) \leftarrow 0$.
For all non-absorbing states $x \in \mathcal{X}$: $C(x) \leftarrow 0$.
For all absorbing states $x \in \mathcal{X}$: $C(x) \leftarrow 1.0$.
$x \leftarrow$ initial state
for $t = 0$ to T **do**
    if $t < ExplorationSteps$ or $\overline{C}(x_t) < 1.0 - \varepsilon$ **then**
        Choose u_t using a uniform probability distribution
        Execute action u_t and observe r and x_{t+1}
        $\delta = r + \gamma \max_{u \in \mathcal{U}} Q(x_{t+1}, u) - Q(x_t, u_t)$
        if $\delta > 0$ **then**
            $Q(x_t, u_t) \leftarrow Q(x_t, u_t) + \delta$
    else
        Choose $\operatorname{argmax}_{u \in \mathcal{U}} \pi(x_t, u)$
        Execute action u_t and observe r and x_{t+1}
        $\delta = r + \gamma \max_{u \in \mathcal{U}} Q(x_{t+1}, u) - Q(x_t, u_t)$
        $L(x_t, u_t) = \exp\left(\dfrac{-1}{k\mathcal{T}_t(x_t, u_t)}\right)$
        $Q(x_t, u_t) \leftarrow \begin{cases} Q(x_t, u_t) + \alpha\delta, & \text{if } \delta \geq 0 \text{ or synchronized } z > L(x_t, u_t) \\ Q(x_t, u_t), & \text{otherwise} \end{cases}$
        $d \leftarrow \mathcal{T}(x_t, u_t) - \nu\mathcal{T}(x_t, u_t)$
        $\mathcal{T}(x_t, u_t) \leftarrow \nu\mathcal{T}(x_t, u_t)$
        $\overline{d}(x_t) = (1.0 - \tau) \times \overline{d}(x_t) + \tau \times d$
        if $\overline{d}(x_t) < \varrho$ **then**
            $C(x_{t+1}) \leftarrow 1.0$
        $\overline{C}(x_t) = (1.0 - \tau) \times \overline{C}(x_t) + \tau \times C(x_{t+1})$
    if $|Q(x, \operatorname{argmax}_{o \in \mathcal{U}} \pi(o)) - \max_{o \in \mathcal{U}} Q(x_t, o)| > \varepsilon$ **then**
        Select a random action $u_{max} \in \operatorname{argmax}_{o \in \mathcal{U}} Q(x_t, o)$ within bounds ε
        $\forall b \in \mathcal{U}\ \pi(x_t, b) \leftarrow \begin{cases} 1, & \text{if } b = u_{max}. \\ 0, & \text{otherwise.} \end{cases}$

increase exponentially with the size of the state space $|\mathcal{X}|$. Therefore a compromise has to be made with ϱ as the size of the state space increases, where sufficiently decaying the temperature value for all follow-on states is infeasible. However, as we shall see in the next section, the full version of DLQ delivers state-of-the art performances in two challenging Markov games from Wei and Luke's [209] empirical evaluation. We outline the full DLQ algorithm (using synchronized updates) in Algorithm 5.

5.4 Addressing Deception in Markov Games

We evaluate the full synchronized and asynchronized versions of DLQ using two Markov games from Wei and Luke's [209] empirical evaluation, namely the *Relative Overgeneralization 3* (RO3) and *Gradient 2* games. Both domains, which are outlined below, proved challenging for LMRL2 (and all other approaches). We choose to conduct our evaluation using these two games because [209]:

i Despite being designed to address the relative overgeneralization pathology, LMRL2 only converged upon correct and complete joint policies in 73.32% of RO3 runs;

ii While converging upon correct joint-policies in 99.97% of runs conducted for Gradient 2, the percentage of complete policies for LMRL2 was only 5.48%.

We hypothesize that DLQ's ability to minimize the impact of the alter-exploration problem, combined with a staggered temperature decay, will result in a higher percentage of complete and correct joint-policies compared to the results reported for LMRL2.

5.4.1 The Relative Overgeneralization Game

Wei and Luke [209] introduced three games to evaluate the impact of confronting independent learners with relative overgeneralization and miscoordination pathologies in domains with multiple states: *Relative Overgeneralization 1 – 3* (RO1, RO2 and RO3). Given that LMRL2, distributed Q-learning, hysteretic Q-learning and SOoN are capable of delivering high convergence rates on complete policies in RO1 and RO2, we focus on arguably the most challenging of the three relative overgeneralization games, RO3, where the learners are confronted with relative overgeneralization through stochastic state transitions. Figure 5.7 provides a state-transition diagram for RO3 (the illustration is taken from Wei and Luke [209]). The game consists of three states. Taking actions in state *State 1* yields rewards of 0. However, the joint-action $\langle A, A\rangle$ is most likely to result in the learners progressing to the optimal *State 2*, from where coordinated actions $\langle A, A\rangle$ and $\langle B, B\rangle$ can result in a reward of 10. However, pairing action A with B or C in *State 1* has an increased likelihood of the learners transitioning into *State 3*, which yields a reward of 0 for each action combination. Joint-actions including C are more likely to transition to *State 2*, but still less likely than when the agents choose $\langle A, A\rangle$. All actions taken in *State 2* and *State 3* result in a transition into an absorbing state. Therefore, the learners are confronted with a Fully Stochastic Climb Game distributed over two transitions. We note that in this game every correct solution is a complete solution.

We first attempt to replicate the results reported for LMRL2 in RO3 using the following hyperparameter configuration [209]:

- $\alpha = 0.1$;
- $\gamma = 0.9$;

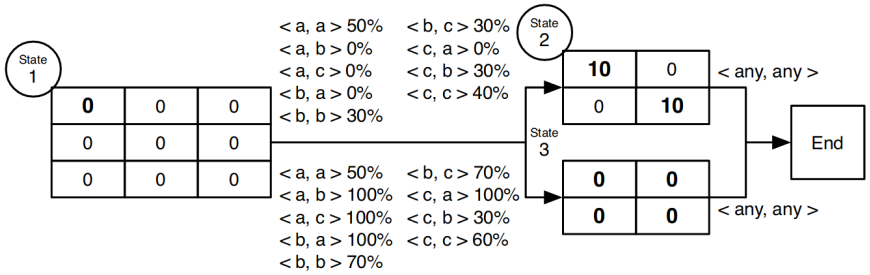


Figure 5.7: Relative Overgeneralization 3 State Transition Diagram (Illustration is taken from Wei and Luke [209]).

- $\tau = 0.1$;
- $\nu = 0.995$;
- $MaxTemp = 50.0$;
- $\omega = 0.3$;
- $k = 1.0$;
- $Q(x, u) \leftarrow 0$ (the lowest reward in the game);

Interestingly we are unable to replicate the 73.32% convergence rate using this setting. However, upon experimenting with the Q-value initialization we do converge on 70.5% using the authors' configuration with an initialization of $Q(x, u) \leftarrow 10$, the maximum reward available in the game. Furthermore, we can improve upon the benchmark reported by Wei and Luke [209] via reducing the size of the learning rate α. In Figure 5.8 we illustrate the correct run percentages for the following hyperparameter combinations:

- $Q_{init} = \{0, 10\}$;
- $\alpha = \{0.001, 0.01, 0.1\}$.

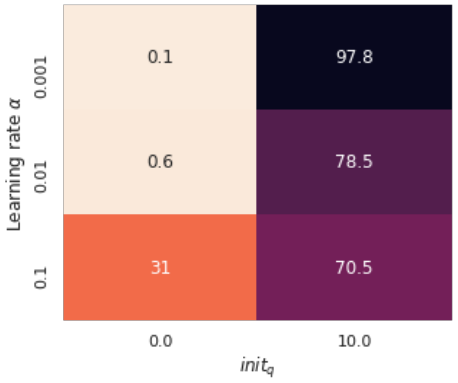
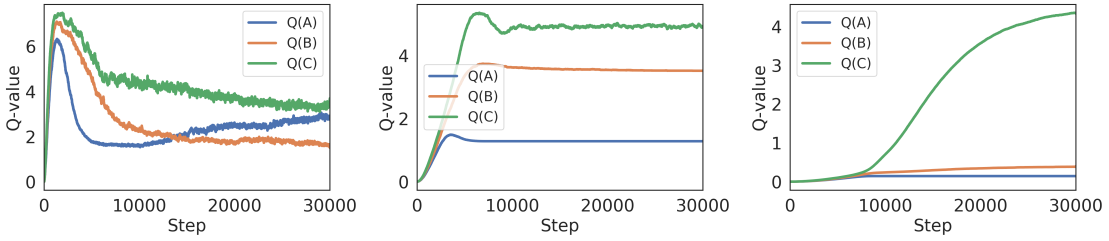


Figure 5.8: Correct run percentage for LMRL2 hyperparameter configurations within RO3. We compare Q-value initialization and learning rate α.

As in previous evaluation we conduct 1,000 training runs per hyperparameter configuration. Each training run consists of 30,000 iterations [209]. We observe a significant increase in the number of optimal (correct) joint-policies upon initializing the Q-values with 10.0. Furthermore, we observe an increase in the percentage of correct runs for $Q_{init} = 10$ upon lowering α. Interestingly lowering the α-values proves detrimental when $Q_{init} = 0$. A closer look at the average Q-values helps shed light on why this is the case. We conduct 100 additional training runs for each of the following learning rate configurations, storing the Q-values for each iteration: $\alpha = \{0.1, 0.01, 0.001\}$. In Figure 5.9 we plot the average Q-values. We observe that when starting with the Q-values at 0, the learners require multiple iterations for the utility value estimates for each action in *State 1* to increase. Using a low leniency moderation factor $k = 1.0$ the learners' leniency decreases rapidly, explaining why the Q-value for action C frequently emerges as the highest utility value estimate. In Figure 5.10 meanwhile we observe that agents using a Q-value initialization of 10.0 benefit from a low learning rate α, with the Q-values from each action being more closely aligned during the initial learning phase of each training run. Due to using a lower learning rate miscoordination has less of an impact, meaning the Q-values for each action remain closely aligned until action A emerges as the optimal action (Sub-Figure 5.10(c)).

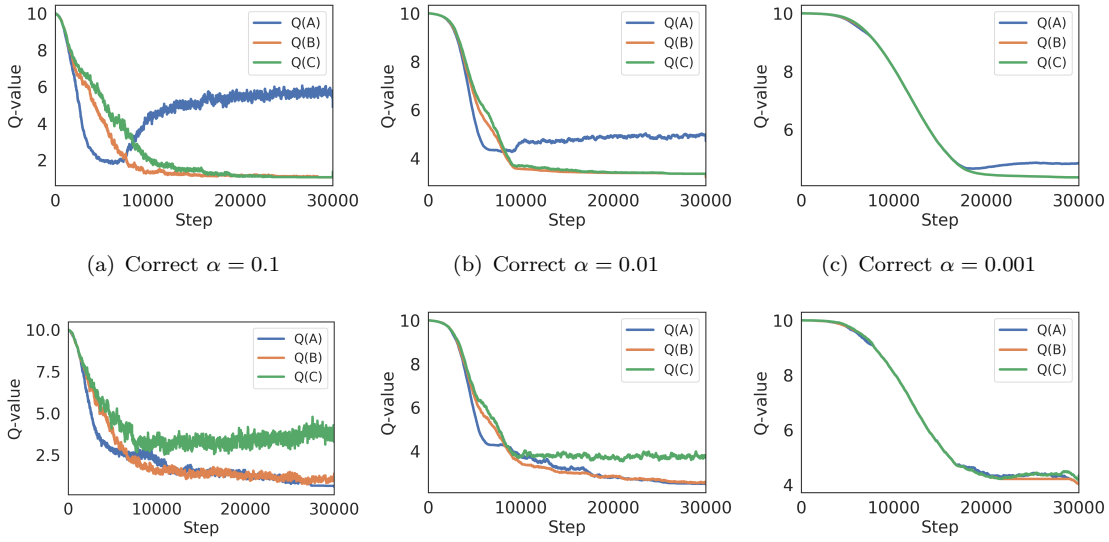


(a) $\alpha = 0.1$ (b) $\alpha = 0.01$ (c) $\alpha = 0.001$

Figure 5.9: Q-value comparison for RO3 *State 1* using LMRL2 with $Q_{init} = 0$.

We conduct a second hyperparameter sweep with $\alpha = 0.001$ to evaluate the impact of using a different leniency moderation factors $k = \{1, 10, 100, 1000\}$, while also gathering Synchronized DLQ (DLQ) and Asynchronized DLQ (ADLQ) runs. For DLQ we use the following hyperparameters: $\varepsilon \leftarrow 0.001$, $ExplorationSteps \leftarrow 500$, $\tau = 0.1$ and $\varrho \leftarrow 0.0001$. We illustrate the results from our runs in 5.11. We find for all approaches the highest convergences rates are achieved when using a low leniency moderation factor $k = 1$, although the drop off in convergence rate is less noticeable for SDLQ and ADLQ. Furthermore, there appears to be no significant difference between the convergence rates of ADLQ and SDLQ across settings. Nevertheless, both approaches outperform LMRL2 for each hyperparameter configuration.

To summarize, we identify a number of hyperparameter configurations for LMRL2 that improve upon the 73.32% convergence rate reported by Wei and Luke [209] for RO3, while observing even higher convergence rates for SDLQ and ADLQ. However, we consider that the high convergence rate for DLQ should not come as a surprise. Upon



(a) Correct $\alpha = 0.1$ (b) Correct $\alpha = 0.01$ (c) Correct $\alpha = 0.001$

(d) Incorrect $\alpha = 0.1$ (e) Incorrect $\alpha = 0.01$ (f) Incorrect $\alpha = 0.001$

Figure 5.10: Q-value comparison for RO3 *State 1* using LMRL2 with $Q_{init} = 10$.

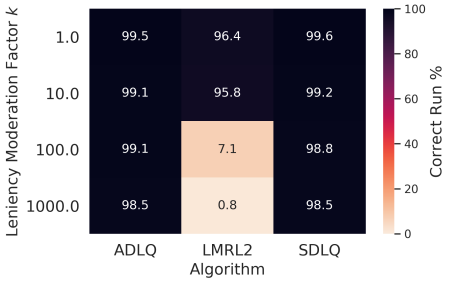


Figure 5.11: Hyperparameter sweep for LMRL2, SDLQ and ADLQ within RO3. For LMRL2 Q-values are initialized to 10. The learning rate $\alpha = 0.001$.

using distributed Q-learning's approach towards solving miscoordination in *State 2*, the learners are essentially being confronted with a less punishing variation of the Fully Stochastic Climb Game. We now move on to the more challenging Gradient 2 game, where, due to an increase in the number of states, considerations are required regarding the number of iterations that learners should be given in order to maximize the likelihood of converging upon a complete joint policy.

5.4.2 The Gradient Game

Wei and Luke [209] introduced two variations of the gradient game. Given that *Gradient 1* can be mastered by maximum based learners (e.g., distributed Q-learning) [209], we focus on the more challenging version, *Gradient 2*, which confronts learners with the deception, stochastic reward and miscoordination pathologies. In Figure 5.12 we provide a state-transition diagram of Gradient 2 (the illustration is taken from Wei and Luke [209]). The game is deceptive in that *State 3* and *State 4* have higher local rewards compared to *State 2*, but ultimately lead to poor future rewards, when compared against the

optimal average reward of 30 that can be obtained in *State 5*. Furthermore, the learners are also confronted with miscoordination in states 5 through 13. All approaches from Wei and Luke's [209] empirical evaluation, with the exception of LMRL2, struggled to converge upon correct solutions in Gradient 2. However, while 99.97% of the LMRL2 runs converged on correct joint-policies, only 5.48% of runs were complete [209]. One potential explanation for the low percentage of complete runs is that Gradient 2 has a larger state-action space compared to the other games used during the authors' empirical evaluation. Therefore, adequately decaying the temperature values to obtain average utility estimates within each state necessitates that each state-action pair is visited a sufficient number of times. However, if the learners converge before this can happen, then the average utility values will never be computed for the less optimal paths, preventing a complete joint-policy from being learnt.

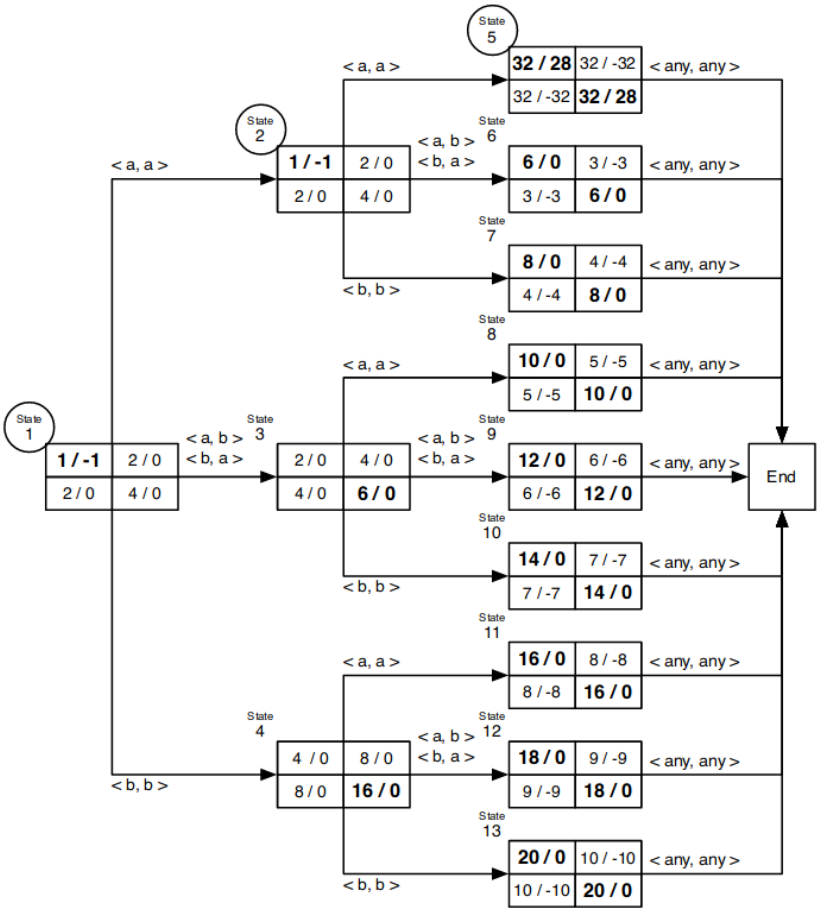


Figure 5.12: State transition diagram for the *Gradient 2* game (Illustration is taken from Wei and Luke [209]).

We hypothesize that applying DLQ with a sufficiently small ϱ to Gradient 2 will result in an increase of the number of complete runs compared to the 5.48% achieved by LMRL2. However, during preliminary trials we observed that an excessive number of iterations are required in order to obtain $\overline{C}(x) > 1 - \varepsilon$ for all $x \in \mathcal{X}$. Increasing ϱ

meanwhile comes at the cost of a decrease in the number of complete solutions. We consequently evaluate SDLQ and ADLQ using the following hyperparameter configuration, with each training run consisting of 240,000 iterations:

- $k \leftarrow 1.0$;
- $\alpha \leftarrow 0.01$;
- $\delta \leftarrow 0.995$;
- $\tau \leftarrow 0.1$;
- $\gamma \leftarrow 0.9$;
- $MaxTemp \leftarrow 50$;
- $ExplorationSteps \leftarrow 1,000$;
- $\varepsilon \leftarrow 0.001$;
- $\varrho \leftarrow 0.0001$.

We gather 1,000 training runs for each SDLQ and ADLQ hyperparameter configuration, resulting in the following convergence rates:

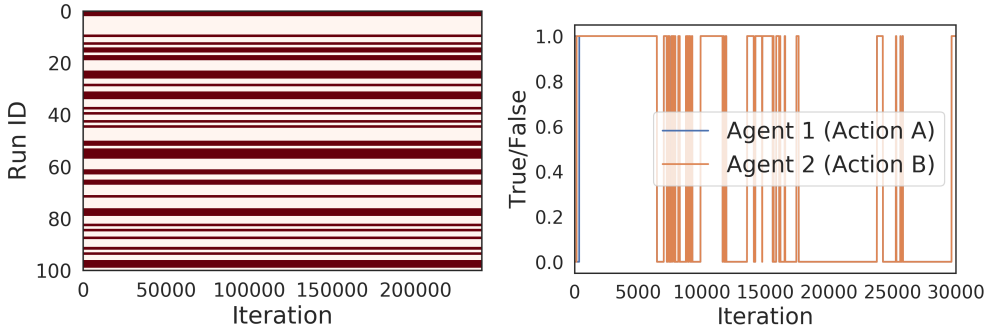
Approach	**Complete Run %**	**Correct Run %**
ADLQ	83.7%	99.6%
SDLQ	48.3%	49.3%

TABLE 5.1: Asynchronized DLQ (ADLQ) and Synchronized DLQ (SDLQ) complete and correct run percentages in the Gradient 2 game.

Upon investigating why SDLQ is outperformed by ADLQ, we find that reward spaces exist where the stochasticity introduced by asynchronous updates is beneficial. We observe that *State 5* in particular represents a challenge for SDLQ, due to the maximum reward for each of the four joint-actions being 32. Each learner's policy table will be set to the first action executed that results in the max reward in *State 5* during the random exploration phase. Due to the action space only consisting of two actions, if the agents happen to lock onto one of the sub-optimal joint-actions $\langle A, B\rangle$ or $\langle B, A\rangle$, then they enter a cycle of switching between incompatible-actions once synchronized leniency updates are enabled. As a result SDLQ learners begin alternating between sub-optimal joint-actions $\langle A, B\rangle$ and $\langle B, A\rangle$. Upon subsequently repeatedly receiving a miscoordination penalty the learners significantly underestimate the utility available for each action in *State 5*.

The barcode plot in Sub-Figure 5.13(a) illustrates the consequences of agents locking onto sub-optimal joint-actions in *State 5*. The plot depicts occurrences of miscoordination in *State 5* for 100 training runs, with dark red lines indicating that the learners'

joint-policy is set to either $\langle A, B \rangle$ or $\langle B, A \rangle$. We observe that the behaviour remains consistent throughout each training run following the initial exploration phase. In Sub-Figure 5.13(b) we illustrate the joint-actions in *State 5* for an incorrect SDLQ run, depicting $\pi_1(5, A)$ and $\pi_2(5, B)$, the policy table entries for *State 5* and actions A and B for *Agent 1* and *Agent 2* respectively. We observe that the changes to $\pi_1(5, A)$ and $\pi_2(5, B)$ are synchronized following an initial exploration phase. In Figure 5.14 we illustrate the Q-values learned as a consequence of not being able to escape this sub-optimal joint-action spiral. ADLQ meanwhile can escape this cycle through agents unilaterally changing their policy table following an asynchronous leniency update. In theory the bounded policy table updates using ε should allow SDLQ to escape this cycle. The use of policy table updates conditioned on ε allowed DLQ learners to master a similar dilemma in *State 1* of RO3, where following *learning phase 1* $\forall x \in \mathcal{X}$ and $\forall u \in \mathcal{U}$ $Q(x, u) = 10.0$. In RO3 we observe a gradual decrease in the convergence rate upon lowering ε. However, we have been unable to identify an ε setting that improves the convergence rate for SDLQ in Gradient 2. We therefore leave this for future work.



(a) Miscoordination Iterations

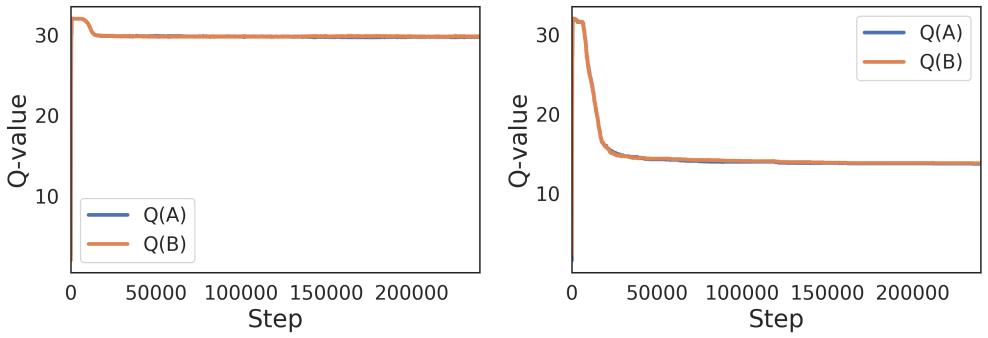
(b) Policy Plot

Figure 5.13: Sub-Figure (a) illustrates instances of miscoordination (dark red) within 100 runs gathered for SDLQ within State 5 of Gradient 2. We observe that learners who end up with a sub-optimal joint-policy during the initial exploration phase often fail to escape the miscoordination cycle. Sub-Figure (b) depicts $\pi_1(5, A)$ and $\pi_2(5, B)$, the policy table entries for *State 5* and actions A and B for *Agent 1* and *Agent 2* respectively, from a failed SDLQ Gradient 2 run. We note that the line for *Agent 1* is obscured by *Agent 2* for the majority of the run, illustrating that the learners switch between sub-optimal joint-action during identical time-steps.

To summarize, while the percentage of correct joint-policies for ADLQ of 99.6% is below the 99.97% achieved by LMRL2, the 83.7% complete policies represents a significant improvement. We hypothesize that these results could be further improved upon through additional hyperparameter tuning.

5.5 Summary

In the previous chapter we observed that even for approaches designed to mitigate the moving target problem (e.g., LMRL2), one agent unilaterally changing their policy can have catastrophic consequences. For example, in the Fully Stochastic Climb Game the



(a) Correct Runs

(b) Incorrect Runs

Figure 5.14: SDLQ Q-values from Gradient 2 State 5 for actions A and B averaged over 59 and 41 correct and incorrect runs respectively. Note: the Q-values for action A are obscured by B due to the Q-values being averaged over multiple runs. For correct runs we observe that SDLQ is able to estimate the average utility for coordinated actions involving actions A and B. For incorrect runs meanwhile the Q-value for each action is significantly underestimated.

moving target problem can lead to miscoordination transitions, resulting in convergence upon a sub-optimal joint-policy due to relative overgeneralization.

In this chapter we introduce Distributed-Lenient Q-learning (DLQ) to address the noise introduced by miscoordination through using two learning phases: a maximum learner phase (*learning phase 1*), where a uniform action selection policy is utilized to establish the maximum reward available for each action, followed by a greedy action selection phase (*learning phase 2*) where the learners use lenient utility value updates, allowing the agents to gradually establish the average utility for each equilibrium, while minimizing the impact of the moving target problem. In addition, we introduce synchronized leniency updates, which help eliminate miscoordination in a number of challenging domains, including the Fully Stochastic Climb Game [90], a domain, where a large number of previous independent learning approaches have struggled [90, 120, 121, 145, 148, 149, 209]. We also find that DLQ achieves state of the art performances in two challenging Markov games: *Gradient 2* and *Relative Overgeneralization 3* [209].

However, we note that the synchronization property upon which SDLQ relies may not always be guaranteed, in particular within asymmetric games. We leave the identification of conditions where synchronisation is guaranteed to occur for future work. Furthermore, the assumption of agents operating on synchronized parameters based on discrete time-scales does not hold in many real world domains, which is the condition necessary for performing synchronized leniency updates. Due to the absence of standards for learning agents, in many multi-agent systems the assumption of all algorithms being implemented with the same hyperparameter configurations, or even using the same algorithm, is unrealistic [91]. Furthermore, in our evaluation of the Markov game *Gradient 2* we have seen that reward spaces exist where performing synchronized updates can actually reduce the likelihood of independent learning agents converging

upon an optimal joint-policy. SDLQ's struggles within *Gradient 2* serve as a reminder that, even for simple strategic-form and low-dimensional Markov games with a small state-space, finding a *silver bullet* approach for independent learning is non-trivial. Finally, the excessive number of iterations required by ADLQ to achieve the above result is concerning, especially given that Gradient 2 only consists of 26 state-action pairs. Our findings therefore reiterate that independent learning often requires compromises in order to achieve convergence within a reasonable amount of time. It should not come as a surprise therefore that similar compromises are required when scaling leniency to multi-agent *deep* reinforcement learning, which is the focus of the next chapter.

Chapter 6

Lenient Multi-Agent Deep Reinforcement Learning

This chapter is based on the following publication:

Gregory Palmer, Karl Tuyls, Daan Bloembergen, and Rahul Savani, Lenient Multi-Agent Deep Reinforcement Learning, In Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, 2018, pp. 443–451.

The field of *deep reinforcement learning* has seen a great number of successes in recent years. Deep reinforcement learning agents have been shown to master numerous complex problem domains, ranging from computer games [98, 129, 160, 203] to robotics tasks [37, 63]. Much of this success can be attributed to using convolutional neural network (*ConvNet*) architectures as function approximators, allowing reinforcement learning agents to be applied to domains with large or continuous state and action spaces [98, 129, 160, 203].

Recently the sub-field of multi-agent deep reinforcement learning has received an increased amount of attention. However, as we have seen in the previous chapters, multi-agent reinforcement learning is challenging even in stateless environments with only two implicit learning agents, lacking the convergence guarantees present in most single-agent learning algorithms [15, 101, 122, 137, 199]. One of the key challenges faced within multi-agent reinforcement learning is the *moving target problem*: Given an environment with multiple agents whose rewards depend on each others' actions, the difficulty of finding optimal policies for each agent is increased due to the policies of the agents being non stationary [15, 17, 25, 66, 78, 90, 137, 157, 174, 181, 194, 197, 200].

Due to the moving target problem reinforcement learning algorithms that converge in a single agent setting (e.g., decentralized Q-learning) often fail in fully-cooperative multi-agent systems with independent learning agents that require implicit coordination strategies. For decentralized learning agents using deep Q-network architectures the moving target problem represents a significant challenge [46, 66, 141]. ConvNets are

often trained to approximate policy and value functions through sampling past state transitions stored by the agent inside an *experience replay memory* $\mathcal{D}$ (See Chapter 2, Section 2.6) [98, 129, 160, 203]. However, the use of an experience replay memory in a multi-agent deep reinforcement learning context amplifies the moving target problem, as a large proportion of the state transitions stored can become deprecated [46, 66, 141].

In Chapter 4 we evaluated a number of independent learning approaches designed to overcome the moving target problem (among other multi-agent learning pathologies). However, we observed that even for approaches designed to mitigate the moving target problem, one learner unilaterally changing their policy can result in convergence upon a sub-optimal joint-policy. We introduced Distributed-Lenient Q-learning (DLQ) and synchronized leniency updates in Chapter 5 to mitigate the noise introduced by miscoordination. However, many multi-agent systems lack standards for learning agents [91]. Therefore, we cannot assume that independent learners can operate on synchronized discrete time-scales, which is the condition necessary for performing synchronized leniency updates. Furthermore, we have seen that reward spaces exist where performing synchronized updates can actually reduce the likelihood of independent learning agents converging upon an optimal joint-policy (see Chapter 5, Section 5.4.2). Nevertheless, in our empirical evaluation lenient learners have proven more robust than other methods towards multi-agent learning pathologies. This raises the question whether leniency can be applied to domains with a high-dimensional state space.

In this chapter we show how *lenient learning* can be scaled to multi-agent deep reinforcement learning via modifying the DQN [129] and *Double-DQN* (DDQN) [203] architectures, introducing the *Lenient (Double) Deep Q-Network* (LDQN and LDDQN respectively). To recap: lenient learners store temperature values that are associated with state-action pairs [17, 18, 148, 149]. Each time a state-action pair is visited the respective temperature value is decayed, thereby decreasing the amount of leniency that the agent applies when performing a policy update for the state-action pair. The stored temperatures enable the agents to gradually transition from optimists to average reward learners for frequently encountered state-action pairs, allowing the agents to outperform optimistic and maximum based learners in environments with misleading stochastic rewards. In this chapter we extend this idea to multi-agent deep reinforcement learning by storing leniency values in the experience replay memory $\mathcal{D}$, and demonstrate empirically that lenient multi-agent deep reinforcement learning agents learning implicit coordination strategies in parallel are able to converge on the optimal joint policy in difficult coordination tasks with stochastic rewards.

Omidshafiei et al. [141] recently applied concepts from hysteretic Q-learning to multi-agent deep reinforcement learning. In Section 6.5 we empirically evaluate our Lenient DDQN against Hysteretic Double Deep Q-Networks (HDDQNs). We find that while HDDQNs and LDDQNs deliver comparable performances in deterministic reward domains, HDDQNs struggle in fully cooperative domains that yield stochastic rewards.

However, we demonstrate that the performance of HDDQNs within stochastic reward environments can be improved with a scheduled approach.

Our main contributions can be summarized as follows.

1) We introduce Lenient (Double) Deep Q-Network, which includes two extensions to leniency: a retroactive *temperature decay schedule* (TDS) that prevents premature temperature cooling, and a $\overline{\mathcal{T}}\,(o)$-Greedy exploration strategy, where the probability of the optimal action being selected is based on the average temperature of the current state. When combined, TDS and $\overline{\mathcal{T}}\,(o)$-Greedy exploration encourage exploration until average rewards have been established for later transitions.

2) We show the benefits of using TDS over *average temperature folding* (ATF).

3) We provide an extensive analysis of leniency-related hyperparameters for LDDQN.

4) We propose a *scheduled*-H(D)DQN that applies less optimism towards state transitions near terminal states compared to earlier transitions within the episode.

5) We introduce two extensions to the Cooperative Multi-agent Object Transportation Problem (CMOTP) [26], including narrow passages that test the agents' ability to master fully-cooperative sub-tasks and stochastic rewards.

6) We empirically evaluate our proposed LDDQN and SHDDQN against standard HDDQNs using the extended versions of the CMOTP. We find that while HDDQNs perform well in deterministic CMOTPs, they are significantly outperformed by SHDDQNs in domains that yield a stochastic reward. Meanwhile LDDQNs comprehensively outperform both approaches within the stochastic reward CMOTP.

6.1 Related Work

A number of methods have been proposed to help deep reinforcement learning agents converge towards an optimal joint policy in cooperative multi-agent tasks. Gupta et al. [66] evaluated policy gradient, temporal difference error, and actor critic methods on cooperative control tasks that included discrete and continuous state and action spaces, using a decentralized parameter sharing approach with centralized learning. More recent successful approaches have focused on centralized training for decentralized execution (CTDE) [157, 174, 181], e.g., decomposing a team value function into agent-wise value functions through the use of a value decomposition network architecture [181]. Others have attempted to help concurrent learners converge through identifying and deleting obsolete state transitions stored in the replay memory. For instance, Foerster et al. [46] used importance sampling as a means to identify outdated transitions while maintaining an action observation history of the other agents. Our current work does not require the agents to maintain an action observation history. In contrast, our current work focuses on independent learning using optimistic agents within environments that require implicit coordination. This decentralized approach to multi-agent systems offers advantages such as speed, scalability and robustness [122]. The motivation for using implicit coordination

is that communication can be expensive in practical applications, and requires efficient protocols [8, 122, 188].

As we have seen in the previous chapters, hysteretic Q-learning is a form of optimistic learning with a strong empirical track record in fully-observable multi-agent reinforcement learning [9, 122, 217]. Originally introduced to prevent the overestimation of Q-values in stochastic games, hysteretic learners use two learning rates: a learning rate α for updates that increase the value estimate (Q-value) for a state-action pair and a smaller learning rate β for updates that decrease the Q-value [120]. However, empirical evaluations (including our own in Chapter 4) have shown that while hysteretic learners perform well in deterministic environments, they tend to perform sub-optimally in games with stochastic rewards [120, 209]. Hysteretic learners' struggles in these domains have been attributed to learning rate β's inter-dependencies with other agents' exploration strategies [122].

As we have established, lenient learners present an alternative to the hysteretic approach, and have empirically been shown to converge towards superior policies in stochastic games with a small state space. Similar to the hysteretic approach, lenient agents initially adopt an optimistic disposition, before gradually transforming into average reward learners. Lenient methods have received criticism in the past for the time they require to converge [209], the difficulty involved in selecting the correct hyperparameters, the additional overhead required for storing the temperature values, and the fact that they were originally only proposed for matrix games [122]. We have encountered these challenges ourselves in Chapters 4 and 5. However, given their success in tabular settings, we here investigate whether leniency can be applied successfully to multi-agent deep reinforcement learning.

6.2 Independent Learner Baseline

As in the previous chapter our proposed algorithms are based upon Q-learning, a form of temporal difference reinforcement learning that is well suited for solving sequential decision making problems that yield stochastic and delayed rewards [11, 207]. Since interesting sequential decision problems frequently have a large state-action space, Q-values are often approximated using function approximators such as tile coding [11] or neural networks [203] (see Section 2.6). Furthermore, learners in this context are frequently confronted with partial observability, where Q-values are computed for observations $o \in \mathcal{O}$ rather than states $x \in \mathcal{X}$. The algorithms evaluated in this chapter are extensions of the *Double-DQN* (DDQN) introduced by Van Hasselt et al. [203] (see Section 2.6). Each agent i is implemented with a *ConvNet* trained to approximate Q-values for observation-action pairs: $Q_i : \mathcal{O}_i \times \mathcal{U}_i \rightarrow \mathbb{R}$ [104]. The learning agents i are also each implemented with a separate experience replay memory $\mathcal{D}_i$ used to store state transitions as tuples (o_i, u_i, r_i, o'_i), consisting of an observation o_i, action u_i, the resulting observation o'_i and the immediate reward r_i. To ensure obsolete transitions are

eventually discarded experience replay memories $\mathcal{D}_i$ are implemented as First-In First-Out (FIFO) queues [104]. The network parameters θ_i are trained using Adam [94] on the mean squared Bellman residual with the expectation taken over state transitions uniformly sampled from an experience replay memory $\mathcal{D}_i$ [106, 129] (Eq. (2.10)) [1].

We now proceed to describe our main algorithmic contributions of this chapter. First we detail our proposed *Lenient (Double) Deep Q-Network*, and thereafter we discuss our extension to hysteretic deep multi-agent reinforcement learning, which we call *Scheduled Hysteretic (Double) Deep Q-Network*.

6.3 Lenient Deep Q-Learning

In this section we outline how *leniency* can be scaled to multi-agent deep reinforcement learning. Given that learners situated within environments with a high-dimensional state-space will rarely receive identical observations, we shall first consider how the temperature values required by lenient learners can be maintained for semantically similar observations. We subsequently consider how the leniency function can be combined with the (Double) DQN architecture, and propose two extensions to leniency: a retroactive *temperature decay schedule* (TDS) designed to prevent premature temperature cooling, and a $\overline{\mathcal{T}}\,(o)$-Greedy exploration strategy, where the probability of the optimal action being selected is based on the average temperature of the current state. We find that when these two extensions are combined, TDS and $\overline{\mathcal{T}}\,(o)$-Greedy exploration encourage exploration until average rewards have been established for later transitions.

6.3.1 Clustering Observations using Autoencoders

In environments with a high-dimensional or continuous observation space, a tabular approach for mapping each possible observation-action pair to the temperature values required by lenient learners is no longer feasible. Binning can be used to discretize low-dimensional continuous observation-spaces, however, further considerations are required regarding mapping semantically similar observations to a decaying temperature value used by lenient learners when dealing with high-dimensional domains, such as image observations. Recently, researchers studying the application of count based exploration to deep reinforcement learning have developed interesting solutions to this problem. For example, Tang et al. [190] used autoencoders to automatically cluster states in a meaningful way in challenging benchmark domains, including Montezuma's Revenge.

An autoencoder is a neural network architecture frequently used to learn reduced encodings for high-dimensional data [133, 204]. Architectures used to learn encodings for images typically consist of convolutional, dense, and transposed convolutional layers, which are trained to minimize the expected reconstruction error [133]. For instance, in a deep reinforcement learning context an autoencoder can be trained to compress and reconstruct the observations stored in the agent's experience replay memory $\mathcal{D}$ [189].

[1] For simplicity we drop the subscript i denoting an individual agent going forward.

The autoencoder subsequently serves as a pre-processing function $g : \mathcal{O} \rightarrow \mathbb{R}^D$, with a dense layer consisting of D neurons with a saturating activation function (e.g. a Sigmoid function) at the centre. SimHash [30], a locality-sensitive hashing (LSH) function, can be applied to the rounded output of the dense layer to generate a hash-key ϕ for an observation o. This hash-key is computed using a constant $n \times D$ matrix A with i.i.d. entries drawn from a standard Gaussian distribution $N(0, 1)$ as

$$\phi(o) = sgn\big(A\, g(o)\big) \in \{-1, 1\}^n, \tag{6.1}$$

where $g(o)$ is the autoencoder pre-processing function, and n controls the granularity such that higher values yield a more fine-grained clustering [190].

We use a dictionary to map each $\langle \phi(o), u \rangle$ pair encountered to a temperature value, where the hash-keys are computed using Tang et al.'s [190] approach described above. If a temperature value does not yet exist for $\langle \phi(o), u \rangle$ within the dictionary, then an entry is created, setting the temperature value equal to $MaxTemperature$. Otherwise the current temperature value is used and subsequently decayed, to ensure the agent will be less lenient when encountering a semantically similar observation in the future.

6.3.2 Combining Leniency with Deep Q-Network Architectures

Combining leniency with (D)DQNs requires careful considerations regarding the use of the temperature values, in particular when to compute the amount of leniency that should be applied to a state transition that is sampled from the replay memory. In our initial trials we used leniency as a mechanism to determine which transitions should be allowed to enter the replay memory $\mathcal{D}$. However, this approach led to poor results, presumably due to the agents developing a bias during the initial random exploration phase where transitions were stored indiscriminately. To prevent this bias we use an alternative approach where we compute and store the amount of leniency $L(o_t, u_t)$ at time t within the transition tuple stored in $\mathcal{D}$: $(o_t, u_t, r_{t+1}, o_{t+1}, L(o_t, u_t)_t)$. The amount of leniency that is stored is determined by the current temperature value $\mathcal{T}$ associated with the hash-key $\phi(o)$ for observation o and the selected action u:

$$L(o, u) = 1 - \exp\big(-k \times \mathcal{T}(\phi(o), u)\big). \tag{6.2}$$

We note that the leniency function in Equation (6.2) differs from the one used by LMRL2 (Equation (3.18) in Chapter 3). While the equation used by LMRL2 requires large temperature values for learners to maintain a lenient disposition, e.g., using an initial $MaxTemp = 50$, Equation (6.2) is sensitive towards temperature values within the range $[0, 1]$, a property that will prove valuable for our $\overline{\mathcal{T}}(o)$-Greedy exploration strategy, which we outline in Sub-Section 6.3.4 below. As in standard deep Q-learning the aim is to minimize the loss function of Equation (2.10), with the modification that for each sample j chosen from the replay memory for which the leniency conditions of Equation (6.2) are not met, are ignored.

Algorithm 6 Application of temperature decay schedule (TDS)

1: Upon reaching a terminal state **do**
2: $n \leftarrow 0$, *steps* $\leftarrow$ steps taken during the episode
3: **for** $j = steps$ to 0 **do**
4: **if** $\beta_n \mathcal{T}_t \left(\phi\left(o_j\right), u_j\right) < \nu_t$ **then**
5: $\mathcal{T}_{t+1}\left(\phi\left(o_j\right), u_j\right) \leftarrow \beta_n \mathcal{T}_t \left(\phi\left(o_j\right), u_j\right)$
6: **else**
7: $\mathcal{T}_{t+1}\left(\phi\left(o_j\right), u_j\right) \leftarrow \nu_t$
8: $n \leftarrow n + 1$
9: $\nu \leftarrow \mu\, \nu$

6.3.3 Retroactive Temperature Decay Schedule

Throughout initial trials we found that temperatures decay rapidly for state-action pairs belonging to challenging sub-tasks in the environment, even when using *Average Temperature Folding* (ATF) (See Equation (3.27) in Chapter 3). In order to prevent this premature cooling of temperatures we developed an alternative approach using a pre-computed temperature decay schedule (TDS) $\beta_0, \ldots, \beta_n$ with a step limit n. The values for β are computed using an exponent ρ which is decayed using a decay rate d:

$$\beta_n = \exp\left(\rho \times d^t\right) \tag{6.3}$$

for each $t, 0 \leq t < n$.

Upon reaching a terminal state the temperature decay schedule is applied to samples j as outlined in Algorithm 6. The aim is to ensure that temperature values of observation-action pairs encountered during the early phase of an episode are decayed at a slower rate than those close to the terminal state transition (line 4). We find that maintaining a slow-decaying maximum temperature ν (lines 5-7) that is decayed using a decay rate μ helps stabilize the learning process when ϵ-Greedy exploration is used. Without the decaying maximum temperature the disparity between the low temperatures in well explored areas and the high temperatures in relatively unexplored areas has a destabilizing effect during the later stages of the learning process. Furthermore, for agents also using the temperature values to guide their exploration strategy (see below), ν can help ensure that the agents transition from exploring to exploiting within reasonable time. The decaying maximum temperature ν is used whenever $\mathcal{T}\left(\phi\left(o_j\right), u_j\right) > \nu_t$, or when agents fail at their task in environments where a clear distinction can be made between success and failure. Therefore TDS is best suited for domains that yield sparse large rewards.

Applying the TDS after the agents fail at a task could result in the repeated decay of temperature values for state-action pairs leading up to a sub-task. For instance, the sub-task of transporting a heavy item of goods through a doorway may only require a couple of steps for trained agents who have learned to coordinate. However, untrained agents may require thousands of steps to complete the task. If a time-limit is imposed

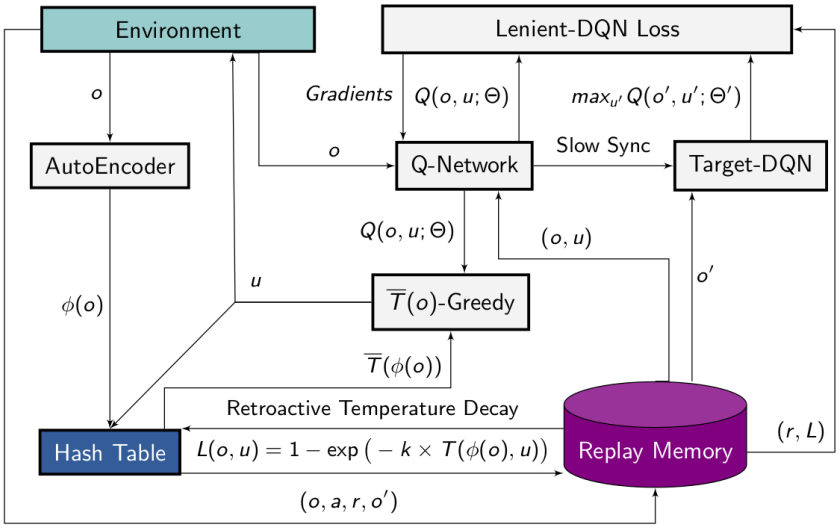


Figure 6.1: Lenient-DQN Architecture. We build on the standard DQN architecture [203] by adding a lenient loss function (top right, see Section 6.3.2). Leniency values are stored in the replay memory along with the state transitions; we cluster semantically similar states using an autoencoder and SimHash (bottom left), and apply our retroactive temperature decay schedule (TDS, Algorithm 6). Actions are selected using the $\overline{\mathcal{T}}(o)$-Greedy exploration method.

for the agents to deliver the goods, and the episode ends prematurely while an attempt is made to solve the sub-task, then the application of the TDS will result in the rapid decay of the temperature values associated with the frequently encountered state-action pairs. We resolve this problem by setting the temperature values $\mathcal{T}_t\left(\phi\left(o_j\right), u_j\right) > \nu$ to ν at the end of incomplete runs instead of repeatedly decaying them, thereby ensuring that the agents maintain a lenient disposition towards one another.

6.3.4 $\overline{\mathcal{T}}(o)$-Greedy Exploration

During initial trials we encountered the same problems discussed by Wei and Luke [209] regarding the selection of the temperature moderation factor for the Boltzmann action selection strategy. This led to the development of a more intuitive $\overline{\mathcal{T}}(o)$-Greedy exploration method where the average temperature value $\overline{\mathcal{T}}(o_t) \in (0, 1]$ for a state o_t replaces the ϵ in the ϵ-Greedy exploration method. An exponent ξ is used to control the pace at which the agents transition from explorers to exploiters. The agent therefore selects action $u = \text{argmax}_{u \in \mathcal{U}} Q(o_t, u)$ with a probability $1 - \overline{\mathcal{T}}(o_t)^{\xi}$ and a random action with probability $\overline{\mathcal{T}}(o_t)^{\xi}$. We outline our lenient deep Q-learning architecture in Figure 6.1.

6.4 Scheduled Hysteretic Deep Q-Learning

In Section 4.4.4, we observe that hysteretic learners are vulnerable towards stochastic rewards. Similarly we find that hysteretic deep Q-learning architectures can be lead astray in deceptive domains with with stochastic rewards. To address this weakness

in domains where agents receive sparse stochastic terminal rewards we introduce the *Scheduled Hysteretic (Double) Deep Q-Network*, which we outline below.

To recap, hysteretic Q-learning [120] is an algorithm designed for decentralised learning in deterministic multi-agent environments, and which has recently been applied to multi-agent deep reinforcement learning as well [141]. Two learning rates are used, α and β, with $\beta < \alpha$. The smaller learning rate β is used whenever an update would reduce a Q-value. This results in an optimistic update function which puts more weight on positive experiences, which is shown to be beneficial in cooperative multi-agent settings. As we have observed in our experiments in Section 4.4.4, given a spectrum with traditional Q-learning at one end and maximum-based learning, where negative experiences are completely ignored, at the other, then hysteretic Q-learning lies somewhere in between depending on the value chosen for β.

Hysteretic deep Q-learning architectures compute the error δ_j for each (o, u, o', r) sample j within the batch drawn from $\mathcal{D}$:

$$\delta_j \equiv \big(r + \gamma Q(o'_j, \underset{u \in U}{\operatorname{argmax}}\, Q(o'_j, u; \theta_t); \theta'_t) - Q\left(o_j, u_j; \theta_t\right)\big), \tag{6.4}$$

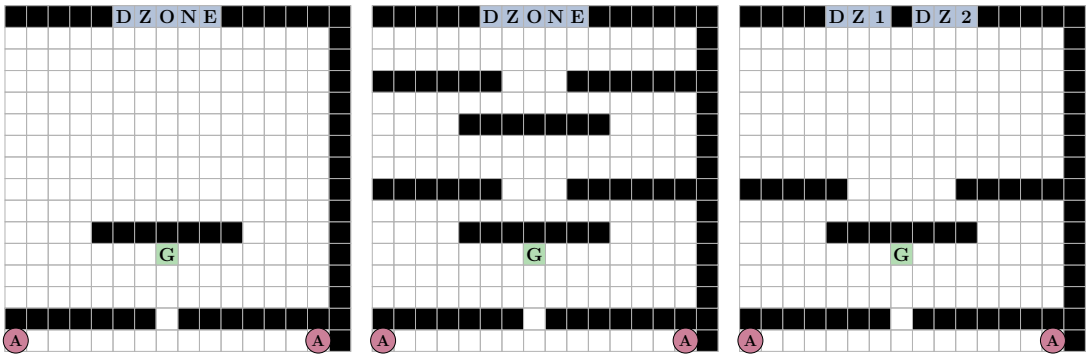
and subsequently, scale each $\delta_j < 0$ using learning rate β:

$$\delta'_j \equiv \begin{cases} \delta_j, & \text{if } \delta \geq 0. \\ \beta\delta_j, & \text{otherwise.} \end{cases} \tag{6.5}$$

before computing the loss:

$$L_t\left(\theta_t\right) = \mathbf{E}_{(o,u,o',r)\sim U(\mathcal{D})}\Big[\left(\delta'\right)^2\Big]. \tag{6.6}$$

For hysteretic deep Q-learning we therefore define β as the percentage of the learning rate α. Hysteretic Q-learners are known to converge towards sub-optimal joint policies in environments that yield stochastic rewards [120]. However, drawing parallels to lenient learning, where it is desirable to decay state-action pairs encountered at the beginning of an episode at a slower rate compared to those close to a terminal state, we consider that the same principle can be applied to hysteretic Q-learning. Subsequently we implemented *Scheduled Hysteretic (Double) Deep Q-Network* (Scheduled-HDDQN) with a pre-computed learning rate schedule $\beta_0, \ldots, \beta_n$ where β_n is set to a value approaching α, and for each $\beta_t, 0 \leq t < n$, we have $\beta_t = d^{n-t}\beta_n$ using a decay coefficient $d \in (0, 1]$. The state transitions encountered throughout each episode are initially stored within a queue data-structure. Upon reaching a terminal state the n state-transitions are transferred to $\mathcal{D}$ as $(o_t,\, o_{t+1},\, r_{t+1},\, u_t,\, \beta_t)$ for $t \in \{0, \ldots, n\}$. Our hypothesis is that storing β values that approach α for state-transitions leading to the terminal state will help agents converge towards optimal joint policies in environments that yield sparse stochastic rewards.



(a) Original (b) Narrow-Passage (c) Stochastic Reward

Figure 6.2: CMOTP Layouts

6.5 Empirical Evaluation

6.5.1 CMOTP Extensions

We subjected our agents to a range of Coordinated Multi-Agent Object Transportation Problems (CMOTPs) inspired by the scenario discussed in Buşoniu et al. [26], in which two agents are tasked with delivering one item of goods to a drop-zone within a grid-world. The agents must first exit a room one by one before locating and picking up the goods by standing in the grid cells on the left and right hand side. The task is fully cooperative, meaning the goods can only be transported upon both agents grasping the item and choosing to move in the same direction. Both agents receive a positive reward after placing the goods inside the drop-zone. The actions available to each agent are to either stay in place or move left, right, up or down. We subjected our agents to three variations of the CMOTP, depicted in Figure 6.2, where each A represents one of the agents, G the goods, and *DZONE* / *DZ* mark the drop-zone(s). The layout in sub-figure 6.2(a) is a larger version of the original CMOTP [26], while the layout in sub-figure 6.2(b) introduces narrow-passages between the goods and the drop-zone, testing whether the agents can learn to coordinate in order to overcome challenging areas within the environment. The layout in sub-figure 6.2(c) tests the agents' response to stochastic rewards. Drop-zone 1 (DZ1) yields a reward of 0.8, whereas drop-zone 2 (DZ2) returns a reward of 1 on 60% of occasions and only 0.4 on the other 40%. DZ1 therefore returns a higher reward on average, 0.8 compared to the 0.76 returned by DZ2. A slippery surface can be added to introduce stochastic state transitions to the CMOTP, a common practice within grid-world domains where the agents move in an unintended direction with a predefined probability at each time-step.

6.5.2 Setup

We conduct evaluations using a Double-DQN architecture [203] as basis for the algorithms. The Q-network consists of 2 convolutional layers with 32 and 64 kernels respectively, a fully connected layer with 1024 neurons and an output neuron for each action.

The agents are fed a 16×16 tensor representing a gray-scale version of the grid-world as input. We use the following pixel values to represent the entities in our grid-world: $Agent1 = 250$, $Agent2 = 200$, $Goods = 150$ and $Obstacles = 50$. *Adam* [94] is used to optimize the networks. Our initial experiments are conducted within a noise free environment, enabling us to speed up the testing of our LDDQN architecture without having to use an autoencoder for hashing; instead we apply python's xxhash. We subsequently test the LDDQN with the autoencoder for hashing in a noisy version of the stochastic reward CMOTP. The autoencoder consists of 2 convolutional Layers with 32 and 64 kernels respectively, 3 fully connected layers with 1024, 512, and 1024 neurons followed by 2 transposed convolutional layers. For our Scheduled-HDDQN agents we pre-compute $\beta_{0 \text{ to } n}$ by setting $\beta_n = 0.9$ and applying a decay coefficient of $d = 0.99$ at each step $t = 1$ to n, i.e. $\beta_{n-t} = 0.99^t \beta_n$, with β_{n-t} being bounded below at 0.4. We summarize the remaining hyper-parameters in Table 6.1. In Section 6.7 we include an extensive analysis of tuning the leniency related hyper-parameters. We note at this point that each algorithm used the same learning rate α specified in Table 6.1.

Component	**Hyper-parameter**	**Setting**
DDQN-Optimization	Learning rate α	0.0001
	Discount rate γ	0.95
	Target network sync. steps	5000
	Experience Replay Memory $\mathcal{D}$ Size	250,000
ϵ-Greedy Exploration	Initial ϵ value	1.0
	ϵ Decay factor	0.999
	Minimum ϵ Value	0.05
Leniency	MaxTemperature	1.0
	Leniency Modification Coefficient k	2.0
	TDS Exponent ρ	-0.01
	TDS Exponent Decay Rate d	0.95
	Initial Max Temperature Value ν	1.0
	Max Temperature Decay Coefficient μ	0.999
	ATF Fold-in Constant τ	0.2
Autoencoder	HashKey Dimensions n	64
	Sigmoidal units in the dense layer D	512

TABLE 6.1: Hyper-parameters

	Original CMOTP Results					
	Hyst. $\beta = 0.5$	**Hyst. $\beta = 0.6$**	**Hyst. $\beta = 0.7$**	**Hyst.$\beta = 0.8$**	**LDDQN ATF**	**LDDQN TDS**
SPE	36.4	36.1	36.8	528.9	36.9	36.8
CSP	92%	92%	92%	91%	92%	92%
SPR	1,085,982	1,148,652	1,408,690	3,495,657	1,409,720	1,364,029

TABLE 6.2: Original CMOTP Results, including average steps per episode (SPE) over the final 100 episodes, coordinated steps percentages (CSP) over the final 100 episodes, and the average steps per training run (SPR).

6.6 Deterministic CMOTP Results

6.6.1 Original CMOTP

The CMOTP represents a challenging fully cooperative task for parallel learners. Past research has shown that deep reinforcement learning agents can converge towards cooperative policies in domains where the agents receive feedback for their individual actions, such as when learning to play pong with the goal of keeping the ball in play for as long as possible [187]. However, in the CMOTP feedback is only received upon delivering the goods after a long series of coordinated actions. No immediate feedback is available upon miscoordination. When using uniform action selection the agents only have a 20% chance of choosing identical actions per state transition. As a result thousands of state transitions are often required to deliver the goods and receive a reward while the agents explore the environment, preventing the use of a small replay memory where outdated transitions would be overwritten within reasonable time. As a result standard Double-DQN architectures struggle to master the CMOTP, failing to coordinate on a significant number of runs, even when confronted with the relatively simple original CMOTP.

We conduct 30 training runs of 5000 episodes per run for each LDDQN and HDDQN configuration. Lenient and hysteretic agents with $\beta < 0.8$ fare significantly better than the standard Double-DQN, converging towards joint policies that were only a few steps shy of the optimal 33 steps required to solve the task. Lenient agents implemented with both ATF and TDS deliver a comparable performance to the hysteretic agents with regards to the average steps per episode and the coordinated steps percentage measured over the final 100 steps of each episode (Table 6.2, left). However, both LDDQN-ATF and LDDQN-TDS average a statistically significant higher number of steps per training run compared to hysteretic agents with $\beta < 0.7$. For the hysteretic agents we observe a statistically significant increase in the average steps per run as the values for β increase, while the average steps and coordinated steps percentage over the final 100 episodes remain comparable.

6.6.2 Narrow Passage CMOTP

Lenient agents using ATF struggle significantly within the narrow passage CMOTP, as evident from the results listed in Table 6.3. We find that the average temperature values cool off rapidly over the first 100 episodes within the Pickup and Middle compartments, as illustrated in Figure 6.3. Meanwhile agents using TDS manage to maintain sufficient leniency over the first 1000 episodes to allow rewards to propagate backwards from the terminal state. We conduct ATF experiments with a range of values for the fold-in constant $\tau = \{0.2, 0.4, 0.8\}$, but always witness the same outcome. Slowing down the temperature decay would help agents using ATF remain lenient for longer, with the side-effects of an overoptimistic disposition in stochastic environments, and an increase in the number of steps required for convergence if the temperatures are tied to the action selection policy. Using TDS meanwhile allows agents to maintain sufficient leniency around difficult sub-tasks within the environment while being able to decay temperatures belonging to later transitions at a faster rate. As a result agents using TDS can learn the average rewards for state transitions close to the terminal state while remaining optimistic for updates to earlier transitions.

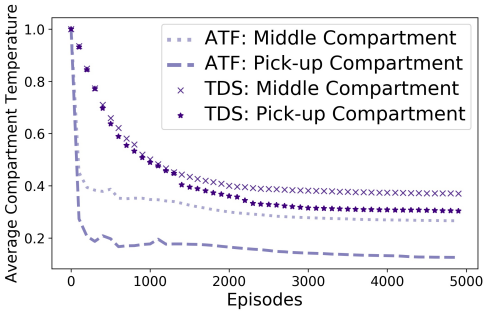


Figure 6.3: Average temperature per compartment

The success of HDDQN agents within the narrow-passage CMOTP depends on the value chosen for β. Agents with $\beta > 0.5$ struggle to coordinate, as we observed over a range of β values (exemplar given in Table 6.3). The only agents that converge upon a near optimal joint-policy are those using LDDQN-TDS and HDDQN ($\beta = 0.5$). We perform a Kolmogorov-Smirnov test with a null hypothesis that there is no significant difference between the performance metrics for agents using LDDQN-TDS and HDDQN ($\beta = 0.5$). We fail to reject the null hypothesis for average steps per episode and percentage of coordinated steps for the final 100 episodes. However, HDDQN ($\beta = 0.5$) average significantly less steps per run while maintaining less overhead, replicating previous observations regarding the strengths of hysteretic agents within deterministic environments.

	Narrow-Passage CMOTP Results			
	Hyst. $\beta = 0.5$	**Hyst. $\beta = 0.6$**	**LDDQN ATF**	**LDDQN TDS**
SPE	45.25	704.9	376.2	45.7
CSP	92%	89%	90%	92%
SPR	1,594,968	4,736,936	3,950,670	2,104,637

TABLE 6.3: Narrow-Passage CMOTP Results, including average steps per episode (SPE) over the final 100 episodes, coordinated steps percentages (CSP) over the final 100 episodes, and the average steps per training run (SPR).

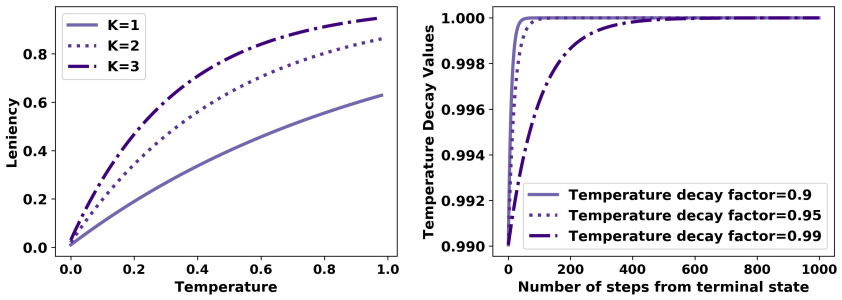
6.7 Stochastic CMOTP Results

In the stochastic setting we are interested in the percentage of runs for each algorithm that converge upon the optimal joint policy, which is for the agents to deliver the goods to dropzone 1, yielding a reward of 0.8, as opposed to dropzone 2 which only returns an average reward of 0.76. We conduct 40 runs of 5000 episodes for each algorithm.

As discussed in Section 6.6, HDDQN agents using $\beta > 0.7$ frequently fail to coordinate in the deterministic CMOTP. Therefore, setting $\beta = 0.7$ is the most likely candidate to succeed at solving the stochastic reward CMOTP for standard HDDQN architectures. However, agents using HDDQN ($\beta = 0.7$) only converge towards the optimal policy on 42.5% of runs. The *scheduled*-HDDQN perform significantly better achieving a 77.5% optimal policy rate. Furthermore, the SHDDQN performs well when an additional funnel-like narrow-passage is inserted close to the dropzones, with 93% success rate. The drop in performance upon removing the funnel suggests that the agents are led astray by the optimism applied to earlier transitions within each episode, presumably around the pickup area where a crucial decision is made regarding the direction in which the goods should be transported.

LDDQN using ϵ-Greedy exploration perform similar to SHDDQN, converging towards the optimal joint policy on 75% of runs. Meanwhile LDDQNs using $\overline{\mathcal{T}}(o)$-Greedy exploration achieve the highest percentages of optimal joint-policies, with agents converging on 100% of runs for the following configuration: $k = 3.0$, $d = 0.9$, $\xi = 0.25$ and $\mu = 0.9995$, which will be discussed in more detail below. However, the percentage of successful runs is related to the choice of hyperparameters. We therefore include an analysis of three critical hyperparameters:

- The leniency modification coefficient k, that determines the speed at which agents transition from optimist to average reward learner (sub-figure 6.4(a)). **Values: 1, 2 and 3**;
- The TDS decay-rate d which controls the rate at which temperatures are decayed n-steps prior to the terminal state (sub-figure 6.4(b)). **Values: 0.9, 0.95 and 0.99**;



(a) Leniency Schedules (b) Temperature Decay Schedules

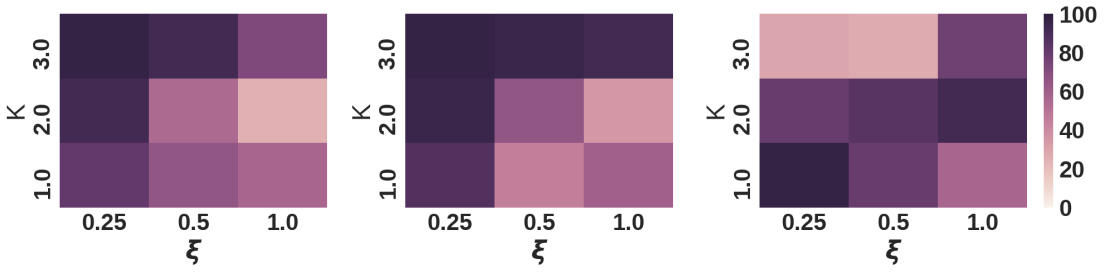
Figure 6.4: TMC and TDS schedules used during analysis.

- $\overline{\mathcal{T}}\,(o)$-Greedy exploration exponent ξ, controlling the agent's transition from explorer to exploiter, with lower values for ξ encouraging exploration. **Values: 0.25, 0.5 and 1.0**.

We conduct 40 simulation runs for each combination of the three variables. To determine how well agents using LDDQN can cope with stochastic state transitions we add a slippery surface where each action results in a random transition with 10% probability. The highest performing agents use a steep temperature decay schedule that maintains high temperatures for early transitions ($d = 0.9$ or $d = 0.95$) with temperature modification coefficients that slow down the transition from optimist to average reward learner ($k = 2$ or $k = 3$), and exploration exponents that delay the transition from explorer to exploiter ($\xi = 0.25$ or $\xi = 0.5$). This is illustrated in the heat-maps in Figure 6.5. When using a TDS with a more gradual incline ($d = 0.99$) the temperature values from earlier state transitions decay at a similar rate to those near terminal states. In this setting choosing larger values for k increases the likelihood of the agents converging upon a sub-optimal policy prior to having established the average rewards available in later states, as evident from the results plotted in sub-figure 6.5(c). Even when setting the exploration exponent ξ to 0.25 the agents prematurely transition to exploiter while holding an overoptimistic disposition towards follow-on states. Interestingly, when $k < 3$ agents often converge towards the optimal joint-policy despite setting $d = 0.99$. However, the highest percentages of optimal runs (97.5%) are achieved through combining a steep TDS ($d = 0.9$ or $d = 0.95$) with the slow transition to average reward learner ($k = 3$) and exploiter ($\xi = 0.25$). Meanwhile, the lowest percentages for all TDSs result from insufficient leniency ($k = 1$) and exploration ($\xi = 1.0$).

Using one of the best-performing configurations ($k = 3.0$, $d = 0.9$ and $\xi = 0.25$) we conduct further trials analyzing the agents' sensitivity to the maximum temperature decay coefficient μ. We conduct an additional set of 40 runs [2] where μ was increased from 0.999 to 0.9995. Combining $\overline{\mathcal{T}}\,(o)$-Greedy with the slow decaying $\mu = 0.9995$ results in the agents spending more time exploring the environment at the cost of requiring longer

[2]The runs were conducted without the slippery surface.



(a) $d = 0.9$ (b) $d = 0.95$ (c) $d = 0.99$

Figure 6.5: Analysis of the LDDQN hyperparameters. The heat-maps show the percentage of runs that converged to the optimal joint-policy (darker is better).

to converge, resulting in an additional 1,674,106 steps on average per run. However, the agents delivered the best performance, converging towards the optimal policy on 100% runs conduct.

Continuous State Space Analysis: Finally, we show that semantically similar state-action pairs can be mapped to temperature values using SimHash in conjunction with an autoencoder. We conduct experiments in a noisy version of the stochastic CMOTP, where at each time step every pixel value is multiplied by a unique coefficient drawn from a Gaussian distribution $X \sim \mathcal{N}(1.0, 0.01)$. A non-sparse tensor is used to represent the environment, with background cells set to 1.0 prior to noise being applied.

Agents using LDDQNs with xxhash converge towards the sub-optimal joint policy after the addition of noise as illustrated in Figure 6.6, with the temperature values decaying uniformly in tune with ν. LDDQN-TDS agents using an autoencoder meanwhile converged towards the optimal policy on 97.5% of runs. It is worth pointing out that the autoencoder introduces a new set of hyperparameters that require consideration, including the size D of the dense layer at the centre of the autoencoder and the dimensions k of the hash-key, raising questions regarding the influence of the granularity on the convergence. We leave this for future work.

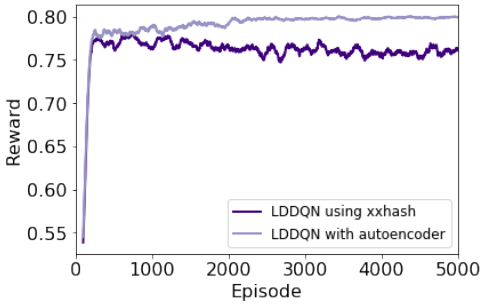


Figure 6.6: Noisy Stochastic CMOTP Average Reward

6.8 Summary

Our work demonstrates that leniency can help multi-agent deep reinforcement learning agents solve a challenging fully cooperative CMOTP using noisy and high-dimensional images as observations. Having successfully merged leniency with a Double-DQN architecture raises the question regarding how well our approach will work with other state of the art components. We have recently conducted preliminary stochastic reward CMOTP trials with agents using LDDQN with a *Prioritized Experience Replay Memory* [160]. Interestingly, the agents consistently converged towards the sub-optimal joint policy. We plan to investigate this further in future work. In addition, our research raises the question how well our extensions would perform in environments where agents receive stochastic rewards throughout the episode. To answer this question we plan to test our LDDQN within a hunter prey scenario where each episode runs for a fixed number of time-steps, with the prey being re-inserted at a random position each time it is caught [122]. Furthermore, we plan to investigate how our LDDQN responds to environments with more than two agent by conducting CMOTP and hunter-prey scenarios with four agents. Finally, the fact that leniency was originally proposed to prevent relative overgeneralization from occurring raises the research question to what extent LDDQN can mitigate this pathology within a high-dimensional temporally extended environments. We shall answer this question in the next chapter.

To summarize the contributions discussed in the current chapter:
1) We have shown how leniency can be applied to multi-agent deep reinforcement learning, enabling agents to converge upon optimal joint policies within fully-cooperative environments that require implicit coordination strategies and yield stochastic rewards.
2) We find that LDDQNs significantly outperform standard and scheduled-HDDQNs within environments that yield stochastic rewards, replicating findings from tabular settings.
3) We introduced two extensions to leniency, including a retroactive temperature decay schedule that prevents the premature decay of temperatures for state-action pairs and a $\overline{\mathcal{T}}(o)$-Greedy exploration strategy that encourages agents to remain exploratory in states with a high average temperature value.
4) Our LDDQN hyperparameter analysis revealed that the highest performing agents within stochastic reward domains use a steep temperature decay schedule that maintains high temperatures for early transitions combined with a temperature modification coefficient that slows down the transition from optimist to average reward learner, and an exploration exponent that delays the transition from explorer to exploiter.
5) We demonstrate that CMOTP [26] can be used as a benchmarking environment for multi-agent deep reinforcement learning, requiring reinforcement learning agents to learn fully-cooperative joint-policies from processing high-dimensional and noisy image observations.

6) Finally, we introduce two extensions to the CMOTP. First, we include narrow passages, allowing us to test lenient agents' ability to prevent the premature decay of temperature values. Our second extension introduces two dropzones that yield stochastic rewards, testing the agents' ability to converge towards an optimal joint-policy while receiving misleading rewards.

Chapter 7

Q-learning with Negative Update Intervals

The work presented in the first half of this chapter is in preparation for a submission to the *Journal of Machine Learning Research*, while the second half is based on the following publication:

Gregory Palmer, Rahul Savani, and Karl Tuyls, Negative Update Intervals in Deep Multi-Agent Reinforcement Learning, In Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, 2019, pp. 43–51.

In the previous chapter we evaluate the extent to which lenient (and modified hysteretic) *deep* Q-learners can overcome the *moving target problem*, *stochasticity* (with regards to rewards and transitions) and *deception* within fully-cooperative multi-agent object transportation problems. However, as we have seen in Chapter 3 (Section 3.1), the *multi-agent reinforcement learning* literature provides a rich taxonomy of learning pathologies that cooperative independent learners must overcome to converge upon an optimal joint-policy [122]. In particular, while searching for an optimal joint-policy, the actions of independent learners influence each others' search space. This can lead to *action shadowing*, where miscoordination due to sub-optimal joint-policies results in utility values of optimal actions being underestimated [50, 147]. In Chapters 4 and 5 we study the above pathologies and a type of action shadowing called *relative overgeneralization* in n-player strategic form games and low-dimensional Markov games with a small state space. Relative overgeneralization can occur when pairing an independent learner's available actions with arbitrary actions by the other agents results in a sub-optimal action having the highest utility estimate [147]. As a result, independent learners can be drawn to sub-optimal but wide peaks in the reward search space due to a greater likelihood of achieving collaboration there [147].

In our Climb Game [33, 90] evaluations in Chapter 4 we observe that relative overgeneralization presents a challenge for independent learners, even in repeated n-player

single-stage strategic-form games. We find that lenient and hysteretic Q-learners maintaining an optimistic disposition can prevent relative overgeneralization within n-player deterministic Climb Game variations [33]. However, a significant amount of hyperparameter tuning is required to increase convergence rates upon adding partially and fully-stochastic rewards. In Chapter 5 we show that lenient learners can in fact master these domains, but only after further modifying the leniency algorithm to mitigate the noise introduced by miscoordination, resulting in our *Distributed-Lenient Q-learning* (DLQ).

However, while DLQ is well suited towards overcoming the above pathologies in repeated n-player single-stage strategic-form games and low-dimensional Markov games with a small state space, in Chapter 6 we observe that compromises are necessary to scale leniency to high-dimensional domains with a large state-space, resulting in our *Lenient (Double) Deep Q-Network* (LDDQN). Our evaluations in the previous chapter focus on mitigating deep Q-learners' amplified moving-target problem in domains with stochastic rewards and transitions. Therefore, for both lenient and hysteretic deep Q-learners questions remain regarding scalability, i.e. can they overcome the same pathologies (in particular relative overgeneralization) in complex domains that suffer from the curse of dimensionality and require reasoning over long time horizons?

To answer this question we evaluate the ability of *leniency* and *hysteretic Q-Learning* to overcome the pathologies outlined in a temporally extended, partially observable version of the *Climb Game* [33, 90]. We call this game the *Apprentice Firemen Game* (AFG). Indeed, to date the majority of multi-agent deep reinforcement learning research on independent learners focuses on stochasticity and mitigating an amplified moving target problem resulting from obsolete state transitions being stored inside *experience replay memories* $\mathcal{D}$ [46, 141, 144, 221]. The AFG, meanwhile, allows us to study the robustness of independent learning algorithms simultaneously facing all of the above pathologies in a system suffering from the curse of dimensionality.

Within the AFG agents must make an irrevocable decision that will determine the outcome of an episode. We find that while hysteretic and lenient learners deliver promising performances in layouts where agents can observe each others' irrevocable decision, both algorithms converge upon sub-optimal joint policies when the same *irrevocable* decision is made in *seclusion*.

To help independent learners overcome the outlined pathologies in this challenging setting, we introduce a novel approach where agents maintain expanding intervals estimating the *min* and *max* of cumulative reward distributions for state-transition trajectories ending without miscoordination. The intervals determine which trajectories are stored and used for sampling, allowing independent learners to discard trajectories resulting in miscoordination. This reduces the impact of noisy utility values occurring in cooperative games with high punishment for uncoordinated behavior, increasing the likelihood of average utility values being established for sequences of actions leading to

coordinated outcomes. We call this approach NUI-DDQN (*Negative Update Intervals Double-DQN*).

Our main contributions in this chapter can be summarized as follows.

1) We design a new environment that simultaneously confronts independent learners with all four of the mentioned pathologies. The environment is based on the Climb Game, which has been used to study relative overgeneralization and stochastic rewards. We embed the Climb Game in a temporally-extended gridworld setting, that we call the *Apprentice Firemen Game* (AFG), in which two fireman need to coordinate to extinguish a fire. Stochastic transitions can be added by introducing randomly moving civilians who obstruct paths.

2) We empirically evaluate hysteretic and lenient approaches in two *AFG* layouts (Figure 7.4). *Layout 1* examines whether the pathologies can be overcome when independent learners can observe each other while making an irrevocable choice that determines the outcome of an episode. In contrast, *layout 2* requires independent learners to independently make the same *irrevocable* decision in *seclusion*. We find that independent learners predominately converge upon superior joint-policies in *layout 1*, providing evidence that independent learners can implicitly learn to avoid miscoordination when able to observe each other during transitions that determine an episode's outcome. *Layout 2* poses a challenge for existing approaches. Lenient learners in particular face the following dilemma: remain lenient and be led astray by misleading stochastic rewards, or estimate average utility values and succumb to relative overgeneralization due oscillating utility values caused by stochastic transitions.

3) We introduce *Q-learning with Negative Update Intervals* (Q-learning with NUI), a novel independent learning approach capable of converging on optimal joint-polices within the n-player repeated strategic-form games used for our empirical re-evaluations in Chapter 4. Furthermore, we identify a hyperparameter configuration for Q-learning with NUI that enables a high convergence rate across settings.

4) We introduce *NUI-DDQN*, a multi-agent deep reinforcement learning algorithm which discards episodes yielding cumulative rewards outside the range of expanding intervals. These intervals are maintained for sequences of transitions (trajectories) equivalent to actions from *cooperative games*. NUI-DDQN reduces the noise introduced by punishing values resulting from miscoordination to utility estimates, allowing independent learners to prevent *relative overgeneralization* and the *alter-exploration problem*. *NUI-DDQN* consistently converges upon the optimal joint-policy in both layouts for deterministic and stochastic rewards.

This chapter is structured as follows. First, we outline and evaluate Q-learning with Negative Update Intervals in Section 7.1 within the n-player strategic-form games used for our empirical evaluations conducted in Chapters 4 and 5. We identify hyperparameter settings that enable Q-learning with NUI to deliver performances comparable with Synchronized-DLQ within the Penalty Game and Climb Game variations. Following our n-layer strategic-form game evaluation we scale Q-learning with NUI to NUI-DDQN

in Section 7.3. We subsequently outline our Apprentice Firemen Game in Section 7.4, before evaluating the learning dynamics of NUI, lenient and hysteretic deep Q-learning architectures in Section 7.5. Finally, under Future Work in Section 7.6 we propose extensions to NUI-DDQN, and discuss interesting preliminary findings regarding agents learning nonverbal communication in the AFT, before concluding with our chapter summary.

7.1 Q-learning with Negative Update Intervals

We introduce *Q-learning with Negative Update Intervals* (Q-learning with NUI) within the context of repeated n-agent single-stage strategic-form team-games. We define a negative update as a Q-value update that lowers a utility value estimate, i.e. an update performed using a temporal difference error (TD-Error) $\delta < 0$. As evident from the utility value update equations for the stateless versions of hysteretic Q-learning and leniency (Equations (3.9) and (3.19) respectively in Chapter 3), both approaches reduce the extent to which negative updates lower utility value estimates over time. However, both approaches apply their update rules to negative TD-Errors indiscriminately. Meanwhile, for Q-learning with NUI we make the following assumption:

Assumption 1. *Within a number of team-games independent learners may compute a* $\delta < 0$ *due to feedback received for joint-actions that resulted in:*

i miscoordination (type 1);

ii coordinated behaviour where the utility is currently overestimated (type 2).

While this is a strong assumption, we can provide a number of examples from multi-agent reinforcement learning literature where Assumption 1 holds, including the stochastic Climb Game variations[90], as well as the Relative Overgeneralization 3 and Gradient 2 [209] games used for our evaluations in Chapters 4 and 5. Neither leniency nor hysteretic Q-learning attempt to distinguish between the two types of δ outlined above. In Chapter 5 we proposed SDLQ as a method for reducing the likelihood of independent learners performing updates using *Type 1* TD-Errors. However, while SDLQ delivers a robust performance across the various Climb Game variations in our empirical evaluation, we observe that reward spaces exist where DLQ performs sub-optimally with synchronized updates (see Section 5.4.2 in Chapter 5). Maximum based learners, meanwhile, avoid negative updates altogether by maintaining utility values for each action $a \in \mathcal{A}$ based on the highest observed reward r_a^{max}.

This raises the research question: to what extent can we implement an algorithm where the independent learners can differentiate between the two types of TD-Errors outlined above. Our aim is to prevent updates using *Type 1* TD-Errors from introducing noise, without discarding updates using *Type 2* TD-Errors, which allows us to establish an accurate average utility value estimate for coordinated behaviour involving each action.

Our approach towards answering this question is to compute intervals for each action $a \in \mathcal{A}$, where the lower endpoint is approximately the *min* reward received for coordinated behaviour involving a. Therefore, receiving a reward less than *min* on the interval indicates miscoordination has occurred. For example, for the bimatrix game outlined in Figure 7.1, due to relative over-generalization and stochastic rewards both maximum and average reward learners will be drawn towards joint actions $\langle C, C \rangle$. Meanwhile, the respective intervals for computing the average reward for coordinated behaviour involving each action would be $[10, 12]$ for action A, $[0, 15]$ for action B and $[0, 16]$ for action C. Having established such intervals we can prevent utility updates resulting from miscoordination behaviour, allowing us to compute noise free utility estimates for each action $a \in \mathcal{A}$, and thereby allow both independent learners to establish that $\langle A, A \rangle$ is the optimal joint-action. Below we formally define *negative update intervals*, and describe how they can be utilized to help independent learners prevent relative overgeneralization.

I \ II	A	B	C
A	$12/10$ $12/10$	$0/-30$ $0/-30$	$5/-5$ $5/-5$
B	$0/-30$ $0/-30$	$14/0$ $14/0$	$15/0$ $15/0$
C	$5/-5$ $5/-5$	$15/0$ $15/0$	$16/0$ $16/0$

Figure 7.1: Example of a stochastic reward game where agents are confronted with relative overgeneralization. Each reward n/m is returned with equal probability.

7.1.1 Negative Update Intervals

While agents guided by r_a^{max} are vulnerable towards stochastic rewards, we consider that for partially and fully stochastic reward spaces where r_a^{min} for coordinated outcomes is greater than the punishment received for miscoordination, there exist intervals $[r_a^{min}, r_a^{max}]$ within which negative updates to utility estimates can be performed while mitigating the noise induced through punishment for miscoordination. We show that maintaining negative update intervals for each action $a \in \mathcal{A}$ increases the likelihood of agents within repeated games converging upon an optimal joint policy $\hat{\boldsymbol{\pi}}^*$.

7.1.2 Maintaining Negative Update Intervals

We establish r_a^{max} for each action $a \in \mathcal{A}$ over n burn-in transitions, where the agents use random exploration. Initially $r_a^{min} = r_a^{max}$. During training r_a^{min} is gradually decayed by subtracting an amount χ. To prevent a premature decay during phases where independent learners are confronted with the alter-exploration problem, we only decay if the reward is large enough. That is, r_a^{min} is only decayed if $r_t \geq r_a^{max}$.

Given that r_a^{min} is monotonically decreased we consider that an interval maintained for an action a will eventually over-expand, leaving learners vulnerable towards the noise introduced by miscoordination rewards. However, assuming that each agent will have a similar estimate regarding the desirability of actions, and therefore periods of miscoordination where agents switch between most desirable actions should be kept to a minimum, we consider that over time the learners should be capable of estimating r_a^{min}. To accomplish this we maintain running mean and standard deviation estimates for the rewards received for each action:

$$\overline{r}_a^{min} \leftarrow \mu \times r + (1 - \mu) \times \overline{r}_a^{min}, \tag{7.1}$$

$$\sigma_{r_a^{min}} \leftarrow \sqrt{(1 - \mu) \times (\sigma_{r_a^{min}} + \mu \times (r - \overline{r}_a^{min})^2)}. \tag{7.2}$$

An update only takes place if reward r is greater than the *max* between r_a^{min} and the $\overline{r}_a^{min} - \sigma_{r_a^{min}} - \varepsilon$ (Equation (7.3)), where ε is an additional small valued term used to expand the interval. Therefore, while leniency is vulnerable towards miscoordination upon cooling temperature values, Q-learning with NUI will continue to discard miscoordination rewards. We outline Q-learning with NUI in Algorithm 7.

$$Q(a) \leftarrow \begin{cases} Q(a) + \alpha\delta, & \text{if } r \geq max(r_a^{min}, \overline{r}_a^{min} - \sigma_{r_a^{min}} - \varepsilon). \\ Q(a), & \text{otherwise.} \end{cases} \tag{7.3}$$

7.1.3 Strategic-Form Game Evaluation

As with the independent learning approaches evaluated in Chapters 4 and 5 we conduct a hyperparameter sweep for NUI-DDQN, aiming to identify configurations that enable a high convergence rate across settings. We evaluate all combinations of:

- $\chi = \{0.01, 0.1, 1.0\}$;
- ϵ decay rates $\{0.99, 0.995, 0.999\}$.

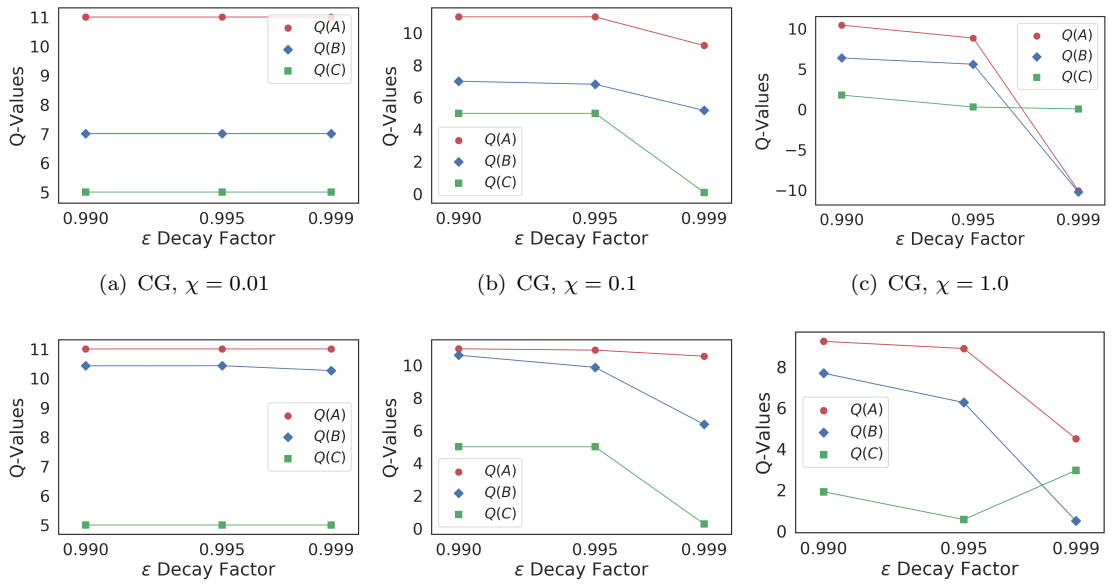
Learners use a burn-in period of 1,000 episodes where random exploration is conducted. This value is excessive for the two-agent variations of each game, but ensures the r_a^{max} is obtained for the four-agent variations. As in previous evaluations we conduct 1,000 training runs of 15,000 iterations per setting. We provide heat-maps to illustrate

Algorithm 7 Q-learning with NUI

Input: Max steps T, learning rate α, $BurnIn$, decay step χ
$Q(a) \leftarrow initialize(a)$
for $t = 0$ to T **do**
    Select a according to the ϵ-greedy selection method based on π
    Apply a and observe reward r
    Update $\overline{r}_a^{min}$ and $\sigma_{r_a^{min}}$ using equations 7.1 and 7.2
    if $r > r_a^{max}$ and $t < BurnIn$ **then**
        $r_a^{max} \leftarrow r$
        $r_a^{min} \leftarrow r$
        $Q(a) \leftarrow r$
    else if $r \geq r_a^{max}$ and $t > BurnIn$ **then**
        $r_a^{min} \leftarrow r_a^{min}$ - χ
    if $r \geq max(r_a^{min}, \overline{r}_a^{min} - \sigma_{r_a^{min}} - \varepsilon)$ **then**
        $Q(a) \leftarrow Q(a) + \alpha(r - Q(a))$

the correct run percentages for each setting in Appendix B. We identify two settings where 100% of runs converged upon the optimal joint policy (across settings): $\chi = 0.01$ combined with either an ϵ decay rate of 0.99 or 0.995. We observe a drop in the convergence rate within the four-agent variations of the *Partially* and *Fully Stochastic Climb Game* for agents using a slow ϵ decay rate of 0.999. Furthermore, we observe a decrease in the correct run percentage across games for larger $\chi = \{1.0, 0.1\}$. Therefore, similar to DLQ, Q-learning with NUI benefits from mitigating the alter-exploration problem through limiting the amount of global exploration. Meanwhile, Q-learning with NUI being more likely to converge upon a correct joint policy when implemented with slowly expanding reward intervals is somewhat equivalent to *Asynchronized DLQ* benefiting from using a lower learning rate α during *learning phase 2*.

In Figure 7.2 we use scatter plots to illustrate the average Q-value for Q-learning with NUI within the two-agent low-penalty version of the Deterministic and Partially Stochastic Climb Games. We observe that Q-value estimates reflecting coordinated behaviour for each action are more likely to be established when using the lower two decay values $\chi = 0.01$ and $\chi = 0.1$. Furthermore, for a large decay value $\chi = 1.0$ we can observe a deterioration in the Q-value estimates upon increasing the ϵ decay rate. For the (deterministic) Climb Game we observe that when using one of the optimal hyper-parameter configurations, e.g., ϵ decay rate 0.99 and $\chi = 0.01$, Q-values for each action approach the reward received for coordinated behaviour. For the Partially Stochastic Climb Game, meanwhile, we observe that the utility for B is being overestimated. However, this indicates that upon discovering that A has a higher utility estimate, the interval for B is no longer expanded. This is desirable, enabling a switch back to B, should A prove sub-optimal.



(a) CG, $\chi = 0.01$ (b) CG, $\chi = 0.1$ (c) CG, $\chi = 1.0$

(d) PSCG, $\chi = 0.01$ (e) PSCG, $\chi = 0.1$ (f) PSCG, $\chi = 1.0$

Figure 7.2: Average Q-values for Q-learning with NUI in the two-agent low-penalty Climb Game (CG) and Partially Stochastic Climb Game (PSCG).

7.1.4 Robustness Towards Noisy Transitions

Wei and Luke [209] observe that even after converging upon an optimal joint-policy, only a small likelihood of global exploration can result in *optimal policy destruction* for Lenient Multi-Agent Reinforcement Learning 2 (LMRL2). Therefore a small number of iterations ending in miscoordination, due to one of the agents taking an exploratory step, is sufficient to destabilize the learning process, and divert the learners towards a sub-optimal joint-policy. For LMRL2 optimal policy destruction typically occurs when the learners have insufficient leniency to recover from repeated miscoordination. The authors demonstrate this property using LMRL2 with a Boltzmann exploration strategy that remains exploratory. As a result a large proportion of iterations end in miscoordination, lowering the Q-value estimates for optimal actions once the learners apply insufficient leniency towards updates.

Throughout this thesis we have observed that LMRL2, SDLQ and Q-learning with NUI are capable of achieving high convergence rates within the two-agent, low-penalty versions of the Penalty Game and the three Climb Game variations. However, in domains with a large state space utility values are frequently backed up from noisy state-transition sequences. Furthermore, within real world domains learners must be robust towards noisy actions, where upon attempting to perform an action, due to disturbances in the environment and noisy observations, a different action is performed than the one intended [192]. In this sub-section we evaluate the robustness of LMRL2, SDLQ and Q-learning with NUI towards noisy actions. We pick the best hyperparameter configuration for each domain based on the performance achieved in noise free evaluations. For SDLQ and Q-learning with NUI this setting remains consistent across domains:

Q-learning with NUI:

- An ϵ decay rate of 0.99,
- decay steps $\chi = 0.01$,
- and 100 burn-in iterations;

Synchronized-DLQ:

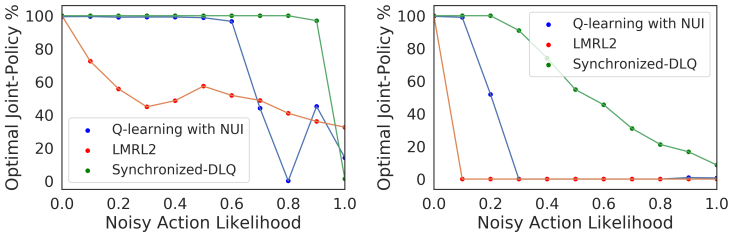
- A leniency moderation factor $k = 1.0$,
- $\alpha = 0.001$,
- and 100 burn-in iterations.

For LMRL2 we use $k = 10^7$ for the Climb Game, Partially Stochastic Climb Game and Penalty Game. For the Fully Stochastic Climb Game we set $k = 10^1$. We gather 1,000 runs for different noisy action likelihoods, as illustrated in Figure 7.3. The x-axis represents the likelihood of a noisy action occurring at each time-step. Therefore, a likelihood of 0.1 means each learner is likely to produce a noisy action every ten episodes. To provide the learners sufficient time to converge we only introduce the noisy actions after 10,000 iterations.

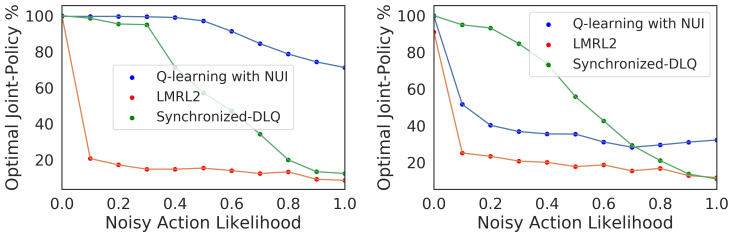
Interestingly, we find that SDLQ outperforms LMRL2 and Q-learning with NUI in all settings, with the exception of the Partially Stochastic Climb Game, and in the Fully Stochastic Climb Game when there is a high likelihood of noisy actions. We believe SDLQ's strong performance during these trials can be attributed towards the learners using a low-learning rate combined by synchronized jumps between equilibria, protecting optimal actions from sequences of miscoordination. However, we observe that even without synchronized updates Q-learning with NUI is robust towards low-amounts of noise in the majority of settings. Furthermore, while scaling SDLQ may prove challenging, we show the remainder of this chapter how Q-learning with NUI can be scaled to help multi-agent deep reinforcement learning agents mitigate the impact of noisy utility values.

7.2 Temporally-Extending Team Bimatrix Games

Negative Update Intervals DDQN (NUI-DDQN), the scaled version of Q-learning with NUI, is designed for partially observable Markov games that are *temporally-extended versions of team bimatrix games*. We call a particular roll-out of a policy π_i for an agent i, i.e. the sequence of resulting states, actions, and associated rewards, a *trajectory* and denote it by τ_i [186]. The outcome of temporally-extended versions of team bimatrix games is determined by joint-trajectories $\boldsymbol{\tau}$ resulting from $\boldsymbol{\pi}$. The reward function has inequalities mirroring those of the corresponding bimatrix game. Therefore, each τ_i belongs to a set of trajectories T^a that implements an action $a \in \mathcal{A}$. Independent learners



(a) Penalty Game (b) Climb Game



(c) Partially Stochastic Climb Game (d) Fully Stochastic Climb Game

Figure 7.3: Correct run percentage for LMRL2, Synchronized-DLQ and Q-learning with NUI within two-player low-penalty Penalty Game and Climb Game variations, where the actions taken by learners are *noisy* after 10,000 episodes.

are tasked with learning a joint policy $\boldsymbol{\pi}$ that results in optimal joint trajectories $\boldsymbol{\tau}$. Throughout this chapter τ_i refers to a trajectory that consists of all the state-transition tuples (o, a, r, o') of an individual episode for an agent. In Section 7.4, we introduce a temporally-extended version of the Climb Game, which serves as the basis for our experiments. First, we define *negative update intervals* within the context of temporally-extended versions of team bimatrix games and describe how they help NUI-DDQN prevent relative overgeneralization.

7.3 Deep Q-Learning with Negative Update Intervals

As in the previous chapter the algorithm which we outline in this section, NUI-DDQN, is an extensions of the *Double-DQN* (DDQN) introduced by Van Hasselt et al. [203]. Each agent i is implemented with a *ConvNet* trained to approximate Q-values for observation-action pairs: $Q_i : \mathcal{O}_i \times \mathcal{U}_i \rightarrow \mathbb{R}$ [104]. As in the stateless version of NUI-DDQN, outlined in Section 7.1, our aim is to compute intervals for each action $a \in \mathcal{A}$, where the lower end-point is approximately the *min* reward received for coordinated behaviour involving a. Therefore, receiving a reward less than *min* on the interval indicates miscoordination has occurred. However, in contrast to the stateless version, an action a in this context is implemented by a trajectory $\tau \in T^a$ within temporally-extended versions of team bimatrix games, as outlined above in Section 7.2.

Classifying Trajectories. The reward yielded by temporally-extended versions of team bimatrix games is determined by joint-trajectories $\boldsymbol{\tau}$. Therefore, given an oracle

$\vartheta : T \rightarrow \mathcal{A}$ capable of determining the set T^a that trajectory τ belongs to, negative update intervals $[r_a^{min}, r_a^{max}]$ can be stored for each action $a \in \mathcal{A}$, thereby increasing the likelihood of independent learners computing noise free *average* utility values for transitions belonging to coordinated joint-trajectories $\boldsymbol{\tau}$. For simplicity our evaluations use temporally-extended versions of team bimatrix games with a predefined $\mathcal{A}$. However, we discuss the potential of using a *theory of mind neural network* [154] for ϑ under Future Work (Section 7.6).

Maintaining Negative Update Intervals. We establish r_a^{max} for each action a during an initial exploration phase used to fill the experience replay memories $\mathcal{D}$. Initially $r_a^{min} = r_a^{max}$. During training r_a^{min} is gradually decayed. To prevent a premature decay during phases where independent learners are confronted with the alter-exploration problem, we only decay if the cumulative reward for the trajectory ($R^\tau = \sum_{t=0}^{|\tau|} r_t$) is large enough. That is, r_a^{min} is only decayed if $R^\tau \geq r_a^{max} - \varepsilon$, where ε is a small constant. We assume a reward space between 1.0 and -1.0, and therefore decay $r_a^{min} + 1.0$ and subsequently subtract 1.0.

Addressing Catastrophic Forgetting. Catastrophic forgetting occurs when trained networks forget how to perform previously learned tasks while learning to master a new task [57]. To allow agents to maintain Q-values for transitions belonging to less frequently observed actions a, without preventing outdated transitions from being discarded, we implement a separate experience replay memory $\mathcal{D}_a$ for each action $a \in \mathcal{A}$. Instead of storing n transitions each $\mathcal{D}_a$ stores n episodes, since traditional experience replay memories may store a significant number of obsolete transitions once independent learners become efficient at solving a task and require less steps. Episodic experience replay memories, meanwhile, are more likely to reflect the current search space. During sampling the $\mathcal{D}_a$ are concatenated.

Storing Trajectories. In addition to r_a^{min} we maintain vectors R^a, which store the most recent n cumulative rewards for each action a. A trajectory is stored *iff* the cumulative reward R^τ is greater than the *max* between r_a^{min} and the R^a's mean $\overline{R^a}$ minus the standard deviation SD_{R^a} (Equation (7.4)). Therefore, while leniency is vulnerable towards miscoordination upon cooling temperature values, NUI-DDQN will continue to discard miscoordination trajectories. We define NUI-DDQN in Algorithm 8.

$$\mathcal{D}_a = \begin{cases} \mathcal{D}_a \cup \tau & \text{if } R^\tau \geq max(r_a^{min}, \overline{R^a} - SD_{R^a}). \\ \mathcal{D}_a & \text{Otherwise.} \end{cases} \tag{7.4}$$

Algorithm 8 NUI-DDQN

Input: Number of episodes E, replay period K, max steps T
Random exploration phase (Init for $\mathcal{D}_a$, r_a^{min} and r_a^{max})
for $e = 1$ to E **do**
    $\tau = \emptyset$
    Observe o_0 and choose $a_0 \sim \pi_\theta(o_0)$
    for $t = 1$ to T **or** until an absorbing state is encountered **do**
        Observe o_t, r_t
        Store transition (o_{t-1}, u_{t-1}, r_t, o_t) in τ
        if $t \equiv 0 \mod K$ **then**
            Optimise Network
        Copy weights from time to time: $\theta'_t \leftarrow \theta_t$
        Choose $a_t \sim \pi_\theta(o_t)$
    $a \leftarrow \vartheta(\tau)$
    $R^a \leftarrow R^a \cup R^\tau$
    if $r_a^{max} < R^\tau$ **then**
        $r_a^{max} \leftarrow R^\tau$
    if $R^\tau \geq max(r_a^{min}, \overline{R^a} - SD_{R^a})$ **then**
        $\mathcal{D}_a \leftarrow \mathcal{D}_a \cup \tau$
    if $R^\tau \geq r_a^{max} - \varepsilon$ **then**
        $r_a^{min} \leftarrow decay(r_a^{min})$

7.4 The Apprentice Firemen Game

The Climb Game is often studied as a repeated game. We are interested in solving an equivalent game extended over the temporal dimension, where joint trajectories $\boldsymbol{\tau}$ result in outcomes comparable to the joint-actions from the Climb Game [33, 90] variations used during our previous empirical evaluations. We formulate a temporally-extended versions of team bimatrix game based on the Climb Game that we call the *Apprentice Firemen Game (AFG)*, where two (or more) agents located within a gridworld are tasked with locating and extinguishing fires. First, however, the agents must locate an equipment *pickup area* and choose one of the items listed in Table 7.1 below. The task is fully cooperative, i.e. both agents are required to extinguish one fire. As outlined in Table 7.1 both agents detonating an explosive device (fighting fire with fire) is the most effective combination, equivalent to the joint action $\langle A, A \rangle$ in the Climb Game. While the fire extinguisher is more effective than the fire blanket, agents choosing one run the risk of being hit by debris if the other agent triggers an explosive device, whereas the fire blanket offers protection. Therefore the fire extinguisher and fire blanket are equivalent to actions B and C respectively.

The independent learners in our evaluation are not explicitly told which actions other agents have performed. However, we hypothesize they can learn to avoid miscoordination in the AFG when able to observe each other during transitions that determine an episode's outcome, reducing the impact of optimal joint action $\langle A, A \rangle$ being a *shadowed equilibrium*. To test this hypothesis we conduct experiments using two layouts outlined

Description	Action ($a \in \mathcal{A}$)	Effectiveness	Risk
Explosive Device	A	High	High
Fire Extinguisher	B	Medium	High
Fire Blanket	C	Weak	None

Table 7.1: Apprentice Firemen Game Equipment

below (and illustrated in Figure 7.4), where equipment pickup decisions are irrevocable for the duration of each episode. At the start of an episode one randomly chosen obstacle in the main area is set on fire. Episodes end when both agents occupy cells next to the fire, upon which a terminal reward is returned. To eliminate confounding factors all non-terminal transitions yield a reward of 0. We introduce a 10,000 step limit upon observing that trained agents delay miscoordination outcomes through avoiding the fire. The agents receive a miscoordination reward of -1.0 upon reaching this limit. The action space is discrete and includes moving up, down, left, right and NO-OP. Pickup actions occur automatically upon independent learners entering an equipment cell empty handed. DDQNs perform well when receiving rewards within $[-1, 1]$, which led us to choose the reward structures listed in Figures 7.5 – 7.7 for the deterministic, partially and fully stochastic reward spaces of the AFG respectively. For stochastic transitions randomly moving civilians can be added who obstruct paths.

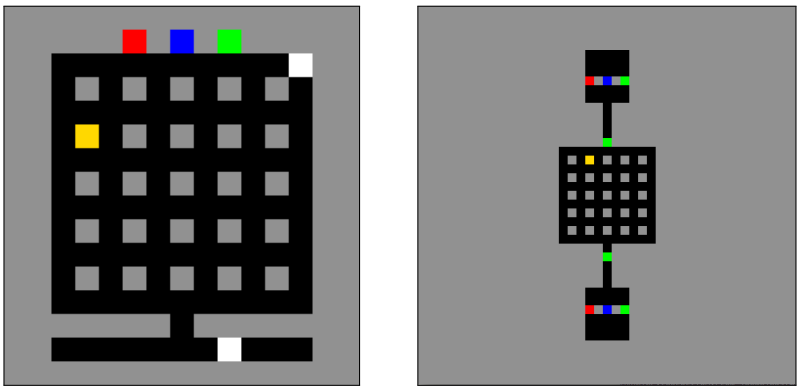
Layout 1: Observable irrevocable decisions Two agents in a 16×15 gridworld begin each episode in opposite corners of a compartment separated from the main area. The agents must exit the compartment, gather equipment from a shared *pickup area* and subsequently extinguish the fire, meaning that agents observe each other during the irrevocable equipment selection process. One agent can, therefore, observe the other agent's choice and subsequently select a best response to avoid miscoordination - in terms of the original Climb (bimatrix) game, this allows agents to act as if it was a perfect-information *commitment* version of the game with a follower and leader.

Layout 2: Irrevocable decisions in seclusion Two agents in a padded 53×53 gridworld begin each episode in separate chambers. To mimic the simultaneity of the choice of actions in the Climb (bimatrix) game, each agent is limited to 13×13 centered observations. Agents are therefore unable to observe each others' equipment selection actions.

7.5 Empirical Evaluation

7.5.1 Implementation Details

Similar to work by Leibo et al. [104] the simple visualisation of our domains helps with regards to training speeds and being able to run multiple training runs in parallel per GPU, while still posing a significant multi-agent deep reinforcement learning challenge. Therefore, given the number of cells within the main area that agents can occupy (90),



(a) Layout 1 (b) Layout 2

Figure 7.4: AFG layouts with fires (yellow), obstacles (grey) and equipment A (red), B (green) & C (blue). Firemen are initially white, but following a pickup adopt the equipment's color. Civilians are also white (Not present in the above images).

I \ II	*A*	*B*	*C*
A	0.8 0.8	−1.0 −1.0	0 0
B	−1.0 −1.0	0.6 0.6	0.5 0.5
C	0.0 0.0	0.0 0.0	0.4 0.4

Figure 7.5: Terminal rewards for the Deterministic AFG.

fire locations (25), agent color combinations (16) and actions (5) we estimate 16,020,000 state-action pairs per layout before factoring in civilians and additional layout specific cells. We therefore follow the example of recent publications by conducting evaluations in gridworlds with sufficient complexity to warrant a multi-agent deep reinforcement learning approach [66, 104, 144]. Networks consist of 2 convolutional layers with 32 and 64 kernels respectively, a fully connected layer (1024 neurons) and an output node for each action[1]. We use learning rate $\alpha = 0.0001$, discount rate $\gamma = 0.95$ and ϵ-Greedy exploration with a ϵ decay rate of 0.999. Each $\mathcal{D}$ stores 250,000 transitions. Regarding algorithm specific configurations:

NUI-DDQN. To determine the set of trajectories that τ belongs to each oracle ϑ queries the respective agent instance regarding the ID of the equipment used. Each $\mathcal{D}_a$ stores 100 episodes, while the decay rate for r_a^{min} is set to 0.995.

LDDQNs. We use a leniency moderation factor $K = 1.0$ and a *retroactive temperature decay schedule* combined with *temperature greedy exploration* strategy as described in the previous chapter.

[1]We make our code available online: `https://github.com/gjp1203/nui_in_madrl`

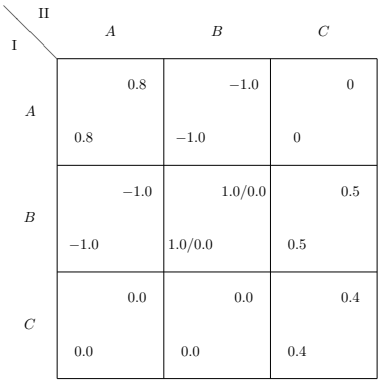


Figure 7.6: Terminal rewards for the Partially Stochastic AFG.
For $\langle B, B\rangle$ 1.0 is yielded on 60% of occasions.

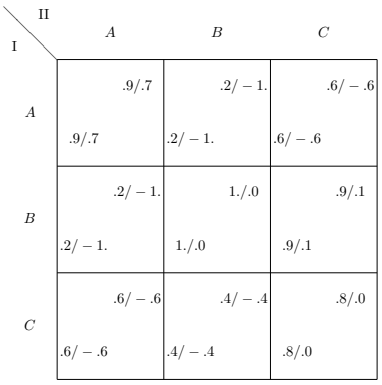


Figure 7.7: Terminal rewards for the Fully Stochastic AFG.
For $\langle B, B\rangle$ 1.0 is yielded on 60% of occasions.

7.5.2 Experiments

To evaluate our hypothesis in Section 7.4 we collect 30 training runs of 5,000 episodes per algorithm within each layout. For LDDQNs an additional 5,000 episodes are required to sufficiently decay the temperatures $\mathcal{T}(o_i, u_i)$. Finally, to evaluate the impact of stochastic transitions we introduce 10 civilians in layout 2 and conduct 30 runs of 10,000 episodes per setting.

7.5.3 Evaluation Using Phase Plots

The ternary phase plots depicted in Tables 7.2 – 7.4 provide insights regarding the learning dynamics of the agents. Each line illustrates the average shift in the trajectory distributions throughout the runs conducted, using a rolling window of 1000 episodes. The black squares at the centre of each plot represent the averaged initial T^a distributions while the red dots represents the final distribution. Each corner represents 100% of trajectories $\tau \in T^a$ for the labelled action $a \in \mathcal{A}$. For example, if both lines end with red dots in the top corner of a simplex, then the two agents are predominately producing trajectories $\tau \in T^A$, and have converged upon the optimal Nash Equilibrium $\langle A, A\rangle$. The agents have therefore learned policies where the optimal equipment is being selected from the pickup area in the AFG, as outlined in section 7.4.

Deterministic rewards. Under pathologies (Chapter 3, Section 3.1) we discuss how maximum based learners can prevent relative overgeneralization in the deterministic reward Climb Game. Similarly the phase plots for deterministic reward settings confirm that with sufficient optimism / leniency, independent learners can prevent relative overgeneralization while facing the curse of dimensionality (HDDQN $\beta = 0.5$, LDDQN and NUI-DDQN in Tables 7.2, 7.3 and 7.4). Meanwhile, HDDQN ($\beta = 0.9$) shows that agents with insufficient optimism gravitate towards the *shadow equilibrium* $\langle C, C\rangle$. Interestingly, in layout 2 with 10 civilians (Table 7.4) HDDQN ($\beta = 0.9$) agents are completing the climb steps discussed in Chapter 4, Section 4.2.2 towards $\langle B, B\rangle$, while $\beta = \{0.5, 0.7\}$ converge towards superior joint policies compared to layout 2 without civilians. Further investigation is required to establish why.

Stochastic Rewards. As evident by the phase plots in Tables 7.2 – 7.4, the optimism that helps HDDQNs prevent relative overgeneralization in the deterministic reward settings can lead agents to converge upon sub-optimal joint policies when learning from partially or fully stochastic rewards. For HDDQN ($\beta = 0.5$), for instance, we observe an increase in $\tau \in T^B$ for partially stochastic, and $\tau \in T^C$ by *Agent 2* for fully stochastic rewards. LDDQNs, meanwhile, are less vulnerable, gravitating towards optimal joint-policies despite stochastic rewards and relative overgeneralization in layout 1 and when receiving partially stochastic rewards in layout 2 with no civilians. However, LDDQNs struggle when receiving fully stochastic rewards in layout 2, and have limited success once civilians are added. NUI-DDQNs, meanwhile, predominately converge upon optimal joint-policies. When receiving partially stochastic rewards NUI-DDQNs are initially tempted by the misleading rewards received for $\langle B, B\rangle$, before converging on a joint-policy with the majority of trajectories $\tau \in T^A$. For fully stochastic rewards a slight increase in $\tau \in T^C$ can be observed.

7.5.4 Learning Best Response Policies

In Section 7.4 we proposed that independent learners should learn to avoid miscoordination trajectories within layout 1 due to observing each other during interactions with the equipment pick-up area. To compare the policies learned in layouts 1 and 2 (without civilians), we compute the average coordinated rewards $\overline{RC}$ for each training run. We compute $\overline{RC}$ using the rewards from the final 1000 episodes that did not end in miscoordination outcomes $\{\langle A, B\rangle, \langle B, A\rangle\}$. Runs with $\overline{RC} \approx 0.8$ have converged upon the optimal joint-policy, where $\langle A, A\rangle$ is the most frequently observed outcome. For the majority of settings higher $\overline{RC}$ values are achieved by agents in *layout 1*. The scatter plots in Table 7.5 provide evidence to support our hypothesis. Each marker within the scatter plots represents the $\overline{RC}$ for an individual run. To provide further clarity we sort the runs by $\overline{RC}$. We observe that HDDQN $\beta = 0.7$ and $\beta = 0.9$ converges upon a policy with $\overline{RC} \approx 0.8$ numerous times in each reward setting in layout 1, while only twice in layout 2 (HDDQN $\beta = 0.7$, Partially Stochastic & Fully Stochastic) [2]. Interestingly, we

[2]We provide additional $\overline{RC}$ scatter plots for each evaluation setting in Appendix B.

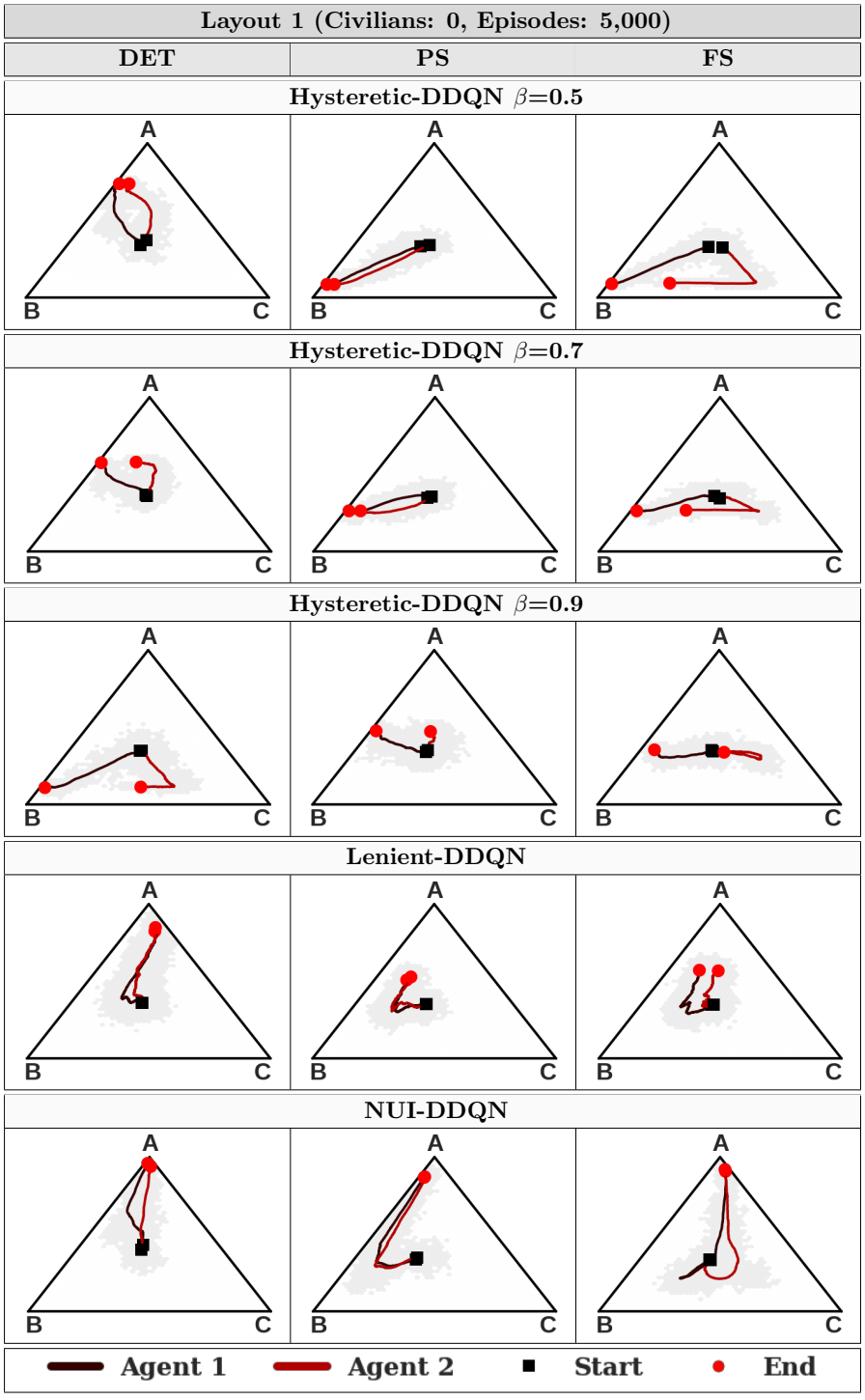


TABLE 7.2: Phase plots for runs conducted within *Layout 1* (0 civilians), illustrating the average shift in the action $\mathcal{A}$ distributions throughout the runs conducted, using a rolling window of 1,000 episodes. The black squares and red dots represent the initial and final distributions, while DET, PS and FS are abbreviations for deterministic, partially stochastic and fully stochastic rewards, respectively.

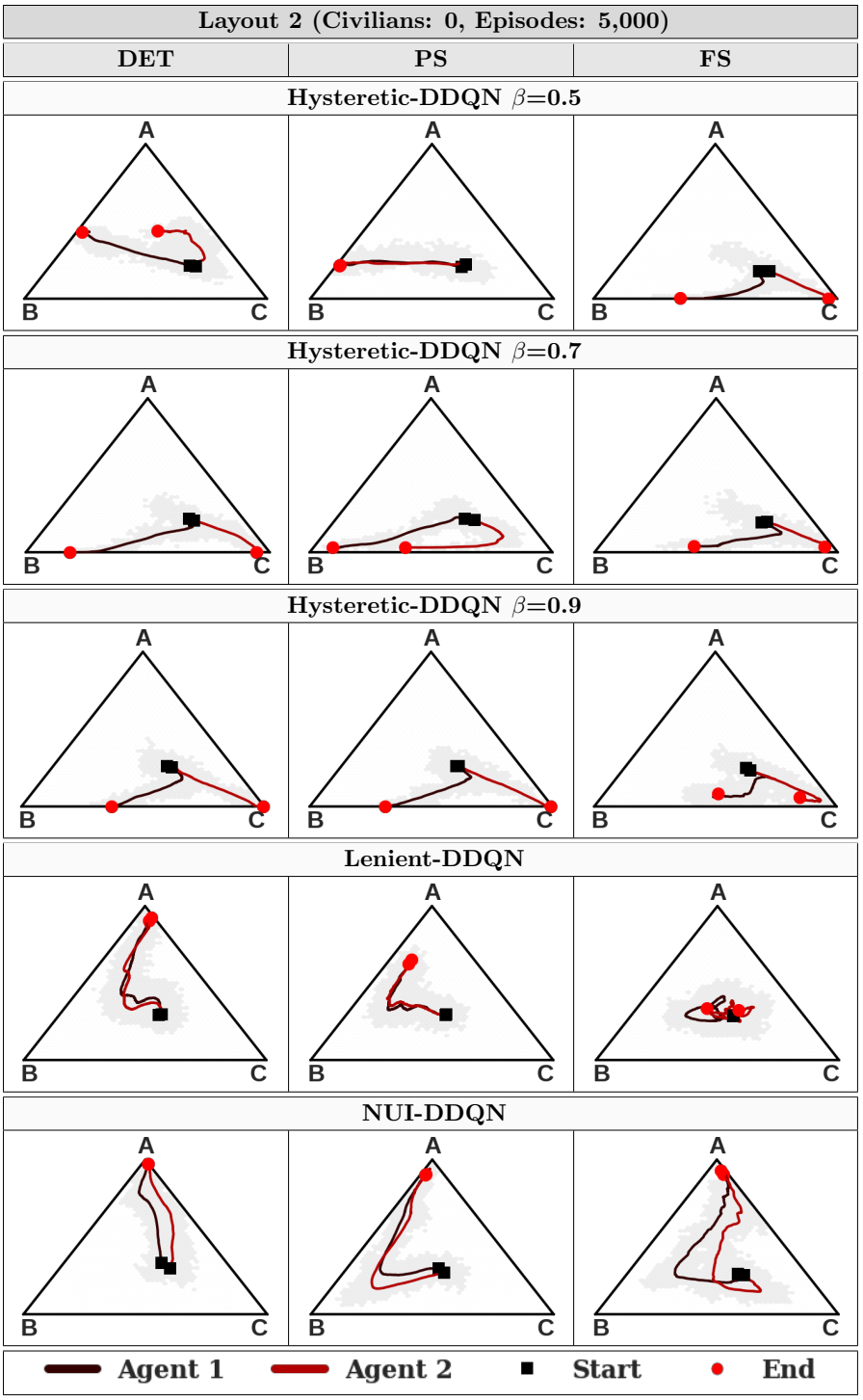


TABLE 7.3: Phase plots for runs conducted within *Layout 2* (0 civilians), illustrating the average shift in the action $\mathcal{A}$ distributions throughout the runs conducted, using a rolling window of 1,000 episodes. The black squares and red dots represent the initial and final distributions, while DET, PS and FS are abbreviations for deterministic, partially stochastic and fully stochastic rewards, respectively.

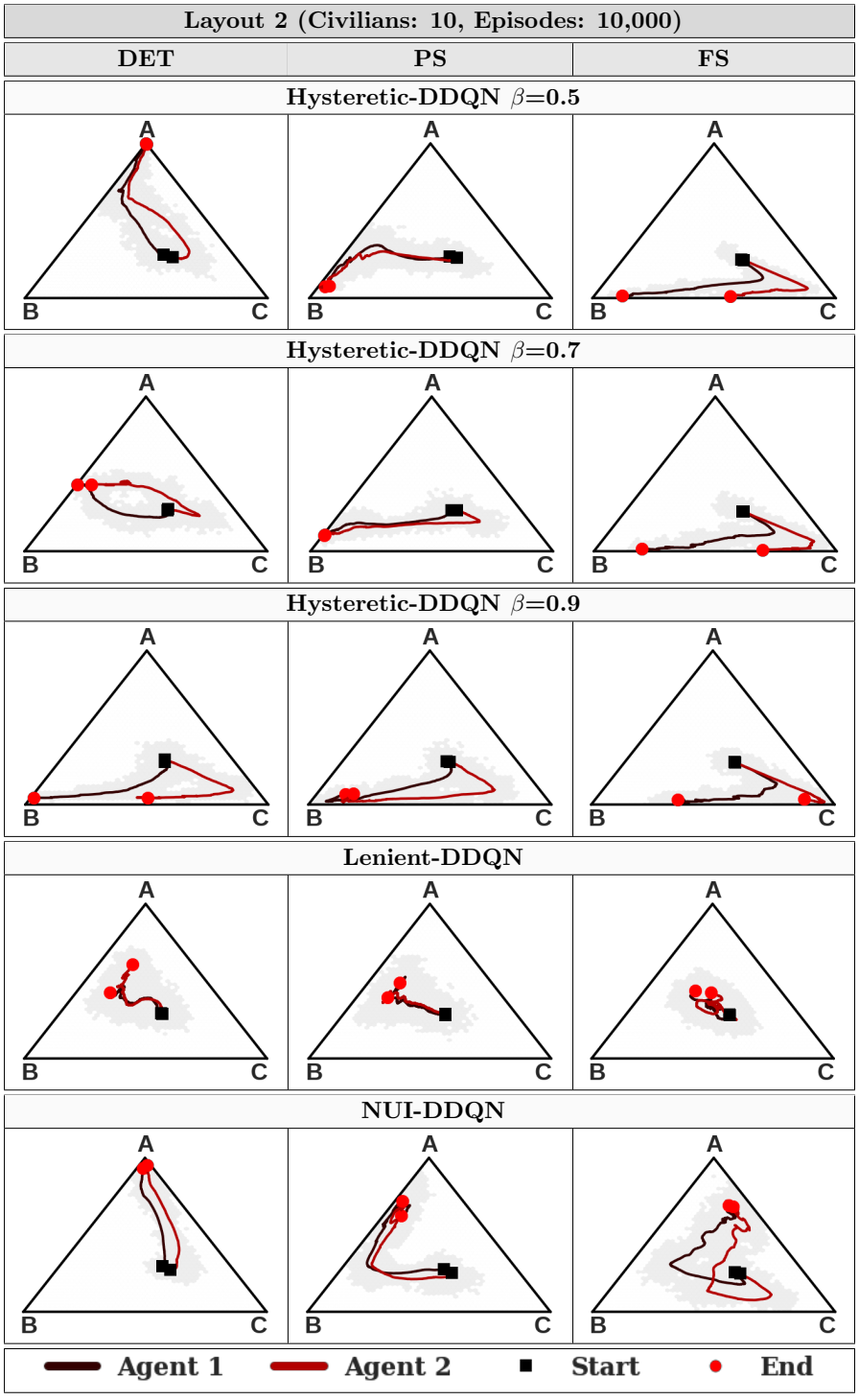


TABLE 7.4: Phase plots for runs conducted within *Layout 2* wtih 10 civilians, illustrating the average shift in the action $\mathcal{A}$ distributions throughout the runs conducted, using a rolling window of 1,000 episodes. The black squares and red dots represent the initial and final distributions, while DET, PS and FS are abbreviations for deterministic, partially stochastic and fully stochastic rewards, respectively.

find that for HDDQN ($\beta = 0.5$, Partially Stochastic) and LDDQN (Deterministic & Partially Stochastic) a larger number of runs converge upon joint-policies where $\overline{RC} \approx 0.8$ in layout 2. NUI-DDQNs, meanwhile, perform consistently when receiving deterministic and partially stochastic rewards in both settings, while a couple of runs faltered for fully stochastic rewards within layout 2. It is worth noting that even for NUI-DDQN runs with low $\overline{RC}$, $\langle A, A \rangle$ remains a frequently observed outcome.

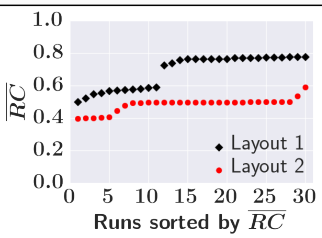
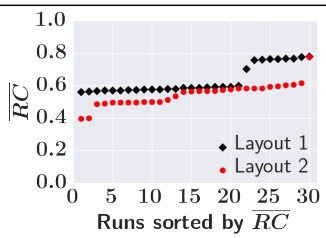
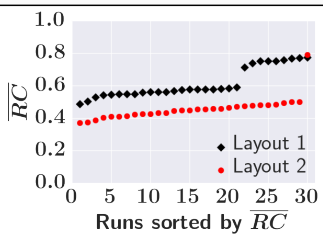
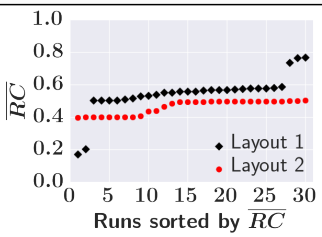
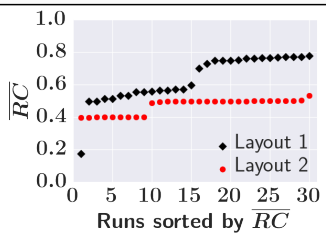
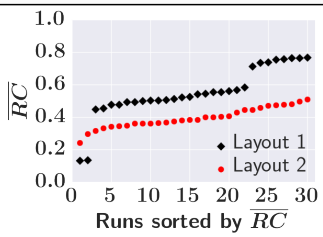
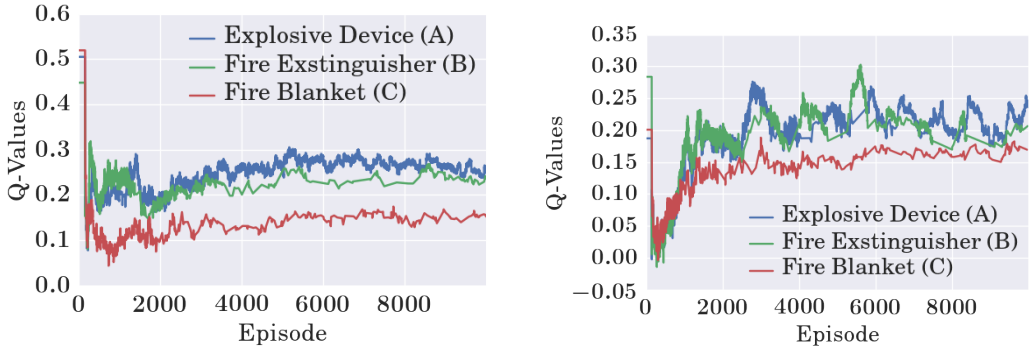
Deterministic	Partially Stochastic	Fully Stochastic
HDQN $\boldsymbol{\beta = 0.7}$		
		
HDQN $\boldsymbol{\beta = 0.9}$		
		

TABLE 7.5: Scatter plots depicting the average coordinated rewards $\overline{RC}$ for HDDQNs with $\boldsymbol{\beta = 0.7}$ and $\boldsymbol{\beta = 0.9}$.

7.5.5 Impact of Stochastic Transitions

Introducing 10 civilians to layout 2 allows us to examine the challenges faced by multi-agent deep reinforcement learners when attempting to prevent relative overgeneralization while making decisions using noisy utility values backed up from stochastic follow-on transitions. In Figure 7.8 we compare the Q-values from actions leading to the selection of equipment within both 0 and 10 civilian settings from two individual NUI-DDQN runs with partially stochastic rewards. We observe that Q-values oscillate significantly upon introducing civilians, with Q-values belonging to sub-optimal equipment B pickups frequently rising above those belonging to A. Stochastic transitions can, therefore, lead to the moving target problem (Section 3.1.5), in this case resulting in extended periods of miscoordination. However, by maintaining negative update intervals, NUI-DDQN can overcome miscoordination and revert back to a policy that generates trajectories $\tau \in T^A$.



(a) 0 Civilians

(b) 10 Civilians

Figure 7.8: NUI-DDQN Pickup Q-values

7.5.6 Considerations Regarding LDDQNs

The phase plots in Tables 7.2 – 7.4 indicate that given further hyperparameter tuning an increase in trajectories $\tau \in T^A$ should be possible for LDDQNs. However, while searching for an optimal set of hyperparameters we encountered the *optimal policy destruction* problem [209]: while LDDQNs fail to converge upon $\langle A, A \rangle$ with insufficiently decayed temperature values, rapidly decaying temperature values leaves LDDQNs vulnerable during the periods of miscoordination discussed in Section 7.5.5. We therefore choose a patient approach, with the consequence that even after 10,000 episodes the agents have still not converged. To illustrate this dilemma we conduct additional runs in a simplified partially stochastic reward version of layout 1 with only 1 fire location [3]. By varying the number of obstacles surrounding this fire, and thereby controlling the number of *Access Points* from which the agents can extinguish it, we observe the rolling percentage of $\langle A, A \rangle$ outcomes increases significantly faster when the number of $\mathcal{T}_t(o, u)$ values that need decaying decreases (See Figure 7.9). We conduct 20 runs for each *access point* setting.

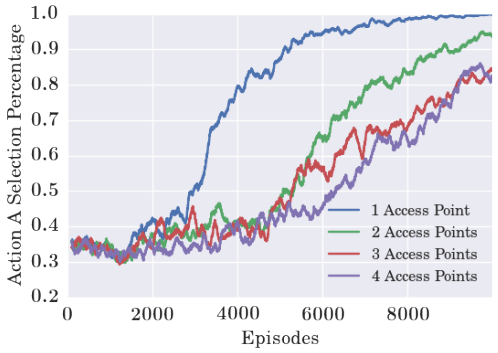


Figure 7.9: Running $\langle \boldsymbol{A}, \boldsymbol{A} \rangle$ % by LDDQNs dependent on fire *Access Points*. Agents could overlap next to the fire for *1 Access Point*.

[3] Illustrations of the layouts can be found in the Appendix B.3.

7.6 Future work

Below we discuss two interesting topics for future work in this area:
1) We conducted trials using 20 sets of fixed policies trained in a simplified layout 1, finding evidence that leader-follower dynamics emerge during training, where one agent waits to observe the other's equipment choice. We assigned the policies to disjoint sets Π^A and Π^B based on the percentage of $\tau \in T^A$ and $\tau \in T^B$ their roll-outs produced. Upon subsequently running trials with each leader-follower combination, we find half of π^B followers choose A when paired with a π^A leader and vice-versa. We are currently investigating why only some independent learners develop this adaptive ability.
2) For simplicity, the actions $a \in \mathcal{A}$ returned by the oracle ϑ for the AFG are predefined. Going forward recent work on the topic of *theory of mind* by Rabinowitz et al. [154] could pave the way for a learned oracle ϑ. The authors build a data-efficient meta-learner that learns models of the agents that it encounters. Through observing trajectories τ their resulting *theory of mind network* architecture is used to predict next-step actions, the consumption of objects within the environment and the successor representation. This opens up the possibility of applying NUI-DDQN to more complex domains where a learning approach is required to identify actions $a \in \mathcal{A}$.

7.7 Summary

Our empirical evaluation highlights the challenges multi-agent deep reinforcement learning agents must overcome to avoid converging upon sub-optimal joint policies when making decisions using noisy approximated utility estimates backed-up from stochastic follow-on state-transitions and rewards.
To summarize our contributions:
1) We presented the Apprentice Firemen Game (AFG), which is a new and challenging environment that simultaneously confronts learners with five pathologies: relative overgeneralization, stochasticity, the moving target problem, the alter exploration problem and deception.
2) We evaluate *hysteretic* [141] and *lenient* [144] learners on the AFG. While both approaches can overcome the pathologies in simpler settings, they fail when required to independently make irrevocable decisions in seclusion determining an episode's outcome.
3) Motivated by this finding we designed a new algorithm NUI-DDQN that is based on negative update intervals. Our algorithm identifies and discards episodes that end in miscoordination. In doing so, it reduces the noise introduced by the large punishments that result from miscoordination. We show that NUI-DDQN consistently converges towards the optimal joint-policy within each setting. Furthermore, we observe that the stateless version of NUI-DDQN, Q-learning with NUI, can prevent relative overgeneralization within the Fully Stochastic Climb Game, while using a hyperparameter configuration that allows for a 100% convergence rate within the remaining games used for the evaluations conducted in Chapters 4 and 5.

Chapter 8

Conclusion

This chapter concludes the thesis. We first provide a summary of the contributions discussed in each of the previous chapters in relation to the research questions formulated in Section 1.4. We subsequently discuss limitations of our work, and conclude with ideas for future research in this area.

8.1 Contributions and Answers to the Research Questions

We proposed five research questions in Section 1.4. Below we shall answer each of these questions in turn based on the findings presented in the respective chapters.

Q1: To what extent can existing independent learning approaches mitigate multi-agent learning pathologies within n-player repeated single-stage strategic-form games?

Chapter 4 & 5

In Chapter 4 we re-evaluate to what extent state of the art independent learners can overcome concurrent learning pathologies within n-player repeated single-stage strategic-form games. The concurrent learning pathologies include: miscoordination, relative overgeneralization, the alter-exploration problem and stochastic rewards. We re-evaluate the following independent learning algorithms: decentralized Q-learning [33], hysteretic Q-learning [120], FMQ [90], Recursive-FMQ [121] and Lenient Multi-Agent Reinforcement Learning 2 (LMRL2) [209]. In contrast to previous work our evaluation focuses on robustness towards scaled penalty values and an increase in the number of independent learners. Our evaluations take place within two and four-player versions of the *Penalty Game* [33], and the deterministic, partially and fully stochastic *Climb Games* [33, 90].

We identify hyperparameter configurations for decentralized Q-learning and hysteretic Q-learning that improve upon the results reported in multi-agent reinforcement learning literature [90, 121, 209]. Furthermore, our findings are in-line with the evaluation conducted by Wei and Luke [209], with LMRL2 emerging as the

most robust existing approach. However, our hyperparameter sweep shows that LMRL2 requires hyperparameter tuning to enable a high convergence rate within each of the games studied. Furthermore, we observe a drop in the convergence rate within the stochastic Climb Game variations upon increasing the number of independent learners, or increasing the scale of the penalty values. In domains suffering from relative overgeneralization and stochastic rewards we observe a deterioration of Q-values following one of the lenient learners unilaterally changing their policy. In Chapter 5 we find this to be a result of miscoordination occurring following asynchronous leniency updates, where only one of the learners lowers the utility value of the current optimal action following a transition. This can result in an alternative action having the highest estimated utility. The subsequent deterioration is caused by a lack of leniency for well explored actions combined with an increased likelihood of miscoordination due to the other agent(s) not having adjusted their policy (i.e., *the moving target problem*). As a result the learners will be unlikely to return to the previous action, even if the new action proves sub-optimal.

Q2: To what extent can *synchronized lenient learners* reduce miscoordination?

Chapters 5 & 7

In Chapter 5 we address two of LMRL2's weakness: (i) the destruction of Q-values following unilateral policy changes; (ii) the optimal policy destruction caused by the alter-exploration problem. In Section 5.1 we introduce Distributed Lenient Q-learning (DLQ) to address the alter-exploration problem. DLQ separates learning into two distinct phases. First the agents establish the maximum reward available for each action during a maximum reward learning phase; equivalent to distributed Q-learning with an uniform action selection policy. The learners subsequently switch to using *leniency* combined with a greedy action selection strategy to prevent the alter exploration problem. In addition we introduce *synchronized leniency updates* to mitigate the moving target problem. We therefore distinguish between Synchronized-DLQ (SDLQ) and Asynchronized-DLQ (ADLQ). In Section 5.2 we empirically show that SDLQ learners mitigate miscoordination through switching between equilibria during the same time-steps. This allows SDLQ to achieve state of the art performances within the *Fully Stochastic Climb Game*. Furthermore, in Section 7.1.4 we observe that SDLQ is robust towards noisy transitions where independent learners execute an unintended action with a probability. We also find that ADLQ outperforms LMRL2 within the majority of fully and partially stochastic Climb Game settings.

In Section 5.3 we scale DLQ to low-dimensional Markov games, introducing a staggered transition from explorer to exploiter and optimistic to average reward learner. We evaluate the full version of DLQ within two Markov games that have

proven challenging for LMRL2: *Relative Overgeneralization 3* (RO3) and *Gradient 2* games [209]. Interestingly, we identify hyperparameter configurations that significantly improve upon the convergence rate previously reported for LMRL2 within RO3 [209]. However, while LMRL2 is sensitive towards the choice of leniency moderation factors within RO3, both types of DLQ deliver convergence rates above 98.5% across settings.

Finally, we evaluate both SDLQ and ADLQ within Gradient 2, a domain where LMRL2 is capable of achieving a high convergence rates on a correct joint-polices, but due to the number of states struggles to find complete policies, where learners behave correctly in each state of the domain. Interestingly, one of the states in Gradient 2 presents a challenge for synchronized leniency updates, due to each action combination resulting in the same potential max reward, and only two actions being available to each agent. However, the rewards in this state are stochastic, meaning the learners are in fact being confronted with miscoordination. We find that, if SDLQ agents lock onto a suboptimal action combination, then the learners are destined to switch between joint-policies that result in miscoordination as the synchronized leniency updates are applied. In contrast ADLQ can break this cycle. Furthermore, a staggered exploration strategy allows ADLQ to converge upon a significantly higher number of complete polices compared to LMRL2.

Q3: To what extent can we design high-dimensional domains for evaluating the susceptibility of deep reinforcement learners towards multi-agent learning pathologies?

Chapters 6 & 7

In Chapters 6 and 7 we introduce two high-dimensional domains with a large state-space for evaluating the susceptibility of multi-agent deep reinforcement learning algorithms towards concurrent learning pathologies:

1.) CMOTP Extensions: In Section 6.5.1 we extend the Coordinated Multi-Agent Object Transportation Problems (CMOTPs) [26], which requires two agents to deliver one item of goods to a drop-zone within a grid-world. The task is fully-cooperative, meaning to shift the goods both agents must move in the same direction. However, first the agents must exit a separate compartment one by one and locate the goods. The learners receive a sparse positive reward upon placing the goods inside the drop-zone. To add to the challenge the learners' observations consist of a bird's eye view of the environment. Therefore, independent learners must first learn to distinguish themselves from their team-mate using the pixel values. We introduce two extensions to the CMOTP: narrow-passages and multiple (deceptive) dropzones that yield stochastic rewards. We find the narrow-passages requiring thousands of transitions for independent learners to reach the drop-zone when using random exploration. Therefore, even when using a large experience

replay memory $\mathcal{D}$, deep reinforcement learners may only store a hand-full of state-transitions with a positive reward. We find that independent learners benefit from maintaining an optimistic disposition in this domain (see Section 6.6). However, within the stochastic reward CMOTP overoptimistic learners can be led astray by stochastic rewards. A slippery surface can be added to the domain to introduce stochastic transitions, a common practice in grid-world domains. Therefore, our CMOTP variations present independent learners with the pathologies of deception, miscoordination, the moving-target problem, the alter-exploration problem and stochasticity.

2) The Apprentice Firemen Game: In Section 7.4 we introduce the *Apprentice Firemen Game*, a temporally-extended version of the team bimatrix game the *Climb Game* [33, 90]. In the AFG two agents located within a grid-world are tasked with locating and extinguishing fires. As with the CMOTP, the AFG is fully cooperative, i.e. two agents are required to extinguish one fire. First, however, the agents must locate an equipment *pickup area* and each choose an item for extinguishing a fire. However, not all items are compatible, resulting in outcomes comparable to the joint-actions from the Climb Game [33, 90]. Therefore, the AFG confronts independent learners with the same pathologies as the CMOTP, with the addition of relative overgeneralization. For additional stochasticity we add civilians to the grid-world environment, which obstruct the learners paths towards the fires.

For single agent deep reinforcement learning the *Arcade Learning Environment* [12] and the OpenAI Gym [21] have established themselves as the most frequently used suits for benchmarking algorithms. The more challenging domains found within these bench-marking suits provide a means for assessing the performance of new approaches when faced with known pathologies. The multi-agent literature meanwhile provides a rich taxonomy of multi-agent learning pathologies. As we have discussed, traditionally multi-agent reinforcement learning pathologies have been studied within the context of strategic-form and stochastic games. Meanwhile, our domains provide a means for evaluating the ability of multi-agent deep reinforcement learning agents to overcome concurrent learning pathologies within high-dimensional domains with a large state-space. Furthermore, through making our CMOTP extensions [144] and the Apprentice Firemen Game (AFG) [143] publicly available [1], our environments have recently been recognised as belonging to a growing list of open source benchmarking environments for multi-agent deep reinforcement learning [54, 79, 80, 110].

Q4: To what extent can *leniency* be scaled to multi-agent deep reinforcement learning?

Chapter 6

[1] `https://github.com/gjp1203/nui_in_madrl`

In Chapter 6 we introduce the *Lenient (Double) Deep Q-Network* (LDDQN), demonstrating that leniency can be scaled to *deep* multi-agent reinforcement learning. In Section 6.7 we show that LDDQN is more likely to converge on correct joint-policies than Hysteretic DDQNs (HDDQNS) within the stochastic reward CMOTP. As with DLQ, we find that staggering the temperature decay to prevent premature temperature cooling helps LDDQNs converge upon optimal joint-policies. Furthermore, we observe that learners benefit from increased exploration within initial states until the average rewards have been established in follow-on states. To accomplish this we introduced two extensions to leniency: (i) a retroactive temperature decay schedule to prevent the premature decay of temperatures for state-action pairs; (ii) a $\overline{\mathcal{T}}(o_t)$-Greedy exploration strategy that allows agents to remain exploratory in states with a high average temperature value. Our LDDQN hyperparameter analysis reveals that the highest performing agents within the stochastic reward CMOTP use a steep temperature decay schedule that maintains high temperatures for early transitions combined with a temperature modification coefficient that slows down the transition from optimist to average reward learner, and an exploration exponent that delays the transition from explorer to exploiter.

Q5: To what extent can independent learners overcome relative overgeneralization while making decisions using noisy utility values backed up from stochastic follow-on transitions?

Chapter 7

In Chapter 7 we turn to the *Apprentice Firemen Game* (AFG) to evaluate to what extent LDDQN can overcome relative overgeneralization within a multi-agent deep reinforcement learning context. However, while LDDQNs are more robust than HDDQNs within this setting, upon increasing the stochasticity the learning dynamics of LDDQNs are less consistent. We hypothesize that stochasticity in this setting is causing *optimal policy destruction* [209], where oscillating Q-Values can cause the moving target problem, a pathology towards which LDDQN is vulnerable upon decaying the temperature values. We introduce (deep) Q-learning with Negative Update Intervals (NUI-DDQN) as a means to mitigate the noise induced by relative overgeneralization and stochasticity on utility values. NUI-DDQN is designed for temporally-extended versions of team bimatrix games. We show that NUI-DDQN is capable of identifying and discarding episodes that end in miscoordination within the AFG. In doing so, NUI-DDQN reduces the noise introduced by the large punishments that result from miscoordination. We show that NUI-DDQN consistently converges towards the optimal joint-policy within each setting. Furthermore, we introduce a stateless version of NUI-DDQN capable of converging on optimal joint-polices within all the n-player repeated strategic-form games used for our empirical re-evaluations in Chapter 4. Similar

to SDLQ we achieve this convergence rate across setting with only one hyperparameter configuration. However, in contrast to SDLQ the learners do not rely on synchronized leniency updates.

8.2 Summarising

This thesis contributes towards a better understanding of the challenges faced by independent learners within temporally extended high-dimensional domains that require a multi-agent deep reinforcement learning approach. Furthermore, our work in Chapters 4 and 5 provides valuable insights regarding the extent to which the moving-target and alter-exploration problems prevent lenient learners from consistent convergence upon correct joint-policies within repeated n-player single-stage strategic-form games suffering from relative overgeneralization and stochastic rewards. Our contributions include a number of novel independent learning approaches to address these challenges, namely *(Synchronized) Distributed Lenient Q-learning* (Chapter 5), *Lenient (Double) DQN* (Section 6.3), *Scheduled Hysteretic (Double) DQN* (Section 6.4), *Q-Learning with Negative Update Intervals* (Section 7.1) and *Negative Update Intervals DDQN* (Section 7.3). In our empirical evaluations we find that each approach improves on the previous state of the art for a number of multi-agent reinforcement learning challenges. However, while significant progress has been made, we are unable to identify an approach that represents a *silver bullet*. Indeed, as discussed throughout this thesis, a compromise is often necessary to enable independent learners to overcome multi-agent learning pathologies within different settings. In the final section of this thesis we consider the limitations of our algorithms, and how these can be addressed in future work.

8.3 Limitations and Future Work

While we introduce novel algorithms towards addressing independent learning pathologies, we find each approach works only under specific conditions. Synchronized-DLQ relies on the synchronized property, which is difficult to enforce within real world domains [91]. Furthermore, in Section 5.4.2 we identify a reward space where synchronized updates result in independent learners switching between sub-optimal policies. Further considerations are required to address this issue. Through introducing a retroactive *temperature decay schedule* (TDS) that prevents premature temperature cooling, and a $\overline{\mathcal{T}}\,(o_t)$-Greedy exploration strategy, our Lenient (Double) DQN approximately replicates the staggered temperature decay used by DLQ. However, we find LDDQN is vulnerable towards relative overgeneralization in domains with increased stochasticity, as evident from our results in Section 7.5.6. Therefore, improving the robustness of DLQ within Markov games, and subsequently scaling the approach to multi-agent deep reinforcement learning presents an interesting challenge for future research.

Q-learning with Negative Update Intervals is currently designed for the set of n-player strategic-form games where Assumption 1 holds. Meanwhile, the scaled version, NUI-DDQN, is limited to temporally-extended versions of team bimatrix games. As discussed in Section 7.6, we feel that the next step for this algorithm is to add extensions capable of identifying the higher level actions that an agent's policy implemented throughout the course of an episode, allowing the learners to maintaining negative update intervals for different types of state-transition trajectories. Furthermore, NUI-DDQN is designed for environments that confront independent learners with relative overgeneralization. We consider that the algorithm is likely to struggle in domains with coordination challenges. For instance, little is to be gained from using NUI-DDQN within the narrow-passage CMOTP (Section 6.5.1), where, due to there only being one type of trajectory, the learners default to (Double) DQNs. Implementing hybrid approaches for domains that include both of these challenges could represent an interesting topic for future work in this area, e.g., combining negative update interval with a lenient loss function.

We consider that within high-dimensional temporally extended domains there exists significant scope for visualizing and interpreting the policies learned by the independent learners. We hypothesize that methods for visualizing and understanding deep reinforcement learning agents provide a means through which to gain insights into the extent to which independent learners are aware of each other, e.g., while independent learning agents are not explicitly aware of each others' actions, they do have the potential to implicitly infer the actions taken by other agents through limited observations. For example, Mordatch and Abbeel [131] observe that decentralized learning agents incapable of explicit communication often learn to communicate via cues.

Recently there have been interesting breakthroughs in the area of visualizing and understanding deep reinforcement learning agents, for instance, by using saliency maps [58, 132]. Via these maps salient features within the environment can be identified that determine an agent's actions. Much can be learned from these saliency maps within a multi-agent deep reinforcement learning context. We have begun our own attempts at visualizing the policies learnt by agents within the *Apprentice Firemen Game* with a shared pickup area. We have interesting preliminary findings, where the saliency maps of learners that we class as followers (who wait for the other agent to make their selection) show higher saliency scores towards the coordinates of the other agent prior to their equipment selection, compared to after the pick-up task has been completed. Upon completing the pick-up task meanwhile we observe an increase in the saliency scores towards the fires within the environment. These preliminary findings raise interesting questions. For instance, can we predict if two agents who have converged upon an optimal joint-policy will remain optimal when separated and paired with different teammates (who have potentially learned a different set of cues)? We hypothesize that this methodology may path a way for further studying the non-verbal communication behaviours discussed by Mordatch and Abbeel [131], and look forward to continuing this line of work.

A further issue that needs addressing within multi-agent reinforcement learning is a more centralized approach towards maintaining scores from benchmarking. This in particular is highlighted by the extensive hyperparameter search conducted by Wei and Luke [209], and the fact that 7 years prior Matignon et al. [122] already found a more optimal set of hyperparameters within the Partially Stochastic Climb Game for decentralized and hysteretic Q-learners. Given that over the coming years there is likely to be an explosion in the number of domains for evaluating different approaches (and paradigms) of multi-agent deep reinforcement learning, efforts are required to ensure that work is not repeated. The community could, for instance, benefit from following the examples of computer vision and single agent deep reinforcement learning, where leader-boards are maintained for standard benchmarking tasks.

We observe that our work has already inspired researchers to investigate the advantages of applying leniency to multi-agent deep reinforcement learning. Zheng et al. [221] introduce a Weighted Double-DQN that makes use of a lenient reward network along with a scheduled replay strategy to improve the convergence rate within stochastic cooperative environments. Gong et al. [54] combined our retroactive temperature decay schedule, $\overline{\mathcal{T}}\,(o_t)$-Greedy exploration strategy and leniency augmented experience replay memory tuples with synchronous n-step methods (advantage actor-critic) to strike a balance between using potentially obsolete state transitions during training, proposing a *lenient ERM-helped synchronous n-step deep Q-network* (LESnDQN), finding that LESnDQN is more sample efficient on the CMOTP. Finally, Lu and Amato [110] propose a distributional reinforcement learning approach towards improving decentralized hysteretic deep reinforcement learning's vulnerability towards stochastic rewards. A time difference likelihood (TDL) measure is used to guide the choice of learning rate for each update, thereby controlling the amount of optimism applied by the learners. The TDL enables the learners to estimate if the transition occurred with an exploratory teammate, and if therefore a lower learning rate should be applied. The authors find their resulting *Likelihood Hysteretic Implicit Quantile Network* (IH-IQN) to be easier to tune and more sample efficient than LDQN within the CMOTP variations and a meeting-in-a-grid task.

Despite the limitations discussed in this section, our work introduces a number of extensions to the start of the art of multi-agent (deep) reinforcement learning research. We propose a multitude of approaches towards mitigating multi-agent learning pathologies, while also visualizing why these methods have an advantage over previous approaches. While we have been unable to provide a silver bullet toward multi-agent (deep) reinforcement learning pathologies, we hope that the material in this thesis paves the way for future research in this vast area, in particular in the exciting young field of multi-agent *deep* reinforcement learning.

Appendix A

Strategic-Form Game Results

In this section we provide visualisations for all hyperparameter configurations used during our n-player repeated strategic form game evaluations of LMRL2, LRML3, Q-learning with NUI, hysteretic Q-learning, FMQ and RFMQ in Chapters 4, 5 and 7. As in previous sections we provide heat-maps illustrating the correct run percentage for each hyperparameter configuration for each algorithm. For each plot we provide the algorithm, game, penalty scaling factor (PSF), and the number of agents.

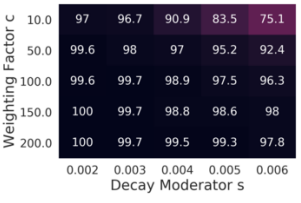
A.1 Frequency Maximum Q-value

(a) Agents: 2, PSF: Low

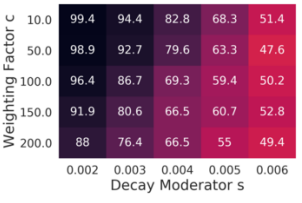
(b) Agents: 2, PSF: Medium

(c) Agents: 2, PSF: High

(d) Agents: 4, PSF: Low

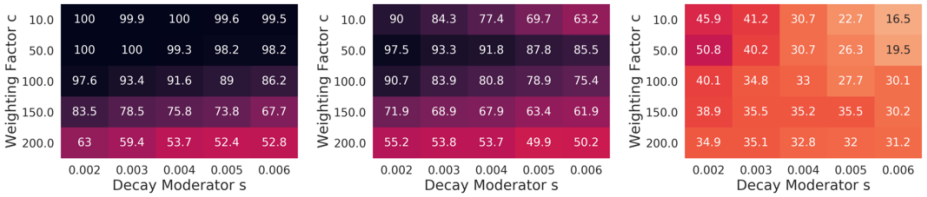


(e) Agents: 4, PSF: Medium

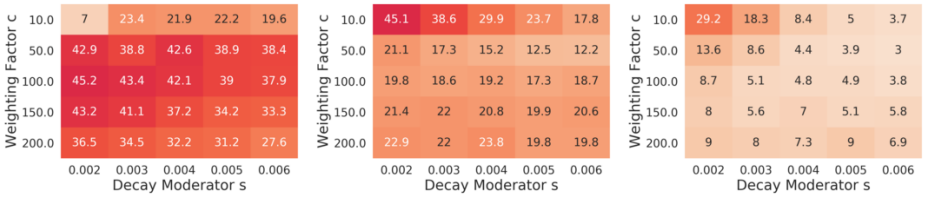


(f) Agents: 4, PSF: High

Figure A.1: Algorithm: FMQ, Game: The Penalty Game

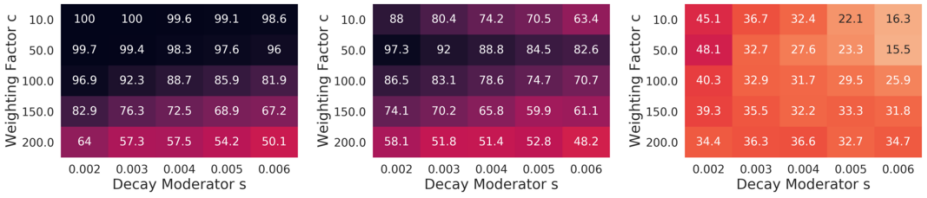


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

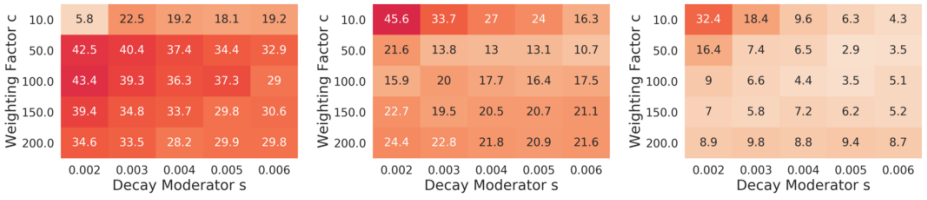


(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.2: Algorithm: FMQ, Game: The Climb Game

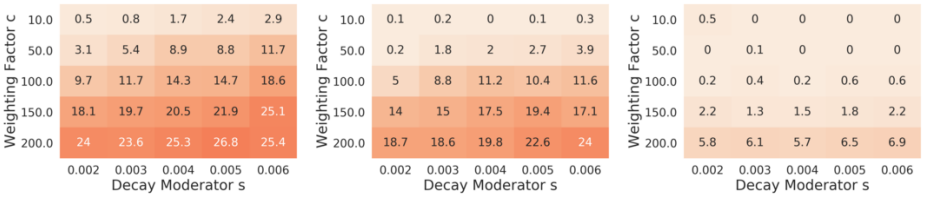


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High



(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.3: Algorithm: FMQ, Game: The Partially Stochastic Climb Game

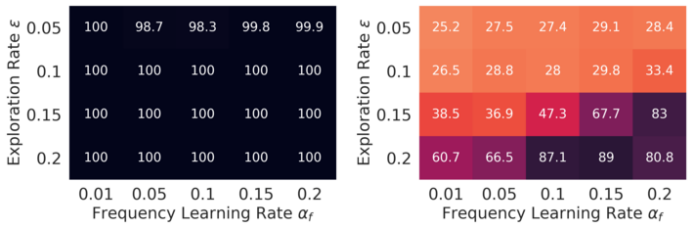


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

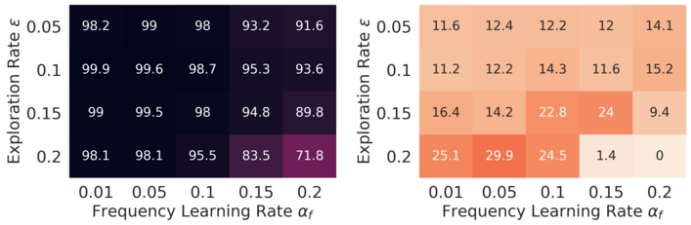
Figure A.4: Algorithm: FMQ, Game: The Fully Stochastic Climb Game

A.2 Recursive Frequency Maximum Q-value



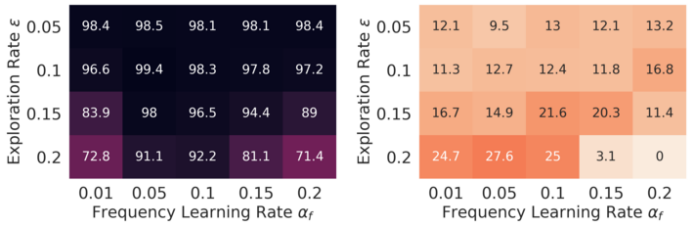
(a) PG, Agents: 2

(b) PG, Agents: 4



(c) CG, Agents: 2

(d) CG, Agents: 4



(e) PSCG, Agents: 2

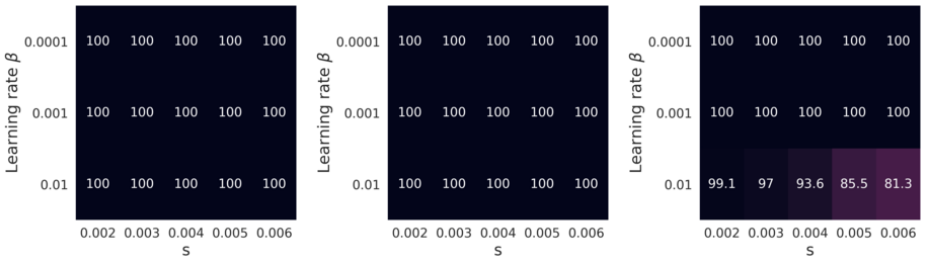
(f) PSCG, Agents: 4

(g) FSCG, Agents: 2

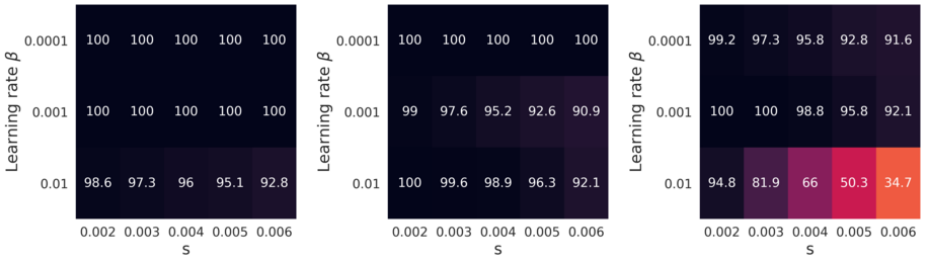
(h) FSCG, Agents: 4

Figure A.5: RFMQ results for the Penalty Game (PG), Climb Game (CG), Partially Stochastic Climb Game (PSCG), and Fully Stochastic Climb Game (FSCG)

A.3 Hysteretic Q-learning

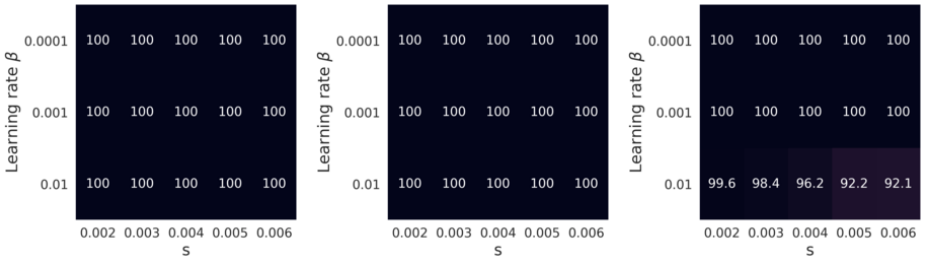


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

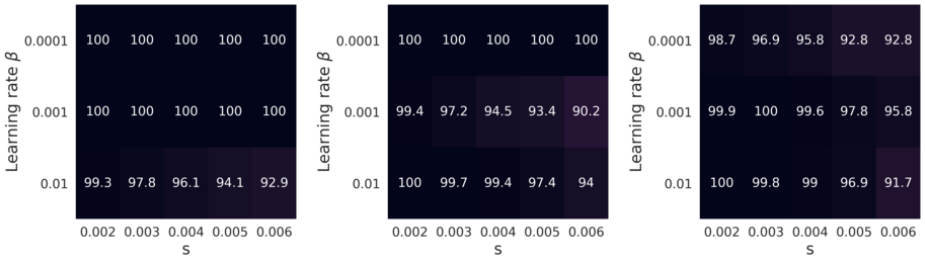


(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.6: Algorithm: Hysteretic Q-learning, Game: The Penalty Game, Exploration: Boltzmann, $MaxTemp = 50$

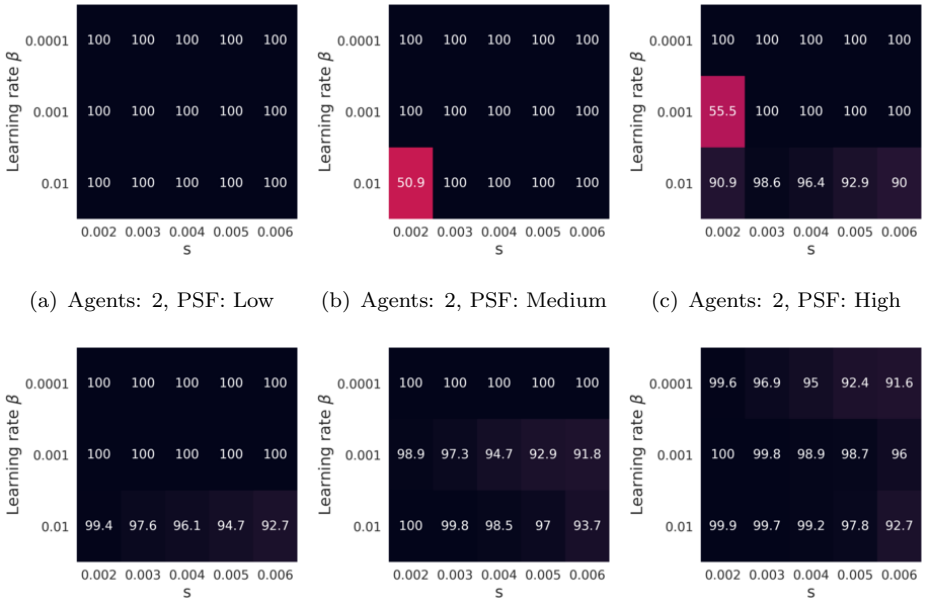


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High



(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

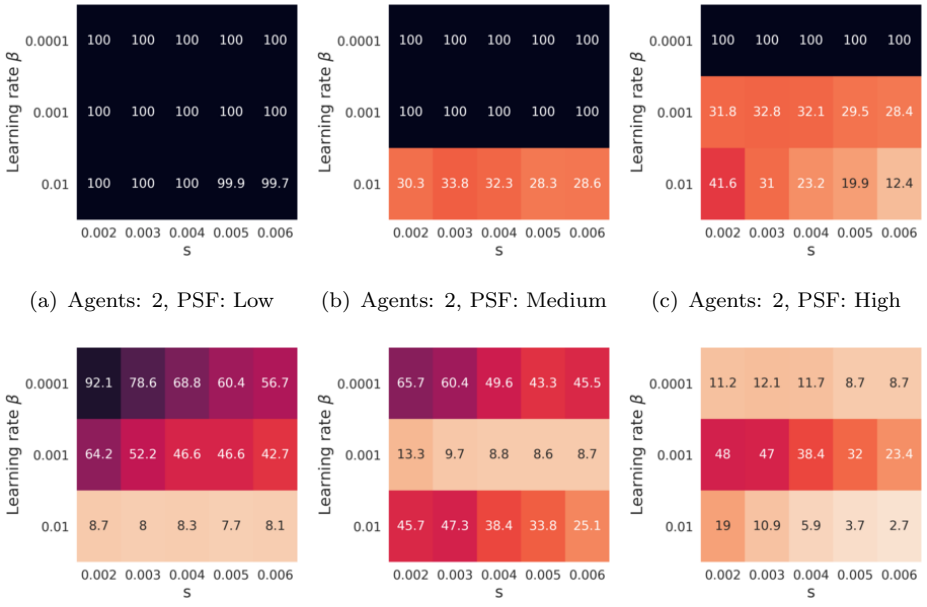
Figure A.7: Algorithm: Hysteretic Q-learning, Game: The Penalty Game, Exploration: Boltzmann, $MaxTemp = 500$



(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

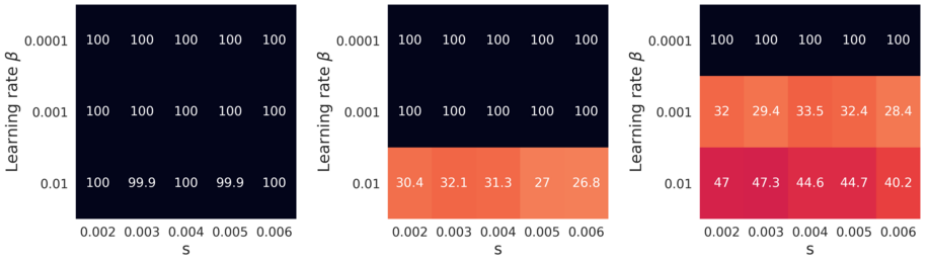
Figure A.8: Algorithm: Hysteretic Q-learning, Game: The Penalty Game, Exploration: Boltzmann, $MaxTemp = 5000$



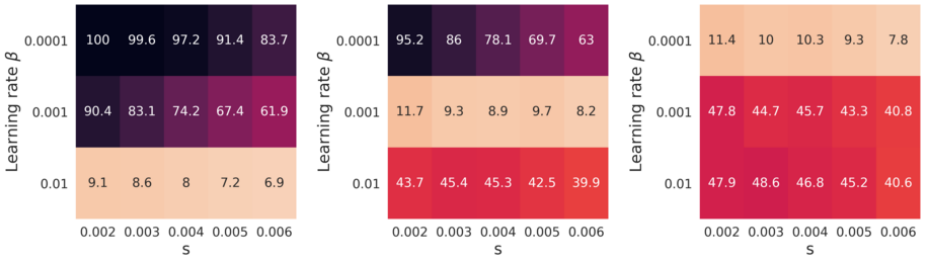
(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.9: Algorithm: Hysteretic Q-learning, Game: The Climb Game, Exploration: Boltzmann, $MaxTemp = 50$

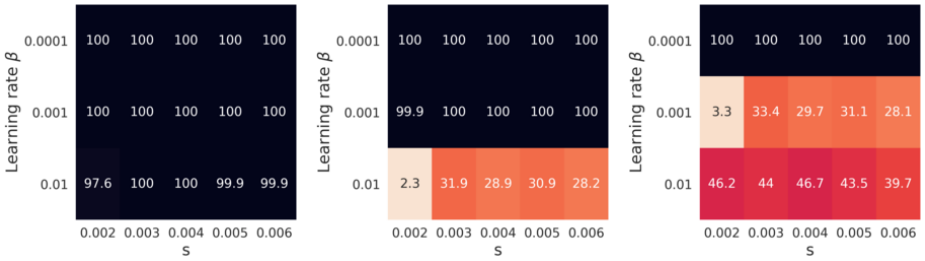


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

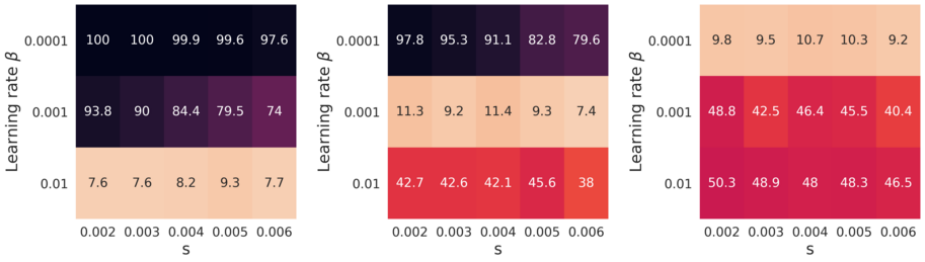


(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.10: Algorithm: Hysteretic Q-learning, Game: The Climb Game, Exploration: Boltzmann, $MaxTemp = 500$

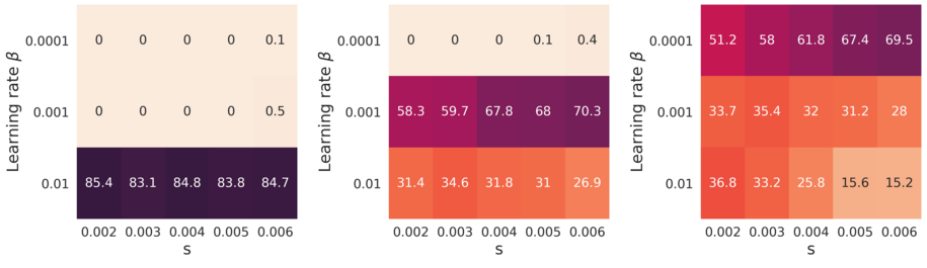


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

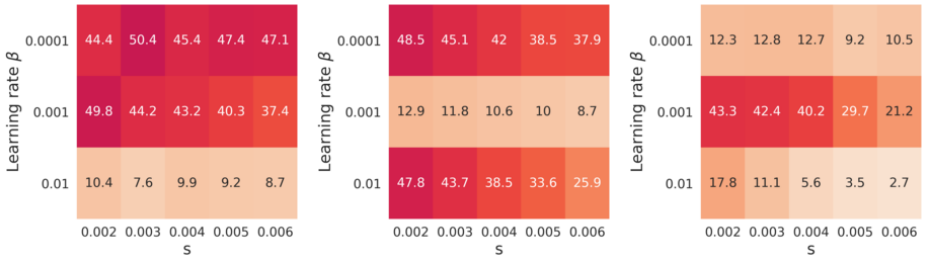


(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.11: Algorithm: Hysteretic Q-learning, Game: The Climb Game, Exploration: Boltzmann, $MaxTemp = 5000$

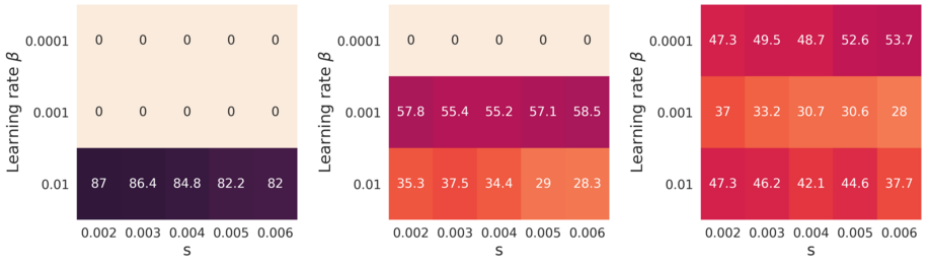


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

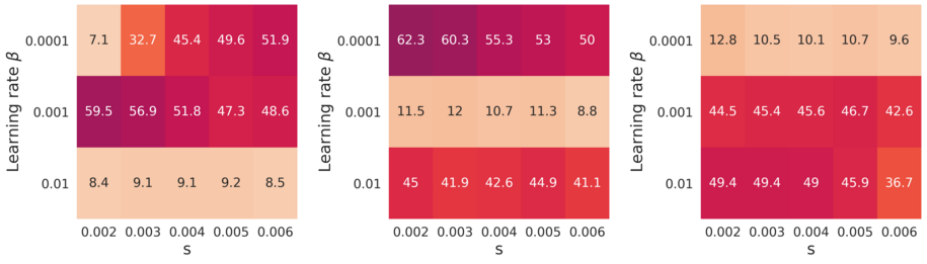


(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.12: Algorithm: Hysteretic Q-learning, Game: The Partially Stochastic Climb Game, Exploration: Boltzmann, $MaxTemp = 50$

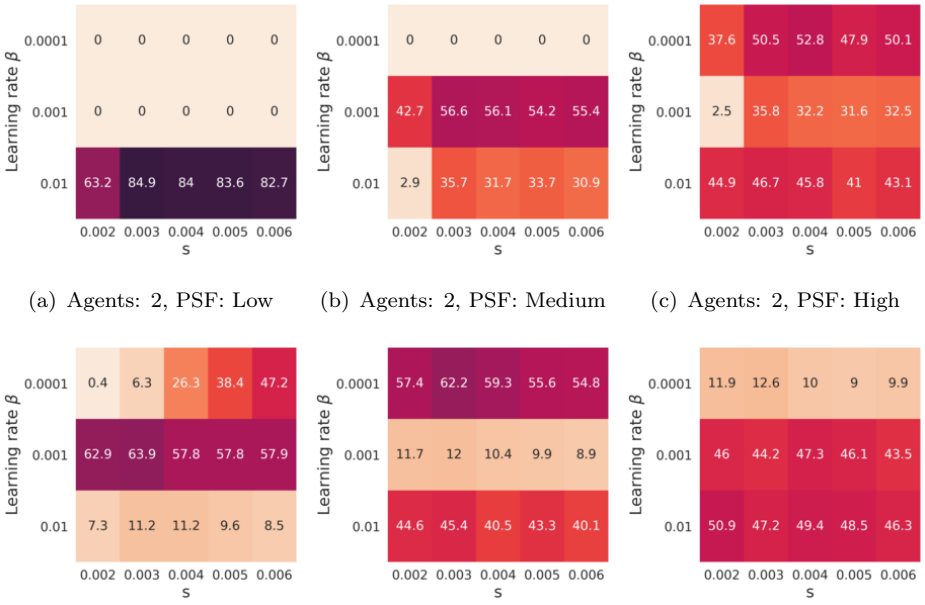


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High



(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

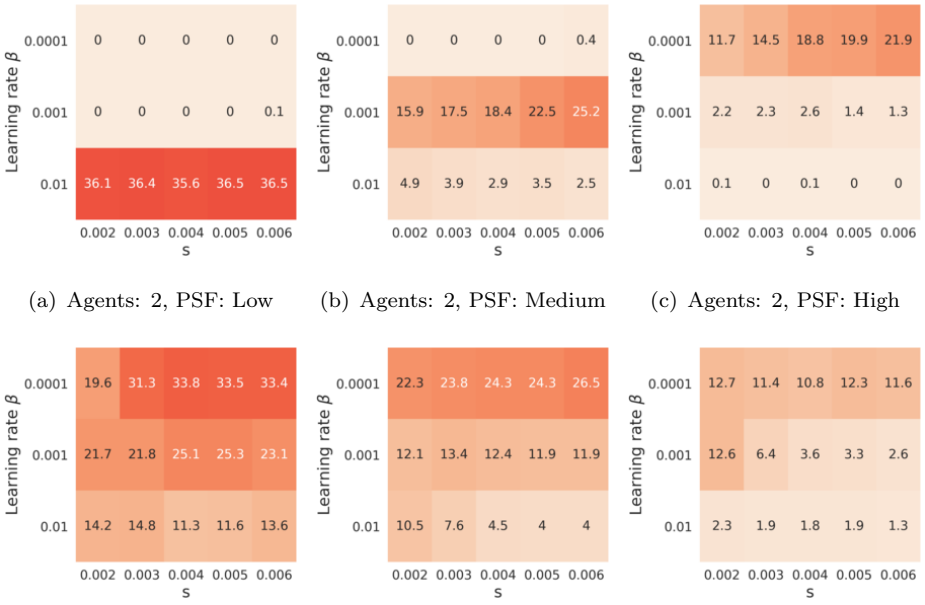
Figure A.13: Algorithm: Hysteretic Q-learning, Game: The Partially Stochastic Climb Game, Exploration: Boltzmann, $MaxTemp = 500$



(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

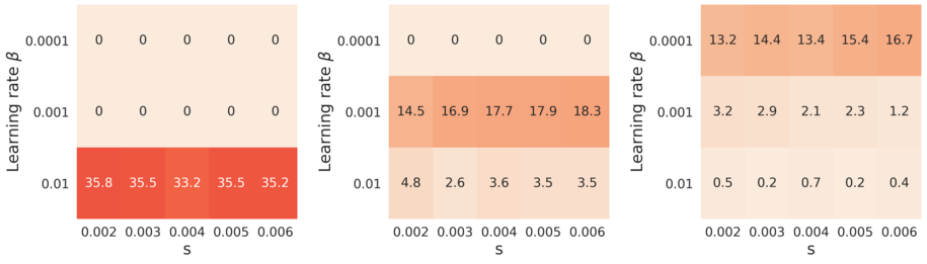
Figure A.14: Algorithm: Hysteretic Q-learning, Game: The Partially Stochastic Climb Game, Exploration: Boltzmann, $MaxTemp = 5000$



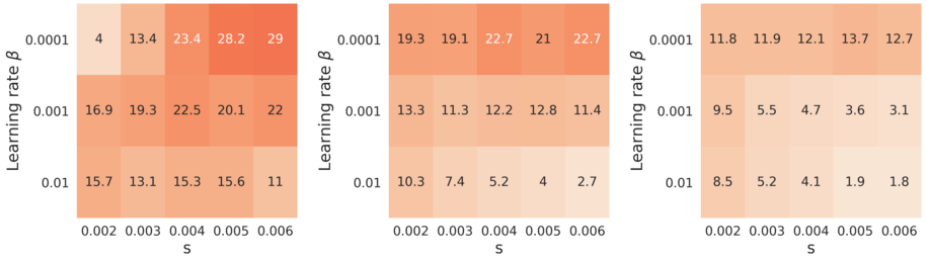
(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.15: Algorithm: Hysteretic Q-learning, Game: The Fully Stochastic Climb Game, Exploration: Boltzmann, $MaxTemp = 50$

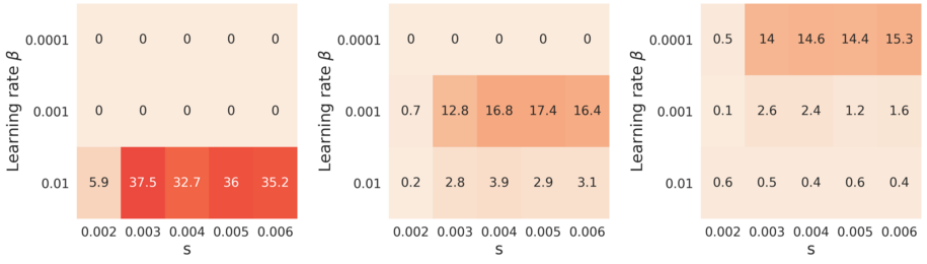


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

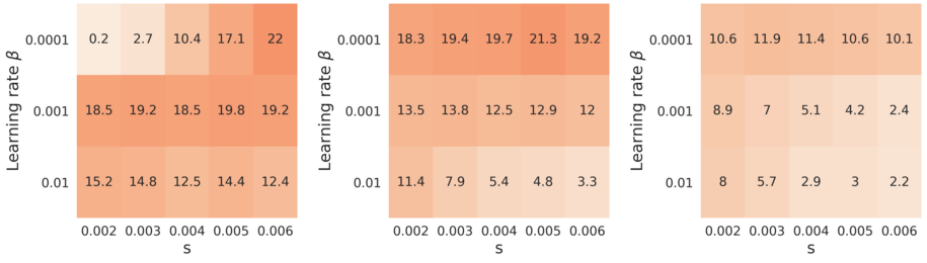


(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.16: Algorithm: Hysteretic Q-learning, Game: The Fully Stochastic Climb Game, Exploration: Boltzmann, $MaxTemp = 500$



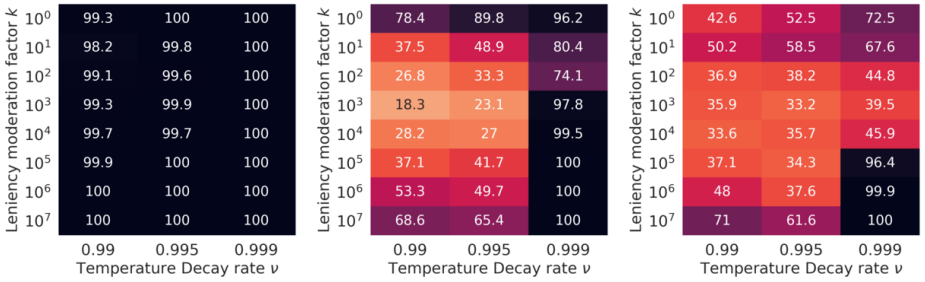
(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High



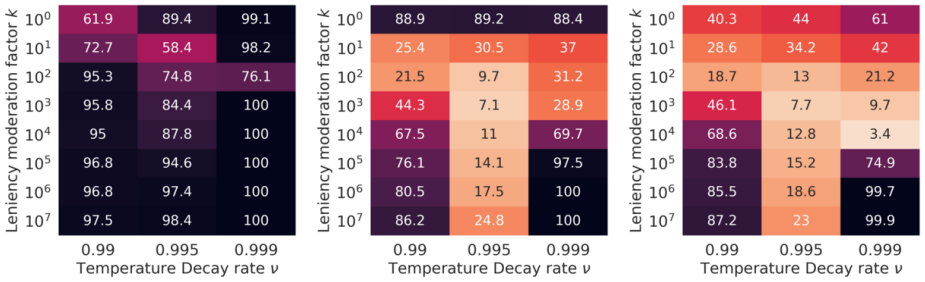
(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.17: Algorithm: Hysteretic Q-learning, Game: The Fully Stochastic Climb Game, Exploration: Boltzmann, $MaxTemp = 5000$

A.4 Lenient Multi-Agent Reinforcement Learning

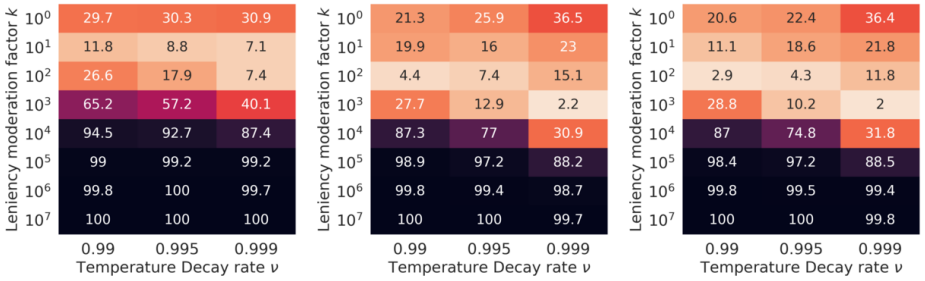


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

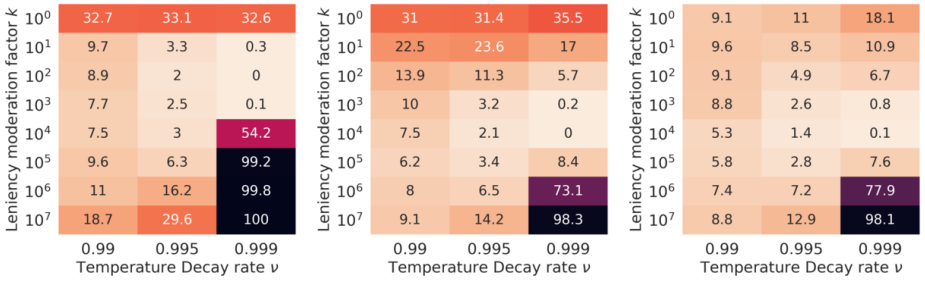


(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.18: Algorithm: LMRL2, Game: The Penalty Game

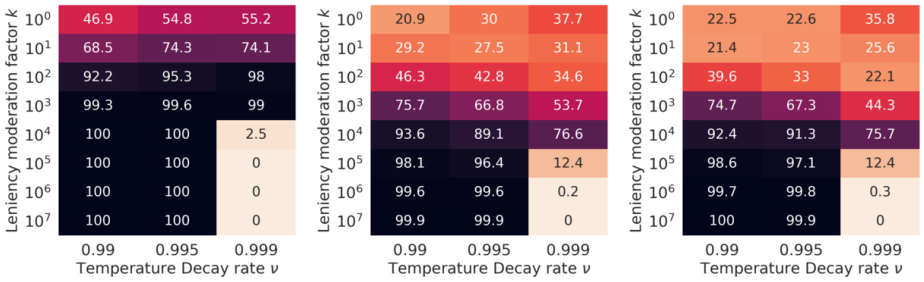


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

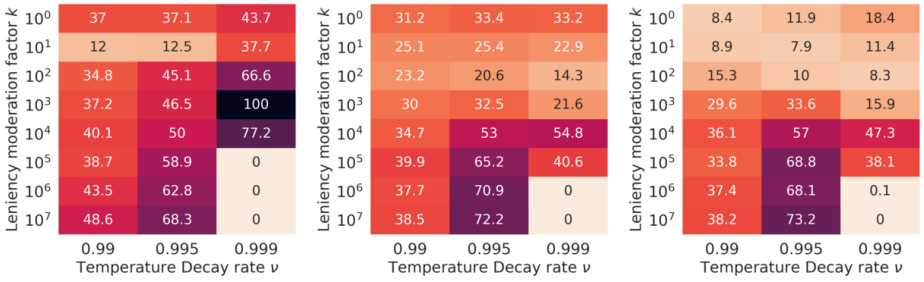


(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.19: Algorithm: LMRL2, Game: The Climb Game

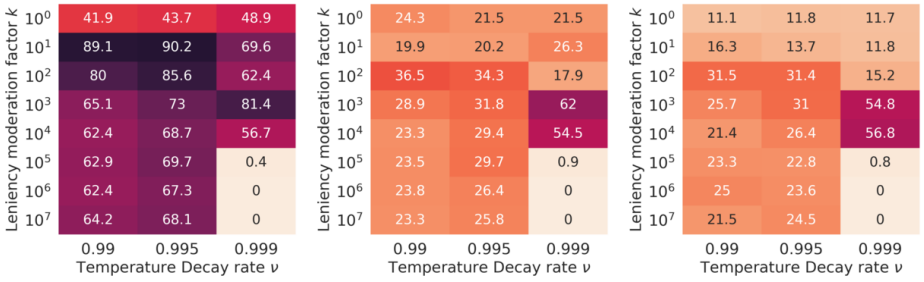


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

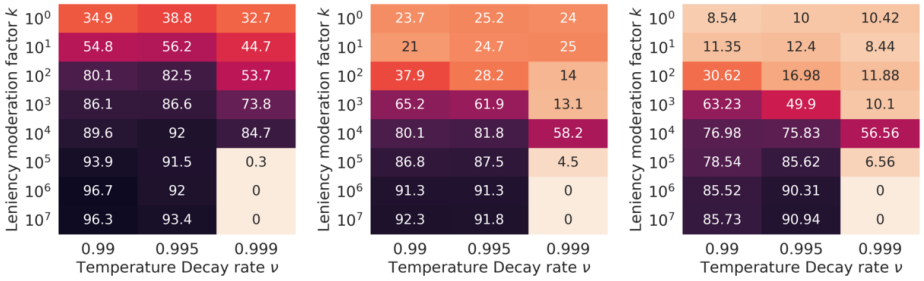


(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.20: Algorithm: LMRL2, Game: The Partially Stochastic Climb Game



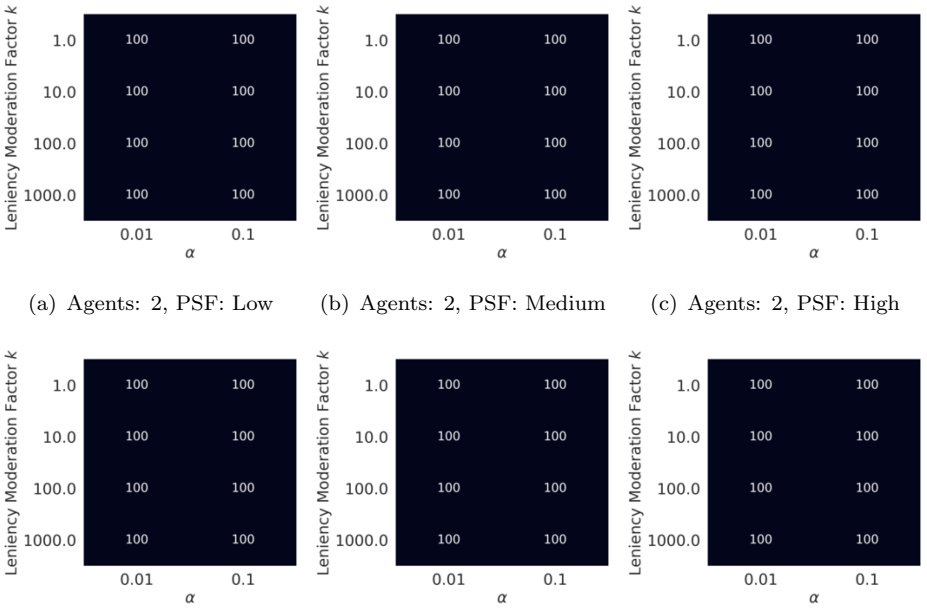
(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High



(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.21: Algorithm: LMRL2, Game: The Fully Stochastic Climb Game

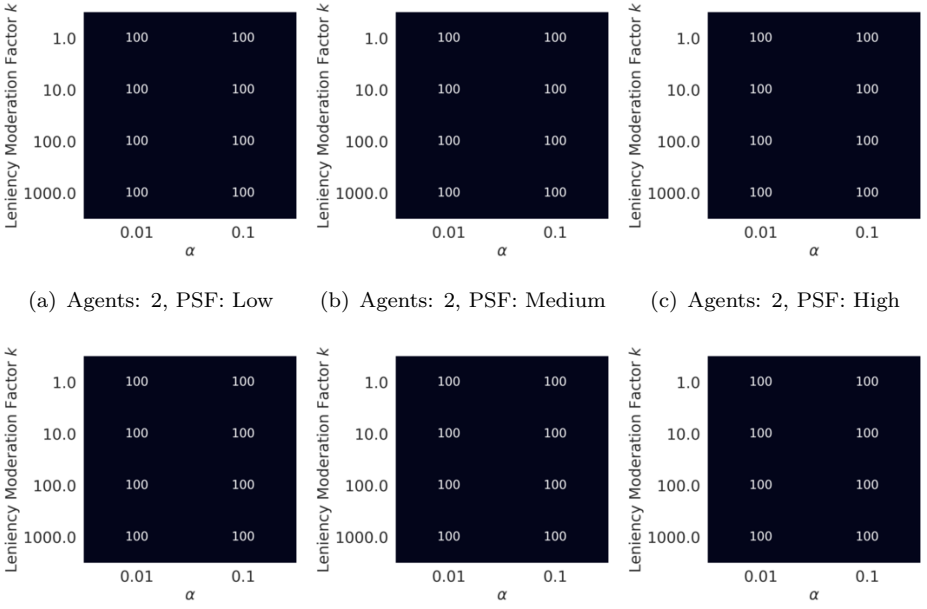
A.5 Synchronized Distributed-Lenient Q-learning



(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

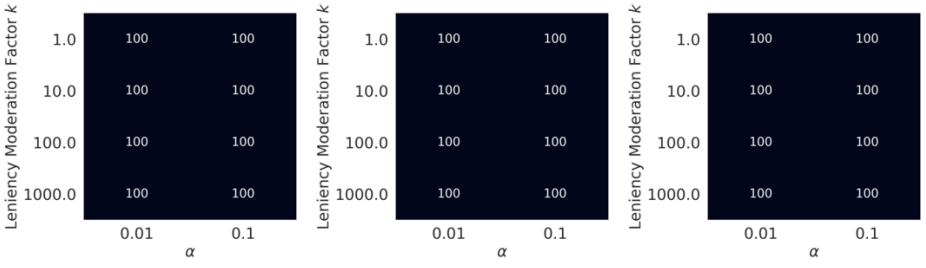
Figure A.22: Algorithm: SDLQ, Game: The Penalty Game



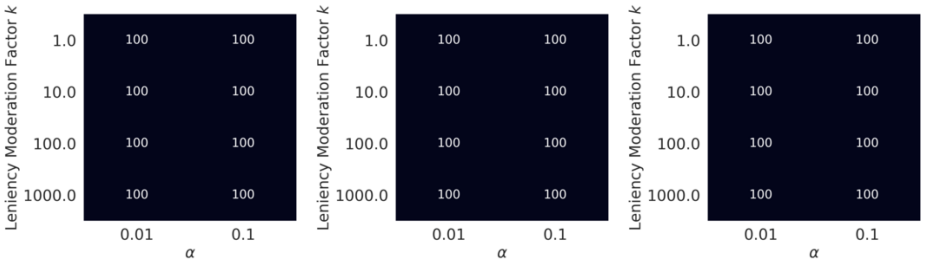
(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.23: Algorithm: SDLQ, Game: The Climb Game

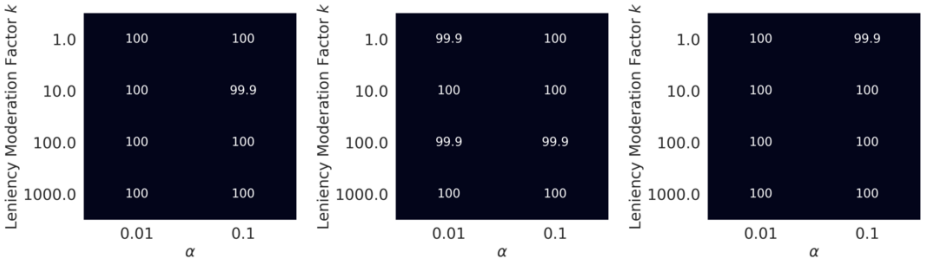


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

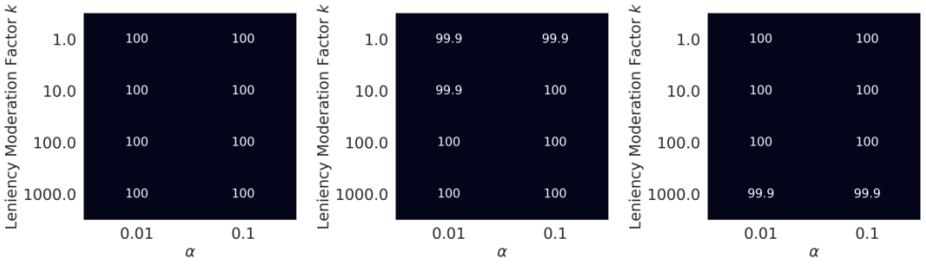


(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.24: Algorithm: SDLQ, Game: The Partially Stochastic Climb Game



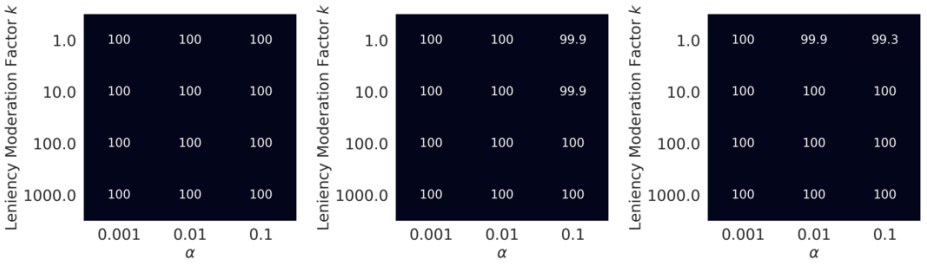
(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High



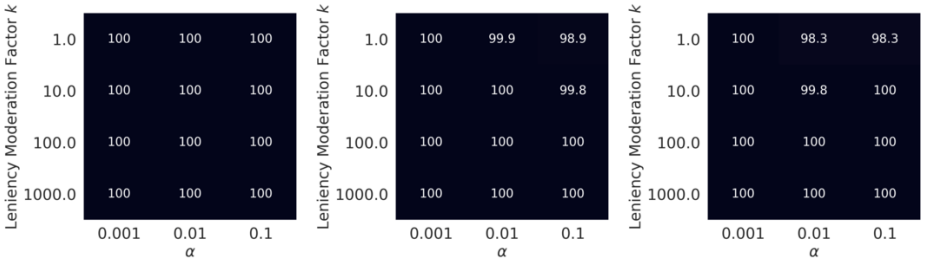
(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.25: Algorithm: SDLQ, Game: The Fully Stochastic Climb Game

A.6 Asynchronized Distributed-Lenient Q-learning

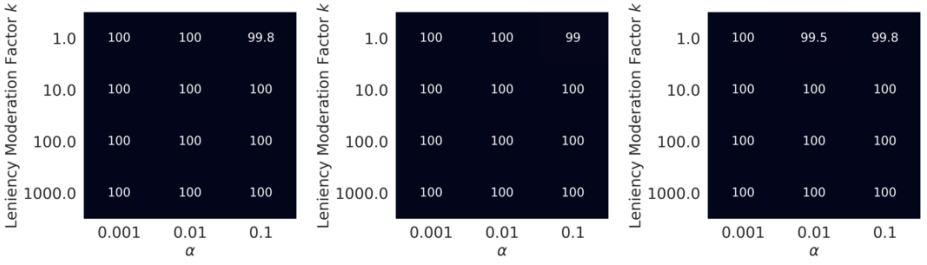


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

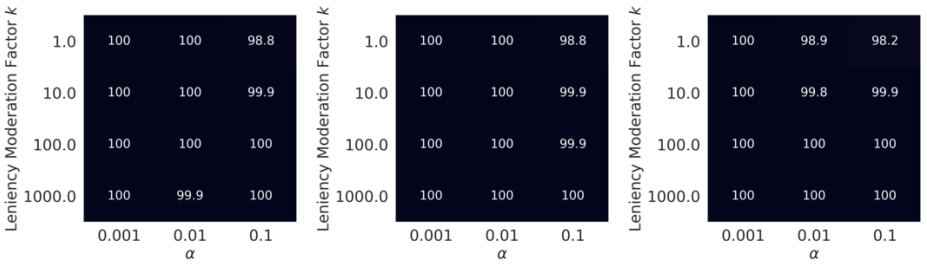


(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.26: Algorithm: ADLQ, Game: The Penalty Game

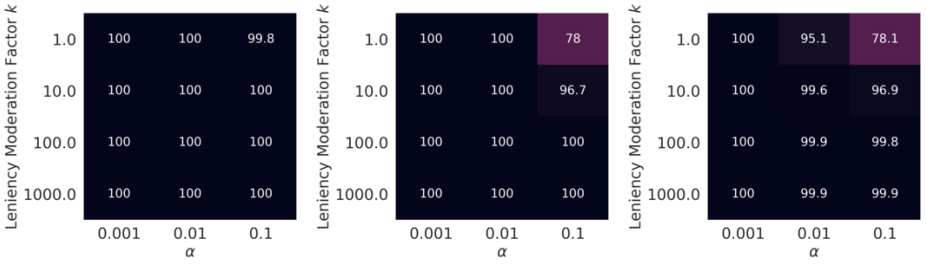


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

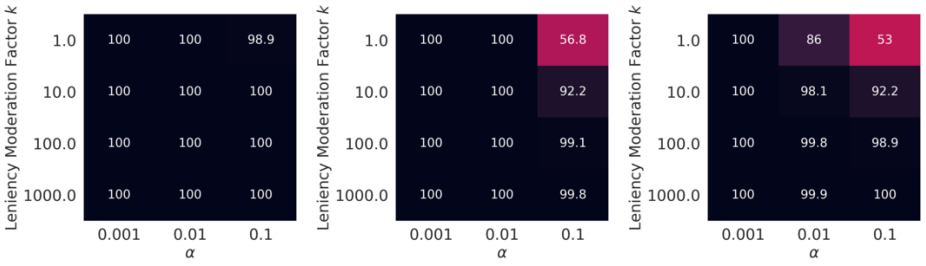


(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.27: Algorithm: ADLQ, Game: The Climb Game

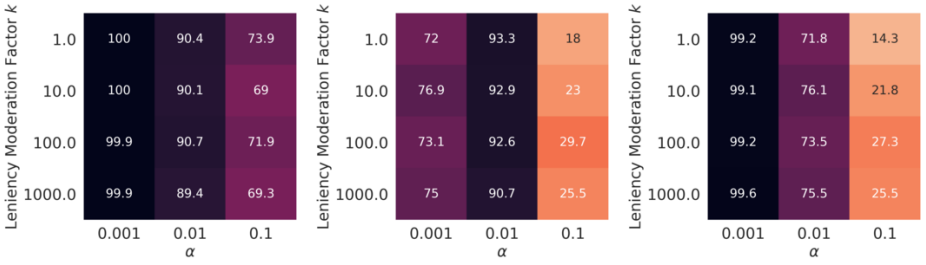


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

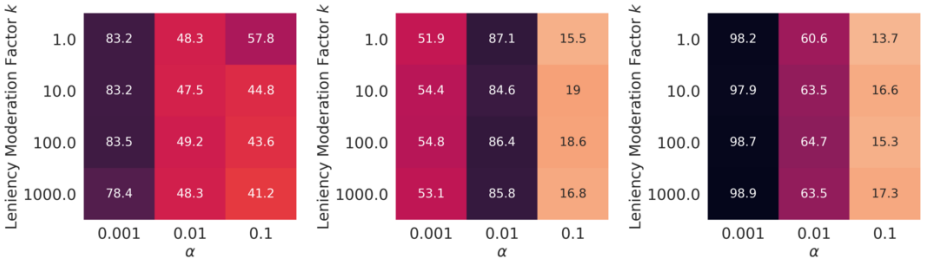


(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.28: Algorithm: ADLQ, Game: The Partially Stochastic Climb Game



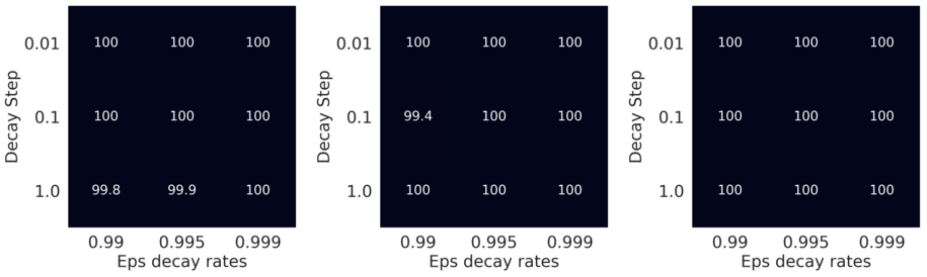
(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High



(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.29: Algorithm: ADLQ, Game: The Fully Stochastic Climb Game

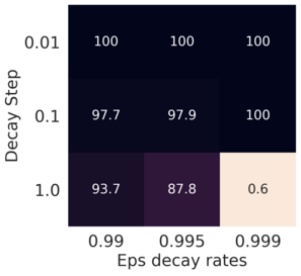
A.7 Q-learning with Negative Update Intervals



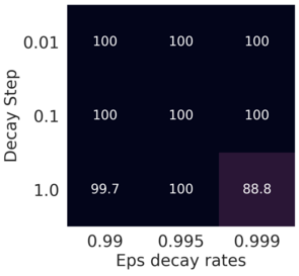
(a) Agents: 2, PSF: Low

(b) Agents: 2, PSF: Medium

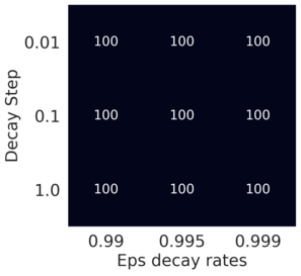
(c) Agents: 2, PSF: High



(d) Agents: 4, PSF: Low

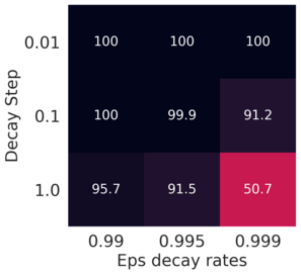


(e) Agents: 4, PSF: Medium

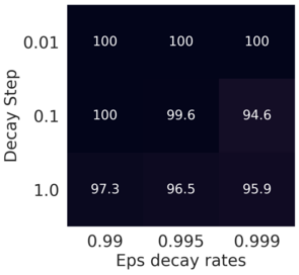


(f) Agents: 4, PSF: Hight

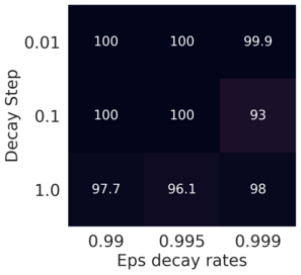
Figure A.30: Algorithm: Q-learning with NUI, Game: The Penalty Game, Burn-In Steps: 1000



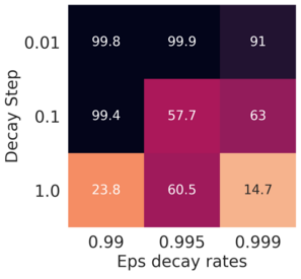
(a) Agents: 2, PSF: Low



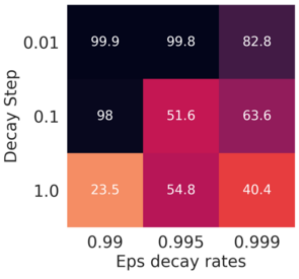
(b) Agents: 2, PSF: Medium



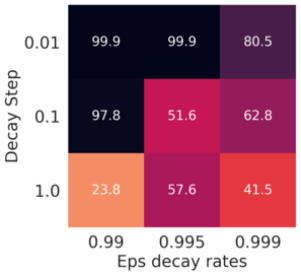
(c) Agents: 2, PSF: High



(d) Agents: 4, PSF: Low

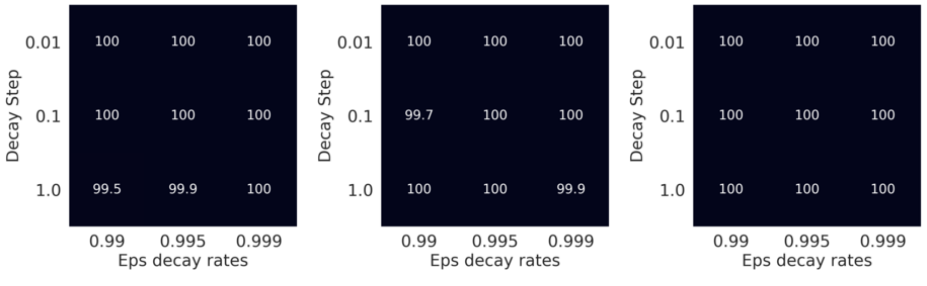


(e) Agents: 4, PSF: Medium



(f) Agents: 4, PSF: High

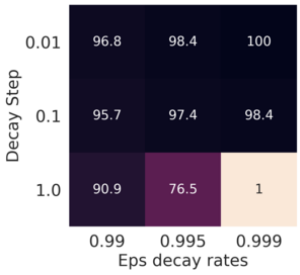
Figure A.31: Algorithm: Q-learning with NUI, Game: The Penalty Game, Burn-In Steps: 500



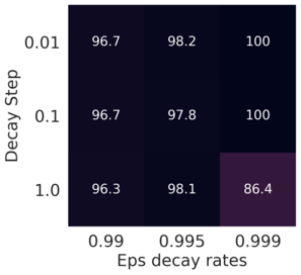
(a) Agents: 2, PSF: Low

(b) Agents: 2, PSF: Medium

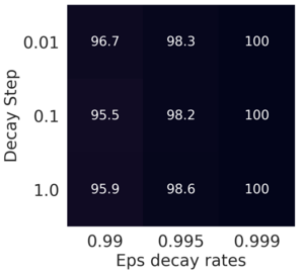
(c) Agents: 2, PSF: High



(d) Agents: 4, PSF: Low

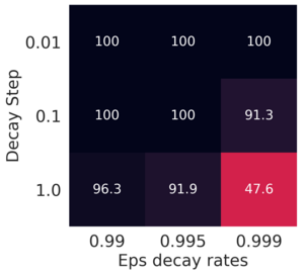


(e) Agents: 4, PSF: Medium

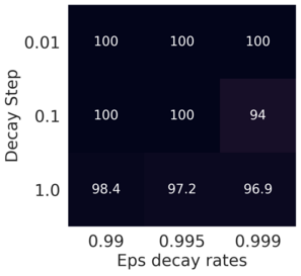


(f) Agents: 4, PSF: High

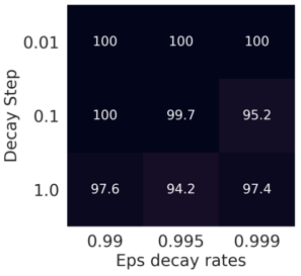
Figure A.32: Algorithm: Q-learning with NUI, Game: The Penalty Game, Burn-In Steps: 100



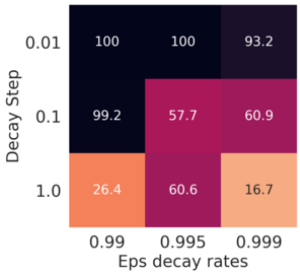
(a) Agents: 2, PSF: Low



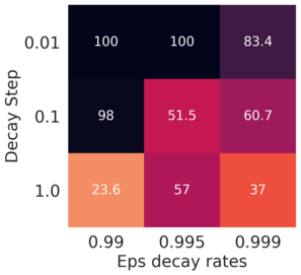
(b) Agents: 2, PSF: Medium



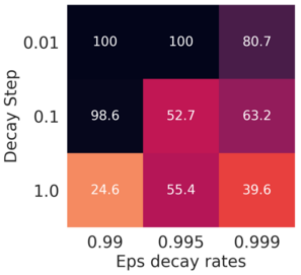
(c) Agents: 2, PSF: High



(d) Agents: 4, PSF: Low

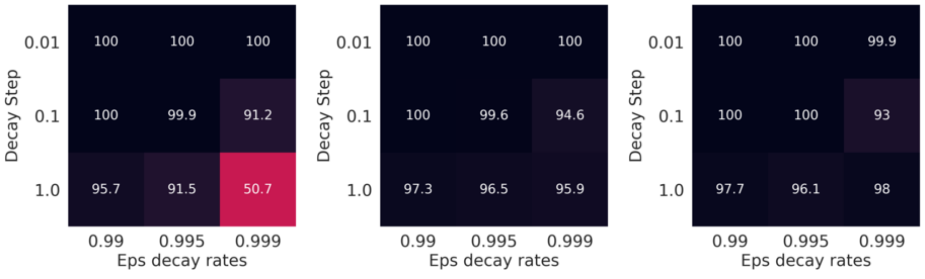


(e) Agents: 4, PSF: Medium

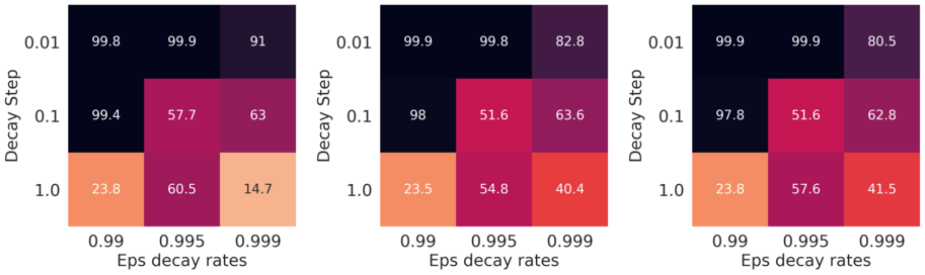


(f) Agents: 4, PSF: High

Figure A.33: Algorithm: Q-learning with NUI, Game: The Climb Game, Burn-In Steps: 1000

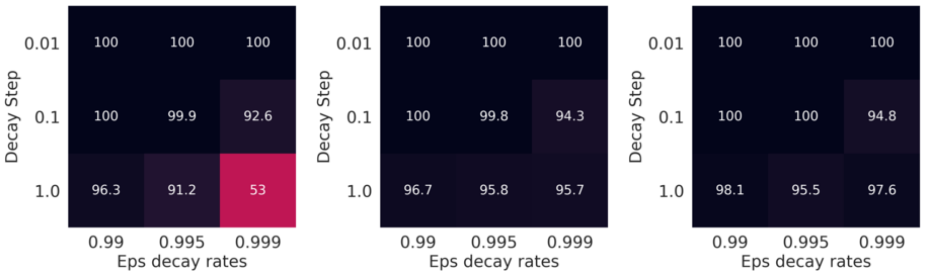


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

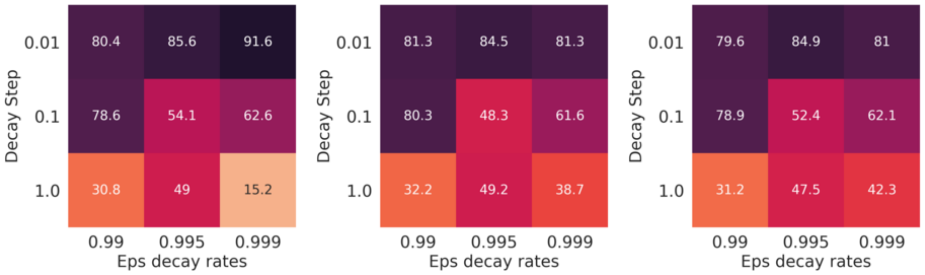


(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.34: Algorithm: Q-learning with NUI, Game: The Climb Game, Burn-In Steps: 500

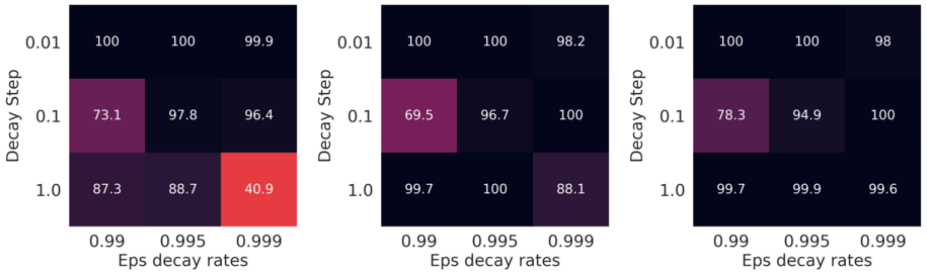


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

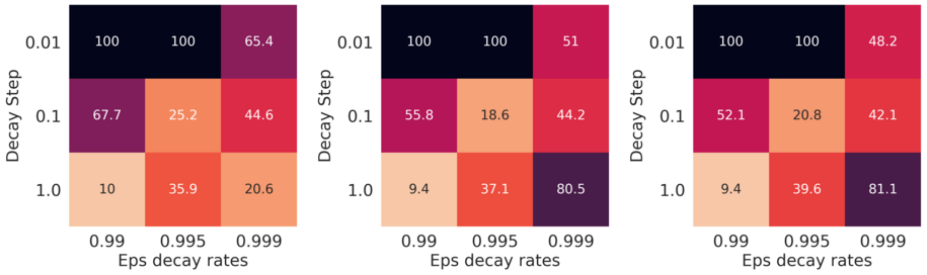


(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.35: Algorithm: Q-learning with NUI, Game: The Climb Game, Burn-In Steps: 100

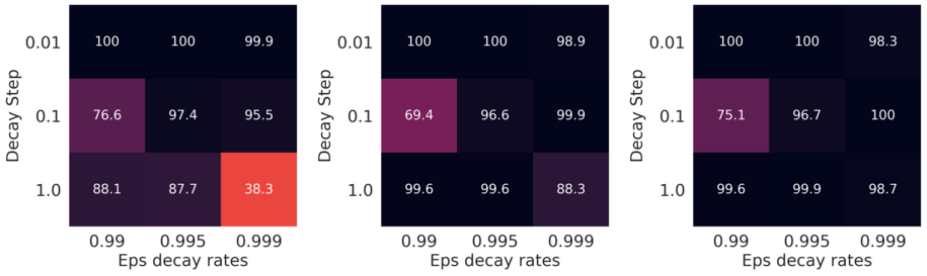


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

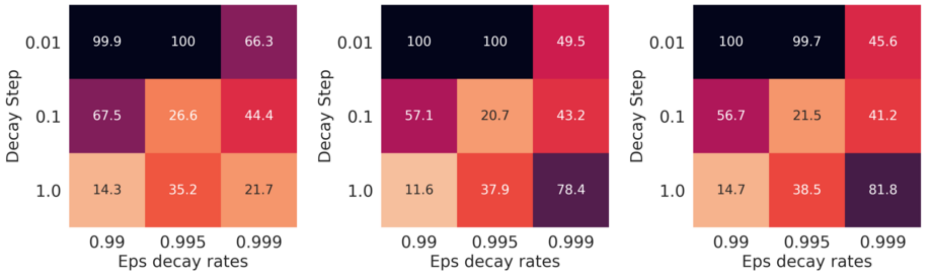


(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.36: Algorithm: Q-learning with NUI, Game: The Partially Stochastic Climb Game, Burn-In Steps: 1000



(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High



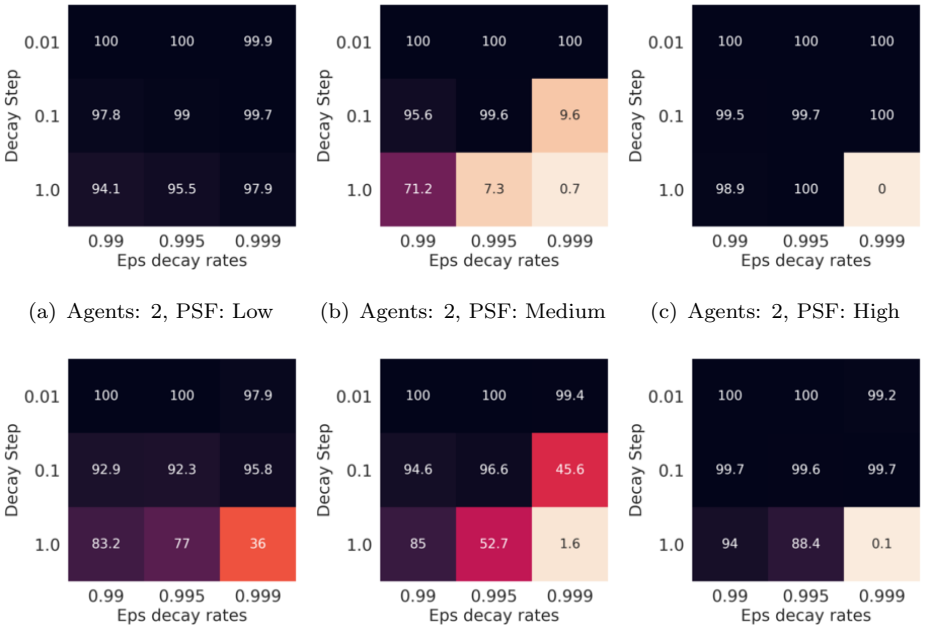
(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.37: Algorithm: Q-learning with NUI, Game: The Partially Stochastic Climb Game, Burn-In Steps: 500

(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

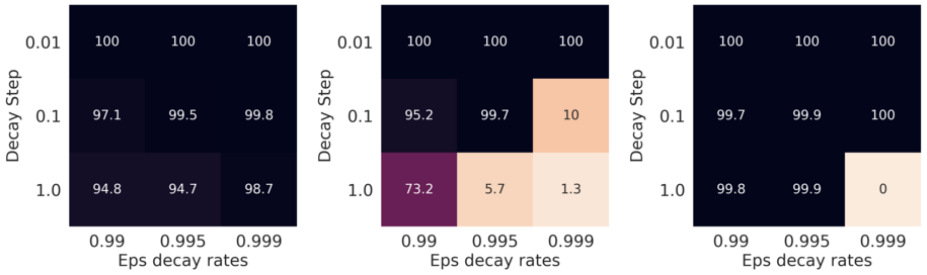
Figure A.38: Algorithm: Q-learning with NUI, Game: The Partially Stochastic Climb Game, Burn-In Steps: 100



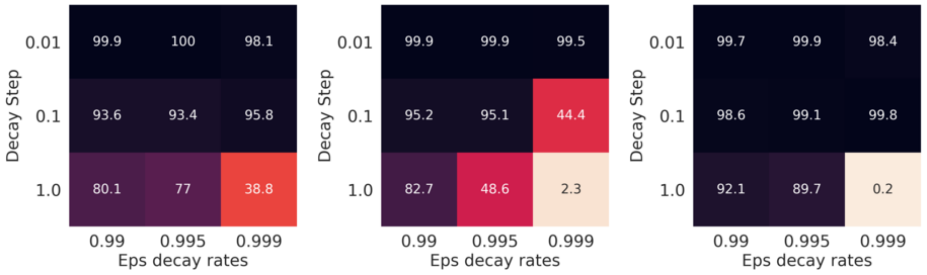
(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.39: Algorithm: Q-learning with NUI, Game: The Fully Stochastic Climb Game, Burn-In Steps: 1000

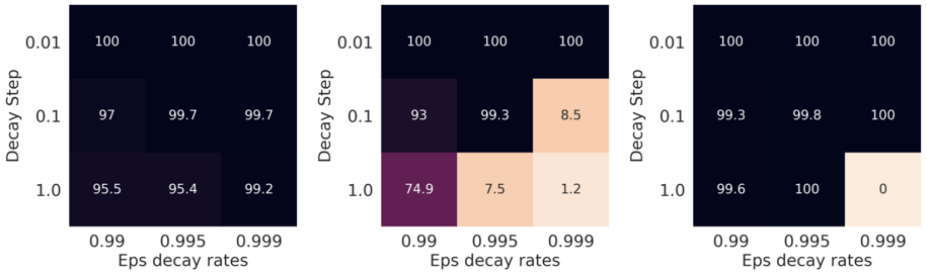


(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High

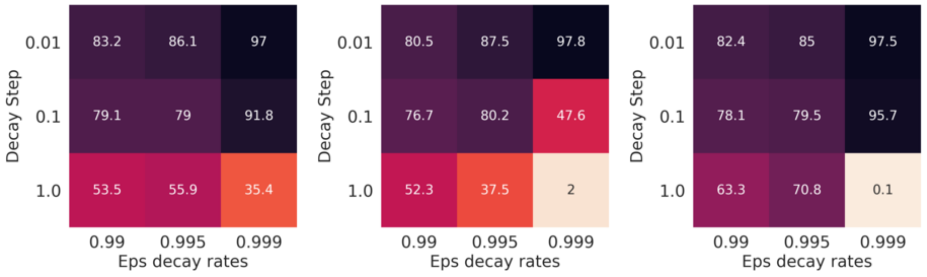


(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.40: Algorithm: Q-learning with NUI, Game: The Fully Stochastic Climb Game, Burn-In Steps: 500



(a) Agents: 2, PSF: Low (b) Agents: 2, PSF: Medium (c) Agents: 2, PSF: High



(d) Agents: 4, PSF: Low (e) Agents: 4, PSF: Medium (f) Agents: 4, PSF: High

Figure A.41: Algorithm: Q-learning with NUI, Game: The Fully Stochastic Climb Game, Burn-In Steps: 100

A.8 Results Summary

In the table below we provide a summary of the highest convergence rate achieved by each algorithm within each setting using tuned hyperparameters:

Game	Agents	Penalty	NUI	SDLQ	ADLQ	LMRL2	Hysteretic	Q-learning	FMQ	RFMQ
Penalty Game	2	Low	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**	**100%**
		Medium	**100%**	**100%**	**100%**	**100%**	**100%**	99.9%	**100%**	91.2%
		High	**100%**	**100%**	**100%**	**100%**	**100%**	NA	**100%**	0.6%
	4	Low	**100%**	**100%**	**100%**	**100%**	**100%**	NA	**100%**	89.0%
		Medium	**100%**	**100%**	**100%**	**100%**	**100%**	NA	**100%**	67.7%
		High	**100%**	**100%**	**100%**	99.9%	**100%**	NA	99.4%	24.8%
Climb Game	2	Low	**100%**	**100%**	**100%**	**100%**	**100%**	24.0%	**100%**	99.9%
		Medium	**100%**	**100%**	**100%**	**100%**	**100%**	NA	97.5%	69.6%
		High	**100%**	**100%**	**100%**	**100%**	**100%**	NA	50.8%	0%
	4	Low	**100%**	**100%**	**100%**	**100%**	**100%**	NA	45.2%	29.9%
		Medium	**100%**	**100%**	**100%**	98.3%	97.8%	NA	45.1%	36.8%
		High	**100%**	**100%**	**100%**	98.1%	50.3%	NA	29.2%	7.3%
Partially Stochastic Climb Game	2	Low	**100%**	**100%**	**100%**	**100%**	87%	28.2%	**100%**	99.4%
		Medium	**100%**	**100%**	**100%**	99.9%	70.3%	NA	97.3%	64.5%
		High	**100%**	**100%**	**100%**	**100%**	69.5%	NA	48.1%	0%
	4	Low	**100%**	**100%**	**100%**	**100%**	63.9%	NA	43.4%	27.6%
		Medium	**100%**	**100%**	**100%**	72.2%	62.3%	NA	45.6%	37.7%
		High	**100%**	**100%**	**100%**	73.2%	50.9%	NA	32.4%	7.9%
Fully Stochastic Climb Game	2	Low	**100%**	**100%**	**100%**	90.2%	37.5%	26.8%	25.4%	64.7%
		Medium	**100%**	**100%**	93.3%	62.0%	25.2%	NA	24%	0%
		High	**100%**	**100%**	99.6%	56.8%	21.9%	NA	6.9%	0%
	4	Low	**100%**	**100%**	83.5%	96.3%	33.4%	NA	8.6%	53.5%
		Medium	**100%**	**100%**	87.1%	92.3%	26.5%	NA	7.3%	1.9%
		High	**100%**	**100%**	98.9%	90.94%	13.7%	NA	7.4%	1.8%

TABLE A.1: Summary of the best results achieved using tuned hyperparameter configurations. Boldface indicates evaluation that resulted in the (joint) highest convergence rate. For decentralized Q-learning we only conduct experiments using the low-reward two-player games, with the exception of the Penalty Game. This is due to decentralized Q-learning's exhibiting low convergence rates, even in the low-reward setting. For RFMQ we note that the results can be improved by (significantly) increasing the number of training iterations, as discussed in Section 4.4.3.

Appendix B

Apprentice Firemen Game Experiment Details & Evaluations

B.1 Hyperparameters

Table B.1 lists the hyperparameters used for our empirical evaluation in Chapter 7. To reduce the time required to evaluate LDDQN we apply python's xxhash to masked observations (i.e., removing civilians).

Component	Hyperparameter	Range of values
DDQN Base	Learning rate α	0.0001
	Discount rate γ	0.95
	Target network sync. steps	5000
	ERM Size	250'000
ϵ-Greedy Exploration	Initial ϵ value	1.0
	ϵ Decay factor	0.999
	Minimum ϵ Value	0.05
Leniency	MaxTemperature	1.0
	Leniency Modification Coefficient K	1.0
	TDS Exponent ρ	-0.1
	TDS Exponent Decay Rate d	0.95
	Initial Max Temperature Value ν	1.0
	Max Temperature Decay Coefficient μ	0.9998
	Action Selection Exponent	0.25
	Hashing	xxhash
NUI-DDQN	ERM_u Capacity	100 Episodes
	Decay threshold	50 Episodes
	r_u^{min} decay rate	0.995

TABLE B.1: Hyper-parameters

B.2 Learning Best Response Policies

Table B.2 provides additional $\overline{RC}$ scatter plots for each evaluation setting. Each marker within the scatter plots represents the $\overline{RC}$ for an individual run. To provide further clarity we sort the runs by $\overline{RC}$. We observe that for the majority of settings higher $\overline{RC}$ values are achieved by agents in *layout 1*. Interestingly only HDDQN ($\beta = 0.5$, PS Rewards) and LDDQN (DET & PS Rewards) achieved higher $\overline{RC}$ values in layout 2. NUI-DDQNs meanwhile perform consistently when receiving deterministic and PS rewards, while a couple of runs faltered for FS rewards within layout 2. It is worth

noting that even for NUI-DDQN runs with low $\overline{RC}$, (A, A) remains a frequently observed outcome:

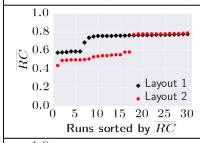
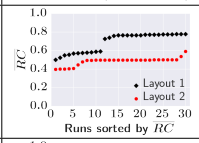
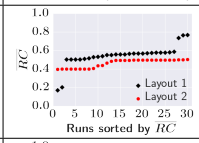
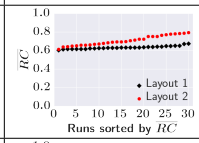
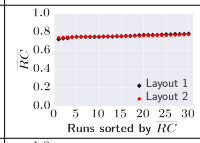
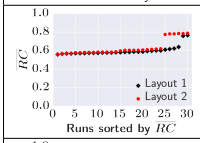
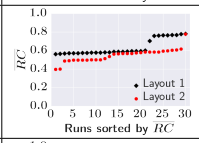
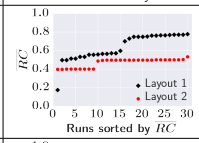
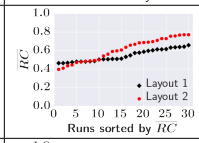
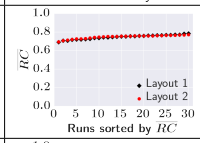
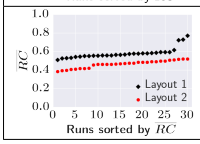
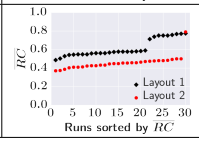
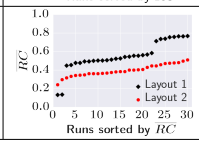
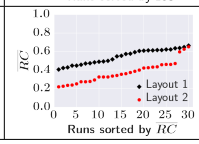
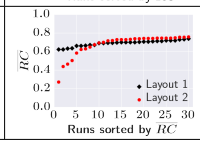
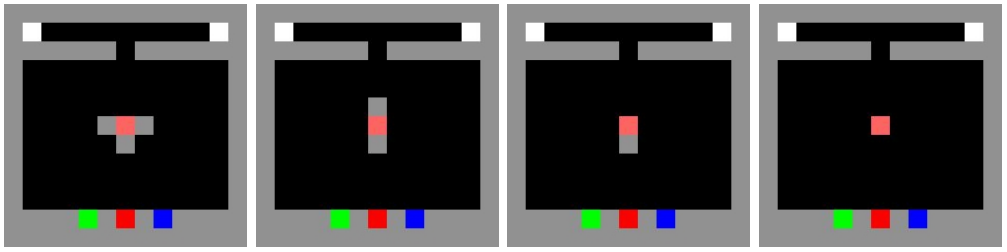
HDQN ($\beta = 0.5$)	**HDQN ($\beta = 0.7$)**	**HDQN ($\beta = 0.9$)**	**LDDQN**	**NUI-DDQN**	
					**DET**
					**PS**
					**FS**

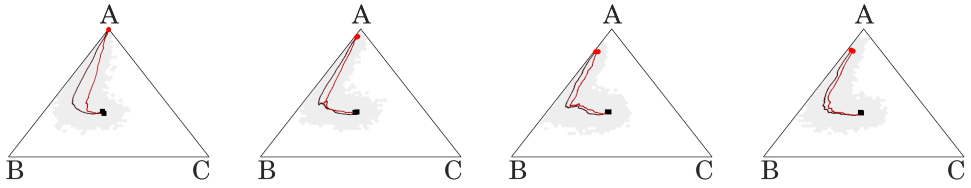
TABLE B.2: Scatter plots illustrating the average coordinated reward $\overline{RC}$ for each training run. The x-axis is sorted by $\overline{RC}$ values.

B.3 LDDQN Variable Access Points Experiments

Figure B.1 illustrates the AFG layouts used for the evaluations discussed in Section 7.5.6. We also provide the resulting phase plot for each layout.



(a) 1 Access Point (b) 2 Access Points (c) 3 Access Points (d) 4 Access Points



(e) 1 Access Point (f) 2 Access Points (g) 3 Access Points (h) 4 Access Points

Figure B.1: Phase plots illustrate delayed convergence of LDDQNs as a result of increasing the number of possible state-action pairs.

Bibliography

[1] Dilip Abreu, David Pearce, and Ennio Stacchetti, *Toward a theory of discounted repeated games with imperfect monitoring*, Econometrica: Journal of the Econometric Society (1990), 1041–1063.

[2] TP Imthias Ahamed, Vivek S Borkar, and S Juneja, *Adaptive importance sampling technique for markov chains using stochastic approximation*, Operations Research **54** (2006), no. 3, 489–504.

[3] Mazda Ahmadi and Peter Stone, *A multi-robot system for continuous area sweeping tasks*, Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006., IEEE, 2006, pp. 1724–1729.

[4] Stefano V Albrecht and Peter Stone, *Reasoning about hypothetical agent behaviours and their parameters*, Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2017, pp. 547–555.

[5] Eduardo Alonso, Mark D'inverno, Daniel Kudenko, Michael Luck, and Jason Noble, *Learning in multi-agent systems*, The Knowledge Engineering Review **16** (2001), no. 3, 277–284.

[6] Kai Arulkumaran, Marc Peter Deisenroth, Miles Brundage, and Anil Anthony Bharath, *A brief survey of deep reinforcement learning*, arXiv preprint arXiv:1708.05866 (2017).

[7] Pierre-Luc Bacon, Jean Harb, and Doina Precup, *The option-critic architecture.*, Proc. of AAAI, 2017, pp. 1726–1734.

[8] Tucker Balch and Ronald C Arkin, *Communication in reactive multiagent robotic systems*, Autonomous robots **1** (1994), no. 1, 27–52.

[9] Nikos Barbalios and Panagiotis Tzionas, *A robust approach for multi-agent natural resource allocation based on stochastic optimization algorithms*, Applied Soft Computing **18** (2014), 12–24.

[10] Enda Barrett, Enda Howley, and Jim Duggan, *Applying reinforcement learning towards automating resource allocation and application scalability in the cloud*,

Concurrency and Computation: Practice and Experience **25** (2013), no. 12, 1656–1674.

[11] Andrew Barto and Richard Sutton, *Reinforcement learning: An introduction*, MIT press, 1998.

[12] Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling, *The arcade learning environment: An evaluation platform for general agents*, Journal of Artificial Intelligence Research **47** (2013), 253–279.

[13] Richard Bellman, *A markovian decision process*, Journal of Mathematics and Mechanics (1957), 679–684.

[14] Huang Bing-Qiang, Cao Guang-Yi, and Guo Min, *Reinforcement learning neural network to the problem of autonomous mobile robot obstacle avoidance.*, 2005 International Conference on Machine Learning & Cybernetics (2005), 85.

[15] Daan Bloembergen, Daniel Hennes, Michael Kaisers, and Karl Tuyls, *Evolutionary dynamics of multi-agent learning: A survey.*, Journal of Artificial Intelligence Research **53** (2015), 659–697.

[16] Daan Bloembergen, Daniel Hennes, Peter McBurney, and Karl Tuyls, *Trading in markets with noisy information: An evolutionary analysis*, Connection Science **27** (2015), no. 3, 253–268.

[17] Daan Bloembergen, Michael Kaisers, and Karl Tuyls, *Lenient frequency adjusted q-learning*, Proc. of 22nd Belgium-Netherlands Conf. on Artif. Intel, 2010.

[18] ———, *Empirical and theoretical support for lenient learning*, Proc. of AAMAS, 2011, pp. 1105–1106.

[19] Daan Bloembergen, Karl Tuyls, Daniel Hennes, and Michael Kaisers, *Evolutionary dynamics of multi-agent learning: A survey*, Journal of Artificial Intelligence Research **53** (2015), 659–697.

[20] Michael Bowling and Manuela Veloso, *Multiagent learning using a variable learning rate*, Artificial Intelligence **136** (2002), no. 2, 215–250.

[21] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba, *Openai gym*, 2016.

[22] Bastian Broecker, Karl Tuyls, and James Butterworth, *Distance-based multi-robot coordination on pocket drones*, 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 6389–6394.

[23] Tim Brys, Anna Harutyunyan, Peter Vrancx, Matthew E Taylor, Daniel Kudenko, and Ann Nowé, *Multi-objectivization of reinforcement learning problems by reward shaping*, 2014 international joint conference on neural networks (IJCNN), IEEE, 2014, pp. 2315–2322.

[24] John Burden and Daniel Kudenko, *Using uniform state abstractions for reward shaping with reinforcement learning*, Workshop on Adaptive Learning Agents (ALA) at the Federated AI Meeting, vol. 18, 2018.

[25] Lucian Busoniu, Robert Babuska, and Bart De Schutter, *A comprehensive survey of multiagent reinforcement learning*, IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2), 2008 (2008).

[26] Lucian Buşoniu, Robert Babuška, and Bart De Schutter, *Multi-agent reinforcement learning: An overview*, Innovations in multi-agent systems and applications-1, Springer, 2010, pp. 183–221.

[27] James Butterworth, Bastian Broecker, Karl Tuyls, and Paolo Paoletti, *Evolving coverage behaviours for mavs using neat*, Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 1886–1888.

[28] Jacopo Castellini, Frans A. Oliehoek, Rahul Savani, and Shimon Whiteson, *The representational capacity of action-value networks for multi-agent reinforcement learning*, Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, AAMAS '19, Montreal, QC, Canada, May 13-17, 2019, 2019, pp. 1862–1864.

[29] Georgios Chalkiadakis and Craig Boutilier, *Coordination in multiagent reinforcement learning: a bayesian approach*, Proceedings of the second international joint conference on Autonomous agents and multiagent systems, ACM, 2003, pp. 709–716.

[30] Moses S Charikar, *Similarity estimation techniques from rounding algorithms*, Proceedings of the thiry-fourth annual ACM symposium on Theory of computing, ACM, 2002, pp. 380–388.

[31] Yu Fan Chen, Miao Liu, Michael Everett, and Jonathan P How, *Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning*, 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2017, pp. 285–292.

[32] Daniel Claes, Daniel Hennes, Karl Tuyls, and Wim Meeussen, *Collision avoidance under bounded localization uncertainty*, 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, IEEE, 2012, pp. 1192–1198.

[33] Caroline Claus and Craig Boutilier, *The dynamics of reinforcement learning in cooperative multiagent systems*, In AAAI/IAAI **1998** (1998), 746–752.

[34] Ronan Collobert and Jason Weston, *A unified architecture for natural language processing: Deep neural networks with multitask learning*, Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 160–167.

[35] Jacob W Crandall, *Just add pepper: extending learning algorithms for repeated matrix games to repeated markov games*, Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems, 2012, pp. 399–406.

[36] ———, *Towards minimizing disappointment in repeated games*, Journal of Artificial Intelligence Research **49** (2014), 111–142.

[37] Tim de Bruin, Jens Kober, Karl Tuyls, and Robert Babuška, *The importance of experience replay database composition in deep reinforcement learning*, Deep Reinforcement Learning Workshop, NIPS, 2015.

[38] Sam Devlin and Daniel Kudenko, *Theoretical considerations of potential-based reward shaping for multi-agent systems*, The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 1, International Foundation for Autonomous Agents and Multiagent Systems, 2011, pp. 225–232.

[39] Sam Devlin, Daniel Kudenko, and Marek Grześ, *An empirical study of potential-based reward shaping and advice in complex, multi-agent systems*, Advances in Complex Systems **14** (2011), no. 02, 251–278.

[40] Sam Devlin, Logan Yliniemi, Daniel Kudenko, and Kagan Tumer, *Potential-based difference rewards for multiagent reinforcement learning*, Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems, International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 165–172.

[41] Martin Duggan, Jim Duggan, Enda Howley, and Enda Barrett, *A reinforcement learning approach for the scheduling of live migration from under utilised hosts*, Memetic Computing **9** (2017), no. 4, 283–293.

[42] Martin Duggan, Kieran Flesk, Jim Duggan, Enda Howley, and Enda Barrett, *A reinforcement learning approach for dynamic selection of virtual machines in cloud data centres*, 2016 Sixth International Conference on Innovative Computing Technology (INTECH), IEEE, 2016, pp. 92–97.

[43] Norbert Elias and Eric Dunning, *Dynamics of group sports with special reference to football*, The British Journal of Sociology **17** (1966), no. 4, 388–402.

[44] Mohamed Elidrisi, Nicholas Johnson, Maria Gini, and Jacob Crandall, *Fast adaptive learning in repeated stochastic games by game abstraction*, Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems, International Foundation for Autonomous Agents and Multiagent Systems, 2014, pp. 1141–1148.

[45] Jakob Foerster, Ioannis Alexandros Assael, Nando de Freitas, and Shimon Whiteson, *Learning to communicate with deep multi-agent reinforcement learning*, Advances in Neural Information Processing Systems, 2016, pp. 2137–2145.

[46] Jakob Foerster, Nantas Nardelli, Gregory Farquhar, Triantafyllos Afouras, Philip HS Torr, Pushmeet Kohli, and Shimon Whiteson, *Stabilising experience replay for deep multi-agent reinforcement learning*, Proc. of ICML, 2017, pp. 1146–1155.

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

[48] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G Bellemare, Joelle Pineau, et al., *An introduction to deep reinforcement learning*, Foundations and Trends® in Machine Learning **11** (2018), no. 3-4, 219–354.

[49] Jordan Frank, Shie Mannor, and Doina Precup, *Reinforcement learning in the presence of rare events*, Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 336–343.

[50] Nancy Fulda and Dan Ventura, *Predicting and preventing coordination problems in cooperative Q-learning systems*, IJCAI, vol. 2007, 2007, pp. 780–785.

[51] Dhiraj Gandhi, Lerrel Pinto, and Abhinav Gupta, *Learning to fly by crashing*, 2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2017, pp. 3948–3955.

[52] Jon Gauthier and Igor Mordatch, *A paradigm for situated and goal-driven language learning*, arXiv preprint arXiv:1610.03585 (2016).

[53] Herbert Gintis, *Game theory evolving: A problem-centered introduction to modeling strategic behavior*, Princeton university press, 2000.

[54] Xudong Gong, Bo Ding, Jie Xu, Huaimin Wang, Xing Zhou, and Dawei Feng, *Parallelized synchronous multi-agent deep reinforcement learning with experience replay memory*, 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), IEEE, 2019, pp. 325–3255.

[55] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, MIT Press, 2016, `http://www.deeplearningbook.org`.

[56] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio, *Generative adversarial nets*, Proc. of NIPS, 2014, pp. 2672–2680.

[57] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio, *An empirical investigation of catastrophic forgetting in gradient-based neural networks*, arXiv preprint arXiv:1312.6211 (2013).

[58] Samuel Greydanus, Anurag Koul, Jonathan Dodge, and Alan Fern, *Visualizing and understanding atari agents*, International Conference on Machine Learning, 2018, pp. 1787–1796.

[59] Marek Grzes and Daniel Kudenko, *Learning potential for reward shaping in reinforcement learning with tile coding*, Proceedings AAMAS 2008 Workshop on Adaptive and Learning Agents and Multi-Agent Systems (ALAMAS-ALAg 2008), 2008, pp. 17–23.

[60] ———, *Plan-based reward shaping for reinforcement learning*, 2008 4th International IEEE Conference Intelligent Systems, vol. 2, IEEE, 2008, pp. 10–22.

[61] ———, *Learning shaping rewards in model-based reinforcement learning*, Proc. AAMAS 2009 Workshop on Adaptive Learning Agents, vol. 115, 2009.

[62] Marek Grześ and Daniel Kudenko, *Online learning of shaping rewards in reinforcement learning*, Neural Networks **23** (2010), no. 4, 541–550.

[63] Shixiang Gu, Ethan Holly, Timothy Lillicrap, and Sergey Levine, *Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates*, arXiv preprint arXiv:1610.00633 (2016).

[64] Carlos Guestrin, Daphne Koller, and Ronald Parr, *Multiagent planning with factored mdps*, Advances in neural information processing systems, 2002, pp. 1523–1530.

[65] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron C Courville, *Improved training of wasserstein gans*, Advances in neural information processing systems, 2017, pp. 5767–5777.

[66] Jayesh K Gupta, Maxim Egorov, and Mykel Kochenderfer, *Cooperative multi-agent control using deep reinforcement learning*, Proceedings of the Adaptive and Learning Agents workshop (at AAMAS 2017), 2017.

[67] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine, *Reinforcement learning with deep energy-based policies*, arXiv preprint arXiv:1702.08165 (2017).

[68] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine, *Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor*, International Conference on Machine Learning, 2018, pp. 1856–1865.

[69] John C Harsanyi, *Games with incomplete information played by "bayesian" players, i–iii part i. the basic model*, Management science **14** (1967), no. 3, 159–182.

[70] Matthew Hausknecht, Prannoy Mupparaju, Sandeep Subramanian, S Kalyanakrishnan, and P Stone, *Half field offense: an environment for multiagent learning and ad hoc teamwork*, AAMAS Adaptive Learning Agents (ALA) Workshop, 2016.

[71] Matthew Hausknecht and Peter Stone, *Deep recurrent q-learning for partially observable mdps*, arXiv preprint arXiv:1507.06527 (2015).

[72] ———, *Deep reinforcement learning in parameterized action space*, arXiv preprint arXiv:1511.04143 (2015).

[73] He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III, *Opponent modeling in deep reinforcement learning*, International Conference on Machine Learning, 2016, pp. 1804–1813.

[74] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, *Deep residual learning for image recognition*, Proc. of CVPR, 2016, pp. 770–778.

[75] Johannes Heinrich and David Silver, *Deep reinforcement learning from self-play in imperfect-information games*, arXiv preprint arXiv:1603.01121 (2016).

[76] Daniel Hennes, Daan Bloembergen, Michael Kaisers, Karl Tuyls, and Simon Parsons, *Evolutionary advantage of foresight in markets*, Proceedings of the 14th annual conference on Genetic and evolutionary computation, ACM, 2012, pp. 943–950.

[77] Pablo Hernandez-Leal and Michael Kaisers, *Towards a fast detection of opponents in repeated stochastic games*, Proc. of AAMAS, Springer, 2017, pp. 239–257.

[78] Pablo Hernandez-Leal, Michael Kaisers, Tim Baarslag, and Enrique Munoz de Cote, *A survey of learning in multiagent environments: Dealing with non-stationarity*, arXiv preprint arXiv:1707.09183 (2017).

[79] Pablo Hernandez-Leal, Bilal Kartal, and Matthew E Taylor, *A survey and critique of multiagent deep reinforcement learning*, arXiv preprint arXiv:1810.05587 (2018).

[80] ———, *Agent modeling as auxiliary task for deep reinforcement learning*, arXiv preprint arXiv:1907.09597 (2019).

[81] Josef Hofbauer and Jörgen W Weibull, *Evolutionary selection against dominated strategies*, Journal of economic theory **71** (1996), no. 2, 558–573.

[82] Zhang-Wei Hong, Shih-Yang Su, Tzu-Yun Shann, Yi-Hsiang Chang, and Chun-Yi Lee, *A deep policy inference q-network for multi-agent systems*, Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 1388–1396.

[83] Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu, Royce Cheng-Yue, et al., *An empirical evaluation of deep learning on highway driving*, arXiv preprint arXiv:1504.01716 (2015).

[84] Satoshi Iizuka, Edgar Simo-Serra, and Hiroshi Ishikawa, *Globally and locally consistent image completion*, ACM Transactions on Graphics (TOG) **36** (2017), no. 4, 107.

[85] Tommi Jaakkola, Michael I Jordan, and Satinder P Singh, *Convergence of stochastic iterative dynamic programming algorithms*, Advances in neural information processing systems, 1994, pp. 703–710.

[86] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra, *Planning and acting in partially observable stochastic domains*, Artificial intelligence **101** (1998), no. 1-2, 99–134.

[87] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore, *Reinforcement learning: A survey*, Journal of artificial intelligence research **4** (1996), 237–285.

[88] Michael Kaisers, *Learning against learning. evolutionary dynamics of reinforcement learning algorithms in strategic interactions*, Ph.D. thesis, Maastricht University, 2012.

[89] Michihiro Kandori, *The use of information in repeated games with imperfect monitoring*, The Review of Economic Studies **59** (1992), no. 3, 581–593.

[90] Spiros Kapetanakis and Daniel Kudenko, *Reinforcement learning of coordination in cooperative multi–agent systems*, AAAI/IAAI **2002** (2002), 326–331.

[91] ———, *Reinforcement learning of coordination in heterogeneous cooperative multi-agent systems*, Adaptive Agents and Multi-Agent Systems II, Springer, 2004, pp. 119–131.

[92] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei, *Large-scale video classification with convolutional neural networks*, Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2014, pp. 1725–1732.

[93] Woojun Kim, Myungsik Cho, and Youngchul Sung, *Message-dropout: An efficient training method for multi-agent deep reinforcement learning*, arXiv preprint arXiv:1902.06527 (2019).

[94] Diederik P. Kingma and Jimmy Ba, *Adam: A method for stochastic optimization*, Proc. of ICLR, 2014.

[95] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton, *Imagenet classification with deep convolutional neural networks*, Proc. of NIPS, 2012, pp. 1097–1105.

[96] Abhishek Kumar, Prasanna Sattigeri, and Tom Fletcher, *Semi-supervised learning with gans: Manifold invariance with improved inference*, Proc. of NIPS, 2017, pp. 5540–5550.

[97] Guillaume Lample and Devendra Singh Chaplot, *Playing fps games with deep reinforcement learning*, arXiv preprint arXiv:1609.05521 (2016).

[98] ———, *Playing fps games with deep reinforcement learning.*, AAAI (2017), 2140–2146.

[99] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel, *A unified game-theoretic approach to multiagent reinforcement learning*, Advances in Neural Information Processing Systems, 2017, pp. 4190–4203.

[100] Martin Lauer and Martin Riedmiller, *An algorithm for distributed reinforcement learning in cooperative multi-agent systems*, In Proceedings of the Seventeenth International Conference on Machine Learning, Citeseer, 2000.

[101] Guillaume J Laurent, Laëtitia Matignon, Le Fort-Piat, et al., *The world of independent learners is not markovian*, International Journal of Knowledge-based and Intelligent Engineering Systems **15** (2011), no. 1, 55–64.

[102] Angeliki Lazaridou, Alexander Peysakhovich, and Marco Baroni, *Multi-agent cooperation and the emergence of (natural) language*, arXiv preprint arXiv:1612.07182 (2016).

[103] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton, *Deep learning*, nature **521** (2015), no. 7553, 436.

[104] Joel Z Leibo, Vinicius Zambaldi, Marc Lanctot, Janusz Marecki, and Thore Graepel, *Multi-agent reinforcement learning in sequential social dilemmas*, Proc. of AAMAS, 2017, pp. 464–473.

[105] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra, *Continuous control with deep reinforcement learning*, arXiv preprint arXiv:1509.02971 (2015).

[106] Long-Ji Lin, *Self-improving reactive agents based on reinforcement learning, planning and teaching*, Machine learning **8** (1992), no. 3-4, 293–321.

[107] ———, *Reinforcement learning for robots using neural networks*, Tech. report, Carnegie-Mellon Univ Pittsburgh PA School of Computer Science, 1993.

[108] Michael L Littman, *Markov games as a framework for multi-agent reinforcement learning*, Machine learning proceedings 1994, Elsevier, 1994, pp. 157–163.

[109] Ryan Lowe, Yi Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch, *Multi-agent actor-critic for mixed cooperative-competitive environments*, Proc. of NIPS, 2017, pp. 6379–6390.

[110] Xueguang Lu and Christopher Amato, *On improving decentralized hysteretic deep reinforcement learning*, arXiv preprint arXiv:1812.06319 (2018).

[111] Sean Luke, Liviu Panait, and Karl Tuyls, *Theoretical advantages of lenient learners: An evolutionary game theoretic perspective*, JMLR **9** (2008), 423–457.

[112] Michael W Macy and Andreas Flache, *Learning dynamics in social dilemmas*, Proceedings of the National Academy of Sciences **99** (2002), no. suppl 3, 7229–7236.

[113] Aleksandra Malysheva, Aleksei Shpilman, and Daniel Kudenko, *Learning to run with reward shaping from video data*, Workshop on Adaptive Learning Agents (ALA) at the Federated AI Meeting, vol. 18, 2018.

[114] Aleksandra Malysheva, Tegg Taekyong Sung, Chae-Bong Sohn, Daniel Kudenko, and Aleksei Shpilman, *Deep multi-agent reinforcement learning with relevance graphs*, arXiv preprint arXiv:1811.12557 (2018).

[115] Patrick Mannion, Jim Duggan, and Enda Howley, *Parallel reinforcement learning for traffic signal control*, Procedia Computer Science **52** (2015), 956–961.

[116] ———, *An experimental review of reinforcement learning algorithms for adaptive traffic signal control*, Autonomic Road Transport Support Systems, Springer, 2016, pp. 47–66.

[117] ———, *Generating multi-agent potential functions using counterfactual estimates*, Proceedings of Learning, Inference and Control of Multi-Agent Systems (at NIPS 2016) (2016).

[118] Ana I Maqueda, Antonio Loquercio, Guillermo Gallego, Narciso García, and Davide Scaramuzza, *Event-based vision meets deep learning on steering prediction for self-driving cars*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 5419–5427.

[119] Michael Mathieu, Camille Couprie, and Yann LeCun, *Deep multi-scale video prediction beyond mean square error*, arXiv preprint arXiv:1511.05440 (2015).

[120] Laëtitia Matignon, Guillaume Laurent, and Nadine Le Fort-Piat, *Hysteretic Q-Learning: an algorithm for decentralized reinforcement learning in cooperative multi-agent teams*, Proc. of IROS, 2007, pp. 64–69.

[121] Laëtitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat, *Coordination of independent learners in cooperative markov games.*, (2009).

[122] Laetitia Matignon, Guillaume J Laurent, and Nadine Le Fort-Piat, *Independent reinforcement learners in cooperative Markov games: a survey regarding coordination problems*, The Knowledge Engineering Review **27** (2012), no. 1, 1–31.

[123] Tambet Matiisen, *Pommerman baselines*, `https://github.com/tambetm/pommerman-baselines`, 2018.

[124] Gábor Melis, Chris Dyer, and Phil Blunsom, *On the state of the art of evaluation in neural language models*, arXiv preprint arXiv:1707.05589 (2017).

[125] Mihail Mihaylov, Karl Tuyls, and Ann Nowé, *A decentralized approach for convention emergence in multi-agent systems*, Autonomous Agents and Multi-Agent Systems **28** (2014), no. 5, 749–778.

[126] W Thomas Miller, Paul J Werbos, and Richard S Sutton, *Neural networks for control*, MIT press, 1995.

[127] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu, *Asynchronous methods for deep reinforcement learning*, International conference on machine learning, 2016, pp. 1928–1937.

[128] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller, *Playing atari with deep reinforcement learning*, arXiv preprint arXiv:1312.5602 (2013).

[129] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al., *Human-level control through deep reinforcement learning*, Nature **518** (2015), no. 7540, 529.

[130] Mehdi Mohammadi, Ala Al-Fuqaha, Mohsen Guizani, and Jun-Seok Oh, *Semisupervised deep reinforcement learning in support of iot and smart city services*, IEEE Internet of Things Journal **5** (2018), no. 2, 624–635.

[131] Igor Mordatch and Pieter Abbeel, *Emergence of grounded compositional language in multi-agent populations*, Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[132] Sajad Mousavi, Michael Schukat, Enda Howley, Ali Borji, and Nasser Mozayani, *Learning to predict where to look in interactive environments using deep recurrent q-learning*, arXiv preprint arXiv:1612.05753 (2016).

[133] Seyed Sajad Mousavi, Michael Schukat, and Enda Howley, *Deep reinforcement learning: an overview*, Proceedings of SAI Intelligent Systems Conference, Springer, 2016, pp. 426–440.

[134] ———, *Traffic light control using deep policy-gradient and value-function-based reinforcement learning*, IET Intelligent Transport Systems **11** (2017), no. 7, 417–423.

[135] Vinod Nair and Geoffrey E Hinton, *Rectified linear units improve restricted boltzmann machines*, Proc. of ICML, 2010, pp. 807–814.

[136] John Nash, *Non-cooperative games*, Annals of mathematics (1951), 286–295.

[137] Ann Nowé, Peter Vrancx, and Yann-Michaël De Hauwere, *Game theory and multi-agent reinforcement learning*, Reinforcement Learning, Springer, 2012, pp. 441–470.

[138] Frans A Oliehoek, Christopher Amato, et al., *A concise introduction to decentralized pomdps*, vol. 1, Springer, 2016.

[139] Frans A Oliehoek, Matthijs TJ Spaan, and Nikos Vlassis, *Optimal and approximate q-value functions for decentralized pomdps*, Journal of Artificial Intelligence Research **32** (2008), 289–353.

[140] Shayegan Omidshafiei, Christos Papadimitriou, Georgios Piliouras, Karl Tuyls, Mark Rowland, Jean-Baptiste Lespiau, Wojciech M Czarnecki, Marc Lanctot, Julien Perolat, and Remi Munos, {*alpha*}*-rank: Multi-agent evaluation by evolution*, arXiv preprint arXiv:1903.01373 (2019).

[141] Shayegan Omidshafiei, Jason Pazis, Christopher Amato, Jonathan P. How, and John Vian, *Deep decentralized multi-task multi-agent reinforcement learning under partial observability*, Proc. of ICML **70** (2017), 2681–2690.

[142] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy, *Deep exploration via bootstrapped dqn*, Advances in neural information processing systems, 2016, pp. 4026–4034.

[143] Gregory Palmer, Rahul Savani, and Karl Tuyls, *Negative update intervals in deep multi-agent reinforcement learning*, arXiv preprint arXiv:1809.05096 (2018).

[144] Gregory Palmer, Karl Tuyls, Daan Bloembergen, and Rahul Savani, *Lenient multi-agent deep reinforcement learning*, Proc. of AAMAS, 2018, pp. 443–451.

[145] Liviu Panait, *Theoretical convergence guarantees for cooperative coevolutionary algorithms*, Evolutionary computation **18** (2010), no. 4, 581–615.

[146] Liviu Panait and Sean Luke, *Cooperative multi-agent learning: The state of the art*, Autonomous agents and multi-agent systems **11** (2005), no. 3, 387–434.

[147] Liviu Panait, Keith Sullivan, and Sean Luke, *Lenience towards teammates helps in cooperative multiagent learning*, Proc. of AAMAS, 2006.

[148] ———, *Lenient learners in cooperative multiagent systems*, Proc. of AAMAS, ACM, 2006, pp. 801–803.

[149] Liviu Panait, Karl Tuyls, and Sean Luke, *Theoretical advantages of lenient learners: An evolutionary game theoretic perspective*, JMLR **9** (2008), no. Mar, 423–457.

[150] Lauren Parker, James Butterworth, and Shan Luo, *Fly safe: Aerial swarm robotics using force field particle swarm optimisation*, arXiv preprint arXiv:1907.07647 (2019).

[151] Emanuele Pesce and Giovanni Montana, *Improving coordination in multi-agent deep reinforcement learning through memory-driven communication*, arXiv preprint arXiv:1901.03887 (2019).

[152] Mitchell A Potter and Kenneth A De Jong, *A cooperative coevolutionary approach to function optimization*, International Conference on Parallel Problem Solving from Nature, Springer, 1994, pp. 249–257.

[153] Martin L Puterman, *Markov decision processes: discrete stochastic dynamic programming*, John Wiley & Sons, 2014.

[154] Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, S. M. Ali Eslami, and Matthew Botvinick, *Machine theory of mind*, Proc. of ICML (Jennifer Dy and Andreas Krause, eds.), vol. 80, PMLR, 10–15 Jul 2018, pp. 4218–4227.

[155] Neil Rabinowitz, Frank Perbet, Francis Song, Chiyuan Zhang, SM Ali Eslami, and Matthew Botvinick, *Machine theory of mind*, International Conference on Machine Learning, 2018, pp. 4215–4224.

[156] Roberta Raileanu, Emily Denton, Arthur Szlam, and Rob Fergus, *Modeling others using oneself in multi-agent reinforcement learning*, International Conference on Machine Learning, 2018, pp. 4254–4263.

[157] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder de Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson, *Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning*, arXiv preprint arXiv:1803.11485 (2018).

[158] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen, *Improved techniques for training gans*, Proc. of NIPS, 2016, pp. 2234–2242.

[159] Larry Samuelson, *Evolutionary games and equilibrium selection*, vol. 1, MIT press, 1998.

[160] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver, *Prioritized experience replay*, arXiv preprint arXiv:1511.05952 (2015).

[161] Benjamin Schnieders, Shan Luo, Gregory Palmer, and Karl Tuyls, *Fully convolutional one-shot object segmentation for industrial robotics*, Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2019, pp. 1161–1169.

[162] Benjamin Schnieders and Karl Tuvls, *Fast convergence for object detection by learning how to combine error functions*, 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, 2018, pp. 7329–7335.

[163] Arturo Servin and Daniel Kudenko, *Multi-agent reinforcement learning for intrusion detection*, Adaptive Agents and Multi-Agent Systems III. Adaptation and Multi-Agent Learning, Springer, 2005, pp. 211–223.

[164] ———, *Multi-agent reinforcement learning for intrusion detection: A case study and evaluation*, German Conference on Multiagent System Technologies, Springer, 2008, pp. 159–170.

[165] Lloyd S Shapley, *Stochastic games*, Proceedings of the national academy of sciences **39** (1953), no. 10, 1095–1100.

[166] Rachael Shaw, Enda Howley, and Enda Barrett, *An advanced reinforcement learning approach for energy-aware virtual machine consolidation in cloud data centers*, 2017 12th International Conference for Internet Technology and Secured Transactions (ICITST), IEEE, 2017, pp. 61–66.

[167] Yoav Shoham, Rob Powers, and Trond Grenager, *If multi-agent learning is the answer, what is the question?*, Artificial Intelligence **171** (2007), no. 7, 365–377.

[168] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al., *Mastering the game of go with deep neural networks and tree search*, nature **529** (2016), no. 7587, 484.

[169] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al., *Mastering the game of go without human knowledge*, Nature **550** (2017), no. 7676, 354.

[170] Satinder Singh, Tommi Jaakkola, Michael L Littman, and Csaba Szepesvári, *Convergence results for single-step on-policy reinforcement-learning algorithms*, Machine learning **38** (2000), no. 3, 287–308.

[171] Leon Sixt, Benjamin Wild, and Tim Landgraf, *Rendergan: Generating realistic labeled data*, Frontiers in Robotics and AI **5** (2018), 66.

[172] John Maynard Smith, *Evolution and the theory of games*, Cambridge university press, 1982.

[173] Richard Socher, Yoshua Bengio, and Christopher D Manning, *Deep learning for nlp (without magic)*, Tutorial Abstracts of ACL 2012, Association for Computational Linguistics, 2012, pp. 5–5.

[174] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi, *Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning*, International Conference on Machine Learning, 2019, pp. 5887–5896.

[175] Thomas Spooner, John Fearnley, Rahul Savani, and Andreas Koukorinis, *Market making via reinforcement learning*, Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems, International Foundation for Autonomous Agents and Multiagent Systems, 2018, pp. 434–442.

[176] Bradly C Stadie, Sergey Levine, and Pieter Abbeel, *Incentivizing exploration in reinforcement learning with deep predictive models*, arXiv preprint arXiv:1507.00814 (2015).

[177] Peter Stone, *Multiagent learning is not the answer. it is the question*, Artificial Intelligence **171** (2007), no. 7, 402–405.

[178] Peter Stone and Manuela Veloso, *Multiagent systems: A survey from a machine learning perspective*, Autonomous Robots **8** (2000), no. 3, 345–383.

[179] Sainbayar Sukhbaatar, Rob Fergus, et al., *Learning multiagent communication with backpropagation*, Advances in Neural Information Processing Systems, 2016, pp. 2244–2252.

[180] Xin Sun, Junyu Shi, Junyu Dong, and Xinhua Wang, *Fish recognition from low-resolution underwater images*, Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI), International Congress on, IEEE, 2016, pp. 471–476.

[181] Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, and Thore Graepel, *Value-decomposition networks for cooperative multi-agent learning*, arXiv preprint arXiv:1706.05296 (2017).

[182] Richard S Sutton, *Generalization in reinforcement learning: Successful examples using sparse coarse coding*, Advances in neural information processing systems, 1996, pp. 1038–1044.

[183] Richard S Sutton and Andrew G Barto, *Introduction to reinforcement learning*, vol. 135, MIT press Cambridge, 1998.

[184] ———, *Reinforcement learning: An introduction*, MIT press, 2018.

[185] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour, *Policy gradient methods for reinforcement learning with function approximation*, Advances in neural information processing systems, 2000, pp. 1057–1063.

[186] Richard S Sutton, Doina Precup, and Satinder Singh, *Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning*, Artificial intelligence **112** (1999), no. 1-2, 181–211.

[187] Ardi Tampuu, Tambet Matiisen, Dorian Kodelja, Ilya Kuzovkin, Kristjan Korjus, Juhan Aru, Jaan Aru, and Raul Vicente, *Multiagent cooperation and competition with deep reinforcement learning*, PloS one **12** (2017), no. 4, e0172395.

[188] Ming Tan, *Multi-agent reinforcement learning: Independent vs. cooperative agents*, Proceedings of the tenth international conference on machine learning, 1993, pp. 330–337.

[189] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, OpenAI Xi Chen, Yan Duan, John Schulman, Filip DeTurck, and Pieter Abbeel, *# exploration: A study of count-based exploration for deep reinforcement learning*, Proc. of NIPS, 2017, pp. 2750–2759.

[190] Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel, *# exploration: A study of count-based exploration for deep reinforcement learning*, arXiv preprint arXiv:1611.04717 (2016).

[191] Gerald Tesauro, *Extending q-learning to general adaptive multi-agent systems*, Advances in neural information processing systems, 2004, pp. 871–878.

[192] Chen Tessler, Yonathan Efroni, and Shie Mannor, *Action robust reinforcement learning and applications in continuous control*, arXiv preprint arXiv:1901.09184 (2019).

[193] Michel Tokic and Günther Palm, *Value-difference based exploration: adaptive control between epsilon-greedy and softmax*, Annual Conference on Artificial Intelligence, Springer, 2011, pp. 335–346.

[194] Karl Tuyls and Ann Nowé, *Evolutionary game theory and multi-agent reinforcement learning*, The Knowledge Engineering Review **20** (2005), no. 01, 63–90.

[195] Karl Tuyls and Simon Parsons, *What evolutionary game theory tells us about multiagent learning*, Artificial Intelligence **171** (2007), no. 7, 406–416.

[196] Karl Tuyls, Julien Pérolat, Marc Lanctot, Georg Ostrovski, Rahul Savani, Joel Z Leibo, Toby Ord, Thore Graepel, and Shane Legg, *Symmetric decomposition of asymmetric games*, Scientific reports **8** (2018), no. 1, 1015.

[197] Karl Tuyls and P Stone, *Multiagent learning paradigms*, Multi-Agent Systems and Agreement Technologies, Springer, 2017, pp. 3–21.

[198] Karl Tuyls, Katja Verbeeck, and Tom Lenaerts, *A selection-mutation model for q-learning in multi-agent systems*, Proceedings of the second international joint conference on Autonomous agents and multiagent systems, ACM, 2003, pp. 693–700.

[199] Karl Tuyls and Gerhard Weiss, *Multiagent learning: Basics, challenges, and prospects*, Ai Magazine **33** (2012), no. 3, 41–41.

[200] Karl Tuyls and Gerhard Weiss, *Multiagent learning: Basics, challenges, and prospects*, AI Magazine **33** (2012), no. 3, 41–52.

[201] Hado Van Hasselt, *Double q-learning*, Proc. of NIPS, 2010, pp. 2613–2621.

[202] Hado Van Hasselt, Arthur Guez, and David Silver, *Deep reinforcement learning with double Q-learning*, CoRR, abs/1509.06461 (2015).

[203] ———, *Deep reinforcement learning with double Q-Learning.*, AAAI (2016), 2094–2100.

[204] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol, *Extracting and composing robust features with denoising autoencoders*, Proceedings of the 25th international conference on Machine learning, ACM, 2008, pp. 1096–1103.

[205] Oriol Vinyals, Timo Ewalds, Sergey Bartunov, Petko Georgiev, Alexander Sasha Vezhnevets, Michelle Yeo, Alireza Makhzani, Heinrich Küttler, John Agapiou, Julian Schrittwieser, et al., *Starcraft ii: A new challenge for reinforcement learning*, arXiv preprint arXiv:1708.04782 (2017).

[206] Okko Jan Vrieze and Frank Thuijsman, *On equilibria in repeated games with absorbing states*, International Journal of Game Theory **18** (1989), no. 3, 293–310.

[207] Christopher JCH Watkins and Peter Dayan, *Q-learning*, Machine learning **8** (1992), no. 3-4, 279–292.

[208] Christopher John Cornish Hellaby Watkins, *Learning from delayed rewards*, (1989).

[209] Ermo Wei and Sean Luke, *Lenient learning in independent-learner stochastic cooperative games*, JMLR **17** (2016), no. 84, 1–42.

[210] Ermo Wei, Drew Wicke, David Freelan, and Sean Luke, *Multiagent Soft Q-Learning*, arXiv preprint arXiv:1804.09817 (2018).

[211] Jörgen W Weibull, *Evolutionary game theory*, MIT press, 1997.

[212] R Paul Wiegand, *An analysis of cooperative coevolutionary algorithms*, Ph.D. thesis, Citeseer, 2003.

[213] David H Wolpert and Kagan Tumer, *Optimal payoff functions for members of collectives*, Modeling complexity in economic and social systems, World Scientific, 2002, pp. 355–369.

[214] Michael J Wooldridge, *An introduction to multiagent systems.*, John Wiley & Sons, 2009.

[215] Zifeng Wu, Chunhua Shen, and Anton Van Den Hengel, *Wider or deeper: Revisiting the resnet model for visual recognition*, Pattern Recognition **90** (2019), 119–133.

[216] Michael Wunder, Michael L Littman, and Monica Babes, *Classes of multiagent q-learning dynamics with epsilon-greedy exploration*, Proceedings of the 27th International Conference on Machine Learning (ICML-10), Citeseer, 2010, pp. 1167–1174.

[217] Yinliang Xu, Wei Zhang, Wenxin Liu, and Frank Ferrese, *Multiagent-based reinforcement learning for optimal reactive power dispatch*, IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews) **42** (2012), no. 6, 1742–1751.

[218] Logan Yliniemi, Adrian K Agogino, and Kagan Tumer, *Multirobot coordination for space exploration*, AI Magazine **35** (2014), no. 4, 61–74.

[219] H Peyton Young, *Monotonic solutions of cooperative games*, International Journal of Game Theory **14** (1985), no. 2, 65–72.

[220] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria, *Recent trends in deep learning based natural language processing*, ieee Computational intelligenCe magazine **13** (2018), no. 3, 55–75.

[221] Yan Zheng, Zhaopeng Meng, Jianye Hao, and Zongzhang Zhang, *Weighted double deep multiagent reinforcement learning in stochastic cooperative environments*, Pacific Rim International Conference on Artificial Intelligence, 2018, pp. 421–429.